TAREA 14 - METODOS NUMERICOS

Guillermo Segura Gómez
Centro de Investigación en Matemáticas
Métodos Numéricos
01 de diciembre de 2023

1 Introducción

Podemos clasificar la solución para ecuaciones diferenciales según las condiciones o restricciones sobre las que son resueltas. Por ejemplo, en la tarea pasada, revisamos problemas de solución de ecuaciones diferenciales ordinarias con una condición inicial. Es decir se especifica el valor de la función de la ecuación diferencial en el inicio del dominio sobre el que se quiere resolver. Para este tipo de problemas, se utilizan métodos como el método de Euler, o el método de RK4, que nos permitían encontrar la solución de las ecuaciones [1].

Existen otros problemas de ecuaciones diferenciales, en los que cada ecuación esta sujeta a dos o mas restricciones, es decir, necesitamos que la solución de la ecuación pase por ciertos puntos específicos. Estos problemas se conocen como, problemas con valores a la frontera, en estos problemas los *valores a la frontera* restringen la solución de la ecuación, y por lo general estos puntos se colocan en la frontera del dominio sobre el cual se soluciona la ecuación diferencial [1]. Este tipo de problemas son mucho mas complejos de resolver que los de valor inicial, y conciernen a problemas en la física y la ingeniería como la solución de la mecánica de la deflexión de una viga. En este trabajo vamos abordar algunos métodos de solución de ecuaciones diferenciales con problemas a la frontera.

2 Problema 1

Investigar en qué consiste el método de disparo para ecuaciones diferenciales ordinarias (shooting method). Describe el método y presenta un ejemplo. No es necesario programarlo, pero si presentar la idea adecuada.

2.1 Solución

Un problema de valores a la frontera de ecuaciones diferenciales de segundo orden tiene la siguiente forma [1]:

$$y'' = f(x, y, y'), \qquad \text{para} \quad a \leq x \leq b \tag{1}$$

junto con las condiciones de frontera

$$y(a) = \alpha \qquad y(b) = \beta \tag{2}$$

El método del disparo es un enfoque para resolver ecuaciones diferenciales con problemas a la frontera. Podemos distinguir entre el disparo lineal y el disparo no lineal. Primero, nos enfocamos en el problema lineal.

El método del disparo lineal se puede aplicar siempre que tengamos un sistema lineal con solución única [1]. Un sistema de ecuaciones diferenciales de segundo orden es lineal si

$$y'' = f(x, y, y'), \tag{3}$$

donde las funciones $p(x)$, $q(x)$ y $r(x)$ existen y cumplen

$$f(x, y, y') = y'p(x) + yq(x) + r(x) \tag{4}$$

[1] Ver libro Burden p 506 para revisar unicidad de las soluciones de las ecuaciones

Para aproximar la solución de este problema lineal utilizando el método del disparo, primero dividimos el problema en dos problemas de valor inicial. Este enfoque simplifica el método, permitiéndonos emplear métodos numéricos más directos diseñados para problemas de valor inicial como el método de Euler o Runge-Kutta. Estos problemas de valor inicial son:

$$\begin{aligned} y_1'' &= p(x)y_1' + q(x)y_1 + r(x), \quad \text{con} \quad y_1(a) = \alpha, \quad y_1'(a) = 0, \\ y_2'' &= p(x)y_2' + q(x)y_2, \quad \text{con} \quad y_2(a) = 0, \quad y_2'(a) = 1. \end{aligned} \tag{5}$$

Estos problemas de valor inicial reflejan dos escenarios diferentes. El primero se ajusta a la condición de frontera en a con una derivada inicial cero, mientras que el segundo establece el valor en a en cero con una derivada inicial de uno. Utilizamos estas dos soluciones, $y_1(x)$ y $y_2(x)$, para construir la solución final que cumple con las condiciones de frontera en a y b. Esto implica hacer una conjetura inicial para las condiciones iniciales.

Una vez resueltos estos problemas de valor inicial, encontramos la solución al problema original de valor en la frontera como una combinación lineal de $y_1(x)$ y $y_2(x)$:

$$y(x) = y_1(x) + \frac{\beta - y_1(b)}{y_2(b)} y_2(x). \tag{6}$$

Este método asegura que la solución satisface las condiciones de frontera en ambos extremos del intervalo. Ajustamos la solución del primer problema de valor inicial con un múltiplo de la solución del segundo problema para cumplir con la condición de frontera en b. Continuamos iterando y ajustando los problemas de valor inicial para que coincidan con las condiciones de frontera de nuestro problema original.

