# Programación básica
# Proyecto 2: Temperatura en una placa, estado estacionario

Guillermo Segura Gómez

30 de octubre de 2018

## 1. Introducción

La temperatura es una magnitud física que nos indica que tan caliente esta un sistema y esta asociada con la energía de dicho sistema. Se puede intercambiar energía entre dos o más cuerpos debido a la diferencia en sus temperaturas. **El equilibrio térmico** es una situación en la que dos objetos no intercambian energía pues se encuentran a la misma temperatura. Esto es lo que nos dice la ley cero de la termodinámica *"Si los objetos A y B estan separadamente en equilibrio térmico con un tercer objeto C, entonces A y B estan en equilibrio térmico entre sí"*. Ahora, si consideramos esto en un caso práctico, podemos decir que una placa de metal, cuyos bordes tienen una temperatura constante, en algun punto llegará a estar en equilibrio térmico; tomando este principio podemos calcular su temperatura en cada punto, sabiendo unicamente las temperaturas iniciales de sus bordes.

## 2. Teoría

Se tiene una placa totalmente aislada excepto en sus bordes, donde la temperatura puede ajustarse a un nivel preestablecido. (Figura 1)

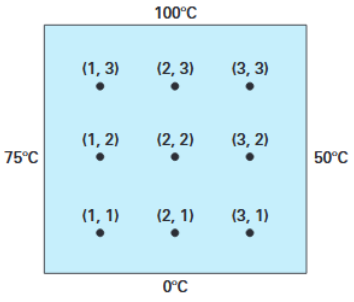


Figura 1: Ejemplo de placa, en donde las temperaturas de los bordes permanecen constantes

El aislamiento y el espesor de la placa permiten que la transferencia de calor este limitada solamente a las dimensiones $x$ y $y$. La temperatura en cada punto de la placa esta dada por la *ecuación de Laplace*:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{1}$$

Para la solución numérica de dicha ecuación, se puede tratar la placa como una matriz de puntos (i, j), tomando como ejemplo la figura 1, en donde cada punto de la matriz satisface la que es llamada *ecuación laplaciana en diferencias*:

$$T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1} - 4T_{i.j} = 0 \tag{2}$$

Se tiene que especificar, como ya había mencionado, y para obtener una solución úncia, que los bordes de la placa tengan una temperatura fija, en lo que se conoce como la *condicion de frontera de Dirichlet*, como en la figura 1. Para la resolución de la ecuación 2, se utilizará el *método Gauss-Seidel*, también llamado *método de Liebmann*. Con esta técnica la ecuación 2 se puede expresar como:

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4} \tag{3}$$

y se resuelve de manera iterativa para $j = 0$ hasta $n$ e $i = 0$ hasta $n$, ya que por simplicidad tomaremos una matriz cuadrada de lado $n$, como la figura 1. El resultado final terminará en una solución estable.

En el método de *Gauss-Seidel*, las iteraciones se repiten hasta que los valores absolutos de todos los errores relativos porcentuales $(\epsilon_a)_{i,j}$ están por debajo de un error especificado de terminación $(\epsilon_s)$. Los errores relativos se calculan mediante la siguiente ecuación:

$$|(\epsilon_a)_{i,j}| = |\frac{T_{i,j}^{nuevo} - T_{i,j}^{anterior}}{T_{i,j}^{nuevo}}|100\,\% \tag{4}$$

