



UNIVERSIDAD  
DE GUANAJUATO

Campus León

División de Ciencias e Ingenierías

Departamento de Ingeniería Química, Electrónica y Biomédica



**Universidad de Guanajuato**  
**Campus León**  
**División de Ciencias e Ingenierías**

Proyecto #2: “Temperatura en una placa, estado estacionario”

**Programación básica**

**1°Semestre**

**Profesora:**

Alma Xochitl González

**Nombre de la alumna:**

Diana Ailed Hernández Bustos

Repositorio GITHUB: dianahern

Fecha de entrega: 30/10/18

## 1. Resumen

Este proyecto consistió en calcular la temperatura de una placa cuadrada y muy delgada (por lo tanto sólo se trabajaron con 2 dimensiones: xy), cuyos bordes tenían una temperatura constante. Dentro de los objetivos de este proyecto se encontraban los de utilizar arreglos de dos dimensiones, el uso de funciones a partir de diferentes archivos, entre otros.

## 2. Introducción y marco teórico:

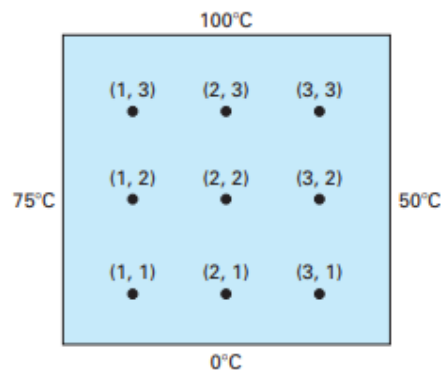
### Ecuación de Laplace

La ecuación de Laplace se utiliza para modelar diversos problemas que tienen que ver con el potencial de una variable desconocida. Por tanto, la forma más exacta para calcular la temperatura en cada uno de los puntos de una placa delgada con una dimensión de xy (bidimensional) está dada por la ecuación de Laplace:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

(Ec. 1)

La ecuación de Laplace está expresada en función de la temperatura de la placa (T), y analiza la placa como una malla de N\*N, cuyos puntos se designan con coordenadas (i, j), de tal forma que las derivadas de T en cada punto pueden ser aproximadas por la temperatura de los puntos vecinos. Por lo tanto por cada punto, se va a obtener una ecuación y como consecuencia, la placa va a tener N\*N ecuaciones.



**Figura 1:** Ejemplo del cálculo de temperaturas en una placa estacionaria

En la mayoría de las soluciones numéricas de la ecuación de Laplace se tienen sistemas que son muy grandes y complejos, ya que por ejemplo para una malla de 10 por 10 se obtienen 100 ecuaciones algebraicas lineales, los cuales ocupan una gran cantidad de memoria de la computadora (al almacenar ceros). Por esta razón, los métodos aproximados representan un mejor procedimiento para obtener soluciones de este tipo. El método comúnmente empleado es el Gauss-Seidel.

### Ecuación de Gauss-Seidel

Es un método iterativo y aproximado que, al igual que la ecuación de Laplace, divide la placa en  $N \times N$  puntos designados con coordenadas  $(i, j)$ , cuyo cálculo es:

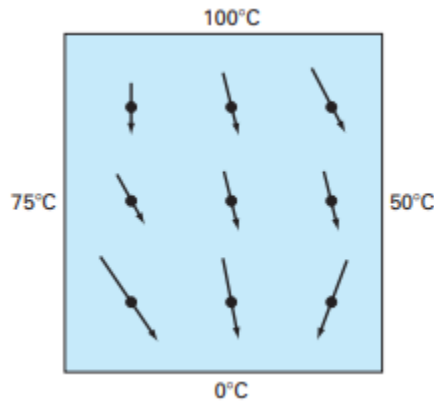
$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4} \quad (\text{Ec. 2})$$

Esta ecuación consiste en encontrar la temperatura de un punto en la placa al calcular el promedio de las temperaturas de los 4 puntos/cuadros vecinos de éste. Esto se facilita cuando los bordes de la placa tienen temperatura fija y los puntos del centro son igual a cero.

