Análisis Númerico: Parcial 3

3 de marzo de 2025

Universidad Nacional de Colombia

Jorge Mauricio Ruiz Vera

Andrés David Cadena Simons	acadenas@unal.edu.co
Sandra Natalia Florez Garcia	sflorezga@unal.edu.co

Problema 1:

Dada la función

$$f(x) = x - \frac{1}{x} - 2, \quad f : \mathbb{R}^+ \to \mathbb{R},$$

se construye el siguiente algoritmo para aproximar la raíz $r = 1$:

$$x_{n+1} = 2 - \frac{1}{x_n}.$$

1. Verificar que si $x_0 > 1$ entonces la sucesión $\{x_n\}$ es monótona decreciente y acotada inferiormente por 1. Concluir que $x_n \to 1$ aunque esta iteración no esta en las hipótesis del teorema del punto fijo. ¿Qué hipótesis no se cumple?

Solución:

Dada la sucesión:

$$x_{n+1} = 2 - \frac{1}{x_n},$$

queremos demostrar que si $x_0 > 1$, entonces $\{x_n\}$ es monótona decreciente y acotada inferiormente por 1.
Demostramos por inducción que $x_n \geq 1$ para todo n:

- El caso base se tiene por hipótesis, ya que $x_0 > 1$.
- Ahora suponga que se tiene para n, es decir.
  Supongamos que $x_n \geq 1$. Veamos si $x_{n+1} \geq 1$:

  $$x_{n+1} = 2 - \frac{1}{x_n}.$$

  Como $x_n \geq 1$, se tiene $\frac{1}{x_n} \leq 1$, lo que implica que:

  $$x_{n+1} = 2 - \frac{1}{x_n} \geq 2 - 1 = 1.$$

  Por lo tanto, la sucesión es acotada inferiormente por 1.

Ahora afirmamos que $x_{n+1} \leq x_n$, ya que:

$$2 \leq x_n + \frac{1}{x_n}.$$

Para todo $x_n > 0$, en particular para todo $x_n > 1$ (los nuestros) ahora si reordenamos la desigualdad:

$$x_{n+1} = 2 - \frac{1}{x_n} \leq x_n$$

Por lo tanto, la sucesión es monótona decreciente.
Ahora, para ver la convergencia note que $\{x_n\}$ es monótona decreciente y acotada inferiormente por 1, por el teorema de monotonía, converge a un límite L. Tomando el límite en la ecuación de recurrencia:

$$L = 2 - \frac{1}{L}.$$

Multiplicamos por L:

$$\begin{aligned} L^2 - 2L + 1 &= 0, \\ (L-1)^2 &= 0 \quad \Rightarrow L = 1. \end{aligned}$$

Por lo tanto, $x_n \to 1$.
Ahora el teorema del punto fijo requiere que la función de iteración sea contractiva en un intervalo alrededor del punto fijo. La derivada de la función de iteración:

$$\varphi(x) = 2 - \frac{1}{x}$$

es

$$\varphi'(x) = \frac{1}{x^2}.$$

Evaluando en $x = 1$:

$$\varphi'(1) = 1.$$

Como $|\varphi'(1)| = 1$, la condición de contractividad ($|\varphi'(x)| < 1$ en un entorno del punto fijo) no se cumple, por lo que no se puede garantizar la convergencia por el teorema del punto fijo.

2. Dar un algoritmo para aproximar la raíz de f que converja cuadráticamente.

Solución:

Ahora veamos un algoritmo de convergencia cuadrática que nos permita aproximar la raíz.
Para obtener un algoritmo que converja cuadráticamente a la raíz $x = 1$, podemos usar el **método de Newton**:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Calculamos $f(x) = x + \frac{1}{x} - 2$ y su derivada:

$$f'(x) = 1 - \frac{1}{x^2}.$$

Aplicamos la fórmula de Newton:

$$x_{n+1} = x_n - \frac{x_n + \frac{1}{x_n} - 2}{1 - \frac{1}{x_n^2}}.$$

Este método tiene **convergencia cuadrática**, lo que significa que el error disminuye aproximadamente como el cuadrado del error anterior en cada iteración.

