ASADES

Avances en Energías Renovables y Medio Ambiente
Vol. 7, N° 2, 2003. Impreso en la Argentina - ISSN: 0329-5184

UN PAQUETE *Mathematico* PARA EL CALCULO DE CONDUCCION EN 1D, 2D y 3D CON ACELERACION POR MULTIGRILLAS

Mónica Cruz, Elda Canterle y Luis Cardón
Facultad de Ciancias Exactas - INENCO
Universidad Nacional de Salta, Buenos Aires 177, 4400 Salta, Argentina
cruz@unsa.edu.ar

RESUMEN. Se describe un paquete para el programa *Mathematica* que permite la resolución de problemas de difusión de calor no estacionaria con fuentes en dominios rectangulares de 1, 2 y 3 dimensiones. El dominio de cálculo puede tener propiedades uniformes, constantes por zonas, no-uniformes dependientes de la posición o de la temperatura. Las fuentes de calor pueden variar de la misma manera que las demás propiedades. Admite condiciones de borde de tipo Dirichlet, Neuman y Robinson. La resolución numérica se realiza con el método de volúmenes de control sobre redes estructuradas uniformes o no uniformes. La resolución de las ecuaciones de discretización se realiza por métodos iterativos acelerados con técnicas de multigrillas. Todos los esquemas básicos de aceleración, tales como ciclos V, W y μ han sido implementados.

Palabras claves: Mathematica, difusión, conducción, multigrillas

INTRODUCCION

Los problemas tipo difusión ocurren frecuentemente en el análisis de las aplicaciones energéticas tradicionales (Hernandez, 2001). Se ha encarado un Proyecto con el fin de desarrollar un programa general para la resolución de problemas de transferencia basado en el programa **Mathematica** con aceleración de convergencia por multigrillas, del cual este paquete es el primer módulo.

METODO DE DISCRETIZACION

La ecuación a resolver está dada por

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + S \tag{1}$$

donde T es la temperatura, sujeta a condiciones de contorno de tipo Dirichlet, Neumann o Robinson, sobre toda la frontera del dominio de cálculo,

$$T = T_B \quad \text{sobre} \quad \partial\Omega_D \qquad k\frac{\partial T}{\partial n} = h(T - T_a) \quad \text{sobre} \quad \partial\Omega_R \qquad q = 0 \quad \text{sobre} \quad \partial\Omega_N \tag{2}$$

Se discretizó con volúmenes de control sobre redes estructuradas no uniformes. La integración temporal se hace en forma totalmente implícita. Las ecuaciones de discretización, para el caso de dos dimensiones, en la nomeclantura de Patankar (1980) que ya es estandar y donde los subíndices en mayúscula indican nodos y en minúscula indican caras entre volúmenes de control, son:

$$a_P T_P = \sum_{nv} a_{nv} T_{nv} + b \tag{3}$$

$$a_E = k_e \frac{A_e}{(\delta x)_e}, a_O = k_o \frac{A_o}{(\delta x)_o}, a_N = k_n \frac{A_n}{(\delta y)_n}, a_S = k_s \frac{A_s}{(\delta y)_s}, a_U = k_u \frac{A_u}{(\delta z)_u}, a_D = k_d \frac{A_d}{(\delta z)_d} \tag{4}$$

con

$$a_P = \sum_{nv=e,o,n,s,u,d} a_{nv} - S_p \Delta x \Delta y, \qquad b = S_C \Delta x \Delta y - a p^0 T_P^0 \qquad y \qquad a_P^0 = \rho c_p \Delta x \Delta y / dt \tag{5}$$

donde la conductividad térmica es $k(x, y)$, el calor específico es $c_p(x, y)$, la densidad es $\rho(x, y)$,todas función de la posición. El término fuente S se admite función lineal de la temperatura de la forma $S = S_C - S_P T_P$.

Para cada paso de tiempo se obtiene entonces un problema lineal dado por

$$\mathbf{A}T = b \tag{6}$$

Las ecuaciones de discretización se resuelven con el método de Jacobi acelerado por multigrillas. Los métodos irerativos convergen muy lentamente cuando la red de discretización es muy grande. Los métodos de multigrilla permiten resolver en redes adecuadas a la frecuencia característica de la solución que se busca acelerando la convergencia. Para ello se resuelve el problema original y varios problemas similares asociados al error sobre un anidamiento de redes, de manera de resolver cada frecuencia de la solución en una red que le resulte óptima.