Este método es iterativo porque para llegar al valor más exacto posible, es necesario que el porcentaje de error (que se designa  $\varepsilon$  en un punto  $(i,j)$ ), sea menor que un valor especificado anteriormente (se recomienda que sea menor de 1%). Este porcentaje de error se calcula así:

$$|(\varepsilon)_{i,j}| = \left| \frac{T_{i,j}^{\text{anterior}} - T_{i,j}^{\text{nuevo}}}{T_{i,j}^{\text{nuevo}}} \right| * 100 \quad (\text{Ec. 3})$$

Es necesario entender que, según la 2ª Ley de la Termodinámica, cuando dos cuerpos de diferentes temperaturas entran en contacto físico, el calor se transfiere del cuerpo con menor temperatura al de mayor. En el caso de una placa de  $N \times N$  puntos, cuyos bordes tienen una temperatura constante, el calor se transfiere de forma vectorial de la siguiente forma:



**Figura 2:** Flujo de calor en una placa sujeta a temperaturas fijas en las fronteras, donde la longitud de las flechas es proporcional a la magnitud del flujo

### 3. Código:

Mi programa “**proyecto2**” se compone de 2 archivos llamados: “**main.c**” y “**función.c**”, cuyo código se va a explicar a continuación.

#### 3.1.-Main.c

Mi programa inicia con la declaración de las librerías usadas que corresponden a “**stdio.h**” y “**stdlib.h**”, ésta última se usa porque en el programa se usó memoria dinámica para reservar espacio para el apuntador bidimensional de la variable matriz. Después de definir las variables, se abre el archivo de texto para escanear la información necesaria sobre las temperaturas de los bordes y el número de filas y columnas (“N”, debido a que se trataba de una placa metálica). Al finalizar esta acción, se cierra el archivo de lectura y se pone una condición para evitar que N sea mayor o igual a 500 con un if.

A continuación, se hace la reservación de memoria para el apuntador de 2 dimensiones llamado matriz que es de tipo **double** y arreglo: **\*\*matriz[i][j]**. Como es de tipo bidimensional, fue necesario primero, hacer la reserva con **Malloc** con espacio de N y después, poner un FOR que llenara cada fila de memoria reservada con otros N elementos usando otra vez la función **Malloc**.

Después se llama a la función inicializar (que tiene elementos de entrada y de salida, que en continuación se explicarán) al igualarla a una variable, la cual coincide con el elemento de salida de esta función.

Posteriormente, se llama a una 2ª función, llamada imprimir (que tiene elementos de entrada pero no de salida) para que calcule la temperatura de la placa en cada iteración y que imprima los resultados en diferentes archivos. Luego se cierra el programa.

#### 3.2.-Funcion.c

En este archivo se encuentran las dos funciones mencionadas anteriormente, las cuales son:

**Double \*\*inicializar(double\*\*matriz, double arriba, double abajo, double izq, double der, int N)**

Esta función toma como elementos de entrada el apuntador bidimensional **\*\*matriz** y las temperaturas constantes y el valor de N escaneados de un archivo de texto en el programa **main.c**. Su objetivo es otorgar los valores de la temperatura de cada punto de la **matriz[i][j]**. Para esto se requieren 2 FOR.

El primer FOR corre de  $i=0$  hasta  $i<N$  y va dando los valores de los puntos que se encuentran en los bordes, al igualar cada uno de estos con las temperaturas dadas en las variables de arriba, abajo, izq y der. Se necesitó poner: **matriz[0][j]**, **matriz[i][0]**, **matriz[N-1][j]** y **matriz[i][N-1]**; los cuales correspondían a los puntos en los bordes.

El segundo FOR sirvió para inicializar todos los puntos centrales de la placa, los que no están en los bordes, con una temperatura de 0.00°C. Para esto, fue necesario hacer que el

FOR corriera de  $i=1$  y llegara hasta  $i=N-1$  (es decir que se saltara toda la primera y la última columna, así como la primera y la última fila).

Al final, esta función regresa el apuntador bidimensional inicial de matriz a la función main.

#### **Void imprimir(double \*\*matriz, int N)**

Esta función toma como elementos de entrada el apuntador bidimensional \*\*matriz ya inicializado y el valor de N del programa main.c. Después de hacer la declaración de las variables usadas en esta función, se abre un while que corra siempre que la variable pt (que corresponde a  $\epsilon$  indicada en la ec. 2) sea mayor a 0.000001% ; esto es para lograr obtener resultados con este pequeñísimo porcentaje de error.

