Resolución de Ecuaciones Diferenciales Elípticas por el Método de las Diferencias Fintas

May 6, 2020

1 Introducción

El método de diferencias finitas no es mas que otra manera de resolver ecuaciones diferenciales con un enfoque numérico. El método es especialmente adecuado para resolver ecuaciones a derivadas parciales y se basa en digitalizar el espacio generando una malla y suponer que en un intervalo la derivada es constante, de manera de poder escribir las derivadas como diferencias entre los valores de ciertos puntos llamados nodos.

Con este método es posible resolver ecuaciones de dinámica de ondas y energía calórica, razón por la cual es muy utilizado en el ámbito de la ingeniería. Estas ecuaciones las podemos clasificar en tres tipos: Elípticas, Parabólicas e Hiperbólicas. En todos los casos el problema contiene dos variables con derivadas parciales, en el espacio y el tiempo (parabólicas e hiperbólicas) o un caso bidimensional en estado estacionario (elípticas). A continuación un resumen de dichas ecuaciones:

Elípticas

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

Parabólicas

$$\frac{\partial^2 U}{\partial x^2} - k\frac{\partial U}{\partial t} = 0 \tag{2}$$

Hiperbólicas

$$\frac{\partial^2 U}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2 U}{\partial t^2} = 0 \tag{3}$$

Con estas ecuaciones es posible resolver todos los problemas de ondas y de cambios en las temperaturas de diferentes materiales interactuando. T es la temperatura, x e y variables espaciales, t el tiempo y U es la perturbación que estamos modelando (temperatura, concentración, movimiento, etc) Las constantes c y k, representan la velocidad de propagación de la onda y la difusión de calor o masa respectivamente.

1.1 Mallado

El mallado no es otra cosa que la digitalización del dominio. Por lo tanto, ahora vamos a saber los valores de las funciones solo en esos puntos que previamente definimos, los nodos. Estos nodos dividen el espacio en elementos y para el caso 1D siempre hay, lógicamente, un nodo mas que elementos (primer elemento dos nodos y todos los demás un solo nodo).

Como vimos las ecuaciones elípticas dependen solo de dos variables espaciales. Es curioso que la solución de estas ecuaciones, dado el carácter estacionario, es independiente del material, por lo que no tenemos constantes físicas. Respecto a la malla, en la figura 1 se muestra un ejemplo para un dominio cuadrado, en el cual se utilizaron 5 nodos por cada variable.

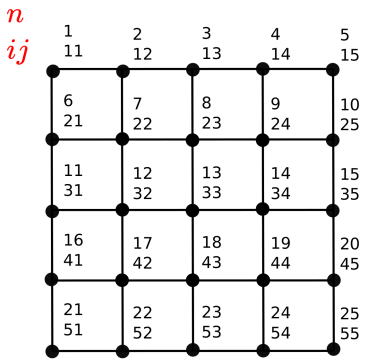

Figure 1: Dominio cuadrado con 5 nodos por dirección, se muestra la numeración n (orden de aparición de la variable) e ij (ubicación en el dominio).

En la figura 1 se muestran las dos maneras en las que comúnmente nos referimos a los nodos para nombrarlos: por orden de aparición de la incógnita (n) o por la ubicación en el dominio (ij).

Ahora, en la figura 2, vamos a aplicar el mallado a un caso real, un problema de calor: disponemos de una placa cuadrada de 1m por 1m, y cada borde está en contacto con una fuente de calor que los mantiene a temperaturas constantes y diferentes: T_1^b, T_2^b, T_3^b y T_4^b. En la misma figura re-enumeramos los nodos ahora que conocemos los valores de los bordes, totalizan 9 incógnitas.

En la figura 2, los círculos huecos son las condiciones de borde mientras que los sólidos las incógnitas. Noten que los subíndices ij incluyen a las condiciones de borde, éstos podría no ser así, aunque al resultar más simple su implementación se las incluye. En este punto ya tenemos nuestro dominio digitalizado con los nodos, solo resta calcular las incógnitas, para lo que utilizaremos las ecuaciones elípticas discretizadas.