Problema 2:

Pregunta

Solución:

Solución

Problema 3:

Sea $f \in C[a,b]$, y sean $x_0 = a, x_1 = a+h, \cdots, x_n = b$, donde $h = \frac{b-a}{n}$. Considerar la poligonal $l(x)$ que interpola a f en los puntos $x_i, i = 0, 1, \cdots, n$. Probar que

1. $$|f(x) - l(x)| \leq \frac{h^2}{2} \max_{x\in[a,b]} |f''(x)|$$

Solución:

Suponga el intervalo $[x_i, x_{i+1}]$, entonces

$$l(x) = \frac{f(x_{i+1}) - f(x_i)}{h}(x - x_i) + f(x_i),$$

luego

$$l(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{f''(\xi_1)}{2}(x_{i+1} - x_i)(x - x_i)$$

de dónde podemos deducir

$$f(x) - l(x) = (f(x) - f(x_i)) - (f'(x_i)(x - x_i) + \frac{f''(\xi_1)}{2}(x_{i+1} - x_i)(x - x_i)$$

Aplicando el teorema de Taylor en $f(x)$

$$f(x) - l(x) = f'(x_i)(x - x_i) + \frac{f''(\xi_2)}{2}(x - x_i)^2$$

de lo que se puede afirmar que

$$\begin{aligned} f(x) - f(x_i) &= \frac{f''(\xi_2)}{2}(x - x_i)^2 - \frac{f''(\xi_1)}{2}(x_{i+1} - x_i)(x - x_i) \\ &= \frac{(x - x_i)}{2}\left(f''(\xi_2)(x - x_i) - f''(\xi_1)(x_{i+1} - x_i)\right) \quad \text{Pero como } f \in C^2[a,b]. \\ &= \frac{(x - x_i)}{2} f''(\xi)((x - x_i) - (x_{i+1} - x_i)) \\ &= \frac{f''(\xi)}{2}(x - x_i)(x - x_{i+1}) \end{aligned}$$

luego usando esto se puede concluir que

$$\begin{aligned} |f(x) - l(x)| &= \left|\frac{f''(\xi)}{2}(x - x_i)(x - x_{i+1})\right| \\ &= \frac{h^2}{2} \max_{x\in[a,b]} |f''(\xi)| \end{aligned}$$

lo que concluye el resultado.

Problema 3 continúa en la siguiente página. . .

2.

$$|f'(x) - l'(x)| \leq h \max_{x\in[a,b]} |f''(x)|.$$

Solución:

Recordemos que $l(x) = \frac{f(x_{i+1})-f(x_i)}{h}(x - x_i) + f(x_i)$, luego $l'(x) = \frac{f(x_{i+1})-f(x_i)}{h} = f'(c)$ para algún $c \in [x_i, x_{i+1}]$. Ahora, aplicando el teorema de Taylor se tiene que:

$$f'(x) = f'(x_i) + f''(\xi_1)(x - x_i),$$

de lo que se sigue que

$$\begin{aligned}
|f'(x) - l'(x)| &= |f'(x_i) + f''(\xi_1)(x - x_i) - f'(c)| \\
&= |f'(x_i) + f''(\xi_1)(x - x_i) - f'(x_i) - f''(\xi_2)(x - x_i)| \\
&= |f''(\xi_1)(x - x_i) - f''(\xi_2)(c - x_i)| \\
&\leq h \max_{x\in[a,b]} |f''|
\end{aligned}$$

lo que concluye el resultado.