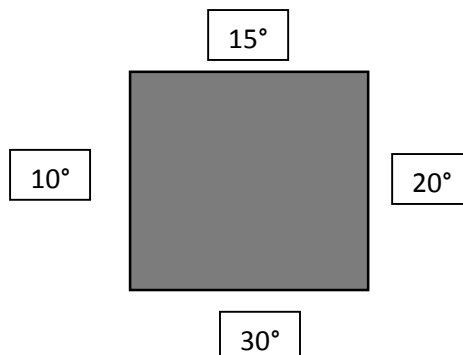
Dentro del while se utiliza un doble FOR que vaya de  $i=1$  hasta  $i=N-1$  de uno en uno y lo mismo para j en el 2° FOR, con la intención de modificar el valor de las temperaturas que corresponden a los puntos centrales de la placa (es decir, exceptuando los que se encuentran en los bordes).

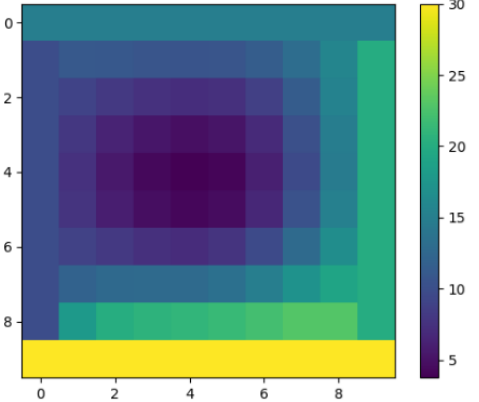
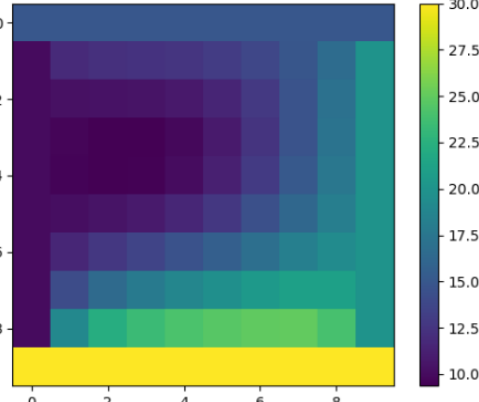
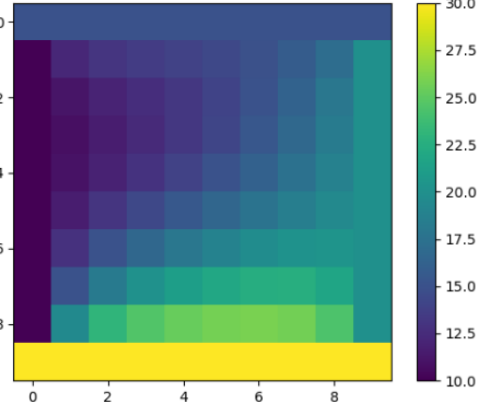
Después, se le asigna una variable a la matriz inicial y posteriormente, se realiza la ecuación de Gauss-Seidel para calcular la nueva temperatura de cada punto como se indica en la ec. 2. En seguida, se calcula  $\epsilon$  como se muestra en la ec. 3, la cual se le otorga la variable de pt, el cual cambia con cada iteración y sirve para lograr obtener resultados lo más exactos posibles.

