EDO con condiciones de frontera

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

Facultad de Ciencias - UNAM

5 de abril de 2018

1. Método de diferencias finitas

1. Método de diferencias finitas
1.1 Definición
1.2 Diferenciación
1.3 Sistemas lineales con matrices en banda
1.4 Propuesta de solución
1.5 Ejercicio EDO-2 con CDF

ED con condiciones de frontera

Entre los métodos numéricos para resolver problemas con CDF, se sabe que los métodos de diferencias finitas tienen las mejores propiedades de estabilidad.

Es cierto, sin embargo, que la estabilidad mejorada se produce, en general, a expensas de un mayor esfuerzo computacional para una precisión comparable.

Método de diferencias finitas

Esencialmente, los métodos de diferencias finitas implican aproximar las derivadas en la EDO y las CDF, mediante esquemas de diferencias finitas definidos en un conjunto discreto de puntos de una malla.

Método de diferencias finitas

En cualquier caso, se supone que los esquemas de discretización particulares utilizados aseguran un orden homogéneo de los errores de truncamiento.

El sistema resultante de ecuaciones algebraicas puede resolverse mediante un método, o en el caso ideal, mediante un método adaptado a la forma particular del sistema.

Forma general de la EDO con CDF

Consideremos el problema lineal con CDF

$$y'' = p(x)\, y' + q(x)\, y + r(x) \tag{1}$$

y con las condiciones

$$\begin{aligned} \alpha_1\, y(x_a) + \beta_1\, y'(x_a) &= 1 \\ \alpha_2\, y(x_b) + \beta_2\, y'(x_b) &= 1 \end{aligned} \tag{2}$$

donde suponemos que $p(x)$, $q(x)$ y $r(x)$ son funciones continuas en el intervalo $[x_a, x_b]$

Condiciones de frontera

Las condiciones de frontera están definidas por los cuatro coeficientes α_1, β_1, α_2 y β_2.

Por lo que debe aplicarse la condición natural adicional: que ambos coeficientes α y β para una frontera dada, no pueden ser iguales a 0, es decir:

$$|\alpha_i| + |\beta_i| \neq 0, \quad i = 1, 2$$

Tipos de CDF

En particular:

1. Para $\beta_i = 0$, se tiene una *condición de frontera de tipo Dirichlet*, la cual define el valor de la solución en la frontera.

Tipos de CDF

En particular:

1. Para $\beta_i = 0$, se tiene una *condición de frontera de tipo Dirichlet*, la cual define el valor de la solución en la frontera.
2. Para $\alpha_i = 0$, se tiene una *condición de frontera de tipo Neumann*, que fija la solución de la derivada en la frontera.

Tipos de CDF

En particular:

1. Para $\beta_i = 0$, se tiene una *condición de frontera de tipo Dirichlet*, la cual define el valor de la solución en la frontera.
2. Para $\alpha_i = 0$, se tiene una *condición de frontera de tipo Neumann*, que fija la solución de la derivada en la frontera.

Tipos de CDF

En particular:

1. Para $\beta_i = 0$, se tiene una *condición de frontera de tipo Dirichlet*, la cual define el valor de la solución en la frontera.
2. Para $\alpha_i = 0$, se tiene una *condición de frontera de tipo Neumann*, que fija la solución de la derivada en la frontera.

Los tipos de condición son distintos en las dos fronteras, dependiendo del problema en cuestión.

Definición de la malla

Consideremos una partición en el dominio $[x_a, x_b]$, definido por una malla de puntos equidistantes:

$$x_m = x_a + (m - 1)\, h, \qquad m = 1, 2, \ldots, M \tag{3}$$

con una separación de igual tamaño

$$h = \frac{x_b - x_a}{M - 1} \tag{4}$$

Diferenciación

Haciendo la notación $y_m \equiv y(x_m)$, recurrimos a una aproximación discreta de la primera y segunda derivada: y'_m y y''_m, a partir del desarrollo de la serie de Taylor en $x_{m+1} = x_m + h$ y $x_{m-1} = x_m - h$:

Diferenciación

$$y_{m+1} = y_m + h\, y'_m + \frac{h^2}{2!}\, y''_m + \frac{h^3}{3!}\, y^3_m + O(h^4) \quad (5)$$

$$y_{m-1} = y_m - h\, y'_m + \frac{h^2}{2!}\, y''_m - \frac{h^3}{3!}\, y^3_m + O(h^4) \quad (6)$$

Primera derivada

Para la primera derivada y'_m, podemos obtenerla mediante aproximaciones directas, al truncar hasta $(h^2/2)\, y''$ y cancelar todos los términos de orden mayor:

$$
\begin{aligned}
y'_m &= \frac{y_m - y_{m-1}}{h} + O(h) \\
y'_m &= \frac{y_{m+1} - y_m}{h} + O(h)
\end{aligned} \tag{7}
$$

que al dividirse entre h, queda solamente $O(h)$.