Para facilitar la resolución numérica, transformamos cada problema de valor inicial en un sistema de dos ecuaciones diferenciales de primer orden. Para el primer problema, definimos $z_1(x) = y_1(x)$ y $z_2(x) = y_1'(x)$, resultando en el siguiente sistema:

$$\begin{aligned} z_1'(x) &= z_2(x) \\ z_2'(x) &= p(x)z_2(x) + q(x)z_1(x) + r(x) \end{aligned} \tag{7}$$

para $a \leq x \leq b$ con $z_1(a) = \alpha$ y $z_2(a) = 0$. De manera similar, transformamos el segundo problema con $z_3(x) = y_2(x)$ y $z_4(x) = y_2'(x)$:

$$\begin{aligned} z_3'(x) &= z_4(x) \\ z_4'(x) &= p(x)z_4(x) + q(x)z_3(x) \end{aligned} \tag{8}$$

para $a \leq x \leq b$ con $z_3(a) = 0$ y $z_4(a) = 1$. Las aproximaciones calculadas en el algoritmo son:

$$u_{1,i} \approx z_1(x_i) = y_1(x_i), \quad u_{2,i} \approx z_2(x_i) = y_1'(x_i) \tag{9}$$

y

$$v_{1,i} \approx z_3(x_i) = y_2(x_i), \quad v_{2,i} \approx z_4(x_i) = y_2'(x_i). \tag{10}$$

Las aproximaciones finales son:

$$w_{1,i} = u_{1,i} + \frac{\beta - u_{1,N}}{v_{1,N}} v_{1,i} \approx y_1(x_i) \tag{11}$$

y

$$w_{2,i} = u_{2,i} + \frac{\beta - u_{1,N}}{v_{1,N}} v_{2,i} \approx y_1'(x_i) \tag{12}$$

Este algoritmo tiene la ventaja adicional de proporcionar aproximaciones tanto para la solución del problema de valor en la frontera como para su derivada. Su aplicación no se limita a los problemas que cumplen con las hipótesis del corolario de unicidad de la solución; es útil para muchos problemas que no satisfacen estas hipótesis. [1]

Por otro lado tenemos al otro tipo de disparo, el disparo no lineal. El método del disparo no lineal aborda ecuaciones diferenciales en las cuales la relación entre y, y', y y'' no es lineal. En estos casos, la función $f(x, y, y')$ en la ecuación diferencial $y'' = f(x, y, y')$ presenta términos no lineales en y o y', o ambas. La aplicación del método del disparo en estos escenarios implica, típicamente, el uso de técnicas iterativas para ajustar las conjeturas de las condiciones iniciales, dado que la solución exacta no puede ser expresada directamente como una combinación lineal de soluciones. Esto requiere un proceso iterativo, como el método de Newton-Raphson, para modificar las condiciones iniciales estimadas hasta que la solución calculada satisfaga adecuadamente las condiciones de frontera. [1]

2.2 Ejemplo del Método de Disparo lineal

Consideremos la siguiente ecuación diferencial:

$$y'' = -2y' - y, \qquad \text{para} \quad 0 \leq x \leq 1 \tag{13}$$

con las condiciones de frontera:

$$y(0) = 1 \qquad y(1) = e^{-2}. \tag{14}$$

Dividimos este problema en dos problemas de valor inicial siguiendo el método del disparo. Primero resolvemos:

$$y_1'' = -2y_1' - y_1, \qquad \text{con} \quad y_1(0) = 1, \quad y_1'(0) = 0, \tag{15}$$

y luego

$$y_2'' = -2y_2' - y_2, \qquad \text{con} \quad y_2(0) = 0, \quad y_2'(0) = 1. \tag{16}$$

Una vez que tengamos las soluciones $y_1(x)$ y $y_2(x)$, las utilizamos para construir la solución del problema de valor en la frontera:

$$y(x) = y_1(x) + \frac{e^{-2} - y_1(1)}{y_2(1)} y_2(x). \tag{17}$$

3 Problema 2

Programar el θ-método y resolver el problema de la ecuación de calor temporal dada por,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad x \in [0,1], \quad t \in [0, 0.4]$$

Valores Iniciales:

$$u(x,0) = u_0(x) = 4x - 4x^2, \quad x \in [0,1]$$

Condiciones de Frontera de Dirichlet Nulas:

$$U(0,t) = 0, \quad t \in [0, 0.4]$$
$$U(1,t) = 0, \quad t \in [0, 0.4]$$

Elige alguno de los método programados durante el curso para resolver sistema de ecuaciones lineales, si utilizas un método iterativo se sugiere utilizar una tolerancia de $\varepsilon = 1 \times 10^{-5}$.