El algoritmo básico es el método de dos redes que se esquematiza en de la figura 1. Se trabaja con dos redes una fina, que denotamos Ω^h o de 2ndo. nivel y otra menos fina, Ω^H o de 1er. nivel. Se resuelve la ecuación 6 en la red Ω^h y se resuelve la ecuación de difusión del error en la red Ω^H. Los resultados de esta última se utilizan para corregir los resultados de la primera. En nuestro caso la red Ω^H tiene el doble de volúmenes de control que la red Ω^h por lo que la designaremos Ω^{2h} en lo sucesivo. Llamaremos aquí u al la variable independiente, en este caso $u = T$ y$f = b$. El error se define $e = u - v$ y la ecuación para el mismo satisface

$$Ae = r = f - Av$$

Esto sugiere que podemos relajar directamente en el error usando la ecuación residual ya que relajar la ecuación original $\mathbf{A}u = f$ con una semilla inicial v es equivalente a relajar en la ecuación residual $Ae = r$ con una semilla inicial $e = 0$. Esta íntima conección entre la ecuación original y la ecuación residual motiva el uso de esta última.

Sea h la grilla de nivel 2 para la cual queremos resolver el problema. Tendríamos siguiente esquema:

-Relajar la ecuación $Au = f$ en sobre Ω^h y obtenemos una aproximación v^h.

-Computar el residuo $r = f - Av^h$.

-Relajar la ecuación residual $Ae = r$ sobre Ω^{2h} para obtener una aproximación del error e^{2h}.

-Corregir la aproximación v^h obtenida en Ω^h con el error estimado obtenido en Ω^{2h} reemplazando v^hpor $v^h + e^{2h}$.

-Relajar $Au = f$ en Ω^hcon semilla inicial v^h obtenida en el paso anterior

LOS OPERADORES DE INTERPOLACION

Para poder pasar de una red Ω^h a la red Ω^{2h} o subir de esta a la primera se definen dos operadores que se denominan de restricción y de prolongación respectivamente. Estos se definen a continuación.

Sea N la cantidad de puntos de la grilla menos 1, el operador I^h_{2h} nos permite pasar de Ω^{2h} a Ω^h, tenemos $I^h_{2h}(v^{2h}) = v^h$, entonces las componentes de v^h están dadas por:

$$\begin{aligned}
v^h_{2i,2j} &= v^{2h}_{ij} \\
v^h_{2i+1,2j} &= \frac{1}{2}\left(v^{2h}_{ij} + v^{2h}_{i+1,j}\right) \\
v^h_{2i,2j+1} &= \frac{1}{2}\left(v^{2h}_{ij} + v^{2h}_{i,j+1}\right) \\
v^h_{2i,2j+1} &= \frac{1}{2}\left(v^{2h}_{ij} + v^{2h}_{i,j+1}\right) \\
v^h_{2i+1,2j+1} &= \frac{1}{4}\left(v^{2h}_{ij} + v^{2h}_{i+1,j} + v^{2h}_{i,j+1} + v^{2h}_{i+1,j+1}\right) \ \text{con} 0 \le i,j \le \frac{N}{2} - 1
\end{aligned}$$

El operador que permite pasar de Ω^h a Ω^{2h} es $I^{2h}_h(v^h) = v^{2h}$ tenemos que

$$v^{2h}_{ij} = \frac{1}{16}\left[\begin{array}{c}[v^h_{2i-1,2j-1} + v^h_{2i-1,2j+1} + v^h_{2i-1,2j-1} + v^h_{2i+1,2j+1} \\ +2\left(v^h_{2i,2j-1} + v^h_{2i,2j+1} + v^h_{2i-1,2j} + v^h_{2i+1,2j}\right) \\ 4v^h_{2i,2j}]\end{array}\right] \qquad \text{con} \quad 1 \le i,j \le \frac{N}{2} - 1$$

Para implementar estos dos operadores hemos definido dos funciones: `interha2h2D` y `inter2hah2D`.

