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 implemeron 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:

$$X_e = b_0 a_w^{b_1} + b_2 a_w^{b_3} \tag{1}$$

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:

$$X_e = b_0 + b_1\chi + b_2\chi^2 + b_3\chi^3 \tag{2}$$
$$\text{donde } \chi = \ln(-\ln(a_w))$$

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:

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

- 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**	40.54	103.12	2.7304	0.1515	1162.1587	47.0871
Roasted	35	**Peleg**	22.74	23.78	2.3557	0.0695	27.4878	12.7665
Roasted	45	**Peleg**	9.64	10.60	2.3200	0.1005	28.5896	10.0415
Medium	25	**Peleg**	105.15	121.98	2.3150	0.2121	47.2828	10.9603
Medium	35	**Peleg**	8.02	19.31	2.3291	0.1821	23.6674	6.2645
Medium	45	**Peleg**	9.21	18.03	1.4873	0.2551	21.4953	4.7650
Fine	25	**Peleg**	82.05	130.55	2.0865	0.4321	38.2058	8.9375
Fine	35	**Peleg**	16.95	36.70	1.7751	0.3524	24.0714	5.3135
Fine	45	**Peleg**	20.83	27.16	20.7949	4.1268	0.8442	0.5007

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.

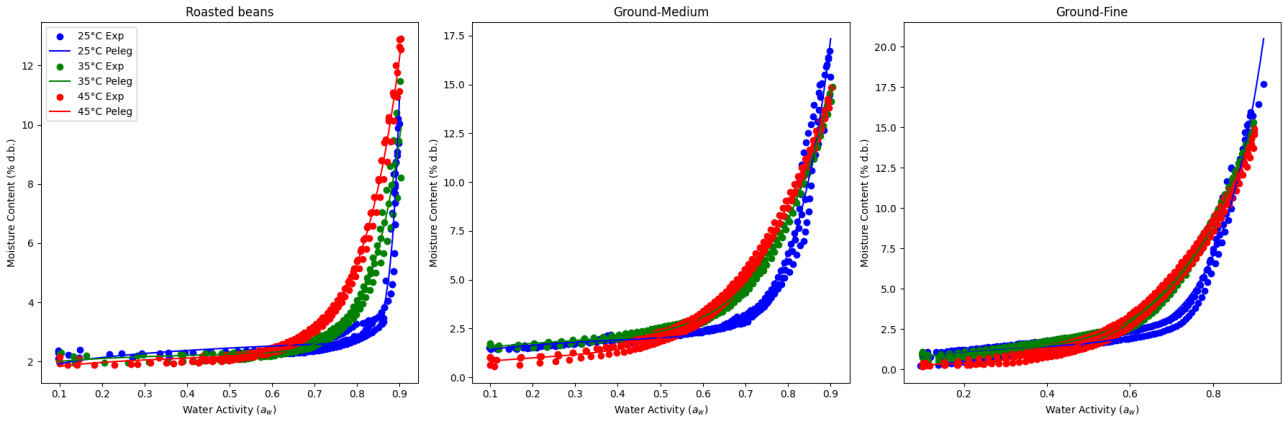

Figura 1: Isotermas de adsorción experimental (puntos) y modelo de Peleg ajustado (líneas) a 25, 35 y 45 °C

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)

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from google.colab import files
import io

# DATOS
print("Por favor, sube el archivo CSV de datos aqui:")
uploaded = files.upload()

# Leer el archivo subido (toma el primero que encuentre)
filename = next(iter(uploaded))
try:
    df = pd.read_csv(io.BytesIO(uploaded[filename]))
    print(f"\nArchivo '{filename}' cargado exitosamente.")
except Exception as e:
    print(f"Error al leer el archivo: {e}")
    print("Asegurate de subir un archivo CSV valido.")

# DEFINICION DE MODELOS

# Modelo Peleg: Xe = b0*aw^b1 + b2*aw^b3
def peleg_model(aw, b0, b1, b2, b3):
    return b0 * (aw ** b1) + b2 * (aw ** b3)

# Modelo DLP: Xe = b0 + b1*chi + b2*chi^2 + b3*chi^3
# Donde chi = ln(-ln(aw))
def dlp_model_func(aw, b0, b1, b2, b3):
    # Evitar log(0) o valores invalidos
    valid_mask = (aw > 0) & (aw < 1)
    chi = np.zeros_like(aw)
    chi[valid_mask] = np.log(-np.log(aw[valid_mask]))
    chi[~valid_mask] = np.nan
    return b0 + b1 * chi + b2 * (chi**2) + b3 * (chi**3)

# PROCESAMIENTO Y AJUSTE

# Configuracion de graficas
tipos_cafe = ['Roasted beans', 'Ground-Medium', 'Ground-Fine']
temperaturas = [25, 35, 45]
colores = {25: 'blue', 35: 'green', 45: 'red'}

fig, axes = plt.subplots(1, 3, figsize=(18, 6))
resultados = []

print("\nRealizando ajustes de curvas...\n")

for i, tipo in enumerate(tipos_cafe):
    ax = axes[i]
    datos_tipo = df[df['Type'] == tipo]

    for temp in temperaturas:
        subset = datos_tipo[datos_tipo['Temperature'] == temp]

```

```
        if subset.empty:
            continue

        aw_exp = subset['Water activity'].values
        me_exp = subset['Moisture content (% dry basis)'].values # Usamos base seca

        chi_exp = np.log(-np.log(aw_exp))
        try:
            # Polyfit devuelve [b3, b2, b1, b0]
            p_dlp = np.polyfit(chi_exp, me_exp, 3)
            params_dlp = [p_dlp[3], p_dlp[2], p_dlp[1], p_dlp[0]]