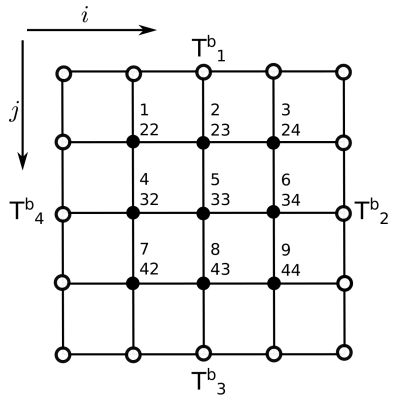

Figure 2: Condiciones de borde e incógnitas del dominio.

1.2 Discretización de la Ecuación Elíptica

La discretización de la ecuación elíptica se realiza, como ya mencionamos, suponiendo que durante un tramo la derivada no cambia. Recordando que los subíndices i y j refieren a moverse en x e y respectivamente, podemos escribir la expresión para las derivadas primeras discretas:

$$\frac{\partial U}{\partial x} = \frac{T_{i+1}^j - T_i^j}{dx} \quad y \quad \frac{\partial U}{\partial y} = \frac{T_i^{j+1} - T_i^j}{dy} \tag{4}$$

Y las derivadas segundas:

$$\frac{\partial^2 T}{\partial x^2} = \frac{\frac{\partial T}{\partial x}_{i+1} - \frac{\partial T}{\partial x}_{i}}{dx} = \frac{\frac{T_{i+1}^j - T_i^j}{dx} - \frac{T_i^j - T_{i-1}^j}{dx}}{dx} = \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{dx^2} \tag{5}$$

$$\frac{\partial^2 T}{\partial y^2} = \frac{\frac{\partial T}{\partial y}_{i+1} - \frac{\partial T}{\partial y}_{i}}{dy} = \frac{\frac{T_{i+1}^j - T_i^j}{dy} - \frac{T_i^j - T_{i-1}^j}{dy}}{dy} = \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{dy^2} \tag{6}$$

Las cuales al ser reemplazadas en la ecuación 1 queda:

$$\frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{dx^2} + \frac{T_i^{j+1} - 2T_i^j + T_i^{j-1}}{dy^2} = 0 \tag{7}$$

Con el objetivo de simplificar el cálculo aun más, supondremos que dx=dy. Esta aproximación es adecuada si ambas longitudes totales son similares, caso contrario conviene optar por valores diferentes de dx y dy permitiendo de este modo la densificación de la malla en zonas de poco espesor. Teniendo esto en cuenta, la ecuación 7 queda (independiente de dx y dy!!):

$$T_i^{j+1} + T_{i+1}^j - 4T_i^j + T_i^{j-1} + T_{i-1}^j = 0 \tag{8}$$

A primera vista, la ecuación 8 resulta incompleta, en el sentido de que tenemos 5 incógnitas en una sola ecuación. Pero recuerden que esta ecuación hay q escribirla para cada uno de los nodos, por lo que vamos a tener 9 ecuaciones, una por nodo. Además, en la ecuación participan la incógnita de cada nodo en particular (T_i^j) y sus 4 vecinos de arriba (T_i^{j-1}), abajo (T_i^{j+1}), derecha (T_{i+1}^j) e izquierda (T_{i-1}^j). Estos vecinos pueden ser en ocasiones condicione de borde, por lo que tendrán valores definidos, o en ocasiones una incógnita.

Finalmente, vamos a poder escribir esta ecuación para los 9 nodos incógnita, resultando en 9 ecuaciones distintas con 9 incógnitas, lo que significa que es posible hallar una solución única (casi un trabalenguas).

El sistema resultante de de 9x9.....

Escribirlo es muy simple aunque tedioso. A continuación escribimos las tres primeras filas, tomando $T_1^b = 10$,$T_2^b = 20$,$T_3^b = 30$ y $T_4^b = 40$. Las incógnitas las vamos a numerar por orden de aparición, ya que resulta mas simple para armar el vector de incógnitas. Se muestra una ampliación del primer nodo para orientarse en el armado de la primer ecuación en la figura 3.

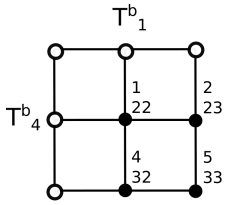

Figure 3: Ampliación del nodo de la incógnita T_1 y sus vecinos contiguos.

