# Proyecto 2: Temperatura de una placa, estado estacionario.

**Por: Gabriel Missael Barco**
**Estudiante de Lic. en Física**

**Programación Básica**
**Profesora: Alma Gonzáles**
29 de octubre de 2018

## 1. Introducción

Se denomina calor a la energía en tránsito que se reconoce solo cuando se cruza la frontera de un sistema termodinámico. Una vez dentro del sistema, o en los alrededores, si la transferencia es de adentro hacia afuera, el calor transferido se vuelve parte de la energía interna del sistema o de los alrededores, según su caso. El término calor, por tanto, se debe de entender como transferencia de calor y solo ocurre cuando hay diferencia de temperatura y en dirección de mayor a menor. De ello se deduce que no hay transferencia de calor entre dos sistemas que se encuentran a la misma temperatura.

Cuando no hay trasferencia de temperatura se dice que un sistema esta en un equilibrio térmico.

## 2. Marco Teórico

En este proyecto se analizará la temperatura de una placa cuadrada muy delgada, que esta aislada por los extremos excepto en los bordes de la superficie, por lo que una buena aproximación será analizar la placa solo en el plano $xy$. La temperatura en cada punto de la placa, a un intervalo de tiempo dado, está determinada por la ecuación de Laplace:

$$\frac{\delta^2 T}{\delta x^2} + \frac{\delta^2 T}{\delta y^2} \tag{1}$$

Uno de los método de solución de está ecuación consiste en representar cada pequeño elemento de área de la placa como un punto de coordenadas $i, j$. Para esto, usaremos el método de Gauss-Seidel, que para esta ecuación se reduce a que la temperatura en cada punto de la placa esta dada por:

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4} \tag{2}$$

donde $i, j$ representa posiciones en los ejes x, y de la placa, respectivamente.

Cada vez que se recorran todos lo puntos de la placa usando esta ecuación la consideraremos una iteración. Para los fines de este proyecto, indicaremos una tolerancia $\epsilon$ en el cambio de temperaturas de los puntos internos, que nos indique que ya encontramos la solución. Es decir, se dejaran de realizar iteraciones cuandos se cumpla que:

$$max(Abs(\frac{T_{i,j}^{nuevo} - T_{i,j}^{viejo}}{T_{i,j}^{viejo}})) < \epsilon \tag{3}$$

Siendo que el valor $\epsilon$ deseado será dado por el usuario.

# 3. Código

El código se encuentra dividido en dos archivos, «1projecto_funciones.c» y « 1projecto_main.c». Como su nombre lo indica, en el primer archivo se encuentran las funciones utilizadas en el programa, mientras que en el segundo se encuentra la estructura principal del programa.

El programa lee las condiciones iniciales de la placa de un archivo titulado «datos.txt», de donde se leen 7 datos, los cuales, en orden de izquierda a derecha son: las temperaturas de los cuatro bordes, el tamaño de la matriz n (el tamaño de su borde). Después se lee un numero que puede ser 1 o 0. En caso de que sea 1, se leerán n*n números, es decir, se leerá la temperatura inicial en cada punto de la placa. en caso de que sea 0, no se leerán dichos números y se inicializaran en 0. Finalmente, se lee $\epsilon$. En este proyecto se analiza la siguiente entrada:

$$100\ 50\ 0\ 75\ 25\ 1\ 0{,}001 \tag{4}$$

## 3.1. Funciones

A continuación se explicaran la funciones contenidas en el archivo «1projecto_ funciones. c».

**double **iniciar(double **aux, int w, float a, float b, float c, float d)**

Esta función recibe una matriz de tipo double, **aux, la cual funciona con apuntadores. También recibe el tamaño del borde de la matriz, $w$, y la temperatura de los cuatro bordes $a, b, c$ y $d$.

Como su nombre lo indica, esta función se encarga de inicializar los valores de la matriz. Primero, se inicializan los valores de los bordes de la placa con los valores dados $a, b, c$ y $d$. Esto se logra mediante un ciclo for. Posteriormente se inicializan todas las casillas de la matriz en 0, esto se logra con dos ciclos for anidados. Finalmente se regresa la matriz obtenida tras realizar las transofrmaciones mencionadas anteriormente.

