UNIVERSIDAD DE GUANAJUATO

Universidad de Guanajuato, División de ciencias e ingenierías, Campus León.

Ingeniería Química Sustentable.

Métodos numéricos.

Proyecto Final: “Simulación numérica de una red de 4 CSTRs en serie para la hidrogenación de etileno”

Reporte final

Realizado por: María Irlanda Álvarez Banda

Fecha de realización: 30/10/2025

Fecha de entrega: 16/11/2025

Simulación numérica de una red de 4 CSTRs en serie para la hidrogenación de etileno.

Resumen

En este trabajo se implementaron y compararon tres métodos numéricos: el método de Gauss-Jordan, el método de Gauss-Seidel y el método de Descomposición LU para resolver un sistema de ecuaciones lineales que se genera en el modelado de una red de cuatro reactores CSTR en serie para la hidrogenación de etileno; los métodos fueron aplicados considerando un caudal constante de 1.0 m^3/s, una concentración de alimentación de 1.0 mol/m^3, así como volúmenes y constantes cinéticas propias de cada reactor. Los resultados mostraron concentraciones de etileno que disminuían a lo largo de la red de reactores: 0.909091, 0.829463, 0.748612 y 0.659570 mol/m^3 para los reactores correspondientes del 1 al 4, alcanzando una conversión global de 34,04%. Todos los métodos analizados mostraron resultados idénticos, verificando la exactitud numérica del modelo. En la comparación de los métodos de solución se observó que el método de Gauss-Seidel fue el más eficiente, alcanzando la convergencia en solo dos iteraciones, mientras que los métodos de Gauss-Jordan y de Descomposición LU también ofrecieron soluciones exactas con distintas ventajas computacionales. Este trabajo se presenta como un gran ejemplo de la eficacia de los métodos de solución numéricos completos para la simulación de sistemas de procesos químicos y como base para futuras optimizaciones del sistema.

Introducción

La hidrogenación de etileno para producir etano es una reacción de gran relevancia en la industria química, esta reacción es un ejemplo clásico y se utiliza frecuentemente en investigaciones para comprender los mecanismos de la catálisis heterogénea y el diseño de catalizadores. Esta puede modelarse y simularse en procesos de reactores ideales para optimizar su optimización y que la eficiencia sea la más alta posible.

Este proyecto tiene como finalidad modelar y resolver una red en serie de cuatro reactores de mezcla completa (CSTR) que operan en estado estacionario, donde se lleva a cabo la reacción antes mencionada. Como suposiciones tenemos un flujo volumétrico constante, cinética de pseudo primer orden y propiedades físicas constantes, se utilizará un sistema de ecuaciones lineales a partir de balances de materia.

La resolución del sistema se resolverá mediante tres métodos numéricos: eliminación de Gauss-Jordan, Gauss-Seidel y descomposición LU. Cada método tiene sus ventajas y desventajas, lo que permitirá analizar con más detenimiento cuál de los tres métodos es el óptimo y certero en este sistema.

Este trabajo busca determinar las concentraciones de etileno en cada reactor y a su vez evaluar el desempeño de cada uno de los métodos numéricos en la resolución de este

problema, contribuyendo a verificar, corroborar y seleccionar el mejor método por medio de programas empleados en el lenguaje C adecuados para su simulación.

Objetivos

- **General**

  Modelar y resolver por medio de métodos numéricos una red en serie de cuatro reactores de mezcla completa (CSTR) para la hidrogenación de etileno, con el fin de determinar las concentraciones de etileno en estado estacionario en cada reactor.

- **Específicos**
  1. Obtener los balances de materia para cada reactor en estado estacionario, a partir del sistema de ecuaciones lineales de la red de reactores.
  2. Implementar los métodos numéricos para la resolución del sistema: Eliminación de Gauss-Jordan, Gauss-Seidel y descomposición LU.
  3. Utilizar valores estandarizados para este sistema reportados en la literatura.
  4. Comparar los resultados obtenidos con cada método en términos de las concentraciones, qué tan rápido converge y la eficiencia del programa.
  5. Analizar y comparar las ventajas y desventajas de cada método en el contexto de este sistema.

Marco teórico

1. Hidrogenación de Etileno

La hidrogenación de etileno (C_2H_4) para formar etano (C_2H_6) es una reacción ampliamente estudiada en ingeniería química, que se lleva a cabo en presencia de un catalizador sólido. Esta reacción es representativa de los procesos de hidrogenación catalítica y se describe mediante la ecuación:

$$C_2H_4 + H_2 \rightarrow C_2H_6 \quad (1)$$

En condiciones de exceso de hidrógeno, la cinética puede aproximarse a un comportamiento de pseudo-primer orden con respecto al etileno, simplificando la velocidad de reacción [1][2].

2. Reactores de Mezcla Completa (CSTR)

Un reactor de mezcla completa (CSTR, por sus siglas en inglés) es un modelo ideal en el que la composición y la temperatura se consideran uniformes en todo el volumen del reactor. Esto implica que la concentración de salida es igual a la concentración en el interior del reactor [3].