Problema 4:

Pregunta

Solución:

Solución

Problema 5:

Observe que

$$I = \int_0^1 \frac{4}{1+x^2} = \pi$$

1. Use las reglas compuestas del punto medio, del trapecio y de Simpson para aproximar I para varios tamaños de paso de integración $h = \frac{1}{n}, n = 10, 50, 100, 250, 500, 1000, 1500, 2000$. Grafique el logaritmo del error absoluto versus n para cada paso. Describa el efecto de redondeo de los errores cuando $i \to 0$.

Solución:

Veamos lo que sucede para cada caso:

- Punto Medio.
  Note que al aplicar el algoritmo para la integral obtenemos la siguiente gráfica de errores:

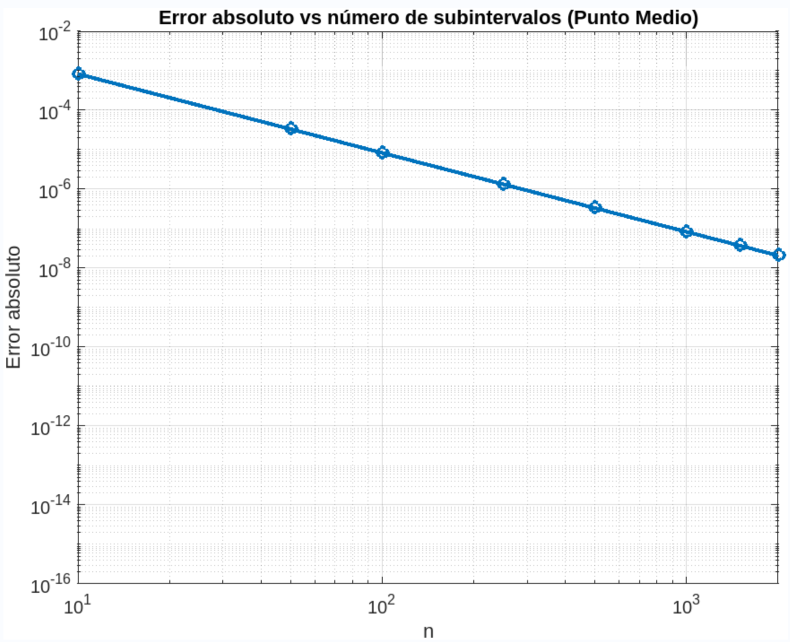

Figura 1: Error absoluto en el método del punto medio.

Aquí cabe recordar que el método del punto medio tiene convergencia de orden $O(h^2)$ y su implementación es bastante sencilla, la podremos ver en el siguiente código que aplicando el método se tomo 0,001995 segundos.

```
format rational

% Creamos la función f
f = @(x) 4 ./ (1 + x.^2);

% Parámetros de la integral y el método
a = 0;
b = 1;
valores_n = [10, 50, 100, 250, 500, 1000, 1500, 2000];
I_exacto = pi;

% Se crea un vector para guardar los errores
errores = zeros(size(valores_n));

% Método del punto medio
tic;
for k = 1:length(valores_n)
    n = valores_n(k);
    % Aquí se encuentra el tamaño del intervalo
    h = (b - a) / n;
    % Se calculan los puntos medios y los guarda en el vector
        x_medios
    x_medios = a + ((2*(0:1:n-1)+1)*h)/2;
    % Regla del punto medio compuesta
    I_punto_medio = h * sum(f(x_medios));
    % Calculo del error
    errores(k) = abs(I_punto_medio - I_exacto);
end
tiempo = toc

% Graficar log-log del error vs n
figure;
loglog(valores_n, errores, '-o', 'LineWidth', 2);
xlabel('n');
ylabel('Error absoluto');
ylim([1e-16, 1e-2]);
title('Error absoluto vs número de subintervalos (Punto
    Medio)');
grid on;
```

- Trapecio.
  Note que al aplicar el algoritmo para la integral obtenemos la siguiente gráfica de errores:

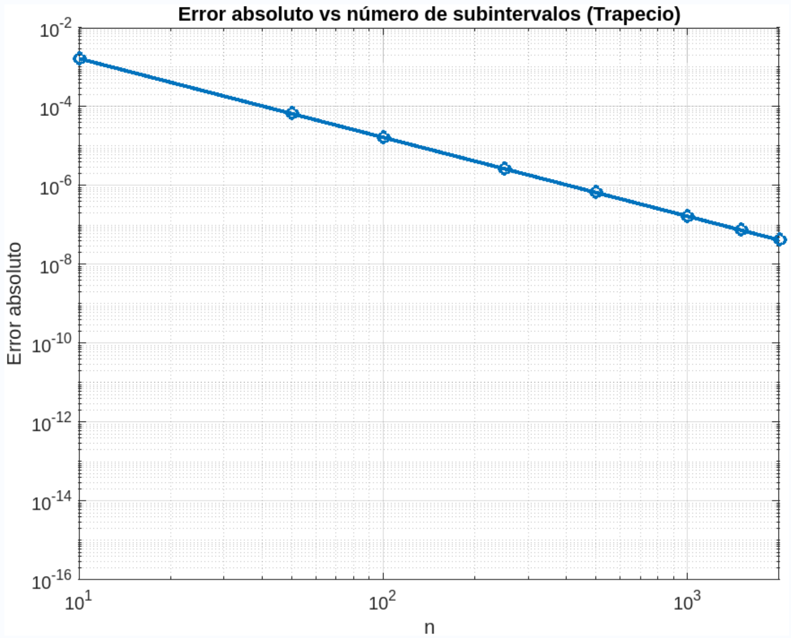

Figura 2: Error absoluto en el método del trapecio.

Aquí cabe recordar que el método del trapecio tiene convergencia de orden $O(h^2)$ y su implementación es (en cuestiones de operaciones) sencilla, la podremos ver en el siguiente código que aplicando el método se tomo 0,002719 segundos.

```matlab
format rational

% Creamos la función f
f = @(x) 4 ./ (1 + x.^2);

% Parámetros de la integral y el método
a = 0;
b = 1;
valores_n = [10, 50, 100, 250, 500, 1000, 1500, 2000];
I_exacto = pi;

% Se crea un vector para guardar los errores
errores_trapecio = zeros(size(valores_n));

% Método del trapecio
for k = 1:length(valores_n)
    n = valores_n(k);
    h = (b - a) / n;
    % Extremos de la partición
    x_extremos = a:h:b;
    m = length(x_extremos);

    % Aplicar la regla del trapecio compuesta
    I_trapecio = (h/2) * (f(a) + 2*sum(f(x_extremos(2:m-1)))
        + f(b));

    % Calculo del error
    errores_trapecio(k) = abs(I_trapecio - I_exacto);
end

% Graficar error del método del trapecio
figure;
loglog(valores_n, errores_trapecio, '-o', 'LineWidth', 2);
xlabel('n');
ylabel('Error absoluto');
ylim([1e-16, 1e-2]);
title('Error absoluto vs número de subintervalos (Trapecio)');
grid on;
```

- Simpson.
  Note que al aplicar el algoritmo para la integral obtenemos la siguiente gráfica de errores:

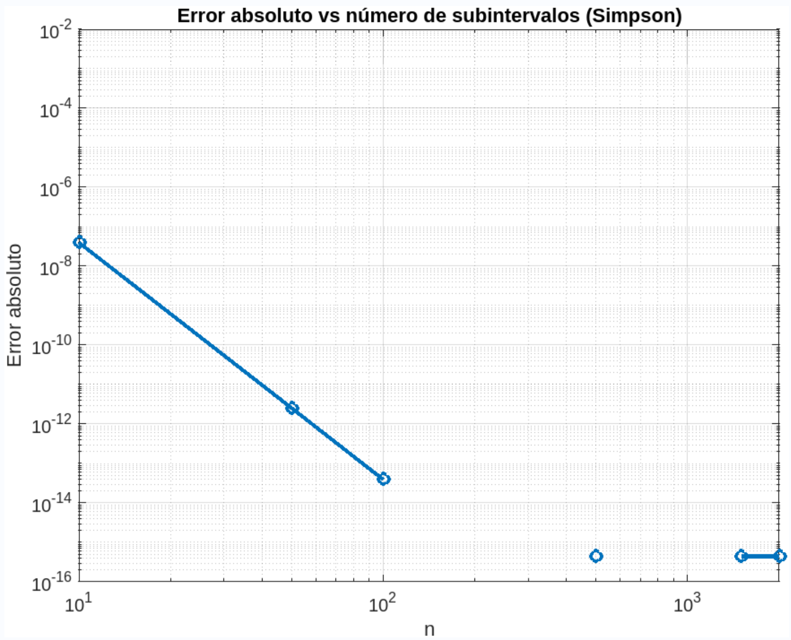

Figura 3: Error absoluto en el método de Simpson.

Aquí cabe recordar que el método tiene convergencia de orden $O(h^4)$ y su implementación es aunque un poco más compleja en la teoría, sigue siendo eficiente cuando hablamos computacionalmente, la podremos ver en el siguiente código que aplicando el método se tomo 0,003878 segundos.

```
format rational
% Creamos la función f
f = @(x) 4 ./ (1 + x.^2);
% Parámetros de la integral y el método
a = 0;
b = 1;
valores_n = [10, 50, 100, 250, 500, 1000, 1500, 2000];
I_exacto = pi;
% Se crea un vector para guardar los errores
errores_simpson = zeros(size(valores_n));
% Método de Simpson
tic;
for k = 1:length(valores_n)
    n = valores_n(k);
    % Aquí nos aseguramos que n sea par (Simpson necesita n
        par)
    if mod(n, 2) == 1
        n = n + 1;
    end
    h = (b - a) / n;
    % Extremos de la partición
    x_extremos = a:h:b;
    m = length(x_extremos);
    % Aplicar la regla de Simpson compuesta
    I_simpson = (h/3) * (f(a) + 4*sum(f(x_extremos(2:2:m-1)))
        + 2*sum(f(x_extremos(3:2:m-2))) + f(b));
    % Calculo del error
    errores_simpson(k) = abs(I_simpson - I_exacto);
end
tiempo = toc
% Graficar error del método de Simpson
figure;
loglog(valores_n, errores_simpson, '-o', 'LineWidth', 2);
xlabel('n');
ylabel('Error absoluto');
ylim([1e-16, 1e-2]);
title('Error absoluto vs número de subintervalos (Simpson)');
grid on;
```

Ahora sería interesante ver la comparación entre los 3 métodos en la siguiente figura y la tabla de tiempos:

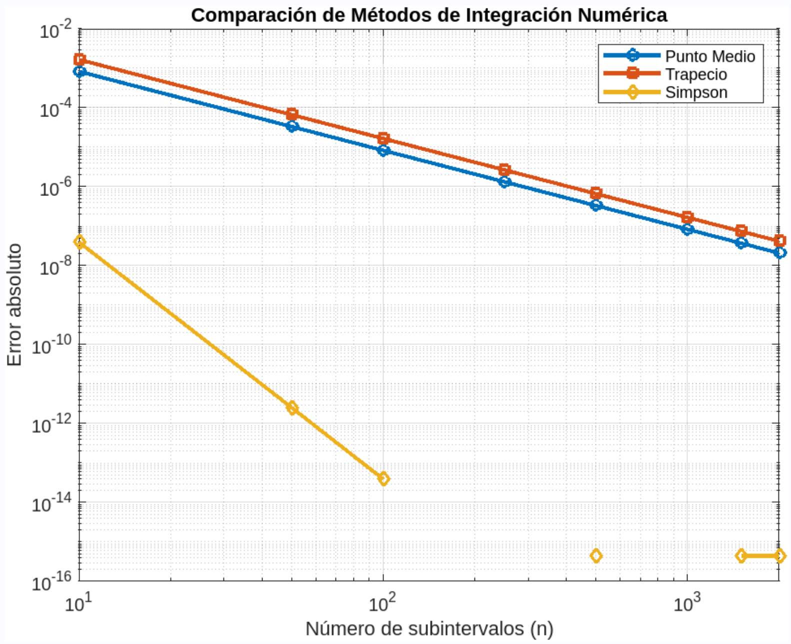

Figura 4: Comparación de los métodos.

