

Conducción de calor en un intercambiador de placas

En una planta industrial se utiliza un intercambiador de calor de placas (fig. 1) para transferir energía térmica entre dos corrientes de proceso. Durante una determinada condición de operación, una de las placas del intercambiador presenta una distribución de temperatura no uniforme en sus bordes debido al contacto con diferentes regiones del equipo y con los fluidos del proceso.

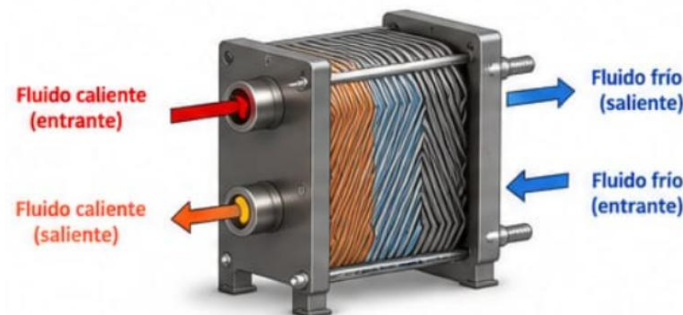


Fig. 1 – Intercambiador de calor de placas.

Para analizar la distribución de temperatura en el interior de la placa, se selecciona la sección rectangular indicada en la figura 2. Se supone que la placa metálica tiene espesor uniforme, conductividad térmica constante, y que el sistema se encuentra en régimen estacionario, sin generación interna de calor.

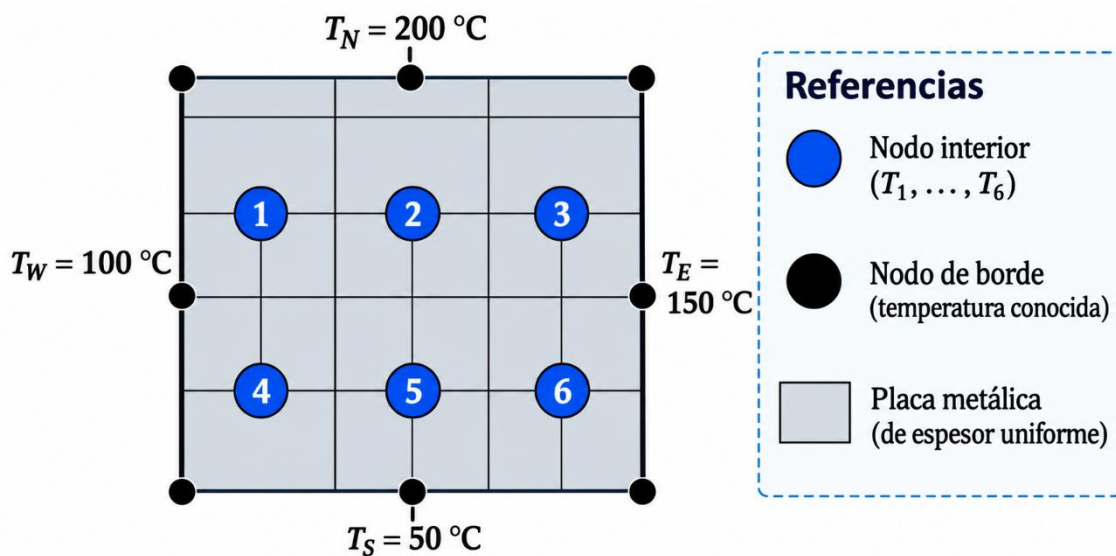


Fig. 2 – Región de la placa (vista superior)

La aplicación de la ecuación de conducción de calor bidimensional mediante el método de diferencias finitas conduce al siguiente sistema de ecuaciones, para las temperaturas desconocidas en los seis nodos interiores T_1 a T_6 :

$$(1) \quad 4T_1 - T_2 - T_4 = T_N + T_W$$

$$(2) \quad -T_1 + 4T_2 - T_3 - T_5 = T_N + T_E$$

$$(3) \quad -T_2 + 4T_3 - T_6 = T_N + T_E$$

$$(4) \quad -T_1 + 4T_4 - T_5 = T_S + T_W$$

$$(5) \quad -T_2 - T_4 + 4T_5 - T_6 = T_S$$

$$(6) \quad -T_3 - T_5 + 4T_6 = T_S + T_E$$

Donde las temperaturas se expresan en °C y T_N, T_S, T_E y T_W son las temperaturas en los bordes de la placa también en °C.

