ASIGNATURA: Fundamentos de Dinámica de Fluidos Computacional
PROFESOR: Christopher Cooper, Nicolás Thiers
FECHA: 14 de octubre de 2016

Tarea 2
Método de volumenes finitos

Conducción de calor en una placa no cuadrada

Digamos que tenemos una placa con la geometría de la Figura 1, por la cual entra calor por abajo, removemos calor por arriba, y está aislada en los costados, y necesitamos saber su distribución de temperatura. Sabemos que en la cara inferior el flujo de calor es $q = q_0$, y queremos encontrar la cantidad de calor que debemos remover en el borde superior para que la temperatura promedio de la placa no supere $T_{\max}$ después de $t = 1$s, y así diseñar el sistema de refrigeración. Para hacer este estudio, resolverán la ecuación de conducción de calor con el método de volúmenes finitos.

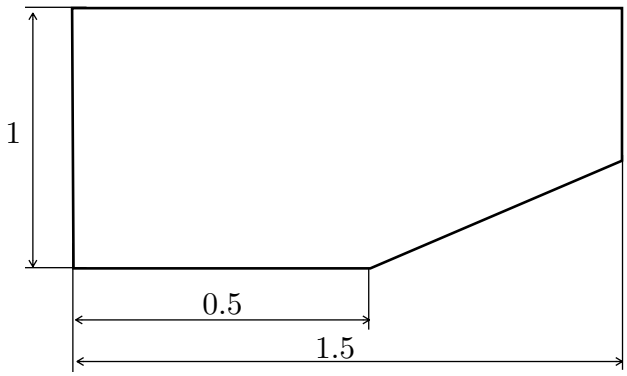

Figura 1: Geometría de la placa. El ángulo abajo es $\theta = 15°$.

Método de volumenes finitos

Ecuación de calor con volumenes finitos

Aprendimos en clases que el método de volumenes finitos resuelve ecuaciones conservativas, en forma integral. En el caso de la ecuación de calor

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T, \tag{1}$$

se puede escribir de forma conservativa como

$$\frac{\partial T}{\partial t} - \nabla \cdot (\alpha \nabla T) = 0 \tag{2}$$

donde al término $\mathbf{F} = \alpha \nabla T$ se le llama flujo. En forma integral, esta ecuación queda como

$$\frac{\partial}{\partial t} \int_{\Omega} T \mathrm{d}V - \oint_{\partial \Omega} \alpha \nabla T \cdot \mathbf{n} \mathrm{d}S, \tag{3}$$

para un volumen de control Ω con contorno $\partial\Omega$, y vector unitario normal $\mathbf{n}$, que siempre apunta hacia afuera.

La Figura 2 presenta una representación bidimensional de un volumen finito, con flujo $\mathbf{F} = (f, g)$. Para este caso, al aproximar las integrales con el método del trapecio en la ecuación (3), quedamos con

$$\begin{aligned} \frac{\partial T_{ij}}{\partial t} A_{ABCD} = \alpha [& f_{i-\frac{1}{2}j} \Delta y_{DA} - g_{i-\frac{1}{2}j} \Delta x_{DA} \\ & + f_{ij-\frac{1}{2}} \Delta y_{AB} - g_{ij-\frac{1}{2}} \Delta x_{AB} \\ & + f_{i+\frac{1}{2}j} \Delta y_{BC} - g_{i+\frac{1}{2}j} \Delta x_{BC} \\ & + f_{ij+\frac{1}{2}} \Delta y_{CD} - g_{ij+\frac{1}{2}} \Delta x_{CD}] \end{aligned} \tag{4}$$

donde $f = \frac{\partial T}{\partial x}$ y $g = \frac{\partial T}{\partial y}$.

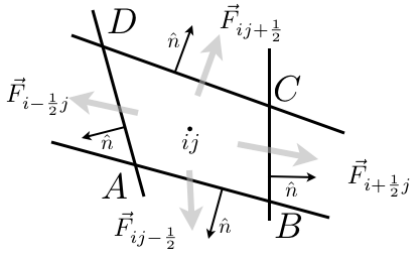

Figura 2: Volumen finito en 2D.