En estado estacionario, el balance de masa para un reactivo A (en este caso el etileno) se expresa como:

$$0 = Qc_{in} - Qc - kcV \tag{2}$$

Donde Q es el flujo volumétrico, c_{in} y c son las concentraciones de entrada y salida, k es la constante cinética y V es el volumen del reactor [4].

3. Parámetros del sistema y valores de referencia

Para la simulación de la red de 4 CSTR en serie, se utilizarán los siguientes valores de referencia, representativos de condiciones típicas de operación en procesos de hidrogenación a escala industrial:

3.1 Flujo volumétrico

$$Q = 1.0\ m^3/s \tag{3}$$

Este valor representa un caudal moderado típico en plantas de procesamiento químico, asegurando condiciones de flujo turbulento que favorecen la mezcla completa en cada reactor [4].

3.2 Concentración de alimentación

$$c_{in} = 1.0\ mol/m^3 \tag{4}$$

Esta concentración inicial de etileno es la más usada en corrientes de alimentación en procesos de hidrogenación industrial [1].

3.3 Volúmenes de reactores

$$V = [1.0,1.2,0.9,1.5]\ m^3 \tag{5}$$

La variación en los volúmenes permite modelar una configuración realista donde los reactores pueden tener tamaños diferentes según su posición en la cadena [3].

3.4 Constantes cinéticas

$$k = [0.10,0.08,0.12,0.09]\ s^{-1} \tag{6}$$

Estos valores de constantes cinéticas de pseudo-primer orden reflejan variaciones típicas en la actividad catalítica entre diferentes reactores, debido a diferencias en temperatura o desgaste del catalizador [1,4].

4. Red de reactores en serie

Cuando múltiples CSTR se conectan en serie, la salida de un rector se convierte en la entrada del siguiente. Esta configuración permite alcanzar mayores conversiones globales en comparación a solo utilizar un reactor del mismo volumen total [4].

Para una red de n reactores, el balance de masa para el reactor i es:

$$Qc_{i-1} - (Q + k_i V_i)c_i = 0 \tag{7}$$

donde:

- Q: caudal volumétrico constante (m^3/s)
- c_{i-1}: concentración de entrada al reactor i
- c_i: concentración de salida del reactor i
- k_i: constante cinética de pseudo-primer orden (s^{-1})
- V_i: volumen del reactor i (m^3)

Para cada reactor i en estado estacionario, el balance de masa para etileno (asumiendo mezclado perfecto, flujo de entrada desde el reactor anterior, reacción de primer orden, y flujo de salida al siguiente reactor) se escribe como:

$$\sum_{\text{entradas } j \to i} Q\, c_j - (Q + k_i V_i)\, c_i \;=\; 0 \tag{8}$$

Para el primer reactor (alimentado directamente) se tiene:

$$Q\, c_{in} - (Q + k_1\, V_1)\, c_1 = 0 \tag{9}$$

Para los reactores 2, 3 y 4:

$$Q\, c_{i-1} - (Q + k_i\, V_i)\, c_i = 0 \text{ para } i = 2,3,4. \tag{10}$$

Esto genera un sistema de ecuaciones lineales que puede resolverse de manera numérica para obtener las concentraciones en cada etapa [4].

5. Métodos numéricos para sistemas de ecuaciones lineales

5.1 Eliminación de Gauss-Jordan

Este método consiste en transformar la matriz aumentada del sistema en una forma escalonada reducida mediante operaciones elementales de fila, lo que permite obtener la solución. Es un método exacto, pero computacionalmente costoso para sistemas grandes [5].

5.2 Método de Gauss-Seidel

Es un método iterativo que actualiza secuencialmente cada variable utilizando los valores más recientes de las demás. Es eficiente para sistemas grandes y dispersos, pero su convergencia depende de la diagonal de la matriz [5].

5.3 Descomposición de LU

Este método factoriza la matriz del sistema en una matriz triangular inferior (L) y una superior (U), facilitando la resolución del sistema mediante sustitución hacia adelante y hacia atrás. Es estable para múltiples resoluciones con la misma matriz [5].

Metodología

1. **Implementación del modelo matemático**

A partir de los balances de materia, se construye el sistema de ecuaciones lineales $Ac = b$ correspondiente a la red de 4 reactores CSTR en serie. La matriz A y el vector b se definen según los parámetros anteriormente mencionados en la sección anterior.

2. **Métodos numéricos implementados**

Se implementarán y compararán los valores obtenidos con los siguientes métodos:

- Eliminación de Gauss-Jordan
- Método Gauss-Seidel
- Descomposición LU

3. **Programación en C**

El desarrollo de los algoritmos y la resolución del sistema se realizarán utilizando C, con apoyo de programas ya realizados anteriormente en sesiones de clase.

4. **Análisis de resultados**

Se evaluará la precisión de cada método comparando las concentraciones obtenidas, el tiempo de convergencia y el número de iteraciones entre cada método. Además, se analizará la estabilidad numérica y la eficiencia de cada programa.

