Método de Diferencias Finitas

Introducción a la Programación

Mecánica Computacional

FICH - UNL

Ecuación de Calor: Difusión con Reacción y Fuente

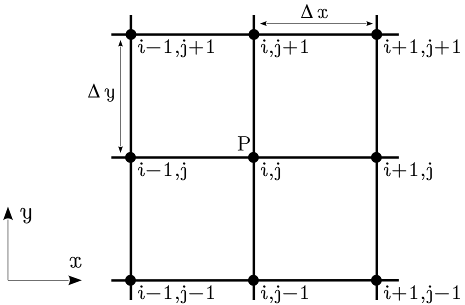
$$-k\left(\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}\right)+c\phi=G \quad (1)$$

Término Difusivo : $-k\left(\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}\right)$

Término Reactivo : $c\phi$

Término Fuente : G

Dominio del Problema 2D



Malla con espaciado uniforme:

$$\begin{aligned} \Delta x &= x_{i+1} - x_i,\ i = 1, 2, \ldots, N \\ \Delta y &= y_{j+1} - y_j,\ j = 1, 2, \ldots, M \end{aligned} \quad (2)$$

Ecuación en diferencias

Stencil general de Diferencias Finitas 2D para nodos interiores de una malla con espaciado uniforme Δx y Δy:

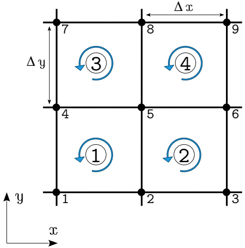
$$-k\left(\frac{\phi_{i+1,j}-2\phi_{i,j}+\phi_{i-1,j}}{\Delta x^2}+\frac{\phi_{i,j+1}-2\phi_{i,j}+\phi_{i,j-1}}{\Delta y^2}\right)+c\phi_{i,j}=G_{i,j} \quad (3)$$

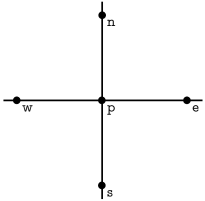
$$\text{Término Difusivo}: -k\left(\frac{\phi_{i+1,j}-2\phi_{i,j}+\phi_{i-1,j}}{\Delta x^2}+\frac{\phi_{i,j+1}-2\phi_{i,j}+\phi_{i,j-1}}{\Delta y^2}\right)$$

$$\text{Término Reactivo}: c\phi_{i,j}$$

$$\text{Término Fuente}: G_{i,j}$$

Representación geométrica





xnode

nro	x	y
1	0,00	0,00
2	0,50	0,00
3	1,00	0,00
4	0,00	0,50
5	0,50	0,50
6	1,00	0,50
7	0,00	1,00
8	0,50	1,00
9	1,00	1,00

neighb

nro	s	e	n	w
1	-1	2	4	-1
2	-1	3	5	1
3	-1	-1	6	2
4	1	5	7	-1
5	2	6	8	4
6	3	-1	9	5
7	4	8	-1	-1
8	5	9	-1	7
9	6	-1	-1	8

icone

nro	i1	i2	i3	i4
1	1	2	5	4
2	2	3	6	5
3	4	5	8	7
4	5	6	9	8

Estructura general del código

Ejemplo de llamada principal al Método de Diferencias Finitas en 2D:

```
function [PHI,Q] = fdm2d(xnode, icone, DIR, NEU, ROB, model)
    %Inicializar variables principales del sistema
    [K,F] = fdm2d_initialize(model.nnodes);

    %Armado de la matriz de vecindad
    [neighb] = fdm2d_neighbors(icone);

    %Ensamble de coeficientes del sistema
    [K,F] = fdm2d_gen_system(K,F,xnode,neighb,model.k,model.c,model.G);

    %Ensamble de nodos frontera Neumann
    [F] = fdm2d_neumann(F,xnode,neighb,NEU);

    %Ensamble de nodos frontera Robin
    [K,F] = fdm2d_robin(K,F,xnode,neighb,ROB);

    %Ensamble de nodos frontera Dirichlet
    [K,F] = fdm2d_dirichlet(K,F,DIR);

    %Resolver el sistema lineal de ecuaciones
    [PHI,Q] = fdm2d_solve(K,F,xnode,neighb,model);
end
```

Estrategia

En ProMetheus:

1. Crear un proyecto nuevo de programación para Diferencias Finitas.
2. Familiarizarse con el entorno de programación e identificar los módulos a programar.

Estrategia (cont.)

