Modelación Numérica I

Iván Vega Gutiérrez

[1]Centro de Investigación en Matemáticas A.C.
Unidad Aguascalientes
E-mail: ivan.vega@cimat.mx

I. Ejercicio 6.7

El objetivo de este ejercicio es estudiar la resolución de la discretización de diferencias finitas de la ecuación de Laplace en dos dimensiones.

Dadas f y g funciones suaves, deseamos encontrar una solución $u(x, y)$ de la siguiente ecuación diferencial parcial:

$$-\Delta u(x, y) = f(x, y), \text{ para } (x, y) \in \Omega = [0, 1] \times [0, 1] \tag{1}$$

las condiciones de frontera son

$$u(x, y) = g(x, y), \text{ para } (x, y) \in \partial\Omega = \text{ frontera de } \Omega \tag{2}$$

donde $\Delta u = \partial^2 u/\partial x^2 + \partial^2 u/\partial y^2$ es el laplaciano de u. Así, el problema (1) es equivalente a hallar $u(x, y)$ que satisfaga:

$$-\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x, y) \text{ para } (x, y) \in \Omega \tag{3}$$

Con las respectivas condiciones de frontera.

A continuación hacemos una discretización del dominio Ω.

Para la variable x tomamos el tamaño de paso $h = \frac{1}{N+1}$, obteniendo N+2 puntos definidos de la siguiente manera:

$$x_i = ih, \qquad \text{con } i = 0, \cdots, N + 1.$$

De manera similar hacemos una partición con respecto a la variable y, tomamos como tamaño de paso $k = \frac{1}{M+1}$ y generamos $M + 2$ puntos definidos como:

$$y_j = jk, \qquad \text{con } j = 0, \cdots, M + 1.$$

En la figura (I) se muestra la discretización del dominio, cuando se toma el tamaño de paso para la variable x como $h = \frac{1}{10}$ y para la variable y se toma $k = \frac{1}{5}$. Los puntos rojos son los valores en la frontera (los cuales conocemos) y los puntos de color verde son las aproximaciones a la función u que queremos hallar.

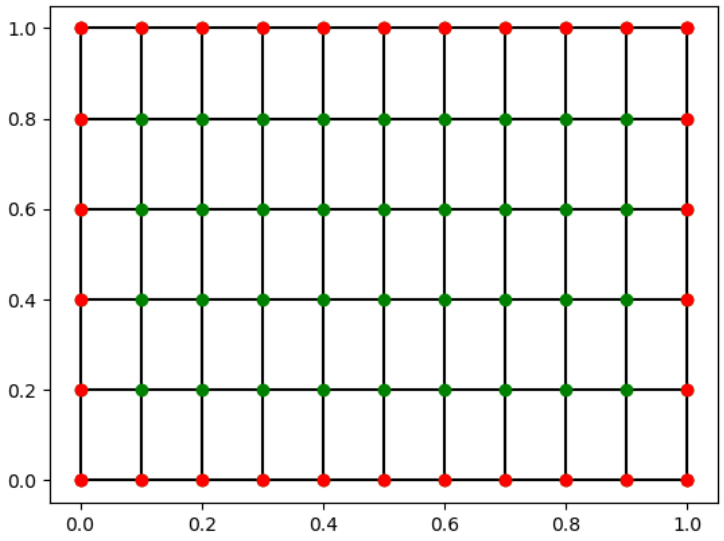

Figura 1. Malla del dominio [0,1]×[0,1]

I.1. Apximación mediante diferencias finitas

Combinando las expansiones de Taylor de $u(x_i - h, y_j)$ y $u(x_i + h, j_j)$, obtenemos que

$$\frac{\partial^2 u}{\partial x^2}(x_i, x_j) = \frac{u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j)}{h^2} + \mathcal{O}(h^2). \tag{4}$$

De manera análoga, obtenemos que

$$\frac{\partial^2 u}{\partial y^2}(x_i, x_j) = \frac{u(x_i, y_{j-1}) - 2u(x_i, y_j) + u(x_i, y_{j+1})}{h^2} + \mathcal{O}(k^2). \tag{5}$$