Se desea determinar la temperatura de los 6 nodos interiores (T_1 a T_6) con una tolerancia de 10^{-6} .

Datos numéricos:

Las temperaturas en los bordes de la región estudiada son:

- Borde Superior: $T_N = 200$ °C
- Borde Inferior: $T_S = 50$ °C
- Borde Izquierdo: $T_W = 100$ °C
- Borde Derecho: $T_E = 150$ °C

Las temperaturas de los nodos exteriores están almacenadas en archivo temp.txt, los resultados obtenidos en cada iteración y el error correspondiente se deben imprimir en pantalla y guardarse en un archivo .txt

Análisis del Problema y Tipo de Problemática

A partir de la discretización de la placa mediante el método de diferencias finitas, se plantean ecuaciones de balance de energía para cada uno de los nodos interiores, considerando la interacción con los nodos adyacentes y las temperaturas conocidas en los bordes de la región. De esta manera, se obtiene **un sistema de ecuaciones lineales** cuya solución permite determinar la distribución de temperaturas en el interior de la placa. Entre los métodos numéricos estudiados para la resolución de este tipo de sistemas se encuentran los métodos iterativos de Jacobi y Gauss-Seidel. En este trabajo se seleccionó

el **método de Gauss-Seidel**, ya que, a diferencia del método de Jacobi, utiliza en cada iteración los valores de las variables que ya fueron actualizados dentro de esa misma iteración. Esto permite aprovechar inmediatamente la información obtenida y, en general, lograr una **convergencia más rápida**, requiriendo una menor cantidad de iteraciones para alcanzar una determinada tolerancia.

En este caso, la elección resulta especialmente conveniente debido a que la discretización de un problema de conducción de calor genera sistemas de ecuaciones de gran tamaño cuando se aumenta el número de nodos, haciendo que los métodos iterativos resulten una alternativa eficiente para obtener una solución aproximada con la precisión requerida.

Reemplazando los valores conocidos se arma el siguiente sistema de ecuaciones:

$$\begin{cases} 4T_1 - T_2 - T_4 = 300 \\ -T_1 + 4T_2 - T_3 - T_5 = 200 \\ -T_2 + 4T_3 - T_6 = 350 \\ -T_1 + 4T_4 - T_5 = 150 \\ -T_2 - T_4 + 4T_5 - T_6 = 100 \\ -T_3 - T_5 + 4T_6 = 225 \end{cases}$$

En forma matricial $A * T = B$, donde:

$$A = \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \quad T = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix} \quad y \quad B = \begin{bmatrix} 300 \\ 200 \\ 350 \\ 150 \\ 100 \\ 225 \end{bmatrix}$$

¿Existe solución para este sistema? ¿Es única?

Para responder estas preguntas, utilizamos el teorema de Rouché-Frobenius. Si el rango de la matriz ampliada (MA) coincide con el rango de la matriz de coeficientes (A), entonces el sistema es compatible. Si, además, este rango común coincide con (n), donde (n) es el número de incógnitas, el sistema es **compatible determinado (SCD)**, es decir, tiene una única solución. En nuestro caso:

$$Rg(MA) = Rg(A) = n = 6$$

Por lo tanto, la solución es única.

Análisis de convergencia

Antes de aplicar el método de Gauss-Seidel se analiza la convergencia del proceso iterativo para el sistema obtenido mediante la discretización por diferencias finitas.