3.1 Pseudocódigo

Para implementar el programa se empleó el siguiente pseudocódigo para la implementación del método de theta, y la solución de la ecuación de calor en una barra. Se utilizaron múltiples funciones auxiliares implementadas en la librería **matrix.h**. El método incluye la solución de un sistema matricial Ax = B, el cual se resuelve utilizando la factorización LU con el método de Crout.

```
Funcion thetaMethod(I, n, u0, f, bc, nu, theta)
    nx := n[0]
    h := (I[2] - I[1]) / nx
    x := crear un arreglo desde I[1] hasta I[2] con paso h
    t := I[3]
    uold := evaluar u0 para todos los puntos en x

    nt := n[1]
    k := (I[4] - I[3]) / nt
    e := crear un arreglo de unos con tamano nx+1

    // Construir las matrices K y B
    K := crear una matriz tridiagonal de tamano (nx+1) x (nx+1)
    B := crear una matriz tridiagonal de tamano (nx+1) x (nx+1)
    para cada punto i en el espacio hacer
        calcular y establecer los valores en las diagonales de K y B

    // Aplicar condiciones de frontera
    K[1][1] := 1; K[1][2] := 0; B[1][1] := 0; B[1][2] := 0
    K[nx+1][nx+1] := 1; K[nx+1][nx] := 0; B[nx+1][nx+1] := 0; B[nx+1][nx] :=
    0

    // Factorizacion LU de K
    [L, U] := factorizacionLU(K)

    // Bucle temporal para resolver la ecuacion
    para cada paso de tiempo desde I[3] hasta I[4] con paso k hacer
```

```
        t := tiempo actual
        fnew := evaluar f en cada punto de x usando t
        fold := multiplicar cada elemento de fnew por h
        fold := agregar condiciones de frontera bc a fold

        // Calcular el lado derecho del sistema
        b := theta * fnew + (1 - theta) * fold + multiplicar B por uold

        // Resolver el sistema lineal
        y := resolver L \ b
        u := resolver U \ y

        // Actualizar uold con u para el siguiente paso de tiempo
        uold := u

    devolver u, x
Fin de la Funcion
```

Listing 1: Método de Crank-Nicolson (theta)

3.2 Ejecución

La ejecución del código se emplea utilizando un archivo makefile para enlazar las librerías.

```
guillermo_sego@MacBook-Air-de-Guillermo Tarea14 % make
gcc -g -o build/Debug/ThetaMethod.o -c ThetaMethod.c
gcc -g -o build/ThetaMethod build/Debug/matrix.o build/Debug/ThetaMethod.o
guillermo_sego@MacBook-Air-de-Guillermo Tarea14 % ./build/ThetaMethod
```

Listing 2: Ejecución método de theta

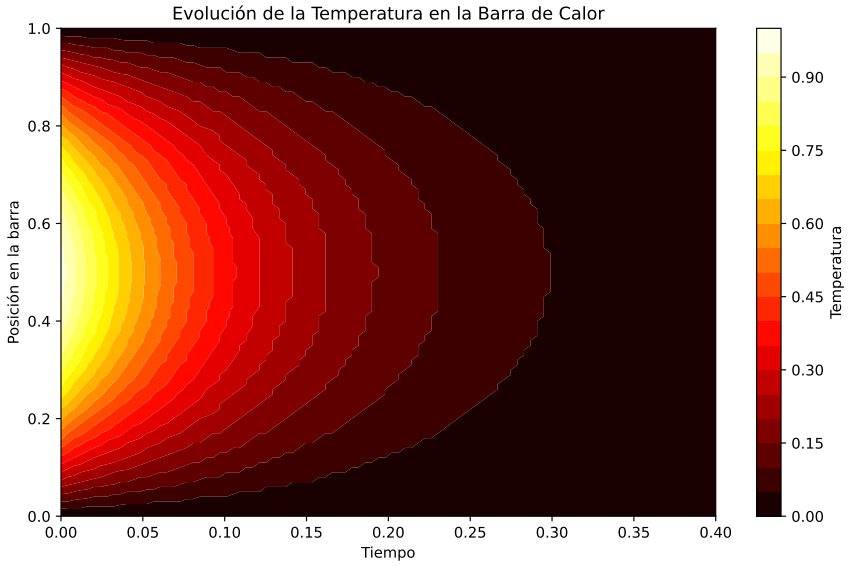


Figure 1: Temperatura de la barra en función del tiempo.