# 3. Código

Gracias a que la solución se resuelve de manera iterativa, es posible programar el problema. Los datos que tenemos incialmente son las temperaturas de los bordes, llamemoslas $a$ $b$ $c$ y $d$ comenzando por el borde izquierdo, después el superior, el derecho y finalmente el inferior; también tenemos el número de lados $n$ de la placa, y el error de terminación, en el programa llamado $ex$, pero en la teoría se le llama $\epsilon_s$. Estos datos son escaneados por el programa de un archivo llamado "base.txt". Para resolver el problema podemos programar la placa utilizando una matriz de $n * n$. Es conveniente utilizar una matriz dinámica, por lo que se tiene que reservar la memoria; ya que la matriz es de dimensiones $(i, j)$ se reserva la memoria primero para filas y después utilizando un ciclo se reserva para las columnas. Ya que los bordes de la matriz tienen un valor fijo, podemos trabajar con una matriz de $(n + 2) * (n + 2)$, esto para tener la matriz de $n * n$ mas los bordes. Si tomamos el caso simple de una matriz de 3x3, nuestro programa reserva memoria para una matriz de 5x5, en donde los lados extras representan los bordes como se muestra a continuación:

$$\begin{pmatrix} 0,0 & 0,1 & 0,2 & 0,3 & 0,4 \\ 1,0 & 1,1 & 1,2 & 1,3 & 1,4 \\ 2,0 & 2,1 & 2,2 & 2,3 & 2,4 \\ 3,0 & 3,1 & 3,2 & 3,3 & 3,4 \\ 4,0 & 4,1 & 4,2 & 4,4 & 4,4 \end{pmatrix}$$

Para poder realizar la matriz se tienen que utilizar dos ciclos *for* al mismo tiempo, el primer ciclo con el contador $i = 0$ crea las filas, y el segundo ciclo $j = 0$ las columnas. Por lo tanto, cada punto en la matriz tiene coordenadas $(i, j)$. Entonces, tomando en cuenta lo anterior, podemos restringir la matriz, los calculos los vamos a realizar en una matriz de $n * n$, por lo que nuestros contadores iran desde $(1, 1)$ hasta $(n+1), (n+1)$, y los bordes de la matriz $(i, 0)$ $(0, i)$ $(i, (n+1))$ y $((n + 1), i)$ se mantendrán con un valor constante. Cabe mencionar que las esquinas de la matriz, tanto las superiores como las inferiores, no importarán en el cálculo, ya que solo se toman en cuenta los valores próximos, como se muestra a continuación:

$$\begin{pmatrix} & i, j-1 & \\ i-1, j & i, j & i+1, j \\ & i, j+1 & \end{pmatrix}$$

Por lo tanto, se declaran dos funciones, una para inicializar todos los puntos de la matriz a un valor de 0.0, llamada *iniciar* y otra para inicializar los bordes de la matriz llamada *lados*. Para dar un ejemplo, podemos tomar el caso de la figura 1, en donde los bordes de la placa valen 75, 100,

50 y 0 respectivamente. La matriz inicializada quedaria asi:

$$\begin{pmatrix} & 100 & 100 & 100 & \\ 75 & 0 & 0 & 0 & 50 \\ 75 & 0 & 0 & 0 & 50 \\ 75 & 0 & 0 & 0 & 50 \\ & 0 & 0 & 0 & \end{pmatrix}$$

el valor que tomen las esquinas no nos importa como explique anteriormente, por lo que omiti dichos valores para este ejemplo.

Una vez con la matriz inicializada, se procede a realizar los cálculos; se tiene que declarar un ciclo, ya que el proceso es iterativo, y con cada iteración nos acercamos mas al resultado final. Por fines prácticos se declara un ciclo *while*, cuya condición de paro sea que el error calculado, (en la teoría llamado $\epsilon_a$ y en el programa llamado $p$) sea menor al error de terminación. Dentro del ciclo se declaran dos $for$ para la manipulación de la matriz; como se explicó anteriormente, se juega con los contadores para que solo se trabaje con una matriz de $n * n$ sin tocar los bordes. Se aplica la ecuación 3, y se sobreescribe cada punto de la matriz en cada iteración. Una vez completado el cálculo de la matriz para la primera iteración del ciclo *while*, se procede a calcular el error; se utilizan dos variables auxiliares, *eps* para guardar la division $(p^{nuevo} - p^{anterior})/p^{nuevo}$ y $e$ para una vez realizada la operación anterior, se guarde como el valor de $p^{nuevo}$ y asi, con la siguiente iteración, el valor de $e$, sea el de $p^{anterior}$, y poder realizar el cálculo.