Si A es diagonalmente dominante por filas se tiene una **condición suficiente** para asegurar la convergencia al aplicar el método de Gauss Seidel. Esto significa que se cumpla la siguiente desigualdad estricta en cada fila de A:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

La matriz de coeficientes es estrictamente diagonal dominante por filas, ya que en todas ellas el valor absoluto del elemento diagonal es mayor que la suma de los valores absolutos de los restantes elementos de la fila. Por lo tanto, La matriz es estrictamente diagonal dominante y, en consecuencia, el método de Gauss-Seidel converge para cualquier aproximación inicial.

Del sistema de ecuaciones original, se despeja T1 de la ec (1), T2 de la ec. (2), así hasta la ecuación 6. Luego se arma el sistema de ecuaciones iterativo correspondiente a la aplicación del método de Gauss Seidel.

$$\left\{ \begin{array}{l} T_1 = \frac{1}{4}(300 + T_2 + T_4) \\ T_2 = \frac{1}{4}(200 + T_1 + T_3 + T_5) \\ T_3 = \frac{1}{4}(350 + T_2 + T_6) \\ T_4 = \frac{1}{4}(150 + T_1 + T_5) \\ T_5 = \frac{1}{4}(100 + T_2 + T_4 + T_6) \\ T_6 = \frac{1}{4}(225 + T_3 + T_5) \end{array} \right. \Rightarrow \left\{ \begin{array}{l} T_1^{k+1} = \frac{1}{4}(300 + T_2^k + T_4^k) \\ T_2^{k+1} = \frac{1}{4}(200 + T_1^{k+1} + T_3^k + T_5^k) \\ T_3^{k+1} = \frac{1}{4}(350 + T_2^{k+1} + T_6^k) \\ T_4^{k+1} = \frac{1}{4}(150 + T_1^{k+1} + T_5^k) \\ T_5^{k+1} = \frac{1}{4}(100 + T_2^{k+1} + T_4^{k+1} + T_6^k) \\ T_6^{k+1} = \frac{1}{4}(225 + T_3^{k+1} + T_5^{k+1}) \end{array} \right.$$

Factor de relajación

Además, en la resolución se incorporará un **factor de relajación** ω con el objetivo de analizar su influencia sobre la velocidad de convergencia del método.

El factor de relajación permite modificar la magnitud de la actualización de cada variable en cada iteración, combinando el valor obtenido mediante Gauss-Seidel con el valor correspondiente a la iteración anterior.

Para $\omega = 1$ se recupera el método de Gauss-Seidel convencional,

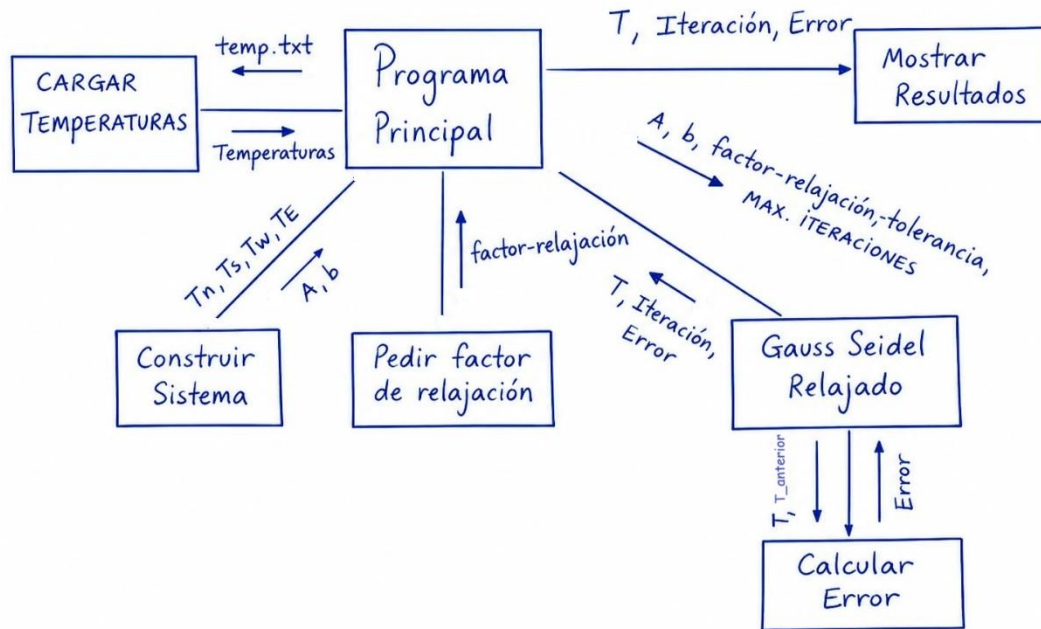
Para valores $0 < \omega < 1$ corresponden a subrelajación y valores $1 < \omega < 2$ a sobrerelajación. En este último caso, una elección adecuada de puede permitir reducir la cantidad de iteraciones necesarias para alcanzar la tolerancia establecida.

