

Análisis Numérico de Isotermas de Adsorción en Café Tostado (Ajuste de Modelos No Lineales)

Métodos Numéricos

Docente: Dra. Alma Xóchitl González Morales

Universidad de Guanajuato – Campus León
División de Ciencias e Ingenierías
León, Guanajuato, México

Noviembre 2025

Análisis Numérico de Isotermas de Adsorción en Café Tostado

Oliva-Villar David Isaac

División de Ciencias e Ingenierías Campus León, Universidad de Guanajuato

(Fecha: 24 de Noviembre de 2025)

Resumen

El presente trabajo analiza las isotermas de adsorción de agua en café tostado (grano entero, molido medio y molido fino) a temperaturas de 25, 35 y 45 °C, tomando resultados experimentales reales de un documento brindado en clase. Los datos experimentales se implementaron para ajustar dos modelos matemáticos: el modelo de Peleg y el modelo Polinomio Doble Logaritmo (DLP). El ajuste se realizó mediante la minimización de la cantidad χ^2 utilizando para esto algoritmos de optimización no lineal y regresión lineal. Los resultados demuestran que el modelo de Peleg ofrece una descripción superior de las curvas experimentales en todos los casos estudiados. Se incluyen implementaciones computacionales en Python y C (usando GSL) para la resolución del problema.

1. Planteamiento del Problema

Las isotermas de adsorción describen la relación de equilibrio entre la actividad de agua (a_w) y el contenido de humedad (X_e) de un material higroscópico a temperatura constante. Para el café de especialidad, comprender este comportamiento es crucial para determinar la vida útil y las condiciones óptimas de almacenamiento.

El objetivo es reproducir los gráficos experimentales y encontrar los parámetros óptimos (b_0, b_1, b_2, b_3) que minimicen el error cuadrático (χ^2) para dos modelos empíricos:

1.1. Modelo de Peleg

El modelo de Peleg es una ecuación no lineal de cuatro parámetros ampliamente utilizada en alimentos:

$$X_e = b_0 a_w^{b_1} + b_2 a_w^{b_3} \quad (1)$$

Donde X_e es el contenido de humedad en base seca y a_w es la actividad de agua.

1.2. Modelo DLP (Double Log Polynomial)

El modelo DLP se basa en una transformación logarítmica doble:

$$X_e = b_0 + b_1 \chi + b_2 \chi^2 + b_3 \chi^3 \quad (2)$$

donde $\chi = \ln(-\ln(a_w))$

Este modelo es lineal en sus parámetros una vez realizada la transformación de la variable independiente.

2. Metodología Numérica

Para determinar qué modelo describe mejor los datos, se define la función objetivo χ^2 (Chi-cuadrado o SSE) a minimizar:

$$\chi^2 = \sum_{i=1}^N (X_{e,\text{exp}}^{(i)} - X_{e,\text{calc}}^{(i)})^2 \quad (3)$$

- Para el **Modelo DLP**, al ser lineal en los parámetros b_i , se puede resolver mediante mínimos cuadrados lineales (o regresión polinómica de grado 3 sobre la variable χ).
- Para el **Modelo Peleg**, al ser no lineal (los parámetros b_1 y b_3 son exponentes), se requiere un método iterativo. Se utilizó el algoritmo de **Levenberg-Marquardt** (disponible en `scipy.optimize` de Python y en la librería GSL de C).

3. Resultados y Discusión

Se procesaron los datos para tres tipos de café (Roasted Beans, Ground-Medium, Ground-Fine) a tres temperaturas. A continuación se presentan los parámetros optimizados y la bondad de ajuste (χ^2).

Cuadro 1: Comparación de ajuste (χ^2) y parámetros obtenidos. El mejor modelo se resalta en negrita.