Fórmulas de diferencias

La primera expresión se le denomia *fórmula de diferencia hacia atrás*, ya que conecta x_m con el punto previo x_{m-1}.

La segunda expresión es la *fórmula de diferencia hacia adelante*, ya que conecta x_m con el siguiente punto x_{m+1}

Aproximaciones de orden superior

Para una aproximación de orden superior de y_m', se obtiene de la diferencia de la expansión de la serie de Taylor (5) y (6), con la cancelación de los términos de tercer orden $(h^3/6)\, y_m^{(3)}$ y la cancelación exacta de los términos de segundo orden $(h^2/2)\, y_m''$:

$$y_m' = \frac{y_{m+1} - y_{m-1}}{2h} + O(h^2) \tag{8}$$

Diferencias centrales

La expresión anterior se le llama *fórmula de diferencias centrales* y ocupa puntos simétricos en la malla alrededor de x_m, donde se desea evaluar la derivada.

Al dividir por h, se reduce el orden del esquema en 1.

Segunda derivada de y_m

Tomando la suma del desarrollo de Taylor (5) y (6), se cancelan los términos de primer y tercer orden, $h\, y'_m$ y $(h^3/6)y_m^{(3)}$, y se obtiene una expresión de diferencias centrales para la segunda derivada:

$$y''_m = \frac{y_{m+1} - 2\, y_m + y_{m-1}}{h^2} + O(h^2) \tag{9}$$

que es de orden $O(h^2)$ dada la división entre h^2.

Sistemas lineales con matrices en banda

Sin embargo, la aplicación de esquemas de diferencias centrales para problemas discretizados de dos puntos, conduce a un sistema lineal con matrices de bandas simétricas, lo que impacta positivamente en la estabilidad de los métodos de solución.

Al usar las fórmulas de diferencias centrales $O(h^2)$ (8) y (9) para aproximar las condiciones de frontera (2), obtenemos el siguiente sistema lineal:

Sistemas lineales con matrices en banda

$$\begin{cases} \dfrac{y_{m+1} - 2\, y_m + y_{m-1}}{h^2} = p_m\, \dfrac{y_{m+1} - y_{m-1}}{2\, h^2} + q_m\, y_m + r_m \\ m = 2, 3, \ldots, M-1 \\ \alpha_1\, y_1 + \beta_1\, \dfrac{y_2 - y_1}{h} = 1 \\ \alpha_2\, y_M + \beta_2\, \dfrac{y_M - y_{M-1}}{h} = 1 \end{cases}$$

Organizado los términos

Reagrupando los valores desconocidos de la solución para y_m, el sistema discreto lineal para un problema con CDF, toma la forma:

Organizado los términos

Reagrupando los valores desconocidos de la solución para y_m, el sistema discreto lineal para un problema con CDF, toma la forma:

$$\begin{cases} (h\,\alpha - \beta)y_1 + \beta_1\,y_2 = h \\ -(2 + h\,p_m)\,y_{m-1} + (4 + 2\,h^2\,q_m)\,y_m + \\ \qquad\qquad\qquad - (2 - h\,p_m)\,y_{m+1} = -2\,h^2\,r_m \\ m = 2, 3, \ldots, M-1 \\ -\beta_2\,y_{M-1} + (h\,\alpha_2 + \beta_2)\,y_M = h \end{cases} \tag{10}$$

Forma matricial

El sistema anterior puede representarse como una matriz triadiagonal:

$$
\begin{bmatrix}
b_1 & c_1 & & & & & \\
a_2 & b_2 & c_2 & & & 0 & \\
& \ddots & \ddots & \ddots & & & \\
& & a_m & b_m & c_m & & \\
& & & \ddots & \ddots & \ddots & \\
& 0 & & & a_{M-1} & b_{M-1} & c_{M-1}
\end{bmatrix}
\begin{bmatrix}
y_1 \\ y_2 \\ \vdots \\ y_m \\ \vdots \\ y_{M-1}
\end{bmatrix}
=
\begin{bmatrix}
d_1 \\ d_2 \\ \vdots \\ d_M \\ \vdots \\ d_{M-1}
\end{bmatrix}
\tag{11}
$$