Procedimiento

1. **Construcción del sistema de ecuaciones**

A partir de los balances de materia para cada reactor en estado estacionario, se derivó el siguiente sistema de ecuaciones lineales:

Para el reactor 1, a partir de la ecuación 9:

$$(Q + k_1 V_1)\, c_1 = Q\, c_{in} \tag{11}$$

Y para los siguientes reactores, se partió de la ecuación 10:

Reactor 2:

$$-Q\, c_1 - (Q + k_2 V_2)\, c_2 = 0 \tag{12}$$

Reactor 3:

$$-Q\, c_2 - (Q + k_3 V_3)\, c_3 = 0 \tag{12}$$

Reactor 4:

$$-Q\, c_3 - (Q + k_4 V_4)\, c_4 = 0 \tag{13}$$

Este Sistema se organizó en forma matricial $Ac = b$, donde:

$$A = \begin{pmatrix} Q + k_1 & 0 & 0 & 0 \\ -Q & Q + k_2 & 0 & 0 \\ 0 & -Q & Q + k_3 & 0 \\ 0 & 0 & -Q & Q + k_4 \end{pmatrix}, c = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}, b = \begin{pmatrix} Q_{c_{in}} \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{14}$$

2. **Sustitución de valores**

Utilizando los valores de referencia

- $Q = 1.0\ \mathrm{m^3/s}$
- $c_{in} = 1.0\ \mathrm{mol/m^3}$
- $V = [1.0, 1.2, 0.9, 1.5]\ \mathrm{m^3}$
- $k = [0.10, 0.08, 0.12, 0.09]\ \mathrm{s^{-1}}$

Se obtuvo las siguientes matrices, basándonos en la 14:

$$A = \begin{pmatrix} 1.10 & 0 & 0 & 0 \\ -1.0 & 1.096 & 0 & 0 \\ 0 & -1 & 1.108 & 0 \\ 0 & 0 & -1 & 1.135 \end{pmatrix}, b = \begin{pmatrix} 1.0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

3. Implementación de los métodos numéricos

3.1 Método de eliminación de Gauss-Jordan

Dentro del programa en C, se incluyó:

- Formación de la matriz $[A|b]$
- Pivoteo
- Eliminación hacia adelante y hacia atrás
- Normalización de la diagonal principal

3.2 Método de Gauss-Seidel

Dentro del algoritmo en el programa, se encuentra:

- Vector inicial: $c^{(0)} = [0,0,0,0]^T$
- Criterio de convergencia: $\epsilon = 1 \times 10^{-6}$
- Fórmula de actualización por componente:

$$c_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} A_{ij} c_j^{(k+1)} - \sum_{j=i+1}^{n} A_{ij} c_j^{(k)}}{A_{ii}} \quad (15)$$

3.3 Método de descomposición LU

Se implemento mediante:

- Factorización $A = LU$
- Resolución de $Ly = b$ por sustitución hacia adelante
- Resolución de $Uc = y$ por sustitución hacia atrás

4. Programas en C

4.1 Método Gauss-Jordan

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

#define n 4  // 4 reactores

```

```c
int main() {
    float A[n][n+1];
    float x[n];
    int i, j, pivote, iter;
    int cambiosFila = 0;
    float determinante = 1.0;

    // Datos del problema
    float Q = 1.0;
    float c_in = 1.0;
    float V[4] = {1.0, 1.2, 0.9, 1.5};
    float k[4] = {0.10, 0.08, 0.12, 0.09};

    //Matriz A y vectro b
    printf("SISTEMA DE 4 REACTORES CSTR EN SERIE\n");
    printf("Datos del sistema:\n");
    printf("Q = %.2f m3/s, c_in = %.2f mol/m3\n", Q, c_in);
    printf("Volumenes: [%.1f, %.1f, %.1f, %.1f] m3\n", V[0], V[1],
V[2], V[3]);
    printf("Constantes k: [%.2f, %.2f, %.2f, %.2f] s-1\n\n", k[0],
k[1], k[2], k[3]);

    //Inicializar matriz A con ceros
    for (i = 0; i < n; i++) {
        for (j = 0; j < n + 1; j++) {
            A[i][j] = 0.0;
        }
    }

    //Reactor 1: (Q + k1*V1)*c1 = Q*c_in
    A[0][0] = Q + k[0] * V[0];
    A[0][4] = Q * c_in;  // vector b

    //Reactor 2: -Q*c1 + (Q + k2*V2)*c2 = 0
    A[1][0] = -Q;
    A[1][1] = Q + k[1] * V[1];
    A[1][4] = 0.0;

    //Reactor 3: -Q*c2 + (Q + k3*V3)*c3 = 0
    A[2][1] = -Q;
    A[2][2] = Q + k[2] * V[2];
    A[2][4] = 0.0;

    //Reactor 4: -Q*c3 + (Q + k4*V4)*c4 = 0
    A[3][2] = -Q;
```

```
    A[3][3] = Q + k[3] * V[3];
    A[3][4] = 0.0;

    //Matriz original
    printf("Matriz del sistema A|c:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%.6f ", A[i][j]);
        }
        printf("| %.6f\n", A[i][4]);
    }
    printf("\n");

    //GAUSS-JORDAN
    for (iter = 0; iter < n; iter++) {
        pivote = iter;

