Resolución de problemas de contorno

Manuel Carlevaro

Departamento de Ingeniería Mecánica

Grupo de Materiales Granulares - UTN FRLP

manuel.carlevaro@gmail.com

Cálculo Avanzado • 2023

Problema de contorno

$$
\begin{aligned}
y'' = f(x, y, y'),\ a \leq x \leq b \\
\alpha_1 y(a) + \alpha_2 y'(a) = \alpha \\
\beta_1 y(b) + \beta_2 y'(b) = \beta
\end{aligned}
$$

Si f es de la forma

$$f(x, y, y') = p(x)y' + q(x)y + r(x)$$

el problema de contorno se llama **lineal**, de otro modo es **no lineal**.

Problema de contorno

$$y'' = f(x, y, y'),\ a \leq x \leq b$$
$$\alpha_1 y(a) + \alpha_2 y'(a) = \alpha$$
$$\beta_1 y(b) + \beta_2 y'(b) = \beta$$

Si f es de la forma

$$f(x, y, y') = p(x)y' + q(x)y + r(x)$$

el problema de contorno se llama **lineal**, de otro modo es **no lineal**.

Condiciones de borde o frontera:

- Dirichlet: $\alpha_2 = \beta_2 = 0$
- Neumann: $\alpha_1 = \beta_1 = 0$
- Robin: combinación lineal de valores de la función y sus derivadas en la frontera

Problema de contorno

$$y'' = f(x, y, y'),\ a \leq x \leq b$$
$$\alpha_1 y(a) + \alpha_2 y'(a) = \alpha$$
$$\beta_1 y(b) + \beta_2 y'(b) = \beta$$

Si f es de la forma

$$f(x, y, y') = p(x)y' + q(x)y + r(x)$$

el problema de contorno se llama **lineal**, de otro modo es **no lineal**.

Condiciones de borde o frontera:

- Dirichlet: $\alpha_2 = \beta_2 = 0$
- Neumann: $\alpha_1 = \beta_1 = 0$
- Robin: combinación lineal de valores de la función y sus derivadas en la frontera

Teorema : Existencia y unicidad.

Sea $f(x, y, y') \in C$ en el conjunto

$$D = \{(x, y, y') \,|\, a \leq x \leq b, -\infty \leq y \leq \infty, -\infty \leq y' \leq \infty\}$$

y que las derivadas parciales f_y y $f_{y'}$ son también continuas en D. Si

- $f_y(x, y, y') > 0,\ \forall (x, y, y') \in D$ *y*
- *existe una constante M tal que*

$$|f_{y'}(x, y, y')| \leq M,\ \forall (x, y, y') \in D$$

entonces el problema con valores de contorno tiene una solución.

Ejemplo: mostrar que

$$\begin{aligned} y'' + e^{-xy} + \operatorname{sen} y' = 0,\ 1 \leq x \leq 2 \\ y(1) = y(2) = 0 \end{aligned}$$

tiene una solución única.
Solución: tenemos

$$f(x, y, y') = -e^{-xy} - \operatorname{sen} y'$$

y para todas las x en $[1, 2]$:

$$\begin{aligned} f_y(x, y, y') = xe^{-xy} > 0 \quad \text{y} \\ \left|f_{y'}(x, y, y')\right| = \left|-\cos y'\right| \leq 1 \end{aligned}$$

Por lo que el problema tiene solución única.

Ejemplo: mostrar que

$$\begin{aligned} y'' + e^{-xy} + \operatorname{sen} y' = 0,\ 1 \leq x \leq 2 \\ y(1) = y(2) = 0 \end{aligned}$$

tiene una solución única.
Solución: tenemos

$$f(x, y, y') = -e^{-xy} - \operatorname{sen} y'$$

y para todas las x en $[1, 2]$:

$$\begin{aligned} f_y(x, y, y') = xe^{-xy} > 0 \quad \text{y} \\ \left|f_{y'}(x, y, y')\right| = \left|-\cos y'\right| \leq 1 \end{aligned}$$

Por lo que el problema tiene solución única.

Teorema : Problema lineal (corolario).

Si el problema lineal

$$\begin{aligned} y'' = p(x)y' + q(x)y + r(x),\ a \leq x \leq b \\ y(a) = \alpha,\ y(b) = \beta \end{aligned}$$

satisface

- $p(x)$, $q(x)$ *y* $r(x)$ *son continuas en* $[a, b]$
- $q(x) > 0$ *en* $[a, b]$

Entonces el problema con valor en la frontera tiene una única solución.