De (4), (5) y (3), obtenemos una solución aproximada a la ecuación de Laplace:

$$\frac{-u_{i-1,j} + 2u_{i,j} - u_{i+1,j}}{h^2} + \frac{-u_{i,j-1} + 2u_{i,j} - u_{i,j+1}}{k^2} = f_{i,j}, \tag{6}$$

donde $u_{i,j}$ denota la aproximación de la función u en el punto (x_i, y_j), es decir, $u_{i,j} = u(x_i, y_j)$, de igual forma $f_{i,j} = f(x_i, y_j)$. La fórmula (6) es llamada la discretización de 5 puntos del Laplaciano, porque acopla 5 valores de u en 5 puntos vecinos. La figura (2) representa gráficamente la relación de los cinco puntos.

I.2. Condiciones de frontera

Observemos que las condiciones de frontera nos da información sobre los puntos cuando $i = 0$, $i = N + 1$, $j = 0$ ó $j = M + 1$, por lo tanto estos puntos los conocemos y están dados por la función $g(x, y)$, luego, $u_{i,0} = g_{i,0}$, $u_{i,M+1} = g_{i,M+1}$, para $0 \leq i \leq N+1$ y $u_{0,j} = g_{0,j}$, $u_{N+1,j} = g_{N+1,j}$, para $0 \leq j \leq M+1$. Por lo tanto a partir de (6) tenemos las aproximaciones en $u_{1,1}, u_{N,1}, u_{1,M}$ y $u_{N,M}$:

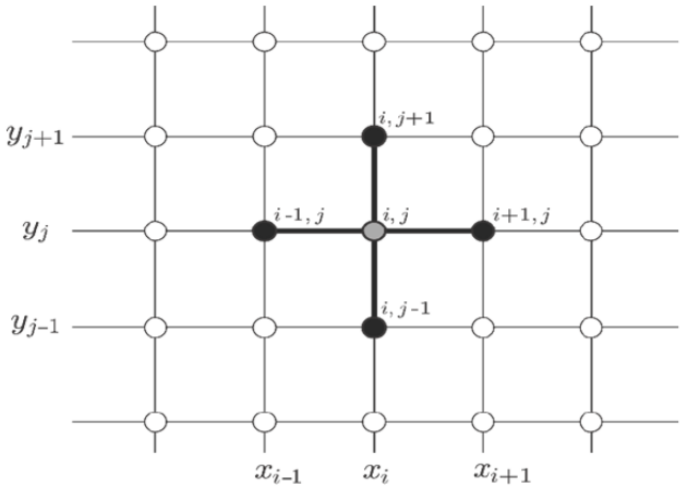

Figura 2. Discretización de cinco puntos

$$\frac{2u_{1,1} - u_{2,1}}{h^2} + \frac{2u_{1,1} - u_{1,2}}{k^2} = f_{1,1} + \frac{g_{1,0}}{k^2} + \frac{g_{0,1}}{h^2}.$$

$$\frac{-u_{N-1,1} + 2u_{N,1}}{h^2} + \frac{2u_{N,1} - u_{N,2}}{k^2} = f_{N,1} + \frac{g_{N,0}}{k^2} + \frac{g_{N+1,1}}{h^2}$$

$$\frac{2u_{1,M} - u_{2,M}}{h^2} + \frac{-u_{1,M-1} + 2u_{1,M}}{k^2} = f_{1,M} + \frac{g_{0,M}}{h^2} + \frac{u_{1,M+1}}{k^2}$$

$$\frac{-u_{N-1,M} + 2u_{N,M}}{h^2} + \frac{-u_{N,M-1} + 2u_{N,M}}{k^2} = f_{N,M} + \frac{g_{N+1,M}}{h^2} + \frac{g_{N,M+1}}{k^2}$$

I.3. Solución

Para simplificar los cálculos, supongamos que $h = k$, en consecuencia tenemos una partición del mismo tamaño para x y y. Además, por (6) se tiene que:

$$\frac{-u_{i-1,j} + 4u_{i,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} = f_{i,j}$$

O bien,

$$\frac{-u_{i,j-1} - u_{i-1,j} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1}}{h^2} = f_{i,j}$$

Así,

$$\frac{-u_{i,j-1}}{h^2} + \frac{-u_{i-1,j} + 4u_{i,j} - u_{i+1,j}}{h^2} + \frac{-u_{i,j+1}}{h^2} = f_{i,j} \tag{7}$$