        //Pivote
        for (i = iter + 1; i < n; i++) {
            if (fabs(A[i][iter]) > fabs(A[pivote][iter])) {
                pivote = i;
            }
        }
        if (fabs(A[pivote][iter]) < 0.0001) {
            printf("\nError: Sistema singular - pivote cero en columna %d\n", iter + 1);
            return 1;
        }

        // Intercambio de filas
        if (pivote != iter) {
            for (j = 0; j < n + 1; j++) {
                float temp = A[iter][j];
                A[iter][j] = A[pivote][j];
                A[pivote][j] = temp;
            }
            cambiosFila++;
            printf("Intercambio de filas %d y %d\n", iter + 1, pivote + 1);
        }

        //Determinante
        determinante *= A[iter][iter];

        //Normalización en la fila del pivote
```

```
97.         float pivot_value = A[iter][iter];
98.         for (j = iter; j < n + 1; j++) {
99.             A[iter][j] /= pivot_value;
100.              }
101.
102.              // Eliminación en otras filas
103.              for (i = 0; i < n; i++) {
104.                  if (i != iter) {
105.                      float factor = A[i][iter];
106.                      for (j = iter; j < n + 1; j++) {
107.                          A[i][j] -= factor * A[iter][j];
108.                      }
109.                  }
110.              }  // ¡AQUÍ FALTABA ESTA LLAVE!
111.          }  // Esta cierra el for principal de Gauss-Jordan
112.
113.          //En caso, ajustar signo del determinante por intercambios
     de fila
114.          if (cambiosFila % 2 == 1) {
115.              determinante = -determinante;
116.          }
117.
118.          //Extraer soluciones
119.          for (i = 0; i < n; i++) {
120.              x[i] = A[i][n];
121.          }
122.
123.          //RESULTADOS
124.          printf("\nRESULTADOS FINALES\n");
125.
126.          printf("\nMatriz identidad | soluciones:\n");
127.          for (i = 0; i < n; i++) {
128.              for (j = 0; j < n; j++) {
129.                  printf("%.6f ", A[i][j]);
130.              }
131.              printf("| %.6f\n", A[i][n]);
132.          }
133.
134.          printf("\nDeterminante: %.6f\n", determinante);
135.
136.          if (fabs(determinante) < 0.0001) {
137.              printf("El sistema no esta determinado.\n");
138.          } else {
139.              printf("El sistema esta determinado.\n");
140.          }
```

```
141.
142.         printf("\nCONCENTRACIONES DE ETILENO EN CADA REACTOR:\n");
143.         for (i = 0; i < n; i++) {
144.             printf("Reactor %d: c%d = %.6f mol/m3\n", i + 1, i + 1,
    x[i]);
145.         }
146.
147.
148.         return 0;
149.     }
```

4.2 Método de Gauss-Seidel

```
5. #include <stdio.h>
6. #include <math.h>
7. #include <stdlib.h>
8.
9. #define n 4  // 4 reactores
10.
11.int main() {
12.     float A[n][n], b[n], x[n] = {0, 0, 0, 0};
13.     float x_anterior[n], suma;
14.     int i, j, iter;
15.     int max_iter = 100;
16.     float tolerancia = 1e-6;
17.
18.     //Datos del problema
19.     float Q = 1.0;
20.     float c_in = 1.0;
21.     float V[4] = {1.0, 1.2, 0.9, 1.5};
22.     float k[4] = {0.10, 0.08, 0.12, 0.09};
23.
24.     //Matriz A y vector b
25.     printf("METODO DE GAUSS-SEIDEL, 4 REACTORES CSTR \n");
26.     printf("Datos del sistema:\n");
27.     printf("Q = %.2f m3/s, c_in = %.2f mol/m3\n", Q, c_in);
28.     printf("Volumenes: [%.1f, %.1f, %.1f, %.1f] m3\n", V[0], V[1],
    V[2], V[3]);
29.     printf("Constantes k: [%.2f, %.2f, %.2f, %.2f] s-1\n\n", k[0],
    k[1], k[2], k[3]);
```

```
30.
31.     // Inicializar matriz A y vector b con ceros
32.     for (i = 0; i < n; i++) {
33.         for (j = 0; j < n; j++) {
34.             A[i][j] = 0.0;
35.         }
36.         b[i] = 0.0;
37.     }
38.
39.     // Llenar matriz A según el sistema de reactores
40.     // Reactor 1: (Q + k1*V1)*c1 = Q*c_in
41.     A[0][0] = Q + k[0] * V[0];
42.     b[0] = Q * c_in;
43.
44.     // Reactor 2: -Q*c1 + (Q + k2*V2)*c2 = 0
45.     A[1][0] = -Q;
46.     A[1][1] = Q + k[1] * V[1];
47.     b[1] = 0.0;
48.
49.     // Reactor 3: -Q*c2 + (Q + k3*V3)*c3 = 0
50.     A[2][1] = -Q;
51.     A[2][2] = Q + k[2] * V[2];
52.     b[2] = 0.0;
53.
54.     // Reactor 4: -Q*c3 + (Q + k4*V4)*c4 = 0
55.     A[3][2] = -Q;
56.     A[3][3] = Q + k[3] * V[3];
57.     b[3] = 0.0;
58.
59.     //Matriz original
60.     printf("Sistema de ecuaciones:\n");
61.     for (i = 0; i < n; i++) {
62.         printf("Ecuacion %d: ", i + 1);
63.         for (j = 0; j < n; j++) {
64.             if (A[i][j] != 0) {
65.                 if (j == 0)
66.                     printf("%.3f*c%d", A[i][j], j + 1);
67.                 else if (A[i][j] > 0)
68.                     printf(" + %.3f*c%d", A[i][j], j + 1);
69.                 else
70.                     printf(" - %.3f*c%d", -A[i][j], j + 1);
71.             }
72.         }
73.         printf(" = %.3f\n", b[i]);
74.     }
```

