Análisis Númerico: Parcial 3

2 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:

Pregunta

Solución:

Solución

Problema 2:

Pregunta

Solución:

Solución

Problema 3:

Pregunta

Solución:

Solución

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:

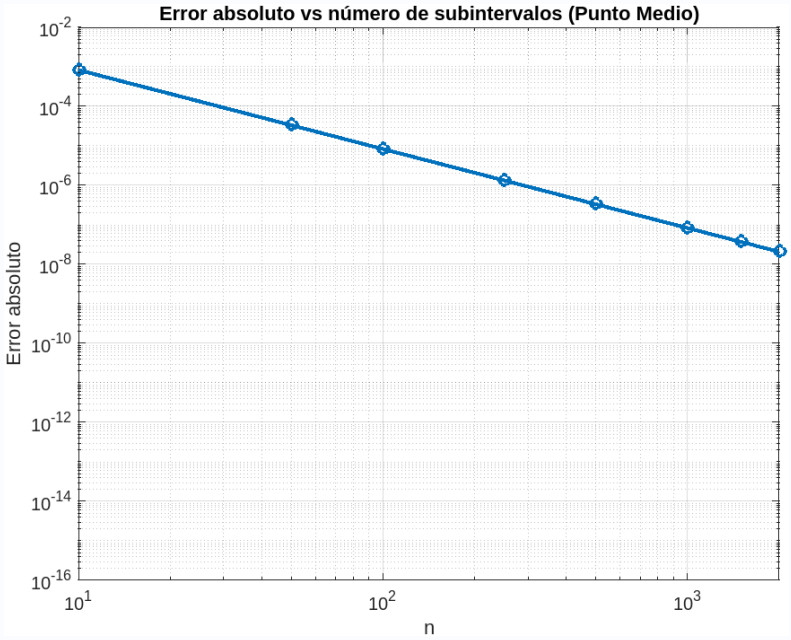

Figura 1: Error absoluto en el método del punto medio.

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.

Problema 5 continúa en la siguiente página...

```
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:

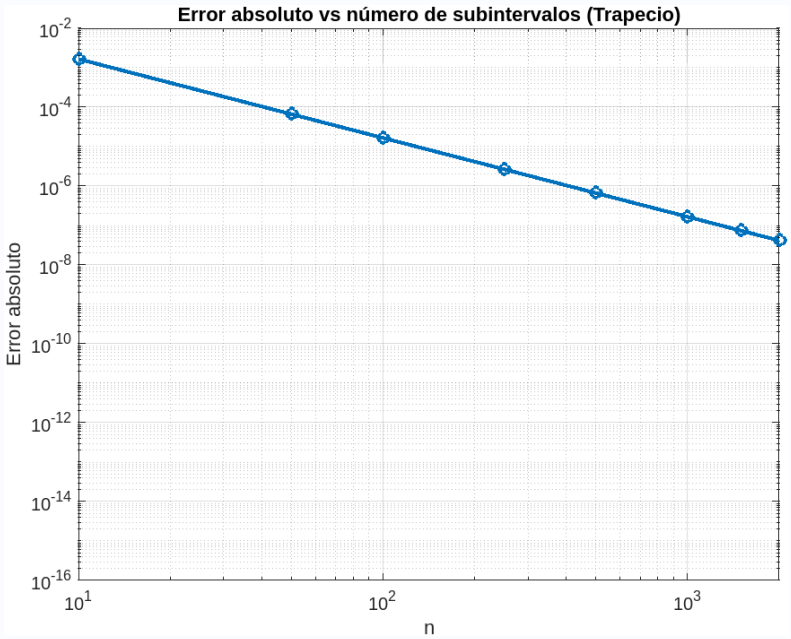

Figura 2: Error absoluto en el método del trapecio.

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.

```
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:

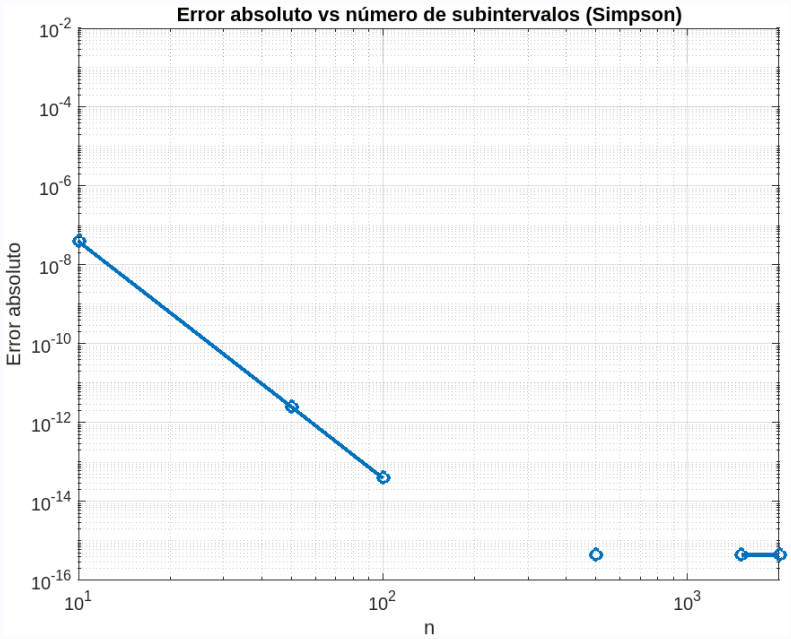

Figura 3: Error absoluto en el método de Simpson.