Ejemplo: mostrar que

$$
\begin{aligned}
y'' + e^{-xy} + \operatorname{sen} y' = 0,\ 1 \leq x \leq 2 \\
y(1) = y(2) = 0
\end{aligned}
$$

tiene una solución única.

Solución: tenemos

$$
f(x, y, y') = -e^{-xy} - \operatorname{sen} y'
$$

y para todas las x en $[1, 2]$:

$$
\begin{aligned}
f_y(x, y, y') = xe^{-xy} > 0 \quad \text{y} \\
\left|f_{y'}(x, y, y')\right| = \left|-\cos y'\right| \leq 1
\end{aligned}
$$

Por lo que el problema tiene solución única.

Teorema : Problema lineal (corolario).

Si el problema lineal

$$
\begin{aligned}
y'' = p(x)y' + q(x)y + r(x),\ a \leq x \leq b \\
y(a) = \alpha,\ \ y(b) = \beta
\end{aligned}
$$

satisface

- $p(x)$, $q(x)$ *y* $r(x)$ *son continuas en* $[a, b]$
- $q(x) > 0$ *en* $[a, b]$

Entonces el problema con valor en la frontera tiene una única solución.

Métodos de solución:

- Método de disparo
- Diferencias finitas

Método de disparo: transformamos el problema de condiciones de contorno (Dirichlet):

$$y'' = f(x, y, y'),\ a \leq x \leq b$$
$$y(a) = \alpha,\ y(b) = \beta$$

en un problema de valor inicial:

$$y'' = f(x, y, y'),\ a \leq x \leq b$$
$$y(a) = \alpha,\ y'(a) = r_0$$

donde r_0 es un parámetro inicial: $y(b) \mapsto y(b, r_0)$
Resolvemos el problema de valor inicial con algún método conocido (por ejemplo Runge-Kutta de cuarto orden).
Si $y(b, r_0)$ no está cerca de β, corregimos la aproximación con $r_1, r_2, \ldots$ hasta que

$$|y(b, r_k) - \beta| < \varepsilon$$

Si $f(x, y, y')$ satisface las condiciones de existencia y unicidad, el problema:

$$y(b, r) - \beta = 0$$

es una ecuación no lineal en la variable r.

Método de disparo: transformamos el problema de condiciones de contorno (Dirichlet):

$$y'' = f(x, y, y'),\ a \leq x \leq b$$
$$y(a) = \alpha,\ y(b) = \beta$$

en un problema de valor inicial:

$$y'' = f(x, y, y'),\ a \leq x \leq b$$
$$y(a) = \alpha,\ y'(a) = r_0$$

donde r_0 es un parámetro inicial: $y(b) \mapsto y(b, r_0)$
Resolvemos el problema de valor inicial con algún método conocido (por ejemplo Runge-Kutta de cuarto orden).
Si $y(b, r_0)$ no está cerca de β, corregimos la aproximación con $r_1, r_2, \ldots$ hasta que

$$|y(b, r_k) - \beta| < \varepsilon$$

Si $f(x, y, y')$ satisface las condiciones de existencia y unicidad, el problema:

$$y(b, r) - \beta = 0$$

es una ecuación no lineal en la variable r.

Procedimiento:

- Seleccionamos aproximaciones iniciales r_0 y r_1 que encierren la solución:

$$y(b, r_0) < \beta < f(b, r_1)$$

- Calculamos la raíz r^* de

$$f_{\text{residuo}}(r) = f(b, r) - \beta$$

  con el método de bisección.
- Resolvemos el problema de valor inicial con

$$y(a) = \alpha,\ y'(a) = r^*$$

Ejemplo:

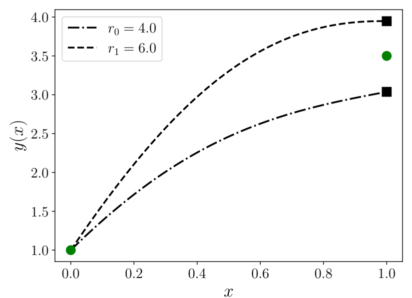
$$y'' = 12x - 4y,\ 0 \le x \le 1$$

$$y(0) = 1,\ y(1) = 3.5$$

Exploramos la solución para los valores $y'(0) = r_1 = 4$ y $y'(0) = r_2 = 6$:

```python
def f(x, y):
    f = np.zeros(2)
    f[0] = y[1]
    f[1] = 12 * x - 4 * y[0]
    return f
```

```python
x_a, x_b = 0.0, 1.0
n_points = 100
beta = 3.5
x = np.linspace(x_a, x_b, n_points)

sol_1 = solve_ivp(f, [x_a, x_b], [1.0, 4.0], dense_output=True)
y_1 = sol_1.sol(x)
sol_2 = solve_ivp(f, [x_a, x_b], [1.0, 6.0], dense_output=True)
y_2 = sol_2.sol(x)
plt.plot(x, y_1.T[:, 0], '-.k', label=r"$r_0 = 4.0$")
plt.plot(x, y_2.T[:, 0], '--k', label=r"$r_1 = 6.0$")
plt.plot([0, 1], [1.0, 3.5], 'go')
```



```python
def residuo(r, a, b, beta):
    x = np.linspace(a, b, 100)
    sol = solve_ivp(f, [a, b], [1, r], dense_output=True)
    z = sol.sol(x)
    resid = z.T[-1, 0] - beta
    return resid
```

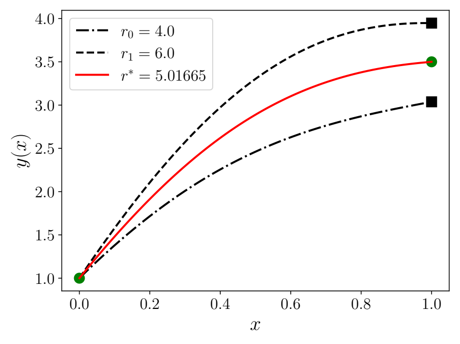
```python
r_opt = bisect(residuo, 4.0, 6.0, args=(x_a, x_b, beta),
    full_output=True)
print(r_opt[1])
sol = solve_ivp(f, [x_a, x_b], [1.0, r_opt[0]],
    dense_output=True)
y = sol.sol(x)
plt.plot(x, y.T[:, 0], '-r', label=r"$r^* = 5.01665$")
```

```
$ ./disparo.py
      converged: True
           flag: 'converged'
 function_calls: 42
     iterations: 40
           root: 5.016654027140248
```

```python
def residuo(r, a, b, beta):
    x = np.linspace(a, b, 100)
    sol = solve_ivp(f, [a, b], [1, r], dense_output=True)
    z = sol.sol(x)
    resid = z.T[-1, 0] - beta
    return resid
```

```python
r_opt = bisect(residuo, 4.0, 6.0, args=(x_a, x_b, beta),
    full_output=True)
print(r_opt[1])
sol = solve_ivp(f, [x_a, x_b], [1.0, r_opt[0]],
    dense_output=True)
y = sol.sol(x)
plt.plot(x, y.T[:, 0], '-r', label=r"$r^* = 5.01665$")
```

```
$ ./disparo.py
      converged: True
           flag: 'converged'
 function_calls: 42
     iterations: 40
           root: 5.016654027140248
```



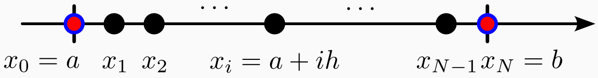
Diferencias finitas: aproximación de derivadas por diferencias finitas. Problema lineal con valores de contorno:

$$y'' = p(x)y' + q(x)y + r(x),\ a \leq x \leq b,\ y(a) = \alpha,\ y(b) = \beta \tag{1}$$

Diferencias finitas: aproximación de derivadas por diferencias finitas. Problema lineal con valores de contorno:

$$y'' = p(x)y' + q(x)y + r(x),\ a \le x \le b,\ y(a) = \alpha,\ y(b) = \beta \tag{1}$$

Malla: dividimos $[a, b]$ en N subintervalos:

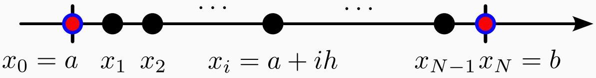


con $h = (b - a)/N$, $x_i = a + ih, i = 0, 1, \ldots, N - 1, N$.

Diferencias finitas: aproximación de derivadas por diferencias finitas. Problema lineal con valores de contorno:

$$y'' = p(x)y' + q(x)y + r(x), \; a \le x \le b, \; y(a) = \alpha, \; y(b) = \beta \tag{1}$$

Malla: dividimos $[a, b]$ en N subintervalos:



con $h = (b-a)/N$, $x_i = a + ih, i = 0, 1, \ldots, N-1, N$.