El algoritmo de dos redes se puede combinar de muchas maneras. La figura 2 muestra lo que se denomina un ciclo μ, o algoritmo de multigrilla completo,

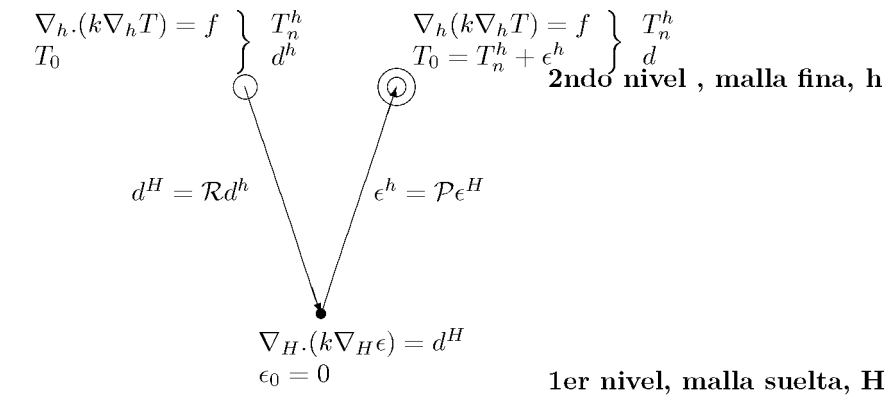

Figure 1: Algoritmo de dos redes. El subíndice cero indica campo de prueba para el comienzo de la iteración

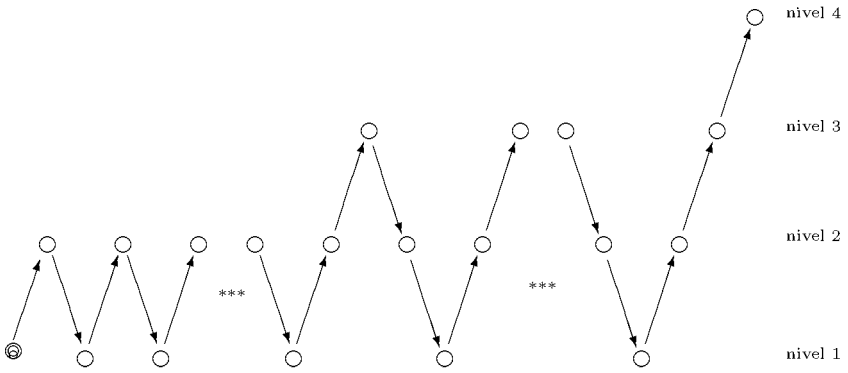

Figure 2: Esquema del algoritmo de multigrilla completo.

LOS PAQUETES DESARROLLADOS

A los efectos de resolver las ecuaciones enunciadas arriba se ha creado una serie de paquetes para el Programa *Mathematica*. Describiremos solo dos de ellos, adecuados a la resolución del caso estacionario bidimensional. En el primero se desarrollan la discretización de las ecuaciones y los métodos iterativos básicos para su resolución, en el segundo se desarrollan los métodos de aceleración de convergencia.

Paquete de discretización y relajación básico

Las funciones principales del primero de ellos son básicamente dos, la función **jacobi2D** que calcula los coeficientes de la matriz de discretización y aplica una paso de iteración con el método de Jacobi. La segunda, la función **relajarMetPon2D**, permite aplicar multiples iteraciones de Jacobi hasta alcanzar la convergencia de la solución. El paquete presenta además otras funciones auxiliares, otros métodos iterativos que no describiremos aquí, entre ellos una serie de métodos ponderados. La importancia del método de relajación empleado es relativa, si se tiene en cuenta que el propósito es usarlo en combinación con un método de aceleración de multigrillas.

La sintaxis de la función jacobi2D es la siguiente

```
jacobi2D[T_List,sc_List,sp_List,KK_List,G_List,Ti_Integer,Tf
Integer,Cbs_List,Cbi_List]
```

donde `T_List` es el campo de temperatura de prueba definido sobre la red `G_List`. Esta matriz contiene las coordenadas de los nodos de los los volumenes de control. Los demás parámetros se usa para dar los valores de la matriz de conductividades, los datos de la matriz de términos fuente con sus dos partes Sc y Sp, las datos de la temperatura sobre los bordes de Dirichlet Ti, Tf, Cbs, Cbi. La subrutina devuelve una matriz con el nuevo valor de campo T luego de la iteración.

