Método de Diferencias Finitas

Julio A. Medina
Universidad de San Carlos
Escuela de Ciencias Físicas y Matemáticas
Maestría en Física
julioantonio.medina@gmail.com

1. Ecuaciones diferenciales parciales elípticas

La ecuación diferencial parcial elíptica a considerar es la ecuación de Poisson

$$\nabla^2 u(x,y) \equiv \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = f(x,y) \tag{1}$$

en $R = \{(x,y) \mid a < x < b, c < y < d\}$, con $u(x,y) = g(x,y) \in S$, donde S denota al contorno de R. Si f y g son continuas en su dominio entonces hay una única solución a la ecuación.

1.1. Seleccionando un retículo

El método a utilizar es una adaptación bidimensional del método de diferencias finitas para problemas con fronteras lineales como se discute en [1]. El primer paso es escoger enteros n y m para definir el tamaño de los pasos(*steps*) $h = (b-a)/n$ y $k = (d-c)/m$ particionando de está manera el intervalo $[a,b]$ en n partes iguales de ancho h y el intervalo $[c,d]$ en m partes iguales con ancho k, formando un retículo o cuadricula como se puede ver en la figura

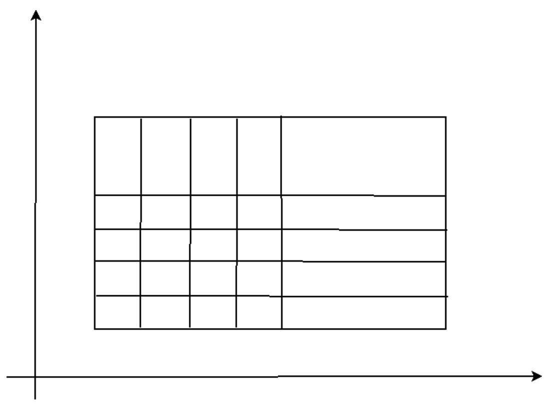

Figura 1: Cuadrícula de $n \times m$

Este retículo se construye formalmente al dibujar lineas verticales y horizontales sobre el dentro del rectángulo R en los puntos con coordenadas (x_i, y_j), donde

$$x_i = a + ih, \text{ para cada } i = 0, 1, 2, \ldots, n \text{ y } y = a + jk, \text{ para cada } j = 0, 1, 2, \ldots, m \tag{2}$$

Las rectas correspondientes a $x = x_i$ y $y = y_i$ son las lineas que forman la cuadricula y sus intersecciones son los puntos del retículo. Para cada punto interior del retículo se (x_i, y_j), para $i = 1, 2, \ldots, n-1$ y $j = 1, 2, \ldots, m-1$, se puede utilizar una serie de Taylor en la variable x alrededor del punto x_i para generar una fórmula de diferencia centrada

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_i) = \frac{u(x_{i+1}, y_j) - 2u(x_i, y_i) + u(x_{i-1}, y_j)}{h^2} - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) \tag{3}$$

donde $\xi_i \in (x_{i-1}, x_{i+1})$. De igual manera se puede encontrar la serie de Taylor en la variable y alrededor del punto y_j para hallar la diferencia centrada

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_i) = \frac{u(x_i, y_{j+1}) - 2u(x_i, y_i) + u(x_i, y_{j-1})}{k^2} - \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \eta_j) \tag{4}$$

donde $\eta_i \in (y_{i-1}, y_{i+1})$, sustituyendo estas fórmulas en las ecuación 1 permite expresar la ecuación de Poisson en los puntos (x_i, y_j) como

$$\begin{aligned}&\frac{u(x_{i+1}, y_j) - 2u(x_i, y_i) + u(x_{i-1}, y_j)}{h^2} + \frac{u(x_i, y_{j+1}) - 2u(x_i, y_i) + u(x_i, y_{j-1})}{k^2} \\ &= f(x_i, y_j) + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_i, y_j) + \frac{k^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, \eta_j)\end{aligned} \tag{5}$$

para cada $i = 1, 2, \ldots, n-1$ y $j = 1, 2, \ldots, m-1$. Las condiciones de contorno son

$$u(x_0, y_j) = g(x_0, y_j) \text{ y } u(x_n, y_j) = g(x_n, y_j), \text{ para cada } j = 0, 1, \ldots, m;$$

$$u(x_i, y_0) = g(x_i, y_0) \text{ y } u(x_i, y_m) = g(x_i, y_m), \text{ para cada } i = 1, 2, \ldots, n-1;$$