```
75.
76.     // Valores iniciales
77.     printf("\nValores iniciales: c1 = %.1f, c2 = %.1f, c3 = %.1f, c4 =
   %.1f\n",
78.            x[0], x[1], x[2], x[3]);
79.     printf("Tolerancia: %.2e\n", tolerancia);
80.     printf("Maximo de iteraciones: %d\n\n", max_iter);
81.
82.     //Método de Gauss-Seidel
83.     int convergio = 0;
84.     for (iter = 0; iter < max_iter; iter++) {
85.         //Guardar valores anteriores
86.         for (i = 0; i < n; i++) {
87.             x_anterior[i] = x[i];
88.         }
89.
90.         //Actualizar cada variable usando Gauss-Seidel
91.         for (i = 0; i < n; i++) {
92.             suma = 0.0;
93.
94.             //Sumar términos con coeficientes distintos de cero
95.             for (j = 0; j < n; j++) {
96.                 if (j != i) {
97.                     suma += A[i][j] * x[j]; // Usa valores nuevos si j
   < i, viejos si j > i
98.                 }
99.             }
100.                 if (fabs(A[i][i]) < 1e-10) {
101.                     printf("\nERROR: Division por cero en ecuacion
   %d\n", i + 1);
102.                     return 1;
103.                 }
104.
105.                 // Nuevo valor de x[i]
106.                 x[i] = (b[i] - suma) / A[i][i];
107.             }
108.
109.             // Calcular error
110.             float error_max = 0.0;
111.             for (i = 0; i < n; i++) {
112.                 float error = fabs(x[i] - x_anterior[i]);
113.                 if (error > error_max) {
114.                     error_max = error;
115.                 }
116.             }
```

```
117.
118.             //Iteracción
119.             printf("Iteracion %2d: ", iter + 1);
120.             for (i = 0; i < n; i++) {
121.                 printf("c%d = %8.6f  ", i + 1, x[i]);
122.             }
123.             printf("Error: %.2e\n", error_max);
124.
125.             //Covengencia
126.             if (error_max < tolerancia) {
127.                 convergio = 1;
128.                 break;
129.             }
130.         }
131.
132.         //RESULTADOS
133.         printf("\nRESULTADOS FINALES\n");
134.
135.         if (convergio) {
136.             printf("Convergencia alcanzada en %d iteraciones\n",
    iter + 1);
137.         } else {
138.             printf("Maximo de iteraciones alcanzado (%d)\n",
    max_iter);
139.         }
140.
141.         printf("\nCONCENTRACIONES DE ETILENO:\n");
142.         for (i = 0; i < n; i++) {
143.             printf("Reactor %d: c%d = %.6f mol/m3\n", i + 1, i + 1,
    x[i]);
144.         }
145.
146.         return 0;
147.
```

4.3 Método de descomposición LU

```
5  #include <stdio.h>
6  #include <math.h>
7  #include <stdlib.h>
8
9  #define n 4  // 4 reactores
10
11 int main() {
12     float A[n][n], L[n][n], U[n][n];
```

```
    float b[n], y[n], x[n];
    int i, j, k;

    //Datos del problema
    float Q = 1.0;
    float c_in = 1.0;
    float V[4] = {1.0, 1.2, 0.9, 1.5};
    float k_const[4] = {0.10, 0.08, 0.12, 0.09};  // Cambié el nombre para evitar conflicto

    //Inicializar matrices L y U
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i == j)
                L[i][j] = 1.0;  //Diagonal de L es 1
            else
                L[i][j] = 0.0;
            U[i][j] = 0.0;  //Inicializar U en 0
        }
    }

    printf("METODO DE DESCOMPOSICION LU - 4 REACTORES CSTR\n");
    printf("Datos del sistema:\n");
    printf("Q = %.2f m3/s, c_in = %.2f mol/m3\n", Q, c_in);
    printf("Volumenes: [%.1f, %.1f, %.1f, %.1f] m3\n", V[0], V[1], V[2], V[3]);
    printf("Constantes k: [%.2f, %.2f, %.2f, %.2f] s-1\n\n", k_const[0], k_const[1], k_const[2], k_const[3]);

    //Construcción automática de la matriz A y vector b
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            A[i][j] = 0.0;
        }
        b[i] = 0.0;
    }