La sintaxis de la función `relajarMetPon2D` es la siguiente

```
relajarMetPon2D[m_,T_List,s_List,KK_List,G_List,Ti_,Tf_,Cbs_List,Cbi_List,j_Integer]
```

permite realizar una cantidad j de iteraciones del método m que puede ser cualquiera de los programados (Jacobi o Jacobi Ponderado) u otro que se programe en el futuro.

Paquete de aceleración por multigrillas

El paquete de multigrillas implementa varios algoritmos de aceleración, entre ellos, el ciclo V, W y μ o multigrilla completa.

El ciclo V es una extensión del algoritmo de dos redes a múltiples redes. De la red fina se puede ir bajando el defecto a una red menos fina, donde se resuelve la ecuación residual, luego nuevamente se baja de nivel de la misma manera y así hasta que sea posible una solución exacta. Luego se invierte el ciclo corrigiendo las varias soluciones intermedias, usando en cada caso, la solución corregida como semilla para iniciar la relajación correspondiente al nivel. El ciclo V se ha implementado en la subrutina `esquemaCicloV2D`. Una vez realizado un ciclo V puede tomarse el resultado como valor de prueba y comenzarse tantos otros ciclos V como se quieran, esto se denomina ciclo W y se ha implementado en la subrutina `esquemaCicloW2D`. Es sabido que los métodos iterativos actúan mejor si se dispone de una buena solución de prueba. Se puede construir una buena solución de prueba partiendo de una red poco densa y prolongando el resultado a una red más fina, y así tantas veces hasta llegar a la red de nivel deseado. En cada nivel puede hacerce ciclos V o W. El algoritmo se esquematiza en la figura 2 y se denomina ciclo μ o de multigrilla completa. Este algoritmo se ha implementado en la subrutina `esquemaCicloMu2D`.

La sintaxis de `esquemaCicloV2D` es la siguiente

```
esquemaCicloV2D[T_List,s_List,KK_List,G_List,Ti_,Tf_,Cbs_List,Cbi
_List,m_,w_?Positive,r1_Integer,r2_Integer,g_?Positive]
```

la función devuelve la corrección de la semilla inicial T al aplicar un esquema de ciclo V. Además de los parámetros que definen la matriz A y que fueron descriptos en el caso de la subrutina `Jacobi2D`, esta subrutina recibe como parámetros a $r1$, $r2$ que definen el número de iteraciones de relajación durante la bajada y subida respectivamente. También recibe el parámetro g relacionado con el número de niveles del ciclo.

La sintaxis de `esquemaCicloW2D` es la siguiente

```
esquemaCicloW2D[T_List,s_List,KK_List,G_List,Ti_,Tf_,Cbs_List,Cbi
_List,m_,w_?Positive,r1_Integer,r2_Integer,g_?Positive]
```

Esta función devuelve la corrección de la semilla inicial al aplicar un esquema de ciclo W,

La sintaxis de `esquemaCicloMu2D` es

```
esquemaCicloMu2D[T_List,s_List,KK_List,G_List,Ti_,Tf_,Cbs_List,Cbi
_List,m_,w_?Positive,r1_Integer,r2_Integer,g_?Positive,u_Integer]
```

Esta función devuelve la corrección de la semilla inicial al aplicar un esquema de ciclo μ u veces.

EJEMPLO DE APLICACION

El siguiente es un pequeño ejemplo que muestra las posibilidades del programa y su modo de uso. Generalmente, como se hace en muchos códigos de propósito general, se divide el programa en dos partes, en una se llama las subrutinas de cálculo y se realiza el posprocesamiento de los resultados, y en otra se definen los datos del problema.