Para la impresión de los archivos se declara un condicional $if$, que hace que se imprima cada 10 iteraciones o cuando el error calculado es menor al error de terminación, ya que esto quiere decir que se tendría el equilibrio. Para imprimir la matriz se llama a una función *imp* la cual va guardando las matrices en archivos con el nombre de *Pn.txt*, en donde el caracter $n$ es variable, y toma el valor de las letras mayúsculas del abecedario. Cuando se tiene el equilibrio final, un condicional dentro de la misma función hace que en lugar de guardarlo en un archivo genérico, lo guarde en uno llamado *Equilibrio_final.txt*.

Dentro del ciclo *while*, también se encuentra un contador $q$, que nos indica el número de iteraciones, y al final del ciclo, se tiene un condicional $if$, el cual establece una condición de paro, si $q$ es mayor a 500, el programa imprime la matriz final aunque el error de terminación siga siendo mayor al error calculado; en este caso, se imprime también en un archivo llamado *Equilibrio_final.txt*, aunque como mencione antes, no se haya alcanzado el equilibrio.

Finalmente, se imprime en la pantalla un diagnóstico con el número total de iteraciones, y los errores calculados. De igual forma, si la condición de paro se cumple, se imprime un diagnostico que nos dice que no se alcanzó el error esperado pues el número de iteraciones fue excedido.

# 4. Resultados

En el programa se resolvió el mismo caso de la figura 1, sin embargo en lugar de hacer una matriz de 3x3, se hizo una de 20x20. Para graficar los resultados se utilizó el programa Excel de Office, en donde al valor mas grande, en este caso 100, se le asigno un color azul, y al valor mas pequeño, en este caso 0, se le asigno un color rojo. Los otros valores, que estan entre 0 y 100, tienen un color de acuerdo a la escala de colores entre estos dos.

Para la primera iteración, se obtuvo la siguiente gráfica:

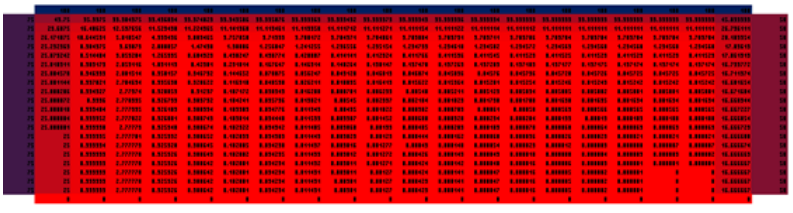

Figura 2: Primera iteración

En la primera iteración los valores no cambian mucho, y todos permanecen con una temperatura cercana a 0.

En la siguiente figura, se muestra la cuadragésima iteración:

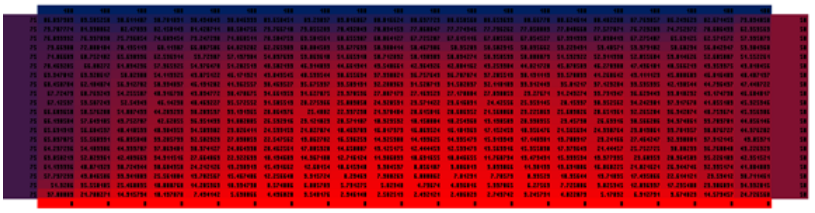

Figura 3: Cuadragésima iteración

Los valores poco a poco van distribuyendose, sin embargo aún se puede observar un color rojo dominante, lo cual no tendría que ser, puesto que tres de los cuatro bordes tienen temperaturas altas.