El siguiente objetivo será llevar nuestro problema a un problema matricial, nuevamente para simplificar supongamos que las condiciones de frontera son cero ($g = 0$), y sean $\bar{u_j} = (u_{1,j}, u_{2,j}, \cdots, u_{N,j})^t$ el vector cuyas entradas son las n variables desconocidas en a columna j, y $\bar{f_j} = (f_{1,j}, f_{2,j}, \cdots, f_{N,j})^t$. Observemos que (7), la podemos expresar de forma matricial:

$$\frac{-\bar{u}_{j-1} + B\bar{u}_j - \bar{u}_{j+1}}{h^2} = \bar{f}_j \tag{8}$$

para $1 \leq j \leq M$. Donde $B = b_{i,j}$ es una matriz cuadrada de $N \times M$, definida de la siguiente manera.

$$b_{i,j} = \begin{cases} -1 & \text{si} \quad j = i-1 \\ 4 & \text{si} \quad j = i \\ -1 & \text{si} \quad j = i+1 \\ 0 & \text{cualquier otro caso} \end{cases}$$

Explícitamente:

$$B = \begin{pmatrix} 4 & -1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 4 & -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 4 & -1 & \cdots & 0 & 0 \\ 0 & 0 & -1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 4 & -1 \\ 0 & 0 & 0 & 0 & \cdots & -1 & 4 \end{pmatrix}$$

Por lo tanto para cada $j = \{1, 2, \cdots, M\}$, se tiene que:

$$-\begin{pmatrix} u_{1,j-1} \\ u_{2,j-1} \\ u_{3,j-1} \\ \vdots \\ u_{N-1,j-1} \\ u_{N,j-1} \end{pmatrix} + \begin{pmatrix} 4 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 4 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 4 \end{pmatrix} \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ u_{3,j} \\ \vdots \\ u_{N-1,j} \\ u_{N,j} \end{pmatrix} - \begin{pmatrix} u_{1,j+1} \\ u_{2,j+1} \\ u_{3,j+1} \\ \vdots \\ u_{N-1,j+1} \\ u_{N,j+1} \end{pmatrix} = h^2 \begin{pmatrix} f_{1,j} \\ f_{2,j} \\ f_{3,j} \\ \vdots \\ f_{N-1,j} \\ f_{N,j} \end{pmatrix}$$

Notemos que si $j = 1$, entonces $\bar{u}_{j-1} = 0$ y si $j = M$, entonces $\bar{u}_{j+1} = 0$

Por lo tanto:

$$\frac{1}{h^2} \begin{pmatrix} B & -I & 0 & \cdots & 0 & 0 \\ -I & B & -I & \cdots & 0 & 0 \\ 0 & -I & B & \cdots & 0 & 0 \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & B & -I \\ 0 & 0 & 0 & \cdots & -I & B \end{pmatrix} \begin{pmatrix} \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \\ \vdots \\ \bar{u}_{M-1} \\ \bar{u}_M \end{pmatrix} = \begin{pmatrix} \bar{f}_1 \\ \bar{f}_2 \\ \bar{f}_3 \\ \vdots \\ \bar{f}_{M-1} \\ \bar{f}_M \end{pmatrix}$$

Donde I denota la matriz identidad de $N \times M$ y B es la matriz definida anteriormente de tamaño $N \times M$, aunque por el supuesto $N = M$.

Por lo tanto el problema original es equivalente a hallar los valores de u tales que

$$A\bar{u} = \bar{f} \tag{9}$$

El código (1) muestra la función Laplacian2dD(n), la cual regresa la matríz A de $n^2 \times n^2$ que aparece en (9). Notemos que la matriz A es una matriz que tiene muchos ceros, a este tipo de matrices se les conoce como matirces ralas o dispersas. En la figura (3) se puede visualizar el patrón de disperción de la matriz A cuando $n = 4$. El patrón de dispersión de la matriz A representa visualmente los valores nulos de la matriz. Los cuadritos negros representan los elementos no nulos mientras que los cuadritos representan los elementos nulos de A.