Para emplear el factor de relajación, primero calculamos el valor de Gauss Seidel y luego lo combinamos con el valor de la iteración anterior:

$$T_i^{k+1} = (1 - \omega)T_i^k + \omega \bar{T}_i^{k+1}$$

Donde $\bar{T}_i^{k+1}$ es la solución obtenida con Gauss Seidel sin factor de relajación

Diagrama de Estructura



Pseudocódigo

ALGORITMO PRINCIPAL

```
temperaturas ← cargar_temperaturas("temp.txt")

TN ← temperaturas["TN"]
TS ← temperaturas["TS"]
TW ← temperaturas["TW"]
TE ← temperaturas["TE"]

A, b ← construir_sistema(TN, TS, TW, TE)

factor_relajacion ← pedir_factor_relajacion()

tolerancia ← 10^-6

max_iteraciones ← 10000

T, iteraciones, error ←
    gauss_seidel_relajado(
        A,
        b,
        factor_relajacion,
        tolerancia,
        max_iteraciones
    )

mostrar_resultados(
    T,
    iteraciones,
    error
)

FIN
```

```
FUNCIÓN cargar_temperaturas(nombre_archivo)
```

```
    Crear diccionario temperaturas
```

```
    Abrir archivo
```

```
    Para cada línea del archivo
```

```
        Separar nombre y valor
```

```
        Guardar valor en el diccionario
```

```
    Fin Para
```

```
    Cerrar archivo
```

```
    Retornar diccionario temperaturas
```

```
FIN FUNCIÓN
```

```
FUNCIÓN construir_sistema(TN, TS, TW, TE)
```

```
    Definir matriz de coeficientes A
```

```
    Construir vector b utilizando:
```

```
        b1 = TN + TW
```

```
        b2 = TN
```

```
        b3 = TN + TE
```

```
        b4 = TW + TS
```

```
        b5 = TS
```

```
        b6 = TS + TE
```

```
    Retornar A y b
```

```
FIN FUNCIÓN
```

```
FUNCIÓN pedir_factor_relajacion()
```

```
    Solicitar valor de  $\omega$ 
```

```
    Mientras ( $\omega \leq 0$ ) ó ( $\omega \geq 2$ )
```

```
        Solicitar nuevamente  $\omega$ 
```

```
    Fin Mientras
```

```
    Retornar  $\omega$ 
```

```
FIN FUNCIÓN
```

```
función calcular_error(T, T_anterior)

    error ← ||T - T_anterior||2 / ||T||2

    Retornar error

FIN FUNCIÓN

función gauss_seidel_relajado(
    A,
    b,
    ω,
    tolerancia,
    max_iteraciones)

    Inicializar:

        T = [0,0,0,0,0,0]

    Crear archivo de salida

    Escribir encabezado

    Para iteracion = 1 hasta max_iteraciones

        T_anterior ← copia de T

        Para i = 1 hasta 6

            suma ← 0

            Para j = 1 hasta 6

                Si j ≠ i

                    suma ← suma + A(i,j) · T(j)

                Fin Si

            Fin Para

            T_GS ← ( b(i) - suma ) / A(i,i)

            T(i) ←
                (1 - ω) · T_anterior(i)
                +
                ω · T_GS

        Fin Para
```

```
error ← calcular_error(  
    T,  
    T_anterior  
)  
  
    Guardar:  
        iteración,  
        T1,  
        T2,  
        T3,  
        T4,  
        T5,  
        T6  
  
    Si error < tolerancia  
        Cerrar archivo  
  
        Retornar:  
            T,  
            iteracion,  
            error  
  
    Fin Si  
  
Fin Para  
  
Cerrar archivo  
  
Retornar:  
    T,  
    max_iteraciones,  
    error  
  
FIN FUNCIÓN  
  
FUNCIÓN mostrar_resultados(  
    T,  
    iteraciones,  
    error)  
  
    Mostrar temperaturas T1 a T6  
  
    Mostrar cantidad de iteraciones  
  
    Mostrar error final  
  
FIN FUNCIÓN
```