Expansión de y: polinomio de Taylor, alrededor de x_i evaluando en x_{i+1} y x_{i-1}, con $y \in C^4[x_{i-1}, x_{i+1}]$:

$$y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \frac{h^3}{6}y'''(x_i) + \frac{h^4}{24}h^{(4)}y^{(4)}(\xi_i^+)$$
$$y(x_i - h) = y(x_i) - hy'(x_i) + \frac{h^2}{2}y''(x_i) - \frac{h^3}{6}y'''(x_i) + \frac{h^4}{24}h^{(4)}y^{(4)}(\xi_i^-)$$

con $\xi_i^+(x_i, x_i + h)$ y $\xi_i^- \in (x_i - h, x_i)$. Restando y sumando:

$$\begin{aligned} y'(x_i) &= \frac{1}{2h}[y(x_{i+1}) - y(x_{i-1})] - \frac{h^2}{6}y'''(\eta_i) \\ y''(x_i) &= \frac{1}{h^2}[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})] - \frac{h^2}{12}y^{(4)}(\xi_i) \end{aligned} \tag{2}$$

con $\eta_i, \xi_i \in (x_{i-1}, x_{i+1})$.
Notación: $y(x_i) \mapsto y_i$,
$f(x_i) \mapsto f_i, f = p, q, r$.

Reemplazando (2) en (1):

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = p_i \left[\frac{y_{i+1} - y_{i-1}}{2h}\right] + q_i y_i + r_i - \frac{h^2}{12}[2p_i y'''(\eta_i) - y^{(4)}(\xi_i)]$$

Reemplazando (2) en (1):

$$\frac{y_{i+1}-2y_i+y_{i-1}}{h^2}=p_i\left[\frac{y_{i+1}-y_{i-1}}{2h}\right]+q_iy_i+r_i-\frac{h^2}{12}[2p_iy'''(\eta_i)-y^{(4)}(\xi_i)]$$

Condiciones de borde $y_0=\alpha$, $y_{N+1}=\beta$, error de truncamiento $\mathcal{O}(h^2)\mapsto$ sistema de ecuaciones lineales:

$$\left(\frac{y_{i+1}-2y_i+y_{i-1}}{h^2}\right)+p(x_i)\left(\frac{y_{i+1}-y_{i-1}}{2h}\right)+q(x_i)y_i=-r(x_i)$$

para los puntos interiores de la malla $i=1,2,\ldots,N-1$. Reordenando:

$$\left(-1-\frac{h}{2}p_i\right)y_{i-1}+\left(2+h^2q_i\right)y_i+\left(-1+\frac{h}{2}p_i\right)y_{i+1}=-h^2r(x_i)$$

Sistema con matriz tridiagonal $(N-1)\times(N-1)$:

$$\boldsymbol{A}\boldsymbol{y}=\boldsymbol{b}$$

Para $i=1$:

$$y_{i-1}=y_0=y(a)=\alpha$$

Para $i=N-1$:

$$y_{i+1}=y_N=y(b)=\beta$$

$$
\boldsymbol{A} = \begin{bmatrix}
2+h^2q_1 & -1+\frac{h}{2}p_1 & & & & \text{\Huge 0} \\
-1-\frac{h}{2}p_2 & 2+h^2q_2 & -1+\frac{h}{2}p_2 & & & \\
 & -1-\frac{h}{2}p_3 & 2+h^2q_3 & -1+\frac{h}{2}p_3 & & \\
 & & & \ddots & & \\
 & & & -1-\frac{h}{2}p_{N-2} & 2+h^2q_{N-2} & -1+\frac{h}{2}p_{N-2} \\
\text{\Huge 0} & & & & -1-\frac{h}{2}p_{N-1} & 2+h^2q_{N-1}
\end{bmatrix}
$$

$$
\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix}, \quad
\boldsymbol{b} = \begin{bmatrix} -h^2r_1 + \left(1+\frac{h}{2}p_1\right)\alpha \\ -h^2r_2 \\ \vdots \\ -h^2r_{N-2} \\ -h^2r_{N-1} + \left(1-\frac{h}{2}p_{N-1}\right)\beta \end{bmatrix}
$$

Ejemplo:

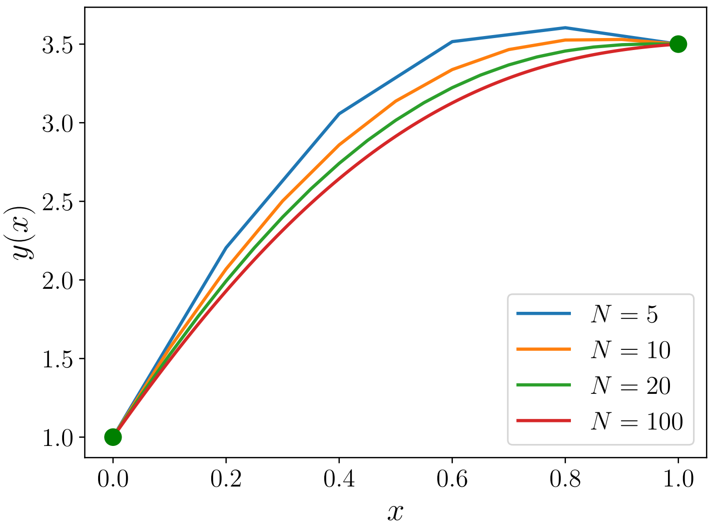
$$y'' = 12x - 4y,\ 0 \le x \le 1$$
$$y(0) = 1,\ y(1) = 3.5$$

```python
def p(x):
    return 0

def q(x):
    return -4

def r(x):
    return 12 * x

def solve_DF(N):
    a, b = 0, 1
    alfa , beta = 1, 3.5
    h = (b - a) / N
    x = np.linspace(a, b, N + 1)
    print("x = ", x, "h = ", h)
```

```python
    # Matriz A
    A = np.zeros((N-1, N-1))
    A[0, 0] = 2 + h**2 * q(x[0])
    A[0, 1] = -1 + h / 2 * p(x[0])
    A[N-2, N - 3] = -1 - h / 2 * p(x[N-1])
    A[N-2, N-2] = 2 + h**2 * q(x[N-1])
    for i in range(1, N-2):
        A[i, i - 1] = -1 - h / 2 * p(x[i])
        A[i, i] = 2 + h**2 * q(x[i])
        A[i, i + 1] = -1 + h / 2 * p(x[i])
    print(A)
    # Vector b
    b = np.zeros(N-1)
    b[0] = -h**2 * r(x[0]) + (1 + h / 2 * p(x[0])) * alfa
    b[1 : -1] = - h**2 * r(x[1 : -3])
    b[-1] = -h**2 * r(x[N-1]) + (1 - h / 2 * p(x[N-1])) * beta
    print("b = ", b)
    # Resolvemos para y
    y = np.zeros(N + 1)
    y[0] = alfa
    y[1:-1] = np.linalg.solve(A, b)
    y[-1] = beta
    return x, y
```

```python
# Graficamos las soluciones
for N in [5, 10, 20, 100]:
    print(f"N = {N}")
    x, y = solve_DF(N)
    plt.plot(x, y, label=f"$N = {N}$")

plt.plot([0, 1], [1.0, 3.5], 'go')
plt.xlabel(r"$x$")
plt.ylabel(r"$y(x)$")
plt.legend(loc=4)
plt.tight_layout()
plt.savefig("dif-finitas.pdf")
```

```
N = 10
x =  [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ] h =  0.1
[[ 1.96 -1.    0.    0.    0.    0.    0.    0.    0.  ]
 [-1.    1.96 -1.    0.    0.    0.    0.    0.    0.  ]
 [ 0.   -1.    1.96 -1.    0.    0.    0.    0.    0.  ]
 [ 0.    0.   -1.    1.96 -1.    0.    0.    0.    0.  ]
 [ 0.    0.    0.   -1.    1.96 -1.    0.    0.    0.  ]
 [ 0.    0.    0.    0.   -1.    1.96 -1.    0.    0.  ]
 [ 0.    0.    0.    0.    0.   -1.    1.96 -1.    0.  ]
 [ 0.    0.    0.    0.    0.    0.   -1.    1.96 -1.  ]
 [ 0.    0.    0.    0.    0.    0.    0.   -1.    1.96]]
b =  [ 1.    -0.012 -0.024 -0.036 -0.048 -0.06  -0.072 -0.084  3.392]
```



3.5
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
$y(x)$
x
$N = 5$
$N = 10$
$N = 20$
$N = 100$

- R.L. Burden, D.J. Faires y A.M. Burden. ***Análisis numérico.*** 10.ª ed. Mexico: Cengage Learning, 2017. Capítulo 11.
- B. Bradie. ***A Friendly Introduction to Numerical Analysis.*** New Jersey, United States: Pearson Education Inc., 2006. Capítulo 8.
- J. Kiusalaas. ***Numerical Methods in Engineering with Python 3.*** Cambridge, United Kingdom: Cambridge University Press, 2013. Capítulo 8.
- J.H. Mathews y K.D. Fink. ***Numerical methods using MATLAB.*** New Jersey, United States: Pearson Education Inc., 2004. Capítulo 9.