Consideremos el caso de un placa metálica que tiene tres franjas de conductividad eventualmente distinta. La placa se somete a conducción unidimensional, para lo cual se fija la temperatura en dos de sus bordes paralelos, en -10 y 10 C respectivamente. Los bordes transversales se hacen adiabáticos. El listado 1 muestra la implementación de los datos del problema para el caso que la conductividades de cada zona sean iguales (línea 19 del listado). Los resultados se muestran en la figura 3 a), obteniéndose un plano como era de esperar. Si cambiamos la conductividad de una franja, la central por ejemplo y le damos un valor sumamente elevado, no se observará gradiente de temperatura en ella, como se observa en la figura 3 b).

```
1   (* treszonas.m *)
2   NNx=17;    (* Nodos *)
3   NNy=17;
4   Nx=NNx-2;  (* V. de C.  interno *)
5   Ny=NNy-2;
6   Nzy1= Ny/3;(* nodos/zona en y *)
7   Nzy2= Ny/3;
8   Nzy3= Ny/3 ;
9   Nzx1= Nx;
10  pnzy1=1;   (* primer nodo zona 1 *)
11  pnzy2= pnzy1 + Nzy1;
12  pnzy3= pnzy2 + Nzy2;
13  pnzy4= pnzy3 + Nzy3;
14  pnzx1=1;
15  pnzx2= pnzx1 + Nzx1;
16  kkk={1,1,1}; (* conductividad/zonas *)
17  k[i_, j_] :=
18  Which[
19  (i >= pnzx1 && i < pnzx2),
20  Which[
21  (j >= (pnzy1-1) && j < pnzy2),kkk[[1]],
22  (j >= pnzy2 && j < pnzy3),kkk[[2]],
23  (j >= pnzy3 && j < (pnzy4+1)),kkk[[3]]],
24  (i==0),0,
25  (i==NNx-1), 0
26  ];
27  KK = Table[
28  k[i, j], {i, 0, NNx-1}, {j, 0, NNy-1}];
29  T = matrizNula[NNx-1,NNy-1];
30  G = Table[
31  {i,j}, {i, 0, NNx-1}, {j, 0, NNy-1}];
32  Ti = 10; Tf = -10;
33  Cbs= Table[0, {j, 0, NNy-1}];
34  Cbi = Table[0, {j, 0, NNy-1}];
35  sss={0,0,0};
36  se[i_, j_] :=
37  Which[
38  (i >= pnzx1 && i < pnzx2),
39  Which[((j == 0) || (j == NNy-1)),0,
40  (j >= pnzy1 && j < pnzy2),sss[[1]],
41  (j >= pnzy2 && j < pnzy3),sss[[2]],
42  (j >= pnzy3 && j < pnzy4),sss[[3]]],
43  (i==0),0,
44  (i==NNx-1),0
45  ];
46  s= Table[
47  se[i, j], {i, 0, NNx-1}, {j, 0, NNy-1}];
48  sp= matrizNula[NNx-1,NNy-1];
```

En el listado 1, de las líneas 2 a 15 se definen los parámetros de la red y las zonas en que ésta se divide. De las líneas 16 a 28 se asignan valores a las conductividades, primero por zonas, luego para cada nodo. En particular se asigna conductividad cero a los nodos sobre el borde, cuando éste es adiabáticos. En las líneas 29 y 31 se da una semilla para la temperatura y se genera la red (una matriz con las posiciones de los nodos de los volúmenes de control, en este caso uniforme). En las líneas 32 a 34 se especifican las temperaturas para los nodos sobre los bordes de Dirichlet. las líneas 35 a 48 se muestra como puede introducirse el término de generación interna, también por zonas. En este caso, los valores fueron puestos a cero en la línea 48. En el archivo `treszonass.m` se reemplaza por $sss = 0, 10^5, 0$ para introducir el término de generación distito de cero en la franja central.

El siguiente listado muestra como se ejecuta el programa cuyos resultados se muestran abajo. En la línea 1 se llama al paquete, en la línea 2 se llama a la definición del problema, en las línea 3 y 4 se efectua el cálculo y en la 5 se generan el gráficos. De las líneas 6 a 9 y 10 a 13 respectivamente se repite para calcular con otro conjunto de datos, los archivos `treszonasK.m` y `treszonasS.m`. La figura 3 es generada por la línea 14.