2. Método de Diferencias Finitas para ecuación de Poisson

2.1. Algoritmo de Generalización Factorización de Crout para matrices tridiagonales por bloques

Algorithm 1 Crout Generalization Algorithm for Tridiagonal Block Matrices

Require: $A \in \mathbb{R}^{N\times N}$, $K \in \mathbb{R}^{N}$, $n \in \mathbb{N}$
Ensure: $sol \in \mathbb{R}^{N}$
$N \leftarrow \lfloor \frac{\text{shape}(A)[1]}{n} \rfloor$
$W \leftarrow []$
$B_1 \leftarrow A_{1:n,1:n}$
$C_1 \leftarrow A_{1:n,n:n+n}$
$W.append(\text{inv}(B_1) \cdot C_1)$
for $i \leftarrow 2$ to $N-1$ **do**
  $B_i \leftarrow A_{(i-1)n:in,(i-1)n:in}$
  $A_i \leftarrow A_{(i-1)n:in,(i-2)n:(i-1)n}$
  $C_i \leftarrow A_{(i-1)n:in,in+1:n+(i-1)n}$
  $W.append(\text{inv}(B_i - A_i \cdot W_{i-2}) \cdot C_i)$
end for
$B_N \leftarrow A_{1:n,1:n}$
$K_1 \leftarrow K_{1:n}$
$G_1 \leftarrow \text{inv}(B_1) \cdot K_1$
$G \leftarrow []$
$G.append(G_1)$
for $i \leftarrow 2$ to N **do**
  $K_i \leftarrow K_{(i-1)n:in}$
  $B_i \leftarrow A_{(i-1)n:in,(i-1)n:in}$
  $A_i \leftarrow A_{(i-1)n:in,(i-2)n:(i-1)n}$
  $G.append(\text{inv}(B_i - A_i \cdot W_{i-2}) \cdot (K_i - A_i \cdot G_{i-2}))$
end for
$Z \leftarrow G[:]$
for $i \leftarrow N-2$ to 0 step -1 **do**
  $Z_i \leftarrow G_i - W_i \cdot Z_{i+1}$
end for
$sol \leftarrow \text{concatenate}(Z)$
return sol

2.2. Método iterativo SOR para resolver sistemas lineales

Algorithm 2 Iterative SOR method for solving linear systems

Require: A es una $n \times n$ matriz, b es un vector de incógnitas de dimensión n, $x^{(0)}$ es una solución propuesta inicial, ω es el parámetro de relajamiento, ϵ es la tolerancia, and max_{iter} es el número máximo de iteraciones.

Ensure: $x^{(k+1)}$ es una solución aproximada de $Ax = b$.

$k \leftarrow 0$
$x^{(k)} \leftarrow x^{(0)}$
while $k < max_{iter}$ and $||x^{(k+1)} - x^{(k)}||2 \geq \epsilon$ **do**
  $k \leftarrow k + 1$
  for $i = 1$ to n **do**
    $s \leftarrow \sum j = 1^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}$
    $x_i^{(k+1)} \leftarrow (1 - \omega) x_i^{(k)} + \frac{\omega}{a_{ii}} (b_i - s)$
  end for
end while
if $k = max_{iter}$ **then**
  print "Se ha alcanzado el número máximo de iteraciones"
end if
return $x^{(k+1)}$

Referencias

[1] Richard L. Burden, J. Douglas Faires *Numerical Analysis*, (Ninth Edition). Brooks/Cole, Cengage Learning. 978-0-538-73351-9