Elementos no nulos

Donde los elementos no nulos tienen las expresiones:

$$b_1 = h\,\alpha_1 - \beta_1, \quad c_1 = \beta_1, \quad d_1 = h$$

$$\begin{cases} a_m = -(2 + h\,p_m), & m = 2, 3, \ldots, M-1 \\ b_m = 4 + 2\,h^2\,q_m \\ c_m = -(2 - h\,p_m) \\ d_m = -2\,h^2\,r_m \end{cases}$$

$$a_M = -\beta_2, \quad b_M = h\,\alpha_2 + \beta_2, \quad d_M = h$$

Propuesta de solución

A continuación se presenta una solución para el problema de la EDO-2 con CDF mediante el uso de funciones que resuelven por etapas:

1. Construir el sistema tridiagonal.

Propuesta de solución

A continuación se presenta una solución para el problema de la EDO-2 con CDF mediante el uso de funciones que resuelven por etapas:

1. Construir el sistema tridiagonal.
2. Resolver el sistema tridiagonal.

Propuesta de solución

A continuación se presenta una solución para el problema de la EDO-2 con CDF mediante el uso de funciones que resuelven por etapas:

1. Construir el sistema tridiagonal.
2. Resolver el sistema tridiagonal.
3. Comparar la aproximación numérica con el valor exacto.

Rutina `Bilocal`

La rutina **`Bilocal`** implementa el algoritmo descrito anteriormente.

La función tiene como "entradas":

- El dominio de solución x_a y x_b.
- El número nx de puntos en la malla.
- Los cuatro coeficientes de CDF: `alfa1`, `beta1`, `alfa2`, `beta2`.
- La función **`Func`**

Código para Bilocal I

Código 1: Función Bilocal

```
def Bilocal(xa, xb, y, nx, alfa1, beta1, alfa2, beta2, Func):
    a = [0] * (nx + 1); b = [0] * (nx + 1); c = [0] * (nx + 1)

    hx = (xb - xa)/(nx - 1); h2 = 2.e0 * hx * hx

    b[1] = hx * alfa1 - beta1; c[1] = beta1; y[1] = hx

    for m in range (2, nx):
```

Código para Bilocal II

```
    x = xa + (m - 1) * hx
    (p, q, r) = Func(x)
    a[m] = -(2.e0 + hx * p); b[m] = 4.e0 + h2 * q; c[m] = -(2.e0 - hx * p)
    y[m] = -h2 * r
a[nx] = -beta2; b[nx] = hx * alfa2 + beta2; y[nx] = hx

TriDiagSys(a, b, c, y, nx)
```

La función `Func`

Los valores de las funciones $p(x)$, $q(x)$ y $r(x)$ que definen la EDO-2, son proporcionados por el usuario en la función `Func`, a través de los argumentos p, q y r.

Código para `Func` I

Código 2: Código para Func

```
def Func(x):
    global n
    p = 2.e0 * x/(1.e0 - x * x); q =- n * (n + 1)/(1.e0 - x * x); r = 0.e0
    return (p, q, r)
```

Solución de la matriz tridiagonal

La manera más efectiva para resolver el sistema tridiagonal es mediante la factorización **LU**.

En la rutina `TriDiagSys` se presenta un algoritmo que resuelve la matriz tridiagonal.

Código para `TrigDiagSys` I

Código 3: Función TriDiagSys

```
def TriDiagSys(a, b, c, d, n):
    if (b[1] == 0.e0): print("
   TriDiagSys: Tenemos una matriz
   singular"); return
    for i in range(2, n + 1):
        a[i] /= b[i - 1]
        b[i] -= a[i] * c[i - 1]
        if (b[i] == 0.e0): print("
   TriDiagSys: Tenemos una matriz
   singular"); return
        d[i] -= a[i] * d[i - 1]

```

Código para TrigDiagSys II

```
    d[n] /= b[n]
    for i in range(n - 1,0, -1): d[i]
     = (d[i] - c[i] * d[i + 1])/b[i]
```

El algoritmo

La función `Bilocal` calcula los coeficientes del sistema discreto, los pasa a la rutina `TriDiagSys`, que resuelve el sistema y devuelve la solución en el elemento $y[\ \ ]$.