Tipo	Temp (°C)	Mejor Modelo	χ^2 (Peleg)	χ^2 (DLP)	Parámetros (Mejor Modelo)			
					b_0	b_1	b_2	b_3
Roasted	25	Peleg	39.81	102.23	2.7310	0.1517	1148.48	46.98
Roasted	35	Peleg	22.77	23.80	2.3577	0.0702	27.59	12.79
Roasted	45	Peleg	9.44	10.45	2.3184	0.1000	28.55	10.03
Medium	25	Peleg	104.91	121.68	2.3149	0.2120	47.26	10.95
Medium	35	Peleg	8.06	19.47	2.3267	0.1813	23.66	6.26
Medium	45	Peleg	9.15	17.95	1.4905	0.2563	21.51	4.77
Fine	25	Peleg	82.15	130.30	2.0882	0.4326	38.24	8.94
Fine	35	Peleg	17.12	37.09	1.7711	0.3508	24.08	5.31
Fine	45	Peleg	20.74	27.14	0.8389	0.4967	20.79	4.12

3.1. Análisis Gráfico

La Figura 1, muestra la reproducción de los gráficos A, B y C correspondientes a la Figura 2 del artículo original. Se observa cómo el contenido de humedad aumenta con la actividad de agua, siguiendo una isoterma, característica de productos alimenticios. Las líneas sólidas representan el ajuste del modelo de Peleg.

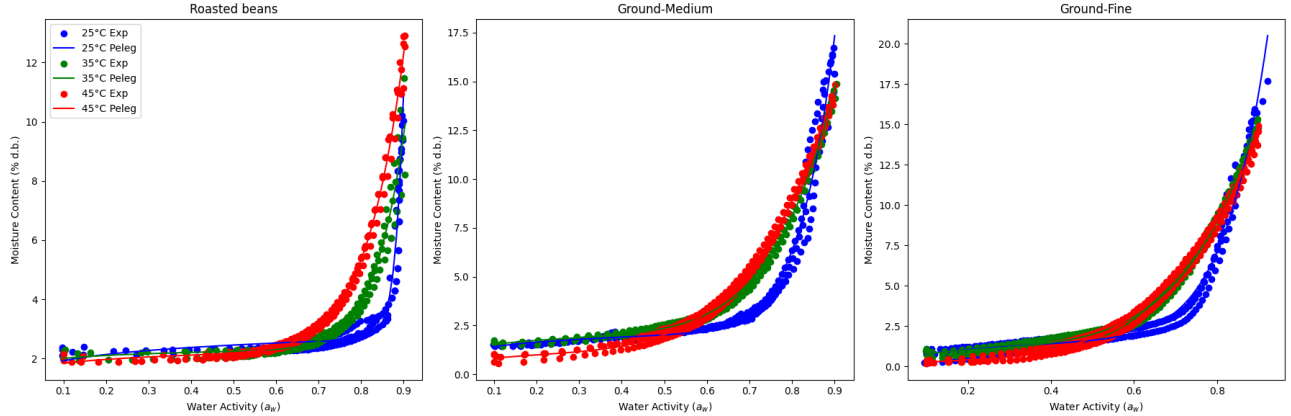


Figura 1: Isotermas de adsorción experimental (puntos) y modelo de Peleg ajustado (líneas) a 25, 35 y 45 °C

4. Conclusión

El análisis comparativo de los métodos numéricos evidenció que el ajuste no lineal (modelo de Peleg), implementado mediante algoritmos iterativos de gradiente, minimiza la función objetivo χ^2 de manera más efectiva que la regresión lineal por mínimos cuadrados (modelo DLP) en todos los escenarios evaluados. Aunque la linealización del modelo DLP permite una solución analítica directa y computacionalmente más sencilla de implementar, puesto que al no ser un método iterativo no mantiene trabajando al equipo como el otro método, pero introduce sesgos que limitan la precisión del ajuste. Por el contrario, la optimización no lineal, validada mediante las implementaciones en Python y C (GSL), demostró ser el método numérico más acertado para este tipo de problemas, logrando una convergencia estable y una reducción superior del error residual a pesar del mayor trabajo en ámbito computacional asociado a los métodos iterativos.

5. Códigos

A continuación se presentan los códigos desarrollados para resolver el problema de minimización planteado.