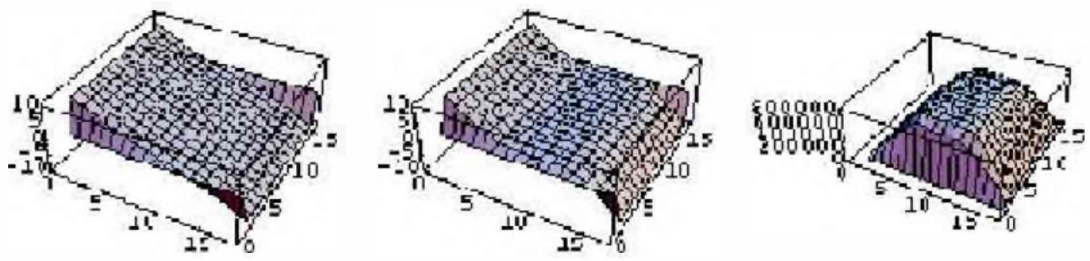

Figure 3: Solución: a) segun datos del listado adjunto b) la línea 19 ha sido cambiado por $kkk = 1, 10^1 0, 1$. c) con $sss = 0, 10^5, 0$

```
1   << Volumenesrym2D1.m
2   << treszonas.m
```

```
Timing[CV1 =
esquemaCicloV2D[T, s , KK, G, Ti, Tf, Cbs, Cbi, jacobi2D, 1, 3, 3,1/2];]
G1 = Graphics3D[ListPlot3D[CV1]];
<< treszonasK.m
Timing[CV2 =
esquemaCicloV2D[T, s , KK, G, Ti, Tf, Cbs, Cbi, jacobi2D, 1, 3, 3,1/2];]
G2 = Graphics3D[ListPlot3D[CV2]]
<< treszonasS.m
Timing[CV3 =
esquemaCicloV2D[T, s , KK, G, Ti, Tf, Cbs, Cbi, jacobi2D, 1, 3, 3,1/2];]
G3 = Graphics3D[ListPlot3D[CV3]]
Show[GraphicsArray[{G1, G2, G3}]]
```

CONCLUSION

Se ha desarrollado un paquete para el programa **Mathematica** para resolver la ecuación de difusión en tres dimensiones con las siguientes características:

- Trabaja sobre dominios rectangulares uni, bi y tridimensionales.

- La discretización se basa en volúmenes de control con redes estructuradas no necesariamente uniformes.

- Implementa el método iterativo de Jacobi y Jacobi ponderado.

- Implementa una variedad de algoritmos de multigrilla: el algoritmo de dos redes como caso especial de ciclo V entre dos únicos niveles de red, ciclos V y W que pueden abarcar varios niveles de redes y algoritmo de multigrilla completa.

- Permite la introducción de fuentes por zonas.

- Permite trabajar con zonas de distintos materiales caracterizados por su coeficiente de difusión.

- Permite resolver el caso estacionario y no estacionario.

- Permite obtener una representación gráfica de los resultados en forma inmediata.

-La organización del programa permite el desarrollo de un archivo para el usuario donde se definen todos los datos concernientes al problema en forma separada del paquete con las subrutinas centrales.

Se resolvieron numerosos problemas que garantizan la corrección de las subrutinas del paquete. El paquete se está utilizando exitosamente para resolver situaciones más complejas.

Los cálculos efectuados muestran que con el paquete desarrollado se pueden resolver problemas de conducción transitoria con el programa *Mathematica* de una forma muy conveniente, ya que el programa es suficientemente veloz y posee todas las capacidades para hacer un posprocesamiento gráfico inmediato de los resultados. Los resultados son suficientemente satisfactorios que alientan el desarrollo de paquete para cubrir problemas de tipo convección difusión y de mecánica de fluidos.

BIBLIOGRAFIA

Alejandro Hernandez. (2001). Simulación numérica de la distribución de temperatura en el suelo por debajo de un edificio. *Memorias del 8vo. Congreso Latinoamericano de Calor y Materia*, pp 242-249, Veracruz, México.

Suhas V. Patankar. (1980). Numerical Heat Transfer. Taylor and Francis.

A *Mathematica* PACKAGE FOR MULTIGRID ACCELERATED 1, 2 and 3D HEAT CONDUCTION EQUATION

ABSTRACT. A *Mathematica* package for the resolution of heat difussion equation is described. It solves non steady state with source problems in an 1, 2 and 3D domain. Uniform, zonal, non-uniform, and temperature dependent properties and sources are considered. Dirichlet, Neuman and Robinson boundary conditions are considered. Non uniform structured control volumes meshes are used. Resolution is done with multigrid acelerated iterative methods. All basic acelerations squemes such as V, W and μ cicles, were implememeted.