Método	Tiempo (s)
Punto medio	$0,001995$
Trapecio	$0,002719$
Simpson	$0,003878$

De lo que podemos concluir lo siguiente:

- En la aplicación el método del punto medio aparenta ser superior al método del trapecio en cuestiones de errores y tiempo, no obstante la diferencia entre estos aparenta ser bastante baja para tenerlo en cuenta a la hora de realizar los cálculos y favorecer alguna de las 2, por lo que no se descarta que el uso del método del trapecio sea más efectivo en otro tipos de problemas en los que quizás sea mejor considerar los términos de borde en la función.
- Es claro que el método de Simpson es mucho más preciso al momento de aproximar el resultado, pues, desde la propia teoría (la convergencia en orden $O(h^4)$) vimos que al refinar la malla el error se reduciría bastante rápido, por lo que vemos que rápidamente con tomar $n = 250$ ya este había superado la precisión numérica de

maquina al ser tan cercano a 0 el error absoluto, situación que con los otros 2 métodos no se alcanzó a concluir de igual forma.

- Si bien en cuestión de tiempo vemos que aún podríamos permitirles a los métodos tener un refinamiento de malla aún más grande, es claro que al método de Simpson le fue suficiente con muy poco y que a los restantes 2 métodos aparenta aún faltarle mucho al refinamiento de la malla para alcanzar la precisión numérica que tiene el método de Simpson.

2. Implemente el método de integración de Romberg para calcular I. Grafiqué el logaritmo del error en los términos diagonales en la tabla de extrapolación versus $log(h)$. Verifique sus resultados con la teoría.

Solución:

Note que al aplicar el algoritmo para la integral obtenemos la siguiente gráfica de errores:

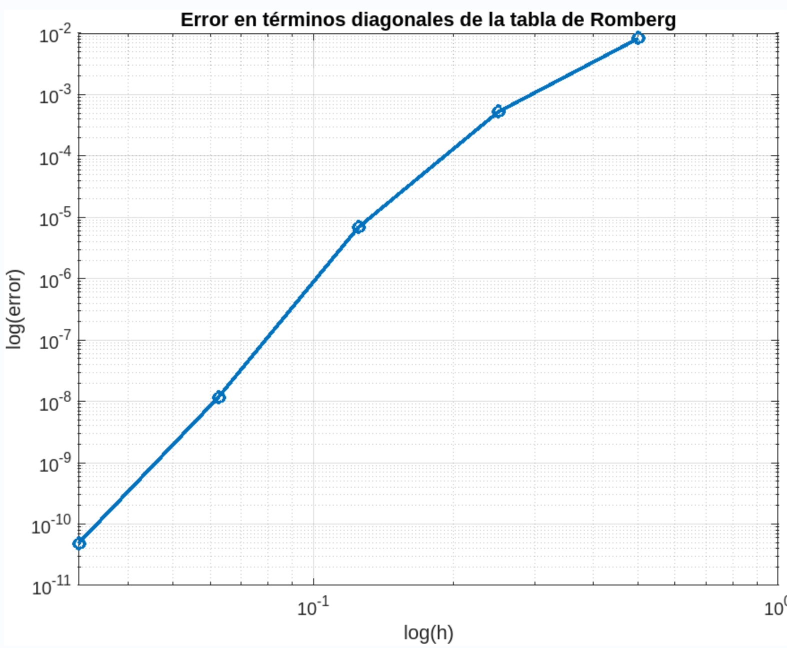

Figura 5: Error en los términos diagonales en la tabla de extrapolación versus $logh$.