Ejercicio

Consideremos el problema de los polinomios de Legendre:

$$\frac{d^2 P_n}{dx^2} = \frac{1}{1 - x^2} \left[2\, x \, \frac{dP_n}{dx} - n(n+1)\, P_n \right] \tag{12}$$

$$P_n(-1) = (-1)^n, \qquad P_n(1) = 1 \tag{13}$$

Completando el problema

Las funciones que definen el lado derecho de la EDO-2, son:

$$p(x) = \frac{x}{1 - x^2} \quad q(x) = -\frac{n(n+1)}{1 - x^2}, \quad r(x) = 0 \tag{14}$$

Las condiciones de frontera de Dirichlet a partir de los coeficientes son:

$$\begin{aligned} \alpha_1 &= (-1)^n, \quad \beta_1 = 0 \\ \alpha_2 &= 1, \qquad \beta_2 = 0 \end{aligned} \tag{15}$$

Solución

Resuelve el problema para $n = 5$, usa la función **`Func`** para obtener los valores de las funciones $p(x)$, $q(x)$ y $r(x)$.

Compara los resultados usando $h = 0.1$ y $h = 0.01$. Discute los resultados.

Archivos de datos

Los resultados se van a guardar en dos archivos de texto plano:

1. Para $h = 0.1$, se llamará "bilocalh0_1.txt"

Archivos de datos

Los resultados se van a guardar en dos archivos de texto plano:

1. Para $h = 0.1$, se llamará “bilocalh0_1.txt”
2. Para $h = 0.01$, se llamará “bilocalh0_01.txt”

Programa completo I

Código 4: Programa completo

```python
def Func(x):
    global n
    p = 2.e0 * x/(1.e0 - x * x)
    q = -n * (n + 1)/(1.e0 - x * x)
    r = 0.e0
    return (p, q, r)

n = 5
xa = -1.e0
xb = 1.e0
hx = 0.1

```

Programa completo II

```
nx = int((xb - xa)/hx + 0.5) + 1

x = [0] * (nx + 1)
y = [0] * (nx + 1)

for m in range(1, nx + 1):
    x[m] = xa + (m - 1) * hx

alfa1 = -1.e0 if n % 2 else 1.e0; 
   beta1 = 0.e0
alfa2 = 1.e0
beta2 = 0.e0

```

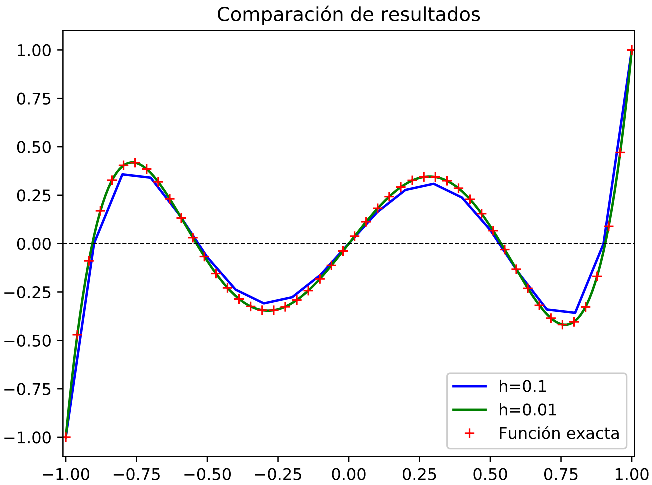
Programa completo III

```
Bilocal(xa, xb, y, nx, alfa1, beta1, alfa2, beta2, Func)

out = open("bilocalh0_1.txt", "w")
out.write("       x          P{0:1d}          err\n".format(n))
print('x \t P{0:1d} \t err\n'.format(n))

for m in range(1, nx + 1):
    (P, d) = Legendre(n, x[m])
```

Programa completo IV

```
    print('{:2.2f} \t {:2.6f} \t {:2.6f}'.format(x[m], y[m], P - y[m]))
    out.write(("{0:10.5f} {1:10.5f} {2:10.5f}\n").format(x[m], y[m], P - y[m]))
out.close()
```

Solución gráfica



¿Cómo graficamos los resultados?

Para graficar los archivos de texto que obtuvimos (se guardó uno con el valor de $h = 0.1$ y el otro con $h = 0.01$), podemos recuperar los valores a listas y ocupar las rutinas conocidas previamente.

La función `genfromtxt`

Para recuperar el conjunto de datos de un archivo de una manera mucho más rápida y sencilla, usamos la función **`genfromtxt`**:

Que carga el conjunto de datos contenido en un archivo de texto plano.