Cálculo del flujo

La mayor complicación de este trabajo es el cálculo del flujo. Existen muchas formas de calcular los flujo, pero en este caso, seguiremos la formulación vista en clases. Para la discusión, fijémonos en la Figura 3, donde queremos calcular el flujo f_e y g_e entre los nodos A y B. Como aproximación, podemos

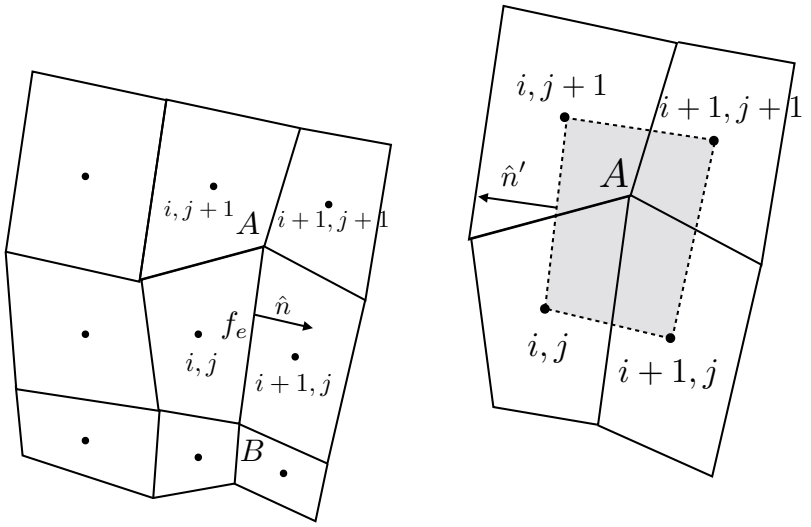

Figura 3: Cálculo del flujo.

decir que el el flujo sobre esa cara es el promedio de los flujos en A y B:

$$f_e = \frac{1}{2}(f_A + f_B)$$
$$g_e = \frac{1}{2}(g_A + g_B). \tag{5}$$

La pregunta es ahora ¿Cómo calcular los flujos en los puntos A y B? Nuevamente, podemos aproximar esto igualándolo al promedio del flujo dentro del área generada por los cuatro puntos contiguos, siendo la definición del promedio

$$f_A = \bar{f} = \frac{\int_{abcd} f \mathrm{d}S}{A_{abcd}}, \tag{6}$$

donde a, b, c, y d, son los puntos i, j; $i+1, j$; $i+1, j+1$; $i, j+1$ en la Figura 3, respectivamente. La integral de área en la Ecuación 6 puede ser pasada a una integral de línea a lo largo del contorno de A_{abcd} por el teorema de la divergencia

$$\int_{abcd} \alpha \frac{\partial T}{\partial x} \mathrm{d}S = \oint_{abcd} T\hat{\mathbf{i}} \cdot \mathbf{n} \mathrm{d}l \tag{7}$$

y aproximando la integral sobre cada lado con el teorema del punto medio, queda

$$f_A = \frac{\partial T}{\partial x} \approx \frac{1}{2A_{abcd}} \left[(T_a + T_b)\Delta y_{ab} + (T_b + T_c)\Delta y_{bc} + \right.$$
$$\left. (T_c + T_d)\Delta y_{cd} + (T_d + T_a)\Delta y_{da} \right]. \tag{8}$$

De la misma forma, podemos obtener una expresión para g_A:

$$g_A = \frac{\partial T}{\partial y} \approx -\frac{1}{2A_{abcd}} \left[(T_a + T_b)\Delta x_{ab} + (T_b + T_c)\Delta x_{bc} + (T_c + T_d)\Delta x_{cd} + (T_d + T_a)\Delta x_{da}\right]. \quad (9)$$

Recuerden que en todas estas derivaciones, hemos usado $\Delta y_{ab} = y_b - y_a$, no al revés!

Condiciones de borde

Afortunadamente, todas las condiciones de borde en este problema son de Neumann, que son más fáciles de lidiar que Dirichlet. Da la coincidencia que la condición de borde de Neumann es exactamente el negativo del flujo de nuestro volumen finito, multiplicado por k. Por lo tanto, en las caras adyacentes a un borde

$$q = -k\frac{\partial T}{\partial \mathbf{n}} = -k\nabla T \cdot \mathbf{n} = -k\frac{\partial T}{\partial x}n_x - k\frac{\partial T}{\partial y}n_y = -kfn_x - kgn_y. \quad (10)$$

Presentación del problema

El método de volumenes finitos nos permite usar mallas no cartesianas, y vamos a aprovecharnos de eso. La Figura 4 muestra la malla que usarán, eso si, ustedes tendrán 30×20 celdas.