Código 1. Función de la matriz laplaciana

```
import numpy as np

def Laplacian2dD(n):
    """
    Esta función calcula la matriz laplaciana de dos dimensiones.

    Argumentos:
    n (entero positivo): número de puntos internos de la malla

    Salida:
    A(matriz cuadra): Matriz dispersa de (n^2)x(n^2)
    """
    #Rellenamos la matriz con cuatros en la diagonal y lo demás con ceros
    A = 4*(np.identity(n*n))
    for i in range(n*n):
        for j in range(n*n):
            #Generamos los menos unos de la matriz B
            if (j==i-1 and (j+1)%n != 0) or (j==i+1 and j%n != 0):
                A[i][j] = -1
            #Generamos los menos unos restantes de la formula de 5 puntos
            elif (j == i + n  or j == i - n):
                A[i][j] = -1
    #Por último dividimos entre h^2 donde h=1/n+1
    A = np.dot(A,(n+1)**2)
    return A
```

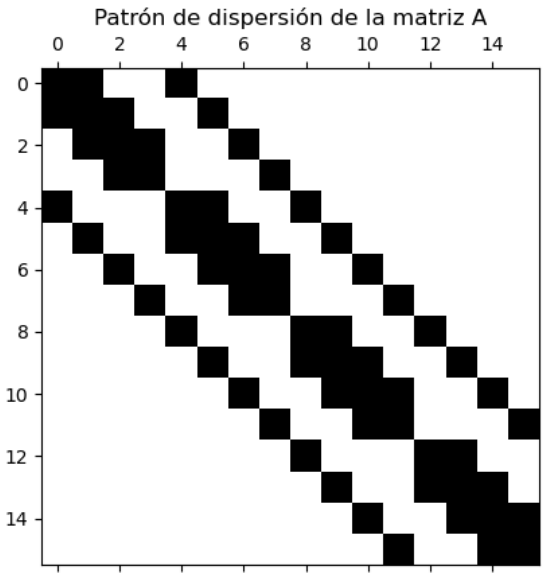

Figura 3. Patrón de dispersión de la matriz A cuando n = 4

Con lo anterior, ya tenemos un algoritmo que nos calcula la matriz A, por tanto, nos gustaría tener un algoritmo que nos calcule los valores del vector $\bar{f}$ que aparece en el lado derecho de (9).El código (2) nos permite obtener los valores de la función f en los puntos (x_i, y_j) de la discretización del dominio.

Código 2. Evaluación de los puntos de la malla

```
import numpy as np

def funcion(x,y):
    #Regresa la función que queramos evaluar en el punto (x,y):
    return x + y

def Laplacian2dDRHS(n,f=funcion):
    """
    Esta función regresa la función en los puntos x,y
    de la partición para resolver la ecuación de Laplace
    """
    #Generamos los puntos de la partición para x e y
    x, y, z =[], [], []
    for j in range(1, n+1):
        x.append(j/(n+1))
        y.append(j/(n+1))
    #Evaluamos la función en los puntos x[i] y y[j]
    for i in range(n):
        for j in range(n):
            z.append(f(x[i],y[j]))
    return z
```

Con todo lo anterior, tenemos las herramientas suficientes para poder resolver la ecuación de Laplace que teníamos inicialmente, todo esto traducido al problema matricial (9).

II. Eejercicio 6.8

Sea A la matriz definida por A=Laplacian2dD(5). El código (3) sirve para hallar la facorización LU de una matriz dada. Si utilizamos LUFacto(A), obtenemos la factorización LU, la figura (4) muestra visualmente los patrones de dispersión de las matrices A, L y U, el código (4) sirve para graficar la figura (4).

Notemos que tanto la matriz L y U tienen una forma muy peculiar. Si observamos el patrón de dispersión de la matriz A, su forma es muy cercana a una matriz triangular superior (o inferior), por ejemplo para $j = 0$ los elementos $(1, 0)$ y $(5, 0)$ son los elementos que "estorban", por lo tanto se procede a eliminarlos mediante operaciones elementales, sin embargo, al tratar de eliminar el punto $(1, 0)$, el elemento $(1, 5)$ deja de ser nulo, lo mismo pasa con elemento $(1, 5)$ y así es como se forma ese pequeño espacio en blanco que aparece en el patrón de dispersión de la matriz U en la parte superior izquierda. Siguiendo este procedimiento de eliminar los elementos no nulos debajo del pivote, obtenemos que:

$$E_n E_2 \cdots E_1 A = U$$

Donde E_i son matrices elementales. Algo análogo se tendría para la matriz L. Hemos de notar que la forma en la que se presenta la matriz A facilita en gran medida los cálculos de la factorización LU, ya que la mayoría de los elementos por debajo de la diagonal son ceros, por lo tanto la eliminación de los elementos por debajo del pivote requiere una menor cantidad de operaciones.