3. Comenzar por las condiciones de borde. Son módulos más pequeños en cantidad de código y focalizados en una sóla tarea en particular. Orden sugerido:
   1. `fdm2d_dirichlet.m`: el más sencillo de todos. Consiste en reemplazar los nodos de borde por ecuaciones triviales del tipo $T_{borde} = \overline{T}$.
   2. `fdm2d_neumann`: siguiente en dificultad. Explicado más adelante.
   3. `fdm2d_robin`: un poco más elaborado, pero continúa la línea de las condiciones de borde anteriores.

Estrategia (cont.)

4. Continuar con el módulo principal del método numérico: `fdm2d_gen_system.m` teniendo en cuenta que:
   - Este módulo se ejecuta antes que las condiciones de borde y sólo modifica los nodos interiores de la malla.
   - Conviene comenzar con las expresiones para mallas con espaciado constante y una vez validados los resultados modificar las expresiones para adaptarlas para los casos de espaciado no constante.
5. Módulo para cálculo del flujo de calor `fdm2d_flux.m` (provisto por la cátedra).

Hasta aquí lo mínimo para poder probar un problema completo en estado estacionario.

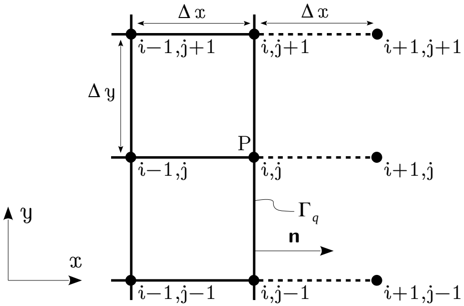
Estrategia (final)

Esquemas temporales:

6. Médodo explícito: `fdm2d_explicit.m`.
7. Médodo implícito: `fdm2d_implicit.m`.

Resulta más cómodo utilizar el planteo vectorial de las expresiones para estado transiente.

Frontera Neumann (ejemplo borde derecho)



Frontera Neumann (ejemplo borde derecho)

Codición de borde Neumann (ecuación diferencial):

$$-k\left(\frac{\partial\phi}{\partial n_e}\right)=\overline{q},\quad n_e\in\Gamma_e \tag{4}$$

Codición de borde Neumann (ecuación en diferencias):

$$-k\frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x}=q,\quad \forall(i,j)\in\Gamma_e \tag{5}$$

Stencil completo (borde derecho):

$$-k\left(\frac{2\phi_{i-1,j}-2\phi_{i,j}}{\Delta x^2}+\frac{\phi_{i,j+1}-2\phi_{i,j}+\phi_{i,j-1}}{\Delta y^2}\right)+c\phi_{i,j}=G_{i,j}-\frac{2q}{\Delta x} \tag{6}$$

FDM 2D - Frontera Neumann

Diferencias Finitas 2D - Frontera Neumann:

```
function [F] = fdm2d_neumann(F,xnode,neighb,NEU)
    %NEUMANN
    M = size(NEU, 1);
    for n = 1 : M
        P = NEU(n, 1);              %centro
        S = neighb(P, 1);           %[1] sur
        E = neighb(P, 2);           %[2] este
        N = neighb(P, 3);           %[3] norte
        W = neighb(P, 4);           %[4] oeste

        q = NEU(n,2);               %calor impuesto

        if (E == -1)
            dx = abs(xnode(W,1) - xnode(P,1));
        end

        if (NEU(n,3) == 2)          %[2] frontera este
            F(P) = F(P) - 2*q/dx;
        end
    end
end
```

Ensamble de la matriz K (gen_system)

Ensamble de la matriz K (gen_system) - Pseudocódigo:

```
function [K,F] = fdm2d_gen_system(K,F,xnode,neighb,k,c,G)
  %N cantidad de nodos de la malla
  for P = 1 : N
    %Averiguar la vecindad de P
    S = ...
    E = ...
    N = ...
    W = ...

    if (E ~= -1)
      de = %distancia entre los nodos P y E
    end
    ...
    %Idem para los nodos E, N, W
    ...
    %Coeficientes ec. en diferencias Eje x
    if (frontera_este)
      ax = 2 / (de * de)   %coef. nodo interior
      bx = -2 / (de * de)  %coef. nodo central
      cx = 0               %coef. nodo ficticio
    else (frontera oeste)
      ...
    else (nodo interior << no pertenece a un borde)
      ...
    end

    %Idem coeficientes  ec. en diferencias Eje y
    ...
```

Ensamble de la matriz K (gen_system) (cont.)

Ensamble de la matriz K (gen_system) - Pseudocódigo (cont.):

```
        ...
        %Contribuciones presentes en cualquier nodo
        K(P,P) = c(P) - k(P)*bx - k(P)*by;
        F(P) = G(P);

        %Si P tiene un vecino al este...sumar su contribucion
        if (E ~= -1)
            K(P,E) = -k(P)*ax;
        end
        ...
        %Idem S, N y W
  end
end
```