Ahora, se muestra la nonagésima iteración:

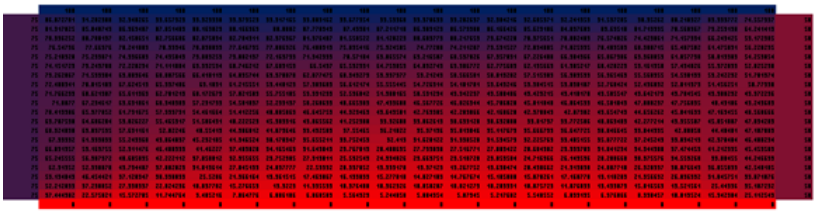

Figura 4: Nonagésima iteración

Cada vez, se ve mas distribuida la temperatura de la placa. El color rojo, domina menos, y ahora un color morado pasa a ser el principal dentro de la placa.

Pasemos ahora a la centésimo cuatrigésima iteración:

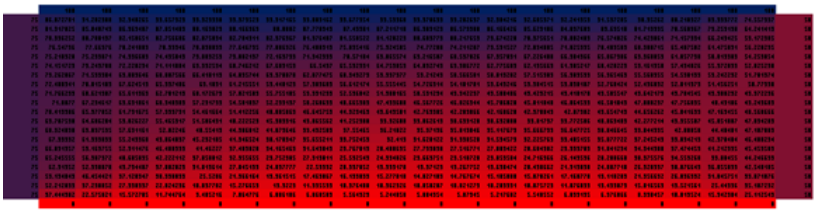

Figura 5: Centésimo cuatrigésima iteración

La figura ya no cambia mucho en comparación con la anterior. En este punto, la temperatura de la placa poco a poco se acerca al equilibrio térmico.

Finalmente, tenemos el equilibrio:

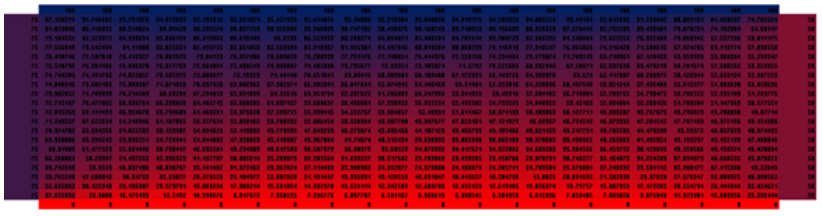

Figura 6: Equilibrio final

Podemos deducir, que el color rojo antes dominante en la placa, paso a ser minoria en comparación del nuevo dominante, un color morado. Se observa también que los colores mas claros, en donde la temperatura es mas pequeña, estan junto al borde inferior, y las temperaturas mas altas, estan junto al borde superior. Y no existen temperaturas iguales a las de los bordes, pues como todos tienen temperaturas diferentes, la temperatura de la placa en equilibrio también es diferente en todos sus puntos.

# 5. Conclusión

Podemos concluir, que gracias al método de aproximaciones, podemos calcular con increible precisión, la temperatura de una placa con solo saber que tan calientes estan sus bordes. El programa funciona para cualquier temperatura, y para cualquier tamaño de la matriz. Solo hay que mencionar, que entre mas grande sea la matriz, mas veces se tendra que iterar para llegar al resultado, pues son mas los puntos que tienen que llegar al equilibrio. También el valor del error de terminación influye mucho en el número de iteraciones, ya que entre mas pequeño sea, mas veces se tendrá que iterar para poder alcanzarlo. Si en algun dado caso, el error es de 0, también es posible llegar al resultado pues los tipos de variables que se estan usando son de tipo *double*, entonces en algun punto, el tamaño de la variable no va a poder guardar valores más pequeños, y por lo tanto se llegara al cero, aunque en si no lo sea, únicamente ya no alcanza para guardar valores mas pequeños.

# Referencias


[1] Steven C. Chapra, Raymond P. Canale *Métodos numéricos para ingenieros*, quinta edición, McGraw-Hill, 2007.

[2] Serway, Raymond A y Jewetl, John W. Jr, *Física, Para ciencias e ingenierias*, sexta edición, Thomson, México, DF, 20