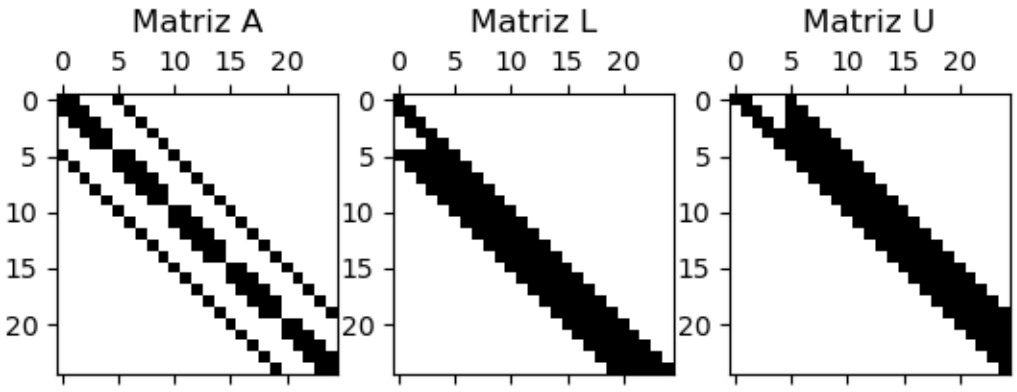

Figura 4. Patrón de dispersión con n=5

Código 3. Factorización LU

```
import numpy as np
def LUfacto(A):
    """
    Esta funcion devuelve las matrices L, U,, las cuales
    son la facotrizacion de A (A = LU). Donde L es una
    matriz triangular inferior y U es triangular superior.

    Argumentos:
    A(matriz): Matriz que se desea factorizar

    Excepciones:
    Division entre cero si U[j][j] =0
    """
    # Hallamaos la dimension de la matriz
    n = A.shape[0]
    #Creamos la matriz U compuesta por ceros
    U = np.zeros((n,n))
    #La matriz L tendra unos en su diagonal
    L = np.identity(n)
    #La primera fila de U será igual a la primera fila de A
    U[0] = A[0]
    #Verificamos que no intentemos una división entre cero
    if A[0][0] == 0:
        print("Error de ejecución: División entre cero")
        return
    else:
        #Hallamos la primera columna de L
        L[:,0] = A[:,0]/A[0][0]
```

```
        #Este ciclo recorrerá las filas i
        for i in range(1,n):
            #El siguiente ciclo recorrerá las columasn j
            for j in range(1,n):
                #Si la siguiente condición se cumple
                #entonces llenaremos la matriz L
                if i>j:
                    suma = 0
                    for k in range(j):
                        suma += L[i][k]*U[k][j]
                    if U[j][j] == 0:
                        print("Error de ejecución: División entre cero")
                        return
                    else:
                        L[i][j] = (A[i][j] - suma)/U[j][j]
                #Si i >= j entonces se llenará la matriz U
                else:
                    suma = 0
                    for k in range(i):
                        suma += L[i][k]*U[k][j]
                    U[i][j] = A[i][j] - suma
    return (L,U)
```

Código 4. Gráficas de los patrones de dispersión

```
import matplotlib.pyplot as plt

#HALLAMOS LAS MATRICES A, L Y U.
A = Laplacian2dD(5)
L,U = LUfacto(A)

#MATRIZ A
plt.subplot(1,3,1)
plt.spy(A)
plt.title('Matriz A')
#MATRIZ L
plt.subplot(1,3,2)
plt.spy(L)
plt.title('Matriz L')
#MATRIZ U
plt.subplot(1,3,3)
plt.spy(U)
plt.title('Matriz U')

plt.show()
```