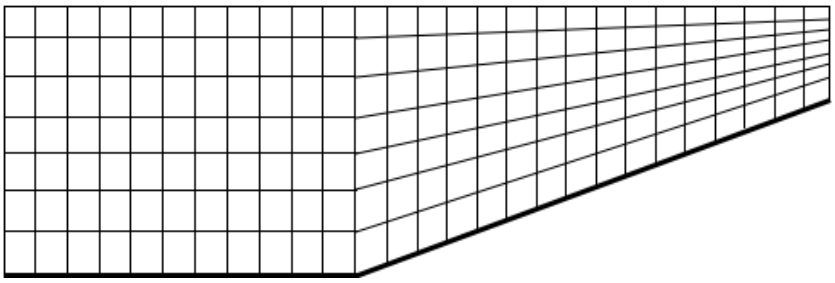

Figura 4: Malla.

Encuentren el calor removido en la parte superior discretizando en el tiempo con Euler explícito, y utilizando $\Delta t = 0{,}001$s, $q_0 = -k\frac{\partial T}{\partial \mathbf{n}} = 10$, $k = 1$, $\alpha = 0{,}1$, $T_{\max} = 24°$C, y condición inicial $T_0 = 20°$C en todas partes. Como referencia, la distribución de temperatura después de 50 pasos de tiempo con $q_{\text{salida}} = -10$ se ve como en la Figura 5, usando el comando `pyplot.contourf`.

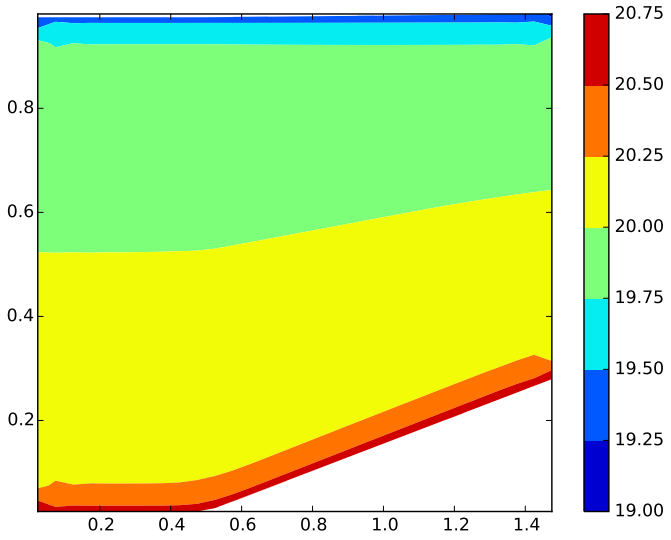

Figura 5: Resultado después de 50 pasos de tiempo.

También, varíen α, Δt y el espaciamiento de malla y encuentren los límites de estabilidad del método ¿Cómo se compara con lo que sabemos teóricamente de diferencias finitas?

Ayuda

Para apoyarlos en el desarrollo de su proyecto, vamos a entregarles código que los ayude a generar la malla. A pesar que la malla no es cartesiana, tiene una estructura clara, y la aprovechamos para guardar la malla en un arreglo bidimensional, que llamamos `mesh`, donde la celda `mesh[j,i]` está abajo de `mesh[j+1,i]`, a la izquierda de `mesh[j,i+1]`, arriba de `mesh[j-1,i]` y a la derecha de `mesh[j,i-1]`. Cada elemento del arreglo bidimensional es una variable del tipo `cell`, creada por nosotros mismos. En el tipo de variable (o clase) `cell` está guardada toda la información de la celda: vértices, área, normales, centro y las temperaturas actual y del tiempo futuro:

- `mesh[j,i].vSW`: la posición del vértice sur-oeste de la celda i, j.
- `mesh[j,i].vSE`: la posición del vértice sur-este de la celda i, j.
- `mesh[j,i].vNE`: la posición del vértice nor-este de la celda i, j.
- `mesh[j,i].vNW`: la posición del vértice nor-oeste de la celda i, j.

- `mesh[j,i].nN`: vector unitario normal a la cara norte de la celda i, j.
- `mesh[j,i].nS`: vector unitario normal a la cara sur de la celda i, j.
- `mesh[j,i].nE`: vector unitario normal a la cara este de la celda i, j.
- `mesh[j,i].nW`: vector unitario normal a la cara oeste de la celda i, j.
- `mesh[j,i].area`: area de la celda i, j.
- `mesh[j,i].T`: guarda el valor de la temperatura en el tiempo futuro.
- `mesh[j,i].Tn`: guarda el valor de la temperatura en el tiempo presente.

Además, el código viene con funciones para calcular el área y llenar las condiciones iniciales. A pesar de ser una forma muy práctica de guardar la data, tiene el problema que no es posible operar en forma vectorial, que sabemos que es muchísimo más rápido en Python. Por ejemplo, `mesh[:,:].T` **no** entrega un arreglo bidimensional con las temperaturas en todos los nodos.