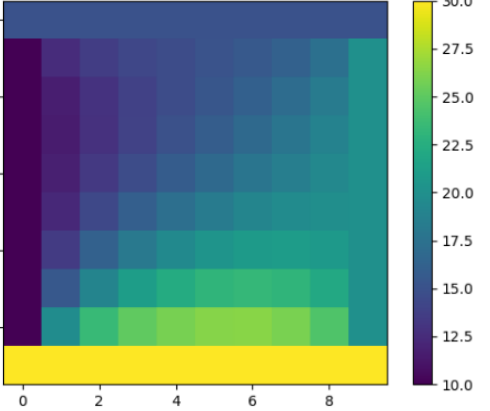
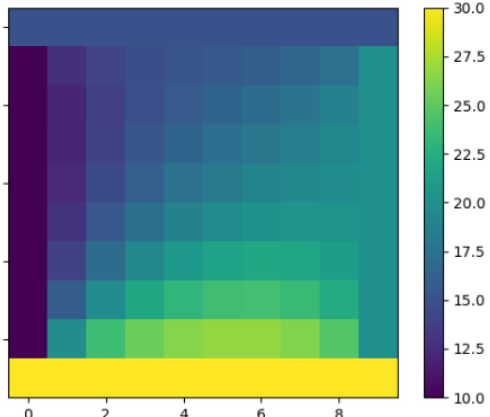
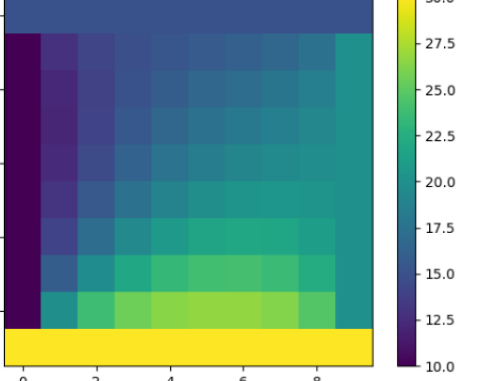
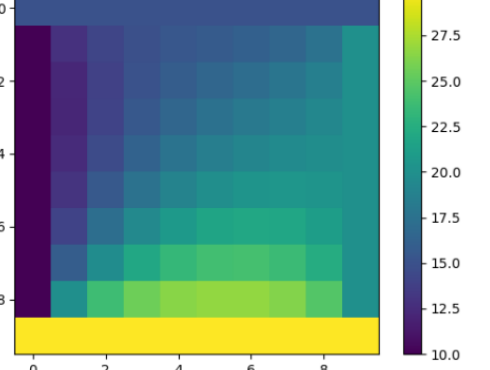
Al final del while se pone un contador que vaya aumentando de uno en uno con cada iteración, es decir cada vez que corra el while. Después, se poner una condición para que el número de iteraciones no supere 1000. A continuación, se pone otra condición para que se vaya imprimiendo de 6 en 6 cada resultado en función del contador. Para crear archivos con diferentes nombres que contengan las matrices de las nuevas temperaturas que cada punto de la placa va adquiriendo con cada iteración, fue necesario usar la función snprintf, que contenga una variable de tipo arreglo. Para poder imprimir los resultados fue necesario poner otro doble FOR (uno para i y otro para j, para así, imprimir  $matriz[i][j]$ ), el cual esta vez, sí abarcara de  $i=0$  hasta  $i=N$ . Al final, sólo se cierra la función, ya que no tiene elementos de salida.

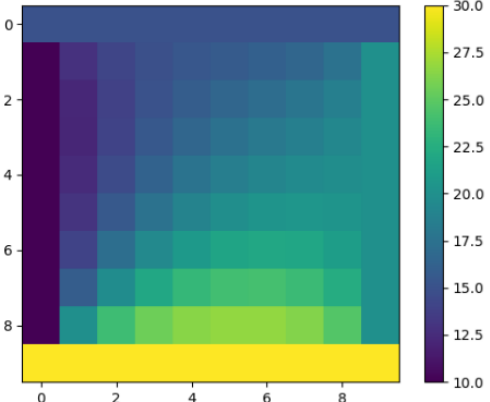
#### **4. Resultados y discusión**

A continuación se presentan los resultados obtenidos al calcular una placa delgada de 10 filas y 10 columnas con las siguientes temperaturas fijas en los bordes:



| Gráfica de la matriz  | Número de iteración | Discusión y comparación con las demás gráficas  |
|---|---------------------|---|
|    | 6                   | <p>Esta gráfica, al ser una de las primeras, tiene un gran porcentaje de error, por lo que está lejos de alcanzar una temperatura promedio en cada un</p> <p>Esta gráfica muestra claramente cómo van adquiriendo poco a poco cierta temperatura los puntos cercanos a los bordes, mientras que el centro permanece frío (aunque ya no es cero como inicialmente lo era).</p> |
|   | 12                  | <p>Se puede apreciar que hubo un gran cambio en esta iteración, ya que el centro de empezó a calentar más.</p> <p>Además se empieza a notar que la esquina noroeste va a tender a ser la sección más fría de la placa, mientras que la sección sureste, la más caliente; debido a las temperaturas fijas en los bordes.</p>   |
|  | 18                  | <p>En esta iteración, el centro empieza a calentarse un poco más y a alcanzar una temperatura más uniforme con sus vecinos, al igual que las esquinas noroeste y sureste.</p>   |