Se resuelve la ecuación de calor en cada paso del tiempo. Los pasos están determinados por el intervalo y la precisión que se desea, a mas precisión, mas costo computacional. Con

cada iteración encontramos los valores de la temperatura U, en ese momento del tiempo. Cada iteración se guarda en un archivo llamado *output.txt*. Posteriormente se graficó la solución utilizando python. La gráfica que se obtuvo se observa en la figura 1, es la temperatura de la barra unidimensional en función del tiempo. Se observa como la temperatura comienza inicialmente en el centro de la barra y con el paso del tiempo se distribuye uniformemente en la barra hasta alcanzar el equilibrio térmico.

4 Problema 3

Elige una función $u(x, y)$, cualquiera de 2 variables, y realiza lo siguiente: a) Calcula el laplaciano de $u(x, y)$, y define $f(x, y) = -\nabla^2 u(x, y)$. b) Evalúa la función $u(x, y)$ sobre el cuadro unitario $[0, 1] \times [0, 1]$, definela como $u_D(x, y)$. c) Resuelve mediante el método de elemento finito el problema,

$$-\nabla^2 u(x, y) = f(x, y), \in \Omega = [0, 1] \times [0, 1]$$
$$u(x, y) = u_D(x, y) \in \partial\Omega$$

NOTA: Para resolver este ejercicio, crea una copia en tu Drive y modifica el código del ejemplo 1 (Ec. de Poisson) en FEniCS que se presentó en Google-Colab. Explica de manera clara los pasos a)-b) para obtener tu problema individual. d) Compara los errores en la norma L_2 y L_∞ con respecto a la función solución $u(x, y)$ que pensaste originalmente.

4.1 Solución

4.1.1 Cálculo del Laplaciano de u y definición de funciones

La función elegida es $u(x, y) = \sin(\pi x) \sin(\pi y)$. El Laplaciano de $u(x, y)$, denotado como $\nabla^2 u(x, y)$, se calcula como la suma de las segundas derivadas parciales con respecto a x e y. Para esta función, el Laplaciano es:

$$\nabla^2 u(x, y) = -\pi^2 \sin(\pi x) \sin(\pi y) - \pi^2 \sin(\pi x) \sin(\pi y) = -2\pi^2 \sin(\pi x) \sin(\pi y) \tag{18}$$

Por lo tanto, la función fuente $f(x, y)$ se define como el negativo del Laplaciano:

$$f(x, y) = 2\pi^2 \sin(\pi x) \sin(\pi y) \tag{19}$$

4.1.2 Evaluación de $u(x, y)$ sobre el Cuadro Unitario y Definición de $u_D(x, y)$

La función $u(x, y)$ se ha evaluado sobre el cuadro unitario $[0, 1] \times [0, 1]$ y se ha utilizado como condición de frontera $u_D(x, y)$. Esto significa que en todos los límites del dominio $[0, 1] \times [0, 1]$, la solución debe coincidir con $u(x, y) = \sin(\pi x) \sin(\pi y)$.

4.1.3 Resolución del Problema con el Método de Elementos Finitos

Utilizando FEniCS, se resuelve el problema variacional asociado a la ecuación de Poisson con la condición de frontera Dirichlet y la función fuente definida en el punto a). La simulación numérica se realizó en una malla de 20×20 para el dominio $[0, 1] \times [0, 1]$.

4.1.4 Comparación de Errores en las Normas L_2 y L_∞

Los errores calculados en comparación con la función solución original $u(x, y)$ son los siguientes:

- Error en la norma L_2: 0.9969254189093316
- Error máximo (norma L_∞): 1.997943205281563

Estos errores son relativamente altos, lo que podría sugerir que la solución numérica obtenida difiere significativamente de la solución analítica.