5.1. Código en Python (Ajuste y Gráficos)

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 from scipy.optimize import curve_fit
5 from google.colab import files
6 import io
7
8 # DATOS
9 print("Por favor, sube el archivo CSV de datos aqui:")
10 uploaded = files.upload()
11
12 # Leer el archivo subido (toma el primero que encuentre)
13 filename = next(iter(uploaded))
14 try:
15     df = pd.read_csv(io.BytesIO(uploaded[filename]))
16     print(f"\nArchivo '{filename}' cargado exitosamente.")
17 except Exception as e:
18     print(f"Error al leer el archivo: {e}")
19     print("Asegurate de subir un archivo CSV valido.")
20
21 # DEFINICION DE MODELOS
22
23 # Modelo Peleg:  $X_e = b_0 * a_w^{b_1} + b_2 * a_w^{b_3}$ 
24 def peleg_model(aw, b0, b1, b2, b3):
25     return b0 * (aw ** b1) + b2 * (aw ** b3)
26
27 # Modelo DLP:  $X_e = b_0 + b_1 * \chi + b_2 * \chi^2 + b_3 * \chi^3$ 
28 # Donde  $\chi = \ln(-\ln(a_w))$ 
29 def dlp_model_func(aw, b0, b1, b2, b3):
30     # Evitar log(0) o valores invalidos
31     valid_mask = (aw > 0) & (aw < 1)
32     chi = np.zeros_like(aw)
33     chi[valid_mask] = np.log(-np.log(aw[valid_mask]))
34     chi[~valid_mask] = np.nan
35     return b0 + b1 * chi + b2 * (chi**2) + b3 * (chi**3)
36
37 # PROCESAMIENTO Y AJUSTE
38
39 # Configuración de graficas
40 tipos_cafe = ['Roasted beans', 'Ground-Medium', 'Ground-Fine']
41 temperaturas = [25, 35, 45]
42 colores = {25: 'blue', 35: 'green', 45: 'red'}
43
44 fig, axes = plt.subplots(1, 3, figsize=(18, 6))
45 resultados = []
46
47 print("\nRealizando ajustes de curvas...\n")
48
49 for i, tipo in enumerate(tipos_cafe):
50     ax = axes[i]
51     datos_tipo = df[df['Type'] == tipo]
52
53     for temp in temperaturas:
54         subset = datos_tipo[datos_tipo['Temperature'] == temp]
```

```

56     if subset.empty:
57         continue
58
59     aw_exp = subset['Water activity'].values
60     me_exp = subset['Moisture content (% dry basis)'].values # Usamos base seca
61
62     chi_exp = np.log(-np.log(aw_exp))
63     try:
64         # Polyfit devuelve [b3, b2, b1, b0]
65         p_dlp = np.polyfit(chi_exp, me_exp, 3)
66         params_dlp = [p_dlp[3], p_dlp[2], p_dlp[1], p_dlp[0]]
67
68         # Calcular Chi2
69         me_pred_dlp = dlp_model_func(aw_exp, *params_dlp)
70         chi2_dlp = np.sum((me_exp - me_pred_dlp)**2)
71     except:
72         params_dlp = [np.nan]*4
73         chi2_dlp = np.inf
74
75     # Ajuste Peleg
76     # Estimacion inicial para ayudar a la convergencia
77     p0_peleg = [2.0, 0.5, 25.0, 10.0]
78     try:
79         popt_peleg, _ = curve_fit(peleg_model, aw_exp, me_exp, p0=p0_peleg,
80                                 maxfev=20000)
81
82         # Calcular Chi2 (SSE)
83         me_pred_peleg = peleg_model(aw_exp, *popt_peleg)
84         chi2_peleg = np.sum((me_exp - me_pred_peleg)**2)
85     except:
86         popt_peleg = [np.nan]*4
87         chi2_peleg = np.inf
88
89     # Determinar el mejor modelo
90     mejor_modelo = "Peleg" if chi2_peleg < chi2_dlp else "DLP"
91
92     # Guardar resultados
93     resultados.append({
94         'Tipo': tipo,
95         'Temp': temp,
96         'Peleg Params': popt_peleg,
97         'Peleg Chi2': chi2_peleg,
98         'DLP Params': params_dlp,
99         'DLP Chi2': chi2_dlp,
100        'Mejor Modelo': mejor_modelo
101    })
102
103     # Graficar
104     # Puntos experimentales
105     ax.scatter(aw_exp, me_exp, color=colores[temp], label=f'{temp} C Exp', marker
106               = 'o')
107
108     aw_suave = np.linspace(min(aw_exp), max(aw_exp), 100)
109
110     if mejor_modelo == "Peleg":
111         y_suave = peleg_model(aw_suave, *popt_peleg)
112         estilo_linea = '-'
113     else:
114         y_suave = dlp_model_func(aw_suave, *params_dlp)
115         estilo_linea = '--'

```