Ecuaciones:
$40 + 10 - 4T_1 + T_2 + T_4 = 0$
$T_1 + 10 - 4T_2 + T_3 + T_5 = 0$
$T_2 + 10 - 4T_3 + 20 + T_6 = 0$

Del mismo modo se procede hasta escribir las 9 ecuaciones, y resulta un sistema que siempre tiene -4 en la diagonal, también hay varios 1 en los valores anterior y siguiente de varias diagonales y otros 1 un poco mas alejados. Los 1 vacantes, que son ocupados por ceros, resultaron de reemplazar la variable por una condición de borde para esa ecuación en particular, por lo que en las filas en las q no estén todos los 1, el vector de términos independientes tendrá como valor la suma de esas condiciones de borde cambiadas de signo.

Luego de armar el sistema solo resta resolverlo por alguno de los métodos de resolución de sistemas lineales que vimos: métodos directos: Jordan y Gauss-Jordan; métodos indirectos: Jacobi y Gauss-Seidel.

Para la resolución del sistema resultante de discretizar una ecuación elíptica, resulta muy ventajosa la implementación de un método indirecto. De esta manera, evitamos tener que armar el sistema de ecuaciones $Ax = b$, lo cual es trabajoso. Recordemos que en el método de Jacobi se utilizaba una solución semilla, la cual era mejorada iterativamente para resolver el sistema y las x^{k+1}

40.000	10.000	10.000	10.000	20.000
40.000	25.000	19.464	17.857	20.000
40.000	30.536	25.000	21.964	20.000
40.000	32.143	28.036	25.000	20.000
30.000	30.000	30.000	30.000	20.000

Figure 4: Valores obtenidos luego de calcular la ecuación elíptica.

de la siguiente iteración se obtenían al despejar la variable i de la fila i del sistema. Pues en nuestro caso ese despeje es muy simple, dado que la ecuación que utilizamos en cada nodo siempre es la 8. Dicho despeje se muestra en la ecuación 9.

$$T_i^j = \frac{T_i^{j+1} + T_{i+1}^j + T_i^{j-1} + T_{i-1}^j}{4} \tag{9}$$

Vemos en la ecuación 9, que las x^{k+1} se obtienen simplemente del promedio entre sus 4 vecinos. Respecto a la convergencia del método indirecto, podemos verificar que el sistema es diagonalmente dominante, ya que siempre tenemos un 4 en la diagonal, y a lo sumo 4 números 1 en el resto de la filas (compruebe esto!), por lo que el sistema siempre converge a una solución correcta.

A continuación, en la figura 4 mostramos el archivo resultante del cálculo de Elípticas utilizando las 4 condiciones de borde propuestas:

A continuación se muestra el código FORTRAN de una subrutina que resuelve ecuaciones elípticas:

Encabezado

```
program elipticas
implicit none
integer,parameter::nx=10,ny=nx
real(8),parameter::lx=10.,ly=lx,iter=10000,tol=1E-5
!vector semilla
u=0.
!condiciones de borde CAMBIAR!!
u(1,:)=0. !(arriba)
u(:,1)=10. !(izq)
u(ny,:)=100.!(abajo)
u(:,nx)=5.!(derecha)
```

Resolución de elípticas

```
subroutine eliptic(u,tol)
real(8) T(nx,ny),Tant(nx,ny),tol,error
integer i,j,k
k=0
```

```
error=2*tol
do while ((error.gt.tol).and.(k.lt.iter)) !Comienza el ciclo de jacobi
   uant=u !Guardo u en iteracion anterior
   do i=2,nx-1
      do j=2,ny-1
         T(i,j) = (T(i-1,j) + T(i+1,j) + T(i,j-1) + T(i,j+1))/4.
      enddo
   enddo
   error=norma(Tant,T) !Recalculo el error
   k=k+1
enddo
end subroutine
```

1.3 Condición de Frontera de Calor (Neumann)

Hasta este punto, solo tomamos en cuenta condiciones de borde que imponen un valor a la temperatura, llamadas de Dirichlet. También podemos encontrar otro tipo de condición de borde, en las que imponemos un valor de la derivada de T. Estas condiciones son conocidas con el nombre de Neumann.

Las condiciones sobre la derivada se utilizan en fronteras donde conocemos el calor que esta fluyendo. Para calcular el calor utilizamos la ecuación 10.

$$k\frac{\partial T}{\partial x} = q \tag{10}$$

Donde q es el calor que fluye y k la conductividad térmica. Particularmente, q puede tomar dos valores: 0 si es una superficie adiabática o, cuando hay convección $q = h(T - T_0)$ donde h es el coeficiente de convección y T_0 la temperatura ambiente.