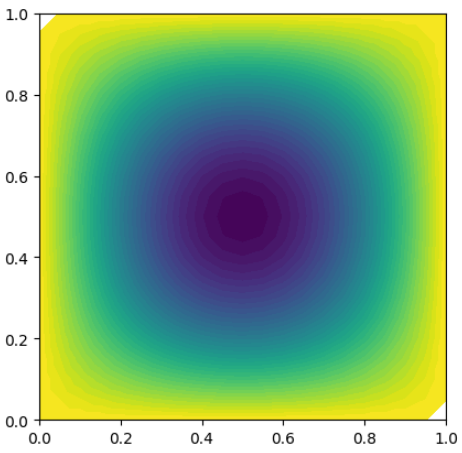
La modificación del código original se encuentra a continuación

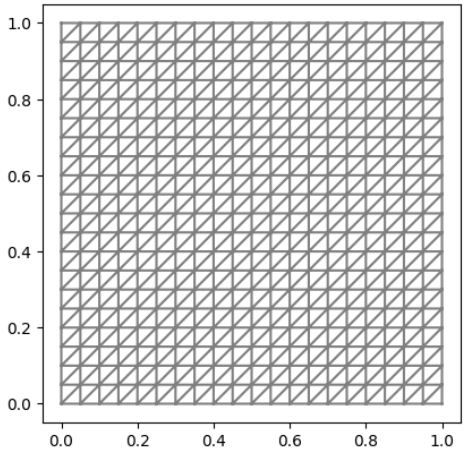
```
from dolfin import *
from mshr import *
import numpy as np
import matplotlib.pyplot as plt

# Crear malla y definir espacio de funciones
mesh = UnitSquareMesh(20, 20)
V = FunctionSpace(mesh, 'P', 1)

# Definir condición de frontera (u_D es tu u(x, y))
u_D = Expression('sin(pi*x[0]) * sin(pi*x[1])', degree=2)

# Definir función fuente f
f = Expression('-2*pow(pi, 2) * sin(pi*x[0]) * sin(pi*x[1])', degree=2)

def boundary(x, on_boundary):
    return on_boundary

bc = DirichletBC(V, u_D, boundary)

# Definir problema variacional
u = TrialFunction(V)
v = TestFunction(V)
a = dot(grad(u), grad(v))*dx
L = f*v*dx

# Calcular solución
u = Function(V)
solve(a == L, u, bc)

# Graficar solución
u.rename('u', 'solution')
plot(u)
plt.show()

plot(mesh)

# Guardar solución en formato VTK
vtkfile = File('poisson_solution.pvd')
vtkfile << u

# Calcular error en norma L2
error_L2 = errornorm(u_D, u, 'L2')
```

```
# Calcular error máximo en vértices
vertex_values_u_D = u_D.compute_vertex_values(mesh)
vertex_values_u = u.compute_vertex_values(mesh)
error_max = np.max(np.abs(vertex_values_u_D - vertex_values_u))

# Imprimir errores
print('error_L2  =', error_L2)
print('error_max =', error_max)

# Mantener gráfico visible
plt.show()
```

Listing 3: Código para calcular el método de elementos finitos

Las gráficas obtenidas del método fueron las siguientes





5 Conclusión

En este trabajo, hemos explorado varios métodos numéricos avanzados para la solución de ecuaciones diferenciales con problemas en la frontera. Los métodos del disparo, el método de theta y una introducción al método de elementos finitos representan herramientas fundamentales en el análisis numérico de ecuaciones diferenciales, de manera particular para los problemas de ecuaciones diferenciales de valor a la frontera que fueron los estudiados en esta tarea.

El método de theta es un método poderoso para la solución de ecuaciones diferenciales parciales con problemas a la frontera como la ecuación de calor, que fue la que solucionamos en este ejercicio. Me sorprendió lo rápido que se generaron los puntos de la solución, y también al graficar la evolución de la temperatura, se convergía al equilibrio térmico, lo cual tiene completo sentido físico.

Cada uno de estos métodos contribuye a nuestro conjunto de herramientas para abordar una amplia gama de problemas prácticos. A medida que avanzamos en nuestro estudio de las ecuaciones diferenciales, queda claro que la elección del método adecuado depende tanto de la naturaleza del problema como de las características específicas de las soluciones que

buscamos. Este conocimiento no solo nos ayuda en nuestra comprensión teórica, sino que también mejora nuestra capacidad para aplicar estas técnicas a problemas del mundo real.

References

[1] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical analysis*. Cengage learning, 2015.