**double **itera(double **aux, int w)**

Esta función es el centro del programa, es la función que resuelve el problema que estamos tratando, que es llegar al punto donde la placa esta en un equilibrio aparente.

Para tal fin, recibe como dato de entrada la matriz **aux y el tamaño de su borde $w$. La matriz se recorre con dos ciclos for anidados, esto partiendo de la casilla (1,1). Por cada vez que se avanza en el eje $i$ se recorre todo el eje $j$, desde 1 hasta $w$-1. El eje $i$ también se recorre desde 1 hasta $w$-1.

Para cada punto dado $i, j$ se obtiene primero la suma de las temperaturas de las 4 casillas colindantes a ella. Este dato se guarda en una variable llamada $sum$ que es de tipo double. Antes de cambiar el valor de la casilla $i, j$, se guarda su valor en la variable $est$. Después de haber echo esto, el valor de la casilla $i, j$ ahora será $sum/4$. Básicamente aplicamos la ecuación 2 para obtener la temperatura de la casilla.

El cambio dado en la casilla $i, j$ estará dado por la ecuación que se encuentra en la condición 3:

$$Abs(\frac{T_{i,j}^{nuevo} - T_{i,j}^{viejo}}{T_{i,j}^{viejo}})$$

Por este motivo guardamos el valor anterior de la casilla en la variable $est$. Este valor de cambio se guarda en la variable $temp$.

Podemos percatarnos que en la condición 3 se usa el termino $max$, esto se refiere a que el valor mas grande de $temp$ de todas las casillas en una iteración dada sea menor a $\epsilon$. Para eso necesitamos guardar el valor máximo de $temp$. Eso lo vamos guardando en cada iteración, comparando el valor maximo que tenemos con el nuevo valor $temp$. Si $temp$ es mayor que nuestro valor maximo anterior, entonces lo renovamos, Esto se logra con una condición tipo $if$. El valor máximo se guarda en una casilla de la matriz que no se utiliza, la que ocupa el lugar $(w-1, w-1)$.

Finalmente la función regresa una matriz (y dentro de ella, el valor máximo de $temp$ logrado en esa iteración).

**void print(double **aux, int w, int a)**

Esta función se encarga de guardar cada iteración en un archivo diferente. Para tal fin, recibe la matriz a guardar **aux, el tamaño de su borde $w$ y el número de iteración que representa $a$. El nombre de todos los archivos tiene la forma general «9a.txt», pero para cada iteración tendrá una letra diferente, y esto estará en función de la variable $a$. Por ejemplo si $a = 0$, entonces la matriz se guarda en el archivo «9a.txt», y si $a = 2$, la matriz se guarda en el archivo «9c.txt».

Se tiene un nombre especial para cuando se ha alcanzado el valor $\epsilon$. Este es «EQUILI BRIO.txt». Esto para que el usuario tenga a la mano el archivo que contiene la solución a la placa en cuestión.

### 3.2. Estructura principal (main)

A continuación explicaremos la estructura principal del programa, localizada en el archivo « 1projecto_main.c». En dicho archivo, se llaman las funciones descritas en la sección anterior, según se van necesitando.

Primero, declaramos lo necesario para el código. Declaramos una apuntador **mat, que de hecho, trabajaremos como una matriz. Declaramos también $A, B, C, D$ que serán las temperaturas de los bordes de la placa; $E\_a$ será el mayor cambio de temperatura en una iteración dada, que se comparará con $\epsilon$; declaramos la variable *matriz* que nos servirá para saber si el usuario quiere dar valor iniciales a los puntos de la placa o no y finalmente también declaramos la variable $n$ que indica el tamaño del borde de la matriz a trabajar.

Después se declara un archivo para la lectura de los datos de «datos.txt». De aquí se leen los parámetros para la placa, por ejemplo, la entrada de datos 4. Una vez hecho esto, reservamos la memoria para la matriz de $n * n$ con *malloc*.

Después, igualamos la matriz **mat a la función iniciar, mandando la matriz reservada **mat, el tamaño del borde de la matriz $n$ y las temperaturas de los bordes. Esto inicializa la matriz.