1.3.1 Condición adiabática

Cuando una condición de borde es adiabática, estamos diciendo que el calor es cero, por lo que luego de discretizar la derivada primera en x nos queda la siguiente condición (mirar ecuaciones 4):

$$\frac{T^j_{i+1} - T^j_i}{dx} = 0 \Rightarrow T^j_{i+1} = T^j_i \tag{11}$$

Lo que quiere decir que en ese borde, hay dos temperaturas iguales,

la del borde y la siguiente. El mismo razonamiento lo podemos realizar para cualquier borde, y obtenidríamos el mismo resultado, solo que cambiando los subíndices, de modo a la izquierda resultará $T_1^j = T_2^j$, a la derecha $T_{n_x}^j = T_{n_x-1}^j$, arriba $T_i^1 = T_i^2$ y abajo $T_i^{n_y} = T_i^{n_y-1}$.

Otra opción algo mas compleja y precisa resulta al discretizar utilizando derivada primera centrada, donde entra el concepto de nodo ficticio, ya que hay que utilizar un nodo que en rigor se encuentra fuera del dominio:

$$\frac{T_{i+1}^j - T_{i-1}^j}{2dx} = 0 \Rightarrow T_{i+1}^j = T_{i-1}^j \tag{12}$$

Resulta evidente que si el nodo de T_i^j esta en un borde, alguno de los nodos $n-1$ o $n+1$ esta fuera del material. De todos modos, reemplazando esta condición de nodo ficticio en la ecuación 8, obtenemos esta nueva ecuación:

$$2T_{i-1}^j - 4T_i^j + T_i^{j+1} + T_i^{j-1} = 0 \tag{13}$$

En ambos casos, el único cambio es que para los nodos de el borde en cuestión, tendremos q agregar una ecuación, ya sea la 11 o la 13. A continuación se muestra como se modifica el código de la subroutine eliptic agregando las líneas que calculan los nodos adiabáticos de los bordes con la ecuación 11, recuerden quitar los ! correspondientes para activar la condición de borde:

Resolución de Elípticas con Condición de Borde Adiabática

```
subroutine eliptic(u,tol)
real(8) T(nx,ny),Tant(nx,ny),tol,error
integer i,j,k
k=0
error=2*tol
do while ((error.gt.tol).and.(k.lt.iter)) !Comienza el ciclo de jacobi
   uant=u !Guardo u en iteracion anterior
   do i=2,nx-1
      do j=2,ny-1
         T(i,j) = (T(i-1,j) + T(i+1,j) + T(i,j-1) + T(i,j+1))/4.
      enddo
   !T(i,ny)= T(i,ny-1) !Condicion adiabática en la direccion y (derecha)
   !T(i,1= T(i,2) !Condicion adiabática en la direccion y (izquierda)
   enddo
   !T(nx,:)= T(nx-1,:) !Condicion adiabática en la direccion x (abajo)
   !T(1:)= T(2,:) !Condicion adiabática en la direccion x (arriba)
   error=norma(Tant,T) !Recalculo el error
   k=k+1
```

enddo
end subroutine

1.3.2 Condición de Calor

Otra posibilidad surge cuando el calor no vale 0, generalmente en casos de convección. En estos casos $q = h(T - T_0)$ donde h es el coeficiente de convección y T_0 la temperatura ambiente. En estos casos, el balance de calor queda:

$$k\frac{\partial T}{\partial x} = h(T - T_0) \tag{14}$$

y discretizando:

$$k\frac{T_{i+1}^j - T_i^j}{dx} = h(T_i^j - T_0) \tag{15}$$

y reordenando:

$$T_{i+1}^j = \frac{1}{1 - \frac{hdx}{k}}(T_i^j - \frac{hdx}{k}T_0) \tag{16}$$

Que es la ecuación que deberíamos agregar al sistema para resolver. Para la implementación, basta utilizar esta ecuación en vez de $T_i^j = T_{i+1}^j$ en el código. Recuerden que los índices j e i, pueden varia dependiendo del borde en cuestión, la ecuación 16 vale para el borde derecho ($i + 1 = n_x$).