El cuál se tardó 0,004838 segundos en ejecutarse.

De aquí podemos concluir lo siguiente:

- Respecto a la teoría, en efecto a valores pequeños de h mejora la aproximación del valor de la integral (más refinamiento, mejor aproximación).
- Note que la curva es çasiüna recta, de lo cuál podemos deducir al estar en escala logaritmica que el error $E(h) \approx Ch^p$ en donde el valor de p aumenta cada vez más con cada nivel de la extrapolación.
- Se alcanzan errores de hasta 10^{-10}, lo que indica que el método logra una precisión bastante alta con relativamente pocas iteraciones.

A continuación podemos ver la matriz:

$$\begin{pmatrix}
3{,}00000000 & 0 & 0 & 0 & 0 & 0 \\
3{,}10000000 & 3{,}13333333 & 0 & 0 & 0 & 0 \\
3{,}13117647 & 3{,}14156863 & 3{,}14211765 & 0 & 0 & 0 \\
3{,}13898849 & 3{,}14159250 & 3{,}14159409 & 3{,}14158578 & 0 & 0 \\
3{,}14094161 & 3{,}14159265 & 3{,}14159266 & 3{,}14159264 & 3{,}14159267 & 0 \\
3{,}14142989 & 3{,}14159265 & 3{,}14159265 & 3{,}14159265 & 3{,}14159265 & 3{,}14159265
\end{pmatrix}$$

Y en la siguiente página se podrá encontrar el código.

```matlab
format long
% Definimos la función
f = @(x) 4 ./ (1 + x.^2);
% Límites de integración
a = 0;
b = 1;
% Valor exacto de la integral
I_exacto = pi;
% Tolerancia
eps = 1e-8;
% Calculamos la integral con Romberg y obtenemos errores
tic;
[I_romberg, hs, errores] = romberg(f, a, b, I_exacto, eps);
toc
% Graficamos log(error) vs log(h)
figure;
loglog(hs, errores, '-o', 'LineWidth', 2);
xlabel('log(h)');
ylabel('log(error)');
title('Error en términos diagonales de la tabla de Romberg');
grid on;
% --- FUNCIONES ---
function [I_romberg, hs, errores] = romberg(f, a, b, I_exacto, eps)
    if nargin < 5
        % Precisión por defecto
        eps = 1e-8;
    end
    % Matriz de Romberg
    R = zeros(1, 1);
    % Primer trapecio
    R(1,1) = 0.5 * (b - a) * (f(a) + f(b));
    print_row(R(1,1));
    % Inicialización de vectores para la gráfica
    hs = [];
    errores = [];
    % Número de iteraciones
    n = 1;
    while true
        % Refinamos h
        h = (b - a) / 2^n;
        % Nueva regla del trapecio
        R(n+1,1) = 0.5 * R(n,1) + h * sum(f(a + (2*(1:2^(n-1)) - 1) * h));
```

Problema 5 continúa en la siguiente página. . .

```
        % Extrapolación de Richardson
        for m = 2:n+1
            R(n+1,m) = R(n+1,m-1) + (R(n+1,m-1) - R(n,m-1)) / (4^(m-1) - 1);
        end
        % Imprimir la fila
        print_row(R(n+1, 1:n+1));
        % Guardamos h y el error absoluto en la diagonal
        hs = [hs, h];
        errores = [errores, abs(R(n+1,n+1) - I_exacto)];
        % Criterio de convergencia
        if abs(R(n+1,n) - R(n+1,n+1)) < eps
            I_romberg = R(n+1,n+1);
            return;
        end
        n = n + 1;
    end
end
function print_row(row)
    fprintf('%11.8f ', row);
    fprintf('\n');
end
```