En caso de que la variable *matriz* sea igual a 1, se leerán la temperatura inicial para cada punto de la matriz.

Antes de entrar al ciclo que realizará las iteraciones, declaramos un contador *count* para saber cuantas llevamos. El ciclo es un bucle *while* con dos condiciones de paro. Ya sea que $E\_a < \epsilon$ o bien, $count > 500$. Dentro del ciclo, primero se pregunta si $E\_a$ es un múltiplo de 10, para imprimir la matriz en un archivo. El motivo de esto se explica en la sección de resultados. Luego de esto, se realiza otra iteración con la función *iterar* y se obtiene el nuevo $E\_a$ de la casilla (n-1,n-1). Antes de volver a realizar el ciclo, se pregunta si este nuevo $E\_a$ es menor a $\epsilon$, que es nuestra condición de paro. En caso de que así sea, se manda a imprimir la matriz en el archivo «EQUILIBRIO.txt», es decir, guardamos la solución. Finalmente imprimimos en pantalla el número de iteraciones necesarias para alcanzar el $\epsilon$ deseado, siempre inferior a 500 este posible número.

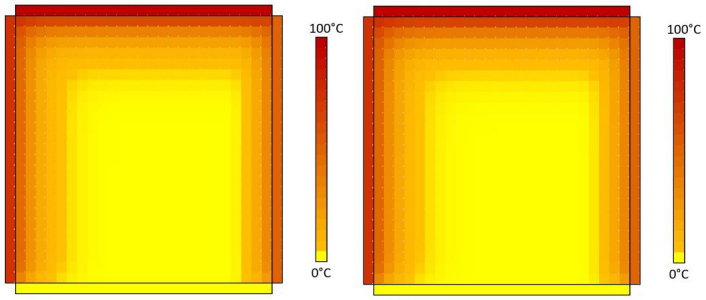
## 4. Resultados y conclusiones

Debemos recordar que analizaremos principalmente la entrada:

100 50 0 75 25 1 0,001

Lo primero que evidente es que el cambio entre iteraciones consecutivas es mínimo:

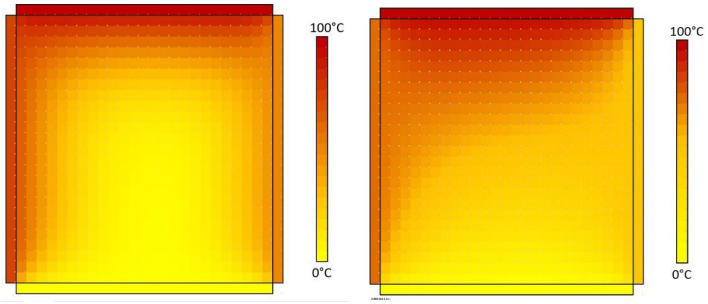
**Figura 1.** Iteraciones 4 y 5




por lo que no es útil ni necesario guardar todas las iteraciones, es por este motivo que solo se guarda una de cada diez iteraciones. Adicional a esto, solo se guardarán 25 archivos máximo. Ya que, en caso de que hubiera mas de 250 iteraciones, se debería muy probablemente a un valor de $\epsilon$ muy pequeño, por lo que después de 250 iteraciones, el cambio de 10 en 10 seria también mínimo. Supongamos que una placa se resuelve en 500 iteraciones (que es nuestro número máximo de iteraciones), quizá la diferencia entre la iteración 250 y 280 sea apenas perceptible. Pero muy probablemente si haya diferencia entre la 250 y la solución, la iteración 500. Es por este motivo que siempre se imprime la matriz que corresponde a la solución en el archivo $EQUILIBRIO.txt$, además de que esta matriz es la que estábamos buscando.

Nos podemos percatar también que el número de iteraciones necesarias para llegar a nuestra solución crece si se incrementa el número de puntos o si el valor de $\epsilon$ es menor. Por ejemplo si a nuestro archivo le modificamos el $\epsilon$ de 0.001 a 0.1 el número de iteraciones necesarias pasa de ser 250 a ser 32. El problema de reducir el $\epsilon$ es que probablemente no se llegue a una solución exacta.