|   |    |  |
|---|----|--|
|    | 24 | La temperatura de la placa empieza a ser cada vez más uniforme y a alcanzar un cierto equilibrio en todos sus puntos, de acuerdo a su posición con respecto a los bordes de diferentes temperaturas.                           |
|   | 42 | No hubo mucho cambio con respecto a la anterior, esto es debido a que $\epsilon$ (%error) ya es muy pequeño. Por lo tanto, sólo va a haber una pequeña variación en las temperaturas a partir de la 42° iteración en adelante. |
|  | 66 | No hubo mucho cambio con respecto a la anterior, esto es debido a que $\epsilon$ (%error) ya es muy pequeño. Por lo tanto, sólo va a haber una pequeña variación en las temperaturas de 0.0001 por ejemplo.                    |
|  | 84 | No hubo mucho cambio con respecto a la anterior, esto es debido a que $\epsilon$ (%error) ya es muy pequeño.   |

|   |                                   |   |
|---|-----------------------------------|---|
|  | 114=<br>Fase de<br>EQUILIBR<br>IO | <p>Esta gráfica muestra la temperatura de cada uno de los puntos de la placa con 0.000001% de error:<br/> <math>\varepsilon = 0.000001\%</math></p> <p>Ésta fue la última iteración en realizarse, y por lo tanto fue la fase de “equilibrio” de las temperaturas en mi placa, según el criterio que tomé de obtener el porcentaje de error anteriormente presentado.</p> |
|---|-----------------------------------|---|

\*NOTA: Es importante apreciar que a partir de la iteración 24, no hubo un cambio muy drástico en las demás gráficas hasta llegar a la de equilibrio (#114). Esto se debe a que la variación de las temperaturas era de decimales muy pequeños, pues mi  $\varepsilon$  establecido era de 0.000001%.

#### Discusión del parámetro $\varepsilon$ con respecto a N e iteraciones

Como se puede apreciar en las gráficas, entre mayor sea el número de iteraciones, se va ir disminuyendo el valor de  $\varepsilon$  (que se calcula como se indica en la ecuación 3). Ya que la  $\varepsilon$  representa el porcentaje de error en relación al equilibrio entre las temperaturas de la placa que se desea encontrar, entre mayor cantidad de iteraciones haya, más exacto será el resultado. Fue necesario poner un parámetro de exactitud para que mi programa se detuviera hasta que los cálculos tuvieran un 0.000001% de error, que resultó ser en la iteración #114. Esto es debido a que con la ecuación de Gauss-Seidel, sólo es posible obtener resultados aproximados, pero no exactos.

Además, entre mayor sea el valor de N, es decir el número de columnas y filas, va a haber mayor número de iteraciones, y por lo tanto va a tardar más tiempo en alcanzar el equilibrio/ promedio de temperaturas en todos los puntos. Esto se debe a que entre mayor sea N, más grande será la matriz y, por lo tanto, será más difícil poner

## **5. Conclusión**

Este método ayuda a demostrar cómo el calor se distribuye uniformemente por toda la placa, siempre moviéndose de los bordes con mayor temperatura a los puntos con menor. Me percaté que el calor fluía de los bordes hacia el centro de la placa (siendo éstos los puntos más fríos al inicio de las iteraciones). Además me di cuenta que si mi placa tuviera la misma temperatura en los 4 bordes, se alcanzaría un equilibrio térmico (es decir, todos los puntos tendrían una temperatura casi igual, pues este método tiene un 0.000001% de error). En el caso analizado en este proyecto, los puntos de la placa nunca llegan a tener una temperatura igual, ya que los bordes de la placa permanecen con la misma temperatura en todo momento. El punto del centro es el que va a tener un promedio de las 4 temperaturas.



## **6. Referencias**

- Chapra S., & Raymond C. (2002), *Métodos para ingenieros* (5th ed., pp.895-900).