```

115         ax.plot(aw_suave, y_suave, color=colores[temp], linestyle=estilo_linea, label
116                 =f'{temp} C {mejor_modelo}')
117
118     ax.set_title(tipo)
119     ax.set_xlabel('Water Activity ($a_w$)')
120     ax.set_ylabel('Moisture Content (% d.b.)')
121     if i == 0:
122         ax.legend()
123
124 plt.tight_layout()
125 plt.show()
126
127 # TABLA DE RESULTADOS
128
129 res_df = pd.DataFrame(resultados)
130
131 tabla_final = res_df[['Tipo', 'Temp', 'Peleg Chi2', 'DLP Chi2', 'Mejor Modelo']].copy()
132
133 print("\n=== COMPARACION DE MODELOS (Menor Chi2 es mejor) ===")
134 print(tabla_final)
135
136 print("\n=== PARAMETROS DEL MODELO PELEG (b0, b1, b2, b3) ===")
137 for index, row in res_df.iterrows():
138     params = row['Peleg Params']
139     if not np.isnan(params[0]):
140         print(f"{row['Tipo']} {row['Temp']} C: [{params[0]:.4f}, {params[1]:.4f}, {
141             params[2]:.4f}, {params[3]:.4f}]")

```

Listing 1: Script Python para ajuste de curvas Peleg y DLP

5.2. Código en C (Minimización GSL)

Para el modelo no lineal de Peleg en C, se utiliza la librería científica GNU (GSL).

```

1  #include <stdio.h>
2  #include <math.h>
3  #include <gsl/gsl_vector.h>
4  #include <gsl/gsl_multifit_nlinear.h>
5
6  struct data {
7      size_t n;
8      double *aw;
9      double *me;
10 };
11
12 // Funcion de residuos para Peleg: f_i = y_model - y_obs
13 int peleg_f(const gsl_vector * x, void *data, gsl_vector * f) {
14     size_t n = ((struct data *)data)->n;
15     double *aw = ((struct data *)data)->aw;
16     double *me = ((struct data *)data)->me;
17
18     double b0 = gsl_vector_get(x, 0);
19     double b1 = gsl_vector_get(x, 1);
20     double b2 = gsl_vector_get(x, 2);
21     double b3 = gsl_vector_get(x, 3);
22
23     for (size_t i = 0; i < n; i++) {
24         double Yi = b0 * pow(aw[i], b1) + b2 * pow(aw[i], b3);
25         gsl_vector_set(f, i, Yi - me[i]);
26     }
27     return GSL_SUCCESS;
28 }

```

```

29
30 void solve_peleg(double *aw_data, double *me_data, size_t n) {
31     // Configuración del solver GSL (Trust Region)
32     const gsl_multifit_nlinear_type *T = gsl_multifit_nlinear_trust;
33     gsl_multifit_nlinear_workspace *w;
34     gsl_multifit_nlinear_fdf fdf;
35     gsl_multifit_nlinear_parameters fdf_params =
36         gsl_multifit_nlinear_default_parameters();
37
38     struct data d = { n, aw_data, me_data };
39     double x_init[4] = { 2.0, 0.5, 20.0, 10.0 }; // Guess inicial
40     gsl_vector_view x = gsl_vector_view_array(x_init, 4);
41
42     fdf.f = peleg_f;
43     fdf.df = NULL; // Derivadas numericas
44     fdf.n = n;
45     fdf.p = 4; // 4 parametros
46     fdf.params = &d;
47
48     w = gsl_multifit_nlinear_alloc(T, &fdf_params, n, 4);
49     gsl_multifit_nlinear_init(&x.vector, &fdf, w);
50
51     int status, info;
52     // Iterar hasta convergencia
53     status = gsl_multifit_nlinear_driver(100, 1e-8, 1e-8, 1e-8, NULL, NULL, &info, w)
54         ;
55
56     printf("Estado: %s\n", gsl_strerror(status));
57     printf("Params Peleg: b0=%g, b1=%g, b2=%g, b3=%g\n",
58         gsl_vector_get(w->x, 0), gsl_vector_get(w->x, 1),
59         gsl_vector_get(w->x, 2), gsl_vector_get(w->x, 3));
60
61     gsl_multifit_nlinear_free(w);
62 }

```

Listing 2: Programa en C usando GSL para minimizar Chi2 en modelo Peleg