            # Calcular Chi2
            me_pred_dlp = dlp_model_func(aw_exp, *params_dlp)
            chi2_dlp = np.sum((me_exp - me_pred_dlp)**2)
        except:
            params_dlp = [np.nan]*4
            chi2_dlp = np.inf

        # Ajuste Peleg
        # Estimacion inicial para ayudar a la convergencia
        p0_peleg = [2.0, 0.5, 25.0, 10.0]
        try:
            popt_peleg, _ = curve_fit(peleg_model, aw_exp, me_exp, p0=p0_peleg,
                maxfev=20000)

            # Calcular Chi2 (SSE)
            me_pred_peleg = peleg_model(aw_exp, *popt_peleg)
            chi2_peleg = np.sum((me_exp - me_pred_peleg)**2)
        except:
            popt_peleg = [np.nan]*4
            chi2_peleg = np.inf

        # Determinar el mejor modelo
        mejor_modelo = "Peleg" if chi2_peleg < chi2_dlp else "DLP"

        # Guardar resultados
        resultados.append({
            'Tipo': tipo,
            'Temp': temp,
            'Peleg Params': popt_peleg,
            'Peleg Chi2': chi2_peleg,
            'DLP Params': params_dlp,
            'DLP Chi2': chi2_dlp,
            'Mejor Modelo': mejor_modelo
        })

        # Graficar
        # Puntos experimentales
        ax.scatter(aw_exp, me_exp, color=colores[temp], label=f'{temp} C Exp', marker
            ='o')

        aw_suave = np.linspace(min(aw_exp), max(aw_exp), 100)

        if mejor_modelo == "Peleg":
            y_suave = peleg_model(aw_suave, *popt_peleg)
            estilo_linea = '-'
        else:
            y_suave = dlp_model_func(aw_suave, *params_dlp)
            estilo_linea = '--'
```

```
        ax.plot(aw_suave, y_suave, color=colores[temp], linestyle=estilo_linea, label
            =f'{temp} C {mejor_modelo}')

    ax.set_title(tipo)
    ax.set_xlabel('Water Activity ($a_w$)')
    ax.set_ylabel('Moisture Content (% d.b.)')
    if i == 0:
        ax.legend()

plt.tight_layout()
plt.show()

# TABLA DE RESULTADOS

res_df = pd.DataFrame(resultados)

tabla_final = res_df[['Tipo', 'Temp', 'Peleg Chi2', 'DLP Chi2', 'Mejor Modelo']].copy
    ()
print("\n=== COMPARACION DE MODELOS (Menor Chi2 es mejor) ===")
print(tabla_final)

print("\n=== PARAMETROS DEL MODELO PELEG (b0, b1, b2, b3) ===")
for index, row in res_df.iterrows():
    params = row['Peleg Params']
    if not np.isnan(params[0]):
        print(f"{row['Tipo']} {row['Temp']} C: [{params[0]:.4f}, {params[1]:.4f}, {
            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).

```
#include <stdio.h>
#include <math.h>
#include <gsl/gsl_vector.h>
#include <gsl/gsl_multifit_nlinear.h>

struct data {
    size_t n;
    double *aw;
    double *me;
};

// Funcion de residuos para Peleg: f_i = y_model - y_obs
int peleg_f(const gsl_vector * x, void *data, gsl_vector * f) {
    size_t n = ((struct data *)data)->n;
    double *aw = ((struct data *)data)->aw;
    double *me = ((struct data *)data)->me;

    double b0 = gsl_vector_get(x, 0);
    double b1 = gsl_vector_get(x, 1);
    double b2 = gsl_vector_get(x, 2);
    double b3 = gsl_vector_get(x, 3);

    for (size_t i = 0; i < n; i++) {
        double Yi = b0 * pow(aw[i], b1) + b2 * pow(aw[i], b3);
        gsl_vector_set(f, i, Yi - me[i]);
    }
    return GSL_SUCCESS;
}
```

```c

void solve_peleg(double *aw_data, double *me_data, size_t n) {
    // Configuracion del solver GSL (Trust Region)
    const gsl_multifit_nlinear_type *T = gsl_multifit_nlinear_trust;
    gsl_multifit_nlinear_workspace *w;
    gsl_multifit_nlinear_fdf fdf;
    gsl_multifit_nlinear_parameters fdf_params =
        gsl_multifit_nlinear_default_parameters();

    struct data d = { n, aw_data, me_data };
    double x_init[4] = { 2.0, 0.5, 20.0, 10.0 }; // Guess inicial
    gsl_vector_view x = gsl_vector_view_array(x_init, 4);

    fdf.f = peleg_f;
    fdf.df = NULL; // Derivadas numericas
    fdf.n = n;
    fdf.p = 4; // 4 parametros
    fdf.params = &d;

    w = gsl_multifit_nlinear_alloc(T, &fdf_params, n, 4);
    gsl_multifit_nlinear_init(&x.vector, &fdf, w);

    int status, info;
    // Iterar hasta convergencia
    status = gsl_multifit_nlinear_driver(100, 1e-8, 1e-8, 1e-8, NULL, NULL, &info, w)
        ;

    printf("Estado: %s\n", gsl_strerror(status));
    printf("Params Peleg: b0=%g, b1=%g, b2=%g, b3=%g\n",
           gsl_vector_get(w->x, 0), gsl_vector_get(w->x, 1),
           gsl_vector_get(w->x, 2), gsl_vector_get(w->x, 3));

    gsl_multifit_nlinear_free(w);
}
```

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