Codificación en Python

```
import numpy as np

def cargar_temperaturas(nombre_archivo):
    temperaturas = {}

    archivo = open(nombre_archivo, "r")

    for linea in archivo:
        nombre, valor = linea.strip().split("=")
        temperaturas[nombre.strip()] = float(valor)

    archivo.close()

    return temperaturas

def construir_sistema(TN, TS, TW, TE):
    A = np.array([
        [4, -1, 0, -1, 0, 0],
        [-1, 4, -1, 0, -1, 0],
        [0, -1, 4, 0, 0, -1],
        [-1, 0, 0, 4, -1, 0],
        [0, -1, 0, -1, 4, -1],
        [0, 0, -1, 0, -1, 4]
    ], dtype=float)

    b = np.array([
        TN + TW,
        TN,
        TN + TE,
        TW + TS,
        TS,
        TS + TE
    ], dtype=float)

    return A, b

def pedir_factor_relajacion():
    factor_relajacion = float(
        input("Ingrese el factor de relajacion (0 < w < 2): ")
    )

    while factor_relajacion <= 0 or factor_relajacion >= 2:
        factor_relajacion = float(
            input(
                "Valor incorrecto. Ingrese un factor de relajacion entre 0 y 2: "
            )
        )

    return factor_relajacion

def calcular_error(T, T_anterior):
    error = np.linalg.norm(T - T_anterior, 2) / np.linalg.norm(T, 2)

    return error
```

```

def gauss_seidel_relajado(A, b, factor_relajacion,
                          tolerancia, max_iteraciones):

    T = np.zeros(len(b))

    archivo = open("temperaturas_iteraciones.txt", "w")

    archivo.write(
        "Iteracion\tT1\tT2\tT3\tT4\tT5\tT6\n"
    )

    for iteracion in range(1, max_iteraciones + 1):

        T_anterior = T.copy()

        for i in range(len(T)):

            suma = 0

            for j in range(len(T)):

                if j != i:

                    suma += A[i, j] * T[j]

            T_GS = (b[i] - suma) / A[i, i]

            T[i] = (1 - factor_relajacion) * T_anterior[i] + \
                factor_relajacion * T_GS

        error = calcular_error(T, T_anterior)

        archivo.write(
            f"{iteracion}\t"
            f"{T[0]:.8f}\t"
            f"{T[1]:.8f}\t"
            f"{T[2]:.8f}\t"
            f"{T[3]:.8f}\t"
            f"{T[4]:.8f}\t"
            f"{T[5]:.8f}\n"
        )

        if error < tolerancia:

            archivo.close()

            return T, iteracion, error

    archivo.close()

    return T, max_iteraciones, error

def mostrar_resultados(T, iteraciones, error):

    print("\nResultados:")
    print("-----")

    for i in range(len(T)):

        print(f"T{i + 1} = {T[i]:.8f} °C")

    print("-----")
    print(f"Iteraciones = {iteraciones}")
    print(f>Error final = {error:.10e}")
  