    //Reactor 1: (Q + k1*V1)*c1 = Q*c_in
    A[0][0] = Q + k_const[0] * V[0];
    b[0] = Q * c_in;

    //Reactor 2: -Q*c1 + (Q + k2*V2)*c2 = 0
    A[1][0] = -Q;
    A[1][1] = Q + k_const[1] * V[1];
    b[1] = 0.0;
```

```
55
56      //Reactor 3: -Q*c2 + (Q + k3*V3)*c3 = 0
57      A[2][1] = -Q;
58      A[2][2] = Q + k_const[2] * V[2];
59      b[2] = 0.0;
60
61      //Reactor 4: -Q*c3 + (Q + k4*V4)*c4 = 0
62      A[3][2] = -Q;
63      A[3][3] = Q + k_const[3] * V[3];
64      b[3] = 0.0;
65
66      //Matriz original
67      printf("Matriz del sistema A | b:\n");
68      for (i = 0; i < n; i++) {
69          for (j = 0; j < n; j++) {
70              printf("%10.6f ", A[i][j]);
71          }
72          printf("| %10.6f\n", b[i]);
73      }
74
75      //Descomposición LU
76      printf("\nDESCOMPOSICION LU\n");
77      for (k = 0; k < n; k++) {
78          //Calculamos elementos de la fila k de U
79          for (j = k; j < n; j++) {
80              float suma = 0.0;
81              for (i = 0; i < k; i++) {
82                  suma += L[k][i] * U[i][j];
83              }
84              U[k][j] = A[k][j] - suma;
85              printf("U[%d][%d] = A[%d][%d] - (L[%d][i]*U[i][%d]) =
    %.6f\n",
86                     k+1, j+1, k+1, j+1, k+1, j+1, U[k][j]);
87          }
88
89          //Calculamos elementos de la columna k de L
90          for (i = k + 1; i < n; i++) {
91              float suma = 0.0;
92              for (j = 0; j < k; j++) {
93                  suma += L[i][j] * U[j][k];
94              }
95              if (fabs(U[k][k]) < 1e-10) {
96                  printf("\nError: Division por cero en U[%d][%d] =
    %.6f\n", k+1, k+1, U[k][k]);
97                  return 1;
```

```
98              }
99              L[i][k] = (A[i][k] - suma) / U[k][k];
100             printf("L[%d][%d] = (A[%d][%d] - (L[%d][j]*U[j][%d])) /
    U[%d][%d] = %.6f\n",
101                     i+1, k+1, i+1, k+1, i+1, k+1, k+1, k+1, L[i][k]);
102         }
103     }
104
105     //Matrices L y U
106     printf("\nMATRIZ L (Triangular Inferior)\n");
107     for (i = 0; i < n; i++) {
108         for (j = 0; j < n; j++) {
109             printf("%10.6f ", L[i][j]);
110         }
111         printf("\n");
112     }
113
114     printf("\nMATRIZ U (Triangular Superior)\n");
115     for (i = 0; i < n; i++) {
116         for (j = 0; j < n; j++) {
117             printf("%10.6f ", U[i][j]);
118         }
119         printf("\n");
120     }
121
122     //Ly = b
123     printf("\n Ly = b\n");
124     for (i = 0; i < n; i++) {
125         float suma = 0.0;
126         for (j = 0; j < i; j++) {
127             suma += L[i][j] * y[j];
128         }
129         y[i] = b[i] - suma;  // L[i][i] siempre es 1
130         printf("y[%d] = b[%d] - (L[%d][j]*y[j]) = %.6f\n", i + 1, i + 1,
    i + 1, y[i]);
131     }
132
133     //Ux = y
134     printf("\nUx = y\n");
135     for (i = n - 1; i >= 0; i--) {
136         float suma = 0.0;
137         for (j = i + 1; j < n; j++) {
138             suma += U[i][j] * x[j];
139         }
140         if (fabs(U[i][i]) < 1e-10) {
```

```
141             printf("\nError: Division por cero en U[%d][%d] = %.6f\n",
   i+1, i+1, U[i][i]);
142             return 1;
143         }
144
145         x[i] = (y[i] - suma) / U[i][i];
146         printf("c[%d] = (y[%d] - (U[%d][j]*c[j])) / U[%d][%d] = %.6f\n",
147                 i + 1, i + 1, i + 1, i + 1, i + 1, x[i]);
148     }
149
150     //Resultados
151     printf("\nRESULTADOS FINALES\n");
152     printf("\nCONCENTRACIONES DE ETILENO:\n");
153     for (i = 0; i < n; i++) {
154         printf("Reactor %d: c%d = %.6f mol/m3\n", i + 1, i + 1, x[i]);
155     }
156     return 0;
157}
```

Resultados

A continuación, se presentan los resultados de cada método:

Método Gauss-Jordan:

```
SISTEMA DE 4 REACTORES CSTR EN SERIE
Datos del sistema:
Q = 1.00 m3/s, c_in = 1.00 mol/m3
Volumenes: [1.0, 1.2, 0.9, 1.5] m3
Constantes k: [0.10, 0.08, 0.12, 0.09] s-1

Matriz del sistema A|c:
1.100000 0.000000 0.000000 0.000000 | 1.000000
-1.000000 1.096000 0.000000 0.000000 | 0.000000
0.000000 -1.000000 1.108000 0.000000 | 0.000000
0.000000 0.000000 -1.000000 1.135000 | 0.000000


RESULTADOS FINALES

Matriz identidad | soluciones:
1.000000 0.000000 0.000000 0.000000 | 0.909091
0.000000 1.000000 0.000000 0.000000 | 0.829463
0.000000 0.000000 1.000000 0.000000 | 0.748612
0.000000 0.000000 0.000000 1.000000 | 0.659570

Determinante: 1.516138
El sistema esta determinado.