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;
```

Problema 5 continúa en la siguiente página...

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

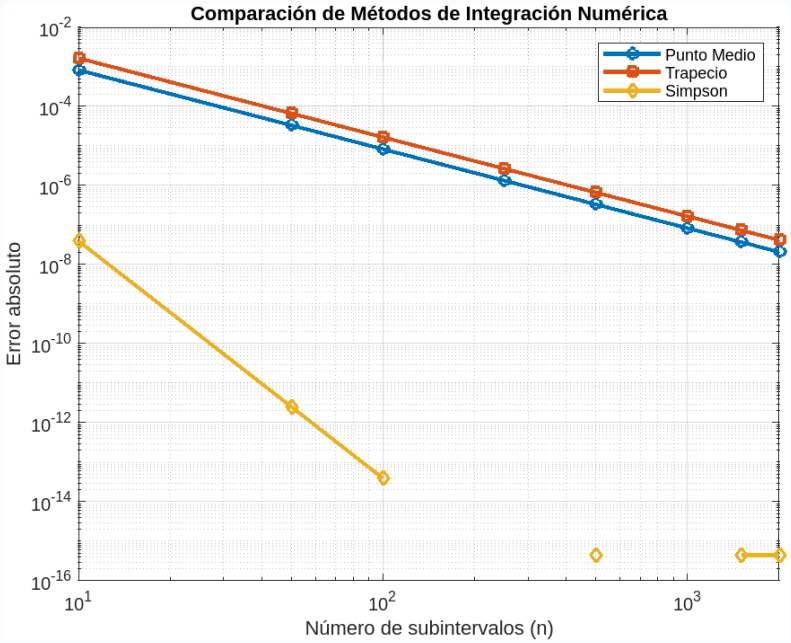

Figura 4: Comparación de los métodos.

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:

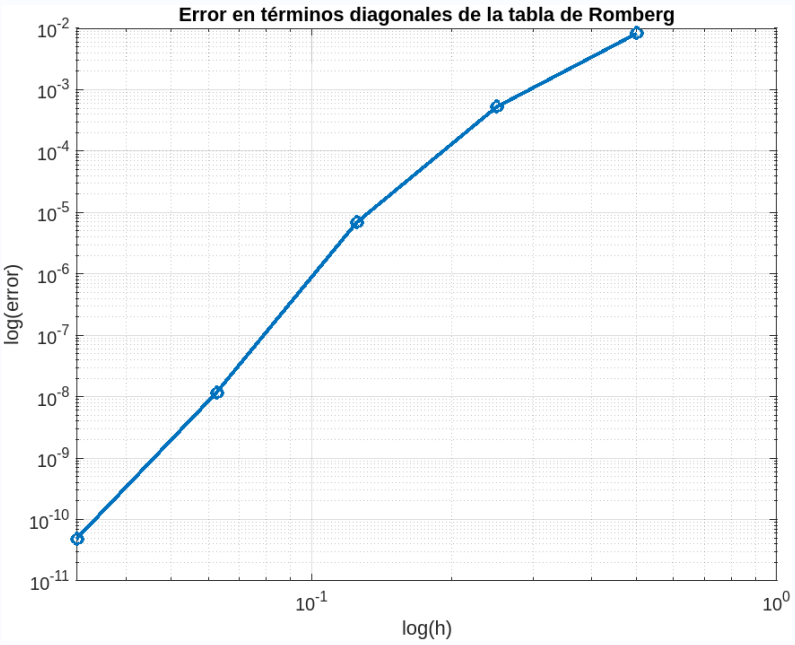

Figura 5: Error en los términos diagonales en la tabla de extrapolación versus $logh$.

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.

Problema 5 continúa en la siguiente página...

```
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));
```

```
43          % Extrapolación de Richardson
44          for m = 2:n+1
45              R(n+1,m) = R(n+1,m-1) + (R(n+1,m-1) - R(n,m-1)) /
                    (4^(m-1) - 1);
46          end
47          % Imprimir la fila
48          print_row(R(n+1, 1:n+1));
49          % Guardamos h y el error absoluto en la diagonal
50          hs = [hs, h];
51          errores = [errores, abs(R(n+1,n+1) - I_exacto)];
52          % Criterio de convergencia
53          if abs(R(n+1,n) - R(n+1,n+1)) < eps
54              I_romberg = R(n+1,n+1);
55              return;
56          end
57          n = n + 1;
58      end
59  end
60  function print_row(row)
61      fprintf('%11.8f ', row);
62      fprintf('\n');
63  end
```