```

```
# -----  
# Programa principal  
# -----  
  
temperaturas = cargar_temperaturas("temp.txt")  
  
TN = temperaturas["TN"]  
TS = temperaturas["TS"]  
TW = temperaturas["TW"]  
TE = temperaturas["TE"]  
  
A, b = construir_sistema(  
    TN,  
    TS,  
    TW,  
    TE  
)  
  
factor_relajacion = pedir_factor_relajacion()  
  
tolerancia = 1e-6  
  
max_iteraciones = 10000  
  
T, iteraciones, error = gauss_seidel_relajado(  
    A,  
    b,  
    factor_relajacion,  
    tolerancia,  
    max_iteraciones  
)  
  
mostrar_resultados(  
    T,  
    iteraciones,  
    error  
)
```

```
Ingrese el factor de relajacion (0 < w < 2): 1  
  
Resultados:  
-----  
T1 = 136.23180410 °C  
T2 = 147.82601871 °C  
T3 = 152.89852160 °C  
T4 = 97.10140101 °C  
T5 = 102.17387185 °C  
T6 = 113.76809836 °C  
-----  
Iteraciones = 15  
Error final = 7.1544995081e-07  
  
Process finished with exit code 0
```

Resultados Contextualizados

Para un factor de relajación $\omega = 1$, en 15 iteraciones y con un error de 7.15×10^{-7} , se obtuvieron los siguientes resultados:

Nodo	Temperatura
(T_1)	136.23 °C
(T_2)	147.83 °C
(T_3)	152.90 °C
(T_4)	97.10 °C
(T_5)	102.17 °C
(T_6)	113.77 °C

Las temperaturas de los bordes están comprendidas entre 50°C y 200°C. Las temperaturas T_i obtenidas para los nodos interiores son físicamente razonables ya que:

$$50^\circ\text{C} \leq T_i \leq 200^\circ\text{C}$$

Por otro lado, se puede verificar que los valores obtenidos satisfacen el sistema original, por ejemplo, en la primera ecuación:

$$(1) \ 4T_1 - T_2 - T_4 = 300$$

Reemplazando:

$$(1) \ 4(136.23180410) - 147.82601871 - 97.10140101 = 299.999$$

Con respecto al factor de relajación, podemos determinar experimentalmente cuál es el más eficiente para este problema. Los resultados muestran exactamente el comportamiento que uno espera cuando se introduce **sobrerrelajación**.

ω	Iteraciones
0.90	19
1.00	15
1.10	11
1.15	10
1.20	11

Se observa que al pasar de $\omega = 0.90$ a $\omega = 1$ la cantidad de iteraciones disminuye de 19 a 15. Esto indica que la subrelajación ralentiza el proceso iterativo, ya que las correcciones aplicadas en cada paso son más pequeñas.

Al utilizar valores de sobrerrelajación ($1 < \omega < 2$), la velocidad de convergencia aumenta significativamente. Con $\omega = 1.1$ el método requiere únicamente 11 iteraciones y alcanza su mejor desempeño para $\omega = 1.15$, donde la convergencia se logra en tan solo 10 iteraciones.

Sin embargo, al incrementar el factor hasta $\omega = 1.2$, la cantidad de iteraciones vuelve a aumentar ligeramente hasta 11. Esto muestra que existe un valor óptimo de relajación para este problema y que aumentarlo indefinidamente no implica necesariamente una mejora en la velocidad de convergencia.

Conclusiones

Si bien el sistema analizado podría resolverse fácilmente mediante un método directo debido a su reducido tamaño, los métodos iterativos como Gauss-Seidel adquieren especial importancia en aplicaciones reales de ingeniería. Al discretizar dominios físicos complejos, suelen obtenerse sistemas con miles o millones de ecuaciones y matrices dispersas, es decir, matrices que contienen una gran cantidad de elementos nulos. En estos casos, los métodos iterativos permiten aprovechar dicha estructura, reduciendo significativamente los requerimientos de memoria y el costo computacional respecto de los métodos directos, lo que los convierte en una herramienta fundamental para la resolución de problemas de gran escala.