**Figura 2.** Solución con $\epsilon$=0.1 y $\epsilon$=0.001



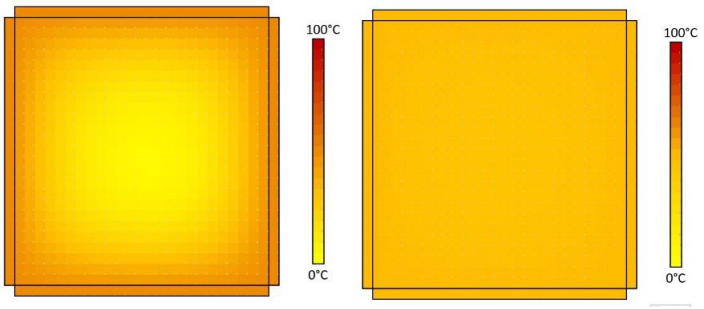
Podemos observar que hay una diferencia considerable entre ambos resultados. Estas diferencia de exactitud es mas notable si analizamos la siguiente entradas:

50 50 50 50 25 0 0,1

50 50 50 50 25 0 0,0001

Como los 4 bordes tienen la misma temperatura, se esperaría que que todos los puntos de la placa llegaran a la misma temperatura.

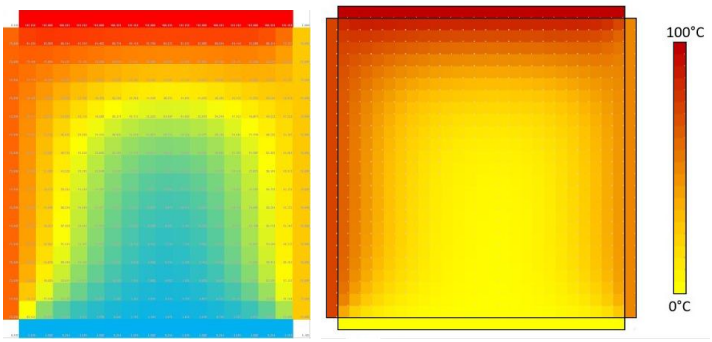
**Figura 3.** Prueba de equilibrio con $\epsilon$=0.1 y $\epsilon$=0.0001




En este caso, evidentemente el primer valor de $\epsilon$ es insuficiente, mientras que el segundo es satisfactorio.

Por otro lado, si reducimos el número de puntos a analizar en la placa, se llega a una solución menos precisa (es decir, con menos resolución) aunque también se reduce el número de iteraciones necesarias para llegar a la solución. Por ejemplo, si modificamos el tamaño del borde de nuestra matriz en la entrada que estamos ejemplificando de ser 25 a ser 16, se observa de la siguiente manera:

**Figura 4.** Iteración 30 con $n$=16 y $n$=25




Y el numero de iteraciones necesarias paso de 250 a 131.

Podemos concluir entonces, que para obtener una solución mas precisa y exacta es recomendable que el valor de $n$ sea alto ($n > 20$), y el valor de $\epsilon$ sea pequeño, preferentemente $\epsilon <= 0{,}005$. También podemos mencionar que un valor de $n > 30$ solo seria deseable si el tamaño real de la placa es grande. Por otro lado, un valor de $\epsilon < 0{,}00001$ genera demasiadas iteraciones con cambios casi nulos, por lo que tampoco son recomendables tales valores.

A continuación se incluyen 7 iteraciones representativas de la entrada analizada:

100 50 0 75 25 1 0,001

**Figura 5.** Estado inicial de la placa

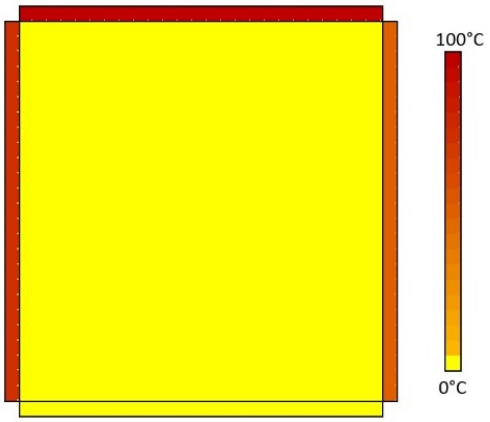



**Figura 6**

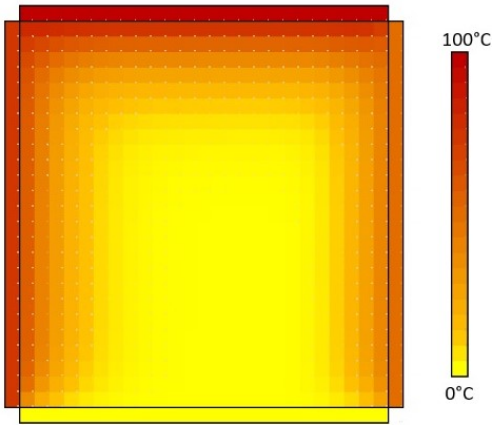


**Figura 7**

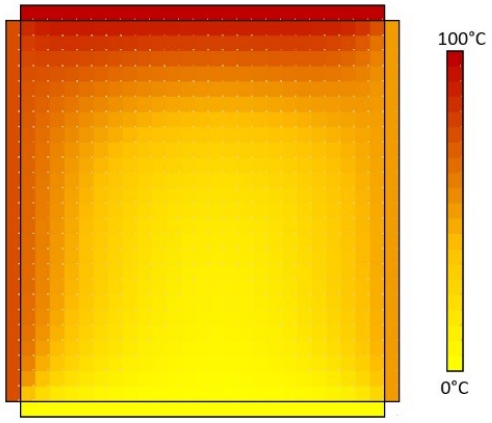



**Figura 8**

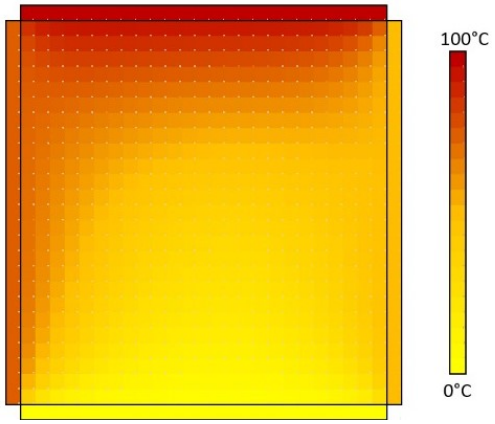



**Figura 9**

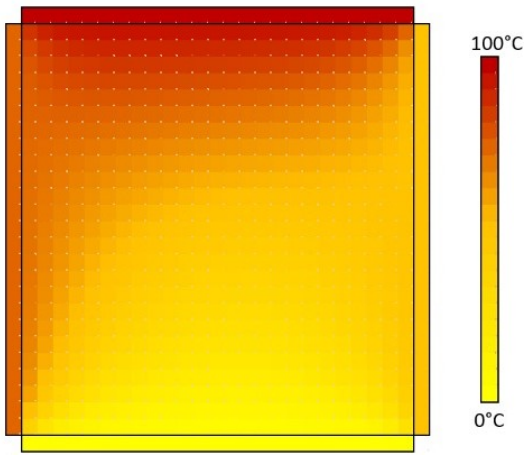


**Figura 10**

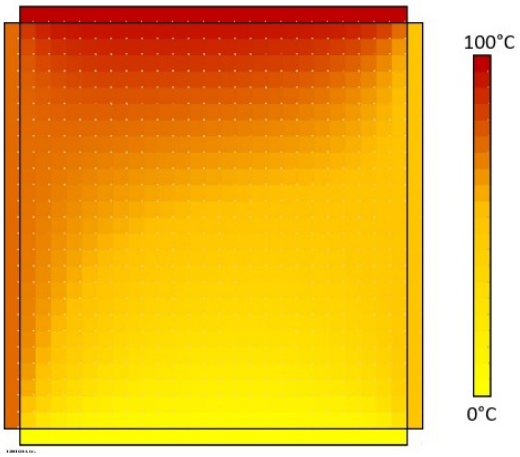



**Figura 11.** Solución encontrada.