CONCENTRACIONES DE ETILENO EN CADA REACTOR:
Reactor 1: c1 = 0.909091 mol/m3
Reactor 2: c2 = 0.829463 mol/m3
Reactor 3: c3 = 0.748612 mol/m3
Reactor 4: c4 = 0.659570 mol/m3
```

Método de Gauss-Seidel:

```
METODO DE GAUSS-SEIDEL, 4 REACTORES CSTR
Datos del sistema:
Q = 1.00 m3/s, c_in = 1.00 mol/m3
Volumenes: [1.0, 1.2, 0.9, 1.5] m3
Constantes k: [0.10, 0.08, 0.12, 0.09] s-1

Sistema de ecuaciones:
Ecuacion 1: 1.100*c1 = 1.000
Ecuacion 2: -1.000*c1 + 1.096*c2 = 0.000
Ecuacion 3:  - 1.000*c2 + 1.108*c3 = 0.000
Ecuacion 4:  - 1.000*c3 + 1.135*c4 = 0.000

Valores iniciales: c1 = 0.0, c2 = 0.0, c3 = 0.0, c4 = 0.0
Tolerancia: 1.00e-06
Maximo de iteraciones: 100

Iteracion  1: c1 = 0.909091  c2 = 0.829463  c3 = 0.748612  c4 = 0.659570  Error: 9.09e-01
Iteracion  2: c1 = 0.909091  c2 = 0.829463  c3 = 0.748612  c4 = 0.659570  Error: 0.00e+00

RESULTADOS FINALES
Convergencia alcanzada en 2 iteraciones

CONCENTRACIONES DE ETILENO:
Reactor 1: c1 = 0.909091 mol/m3
Reactor 2: c2 = 0.829463 mol/m3
Reactor 3: c3 = 0.748612 mol/m3
Reactor 4: c4 = 0.659570 mol/m3
```

Método de descomposición LU:

```
METODO DE DESCOMPOSICION LU - 4 REACTORES CSTR
Datos del sistema:
Q = 1.00 m3/s, c_in = 1.00 mol/m3
Volumenes: [1.0, 1.2, 0.9, 1.5] m3
Constantes k: [0.10, 0.08, 0.12, 0.09] s-1

Matriz del sistema A | b:
  1.100000   0.000000   0.000000   0.000000 |   1.000000
 -1.000000   1.096000   0.000000   0.000000 |   0.000000
  0.000000  -1.000000   1.108000   0.000000 |   0.000000
  0.000000   0.000000  -1.000000   1.135000 |   0.000000

DESCOMPOSICION LU
U[1][1] = A[1][1] - (L[1][i]*U[i][1]) = 1.100000
U[1][2] = A[1][2] - (L[1][i]*U[i][2]) = 0.000000
U[1][3] = A[1][3] - (L[1][i]*U[i][3]) = 0.000000
U[1][4] = A[1][4] - (L[1][i]*U[i][4]) = 0.000000
L[2][1] = (A[2][1] - (L[2][j]*U[j][1])) / U[1][1] = -0.909091
L[3][1] = (A[3][1] - (L[3][j]*U[j][1])) / U[1][1] = 0.000000
L[4][1] = (A[4][1] - (L[4][j]*U[j][1])) / U[1][1] = 0.000000
U[2][2] = A[2][2] - (L[2][i]*U[i][2]) = 1.096000
U[2][3] = A[2][3] - (L[2][i]*U[i][3]) = 0.000000
U[2][4] = A[2][4] - (L[2][i]*U[i][4]) = 0.000000
L[3][2] = (A[3][2] - (L[3][j]*U[j][2])) / U[2][2] = -0.912409
L[4][2] = (A[4][2] - (L[4][j]*U[j][2])) / U[2][2] = 0.000000
U[3][3] = A[3][3] - (L[3][i]*U[i][3]) = 1.108000
U[3][4] = A[3][4] - (L[3][i]*U[i][4]) = 0.000000
L[4][3] = (A[4][3] - (L[4][j]*U[j][3])) / U[3][3] = -0.902527
U[4][4] = A[4][4] - (L[4][i]*U[i][4]) = 1.135000
```

```
MATRIZ L (Triangular Inferior)
  1.000000   0.000000   0.000000   0.000000
 -0.909091   1.000000   0.000000   0.000000
  0.000000  -0.912409   1.000000   0.000000
  0.000000   0.000000  -0.902527   1.000000