Usando la función `genfromtxt`

Debemos de asignar en una variable el contenido de datos que guardamos en cada archivo, mediante **`genfromtxt`**:

Código 5: Función `genfromtxt`

```
mat0 = np.genfromtxt("bilocalh0_1.txt")

mat1 = np.genfromtxt("bilocalh0_01.txt")
```

Comparando con la función exacta

El problema pide que comparemos la solución obtenida por el procedimiento de diferencias finitas contra el valor de la función exacta, por ello, nos apoyamos con la función especial `scipy.special.legendre`:

Código 6: Evaluando la función de Legendre

```
from scipy.special import legendre
   as splegendre

xj5 = np.linspace(-1., 1.)

j5 = np.polyval(splegendre(5), xj5)
```

Graficando las funciones

Por último nos resta incluir la rutina de graficación que ya manejamos con facilidad y así obtenemos la gráfica con las tres curvas:

Graficando las funciones I

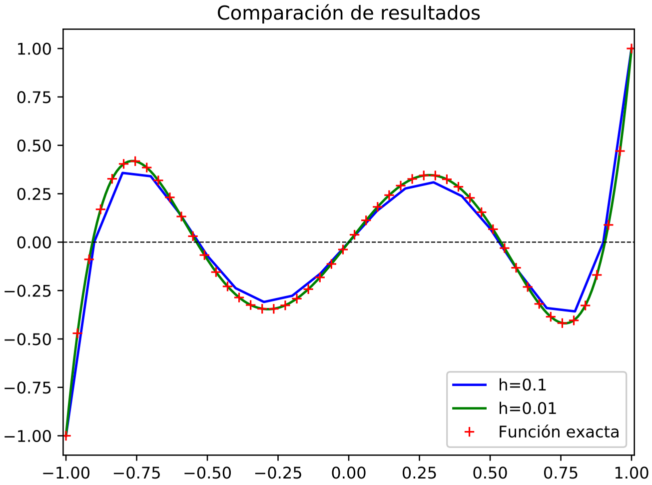
Código 7: Rutina de graficación

```
import matplotlib.pyplot as plt

plt.plot(mat0[:,0], mat0[:,1], 'b',
    label = "h=0.1")
plt.plot(mat1[:,0], mat1[:,1], 'g',
    label = "h=0.01")
plt.plot(xj5, j5, 'r+', label='
    Funcion exacta')
plt.axhline(y=0, ls='dashed', lw=0.7
    , color='k')
plt.title('Comparacion de resultados
    ')
```

Graficando las funciones II

```
plt.xlim([-1.01, 1.01])
plt.legend(loc=’lower right’)
plt.show()
```

Comparando los resultados obtenidos



Ejercicios a cuenta de examen

Los siguientes ejercicios son a cuenta del examen parcial:

1. Usando el método de disparo resuelve para el polinomio de Legendre de orden 5, es decir $P_5(x)$ y compara la solución con el método de diferencias finitas que se menciona en el ejercicio anterior.

Ejercicios a cuenta de examen

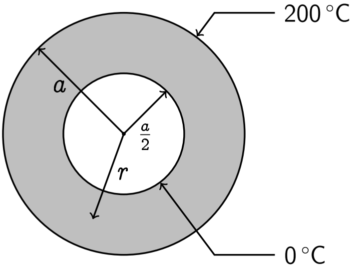
2. Usando el método de diferencias finitas con CDF, resuelve para los polinomios de Chebychev de primera clase

$$\frac{d^2 T_n}{dx^2} = \frac{1}{1 - x2} \left[x \, \frac{dT_n}{dx} - n^2 \, T_n \right]$$

$$T_n(-1) = (-1)^n \quad T_n(1) = 1$$

Usando un tamaño de paso de $h = 10^{-4}$. Calcula y gráfica los polinomios T_n de primera clase de orden $n = 1, 2, \ldots, 5$.

3. Un cilindro grueso transporta un fluido con una temperatura de $0\,^\circ$C. Al mismo tiempo, el cilindro se sumerge en un baño que se mantiene a $200\,^\circ$C.



EDO y CDF

La ecuación diferencial y las CDF que determinan la conducción de calor en estado estacionario en el cilindro son

$$\frac{d^2T}{dr^2} = -\frac{1}{r}\frac{dT}{dr} \quad T_{|r=a/2} = 0\,^{\circ}\text{C} \quad T_{|r=a} = 200\,^{\circ}\text{C}$$

donde T es la temperatura.

Problema a resolver

Determina el perfil de temperatura a través del cilindro usando el método de diferencias finitas, compara tu resultado con la solución analítica:

$$T = 200\left(1 - \frac{\ln r/a}{\ln 0.5}\right)$$