MATRIZ U (Triangular Superior)
  1.100000   0.000000   0.000000   0.000000
  0.000000   1.096000   0.000000   0.000000
  0.000000   0.000000   1.108000   0.000000
  0.000000   0.000000   0.000000   1.135000

 Ly = b
y[1] = b[1] - (L[1][j]*y[j]) = 1.000000
y[2] = b[2] - (L[2][j]*y[j]) = 0.909091
y[3] = b[3] - (L[3][j]*y[j]) = 0.829462
y[4] = b[4] - (L[4][j]*y[j]) = 0.748612

Ux = y
c[4] = (y[4] - (U[4][j]*c[j])) / U[4][4] = 0.659570
c[3] = (y[3] - (U[3][j]*c[j])) / U[3][3] = 0.748612
c[2] = (y[2] - (U[2][j]*c[j])) / U[2][2] = 0.829463
c[1] = (y[1] - (U[1][j]*c[j])) / U[1][1] = 0.909091

RESULTADOS FINALES

CONCENTRACIONES DE ETILENO:
Reactor 1: c1 = 0.909091 mol/m3
Reactor 2: c2 = 0.829463 mol/m3
Reactor 3: c3 = 0.748612 mol/m3
Reactor 4: c4 = 0.659570 mol/m3
```

Análisis de resultados

Los tres métodos numéricos implementados arrojaron resultados idénticos para las concentraciones de etileno en los cuatro reactores. Las concentraciones calculadas muestran una disminución en el transcurso que va pasando de reactor a reactor, partiendo de 0.909091 mol/m^3 en el primer reactor hasta 0.0659570 mol/m^3 en el cuarto, si calculamos la eficiencia del sistema:

$$X_{global} = \frac{c_{in} - c_4}{c_{in}} \times 100\% = \frac{1.0 - 0.659570}{1.0} \times 100\% = 34.04\% \qquad (16)$$

Es decir, el valor pasado es la conversión global que pertenece al sistema. Esta tendencia es consistente con el comportamiento esperado en reactores CSTR en serie, donde cada unidad adicional del sistema contribuya a una mayor conversión de los reactivos. En términos de qué método es el más eficiente sería el de Gauss-Seidel ya que demostró ser el más rápido, alcanzando la convergencia en dos iteraciones con error cero, gracias a la matriz diagonal. El método de Gauss-Jordan, aunque el programa fue más costoso, proporcionó una solución exacta mediante eliminación completa, mientras que la Descomposición LU mostró utilidad al descomponer el sistema en matrices triangulares que facilitan la resolución por sustitución. El análisis de los resultados confirmó que todas las soluciones satisfacen el sistema original con errores numéricos despreciables,

demostrando la importancia de los métodos numéricos en problemas de la Ingeniería Química.

Conclusiones

El proyecto demuestra de forma exitosa la aplicabilidad de los métodos numéricos en la resolución de sistemas de ecuaciones lineales que surgen del modelado de reactores químicos, Los tres métodos que se implementaron en este trabajo, ofrecieron resultados coherentes y precisos en las concentraciones de etileno en la red de cuatro reactores CSTR en ser, esto demuestra que el modelo matemático es correcto. Se mostró que el sistema tiene características numéricas que favorecieron la convergencia, como ser diagonalmente dominante y estar bien establecido, esto explica las rápidas convergencias vistas, sobre todo al aplicar el método de Gauss-Seidel que requirió tan solo dos iteraciones. Desde la perspectiva de ingeniería química, los resultados confirman el comportamiento esperado de una disminución progresiva en la concentración de reactivo a través de la cadena de reactores, alcanzando una conversión global del 34.04%. La elección del método numérico debe considerar el contexto a donde se quiera aplicar: Gauss-Jordan para precisión en sistemas pequeños, Gauss-Seidel para simulaciones repetitivas y Descomposición LU para sistemas que requieren muchas soluciones. Este trabajo verifica y pone las bases para ser trabajado futuramente en tratamiento de datos más extensos, también ayudas a análisis de sensibilidad, optimización de parámetros de operación o la consideración de estas variables, como las cinéticas pero más complejas.

Referencias bibliográficas

1. Dooley, K. M., & Benton, M. G. (s. f.). *Catalytic Reactor: Hydrogenation of Ethylene* [Video]. JoVE. Recuperado de https://app.jove.com/v/10427/heterogeneous-catalytic-reactor-and-hydrogenation-of-ethylene
2. LibreTexts. (n.d.). 15.9: Catálisis - Hidrogenación de Etileno* [Texto educativo]. Español LibreTexts. Recuperado de https://espanol.libretexts.org/Química/Química_General/Mapa:_Química_(Zumdahl_y_Decoste)/15:_Cinética_química/15.9:_Catálisis
3. Levenspiel, O. (1986). *Ingeniería de las reacciones químicas* (2ª ed.). Editorial Reverte.
4. Felder, R. M. & Rousseau, R. W. (2005). *Elementary Principles of Chemical Processes* (3rd ed.). Wiley.
5. Chapra, S. C., & Canale, R. P. (2010). *Métodos numéricos para ingenieros* (6ª ed.). McGraw-Hill.