

Resumen

En este trabajo, se explica el método de Newton Cotes para integración numérica, y la extrapolación de Richardson y Romberg para mejorar la precisión de la solución.

1. Introducción

La integración numérica son algoritmos que calculan el valor numérico de una integral definida, así, el problema básico, es resolver la integral en una dimensión [1]:

$$\int_a^b f(x)dx, \quad (1)$$

una ventaja de estos métodos, es que en algunas ocasiones la función no tiene una antiderivada explícita o esta no es fácil de obtener. Así, en este trabajo se desarrollan los métodos de Newton Cotes y extrapolaciones para resolver integrales de forma numérica.

2. Metodología

2.1. Newton Cotes

Los métodos numéricos de Newton Cotes, son métodos numéricos que ajustan las funciones a polinomios de grado n , dividiendo el intervalo de integración en n intervalos iguales $a \leq x_0 < x_1 < x_2 < \dots \leq b$. Así, el área de la integral se puede aproximar:

$$\int_a^b f(x)dx \approx \sum_{i=0}^n w_i f(x_i), \quad (2)$$

donde el término w_i es el peso que depende del grado del polinomio que se escogerá para ajustar la curva. Los intervalos tendrán tamaños de:

$$h = \frac{b-a}{n}, \quad (3)$$

$$x_i = a + ih. \quad (4)$$

En el Cuadro. 1 se muestran las fórmulas para distintos grados n del polinomio que se ajustará a la curva.

n	Nombre común	Fórmula
1	Regla del trapecio	$\frac{h}{2} [f(x_0) + f(x_1)]$
2	Regla Simpson 1/3	$\frac{h}{3} [f(x_0) + 4f(x_1) + f(x_2)]$
3	Regla Simpson 3/8	$\frac{3h}{8} [f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$
4	Regla de Boole	$\frac{2h}{45} [7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)]$
5	-	$\frac{45h}{288} [19f(x_0) + 75f(x_1) + 50f(x_2) + 50f(x_3) + 75f(x_4) + 19f(x_5)]$
6	-	$\frac{h}{140} [41f(x_0) + 216f(x_1) + 27f(x_2) + 272f(x_3) + 27f(x_4) + 216f(x_5) + 41f(x_6)]$

Cuadro 1: Fórmulas cerradas de Newton-Cotes.

Para mejorar la aproximación de la integral, en lugar de aplicar alguna de las reglas a toda el área de integración, se puede aplicar a cada subintervalo y sumarse las contribuciones, de esta forma, mientras más subdivisiones se hagan, el resultado se aproximará más al resultado teórico.

Algorithm 1 Newton-Cotes integración

Entrada: Función a integrar; límite inferior: double a; límite superior: double b; cantidad de subintervalos:

int cantIntervalos; grado del polinomio: n.

Salida: Solución aproximada de la integral

```
1: function FUNCTION(double x)
2:   Función a integrar.
3: function SOLVEINTEGRAL(int cantIntervalos, int n, double a, double b)
4:   Para cada n se escribirán sus coeficientes:
5:   double Coef[n+1].
6:   switch (n)
7:   case 1:
8:     Coef[0] = 1.0, Coef[1] = 1.0.
9:   ...

10: Función de Newton-Cotes para n = 1, ..., 6.
11: hspace = (b-a)/cantIntervalos; h = hspace/n; sumaTotal = 0, suma = 0.
12: Pesos para cada polinomio: newtonCoef.
13: newtonCoef[6] = {1/2, 1/3, 3/8, 2/45, 5/288, 1/140}.
14: for i = 0; i < cantIntervalos; i ++ do
15:   suma = 0.
16:   for j = 0; j < (n + 1); j ++ do
17:     x = a + j*h
18:     suma += coef[j]*function(x);
19:   sumaTotal += suma

20: return: newtonCoef[n-1]*h*sumaTotal;
```

2.2. Extrapolación de Richardson

Como se mencionó, el resultado que se obtiene con el método de Newton Cotes es una aproximación para la solución, y el resultado exacto tendría términos de error de mayor orden con respecto a h, es decir, para la fórmula del trapecio:

$$\int_a^b f(x)dx \approx \frac{h}{2}f(a) + \frac{h}{2}f(b), \quad (5)$$

$$= \frac{h}{2}f(a) + \frac{h}{2}f(b) + O(h^2), \quad (6)$$

el término $O(h^2)$ es una estimación del error, en este caso, su orden es h^2 . Así, si se quiere estimar el resultado de la integral y este depende del parámetro h, se esperaría que conforme el valor de h tienda a cero, esta converge al valor exacto de la integral. Si se asume que el resultado de la integral es continuo y diferenciable en h, se puede expandir en términos de series de Taylor:

$$q(h) = g + c_1h + c_2h^2 + \dots, \quad (7)$$

$$q\left(\frac{h}{2}\right) = g + c_1\frac{h}{2} + c_2\frac{h^2}{4} + \dots, \quad (8)$$

donde c_1, c_2, \dots, c_n son constantes y g es el resultado obtenido con el método de Newton Cotes, al multiplicar ec. (8) por 2, y restar el resultado de ec. (7), se obtiene:

$$\begin{aligned} g_1 &= 2q\left(\frac{h}{2}\right) - q(h), \\ &= g - \frac{1}{2}c_2h^2 - \frac{3}{4}c_3h^3 - \dots, \end{aligned}$$

de esta forma, se eliminó el término lineal con h, si se hace un procedimiento similar, se encuentra una generalización del resultado de la integral:

$$g_n = \frac{2^n g_{n-1}\left(\frac{h}{2}\right) - g_{n-1}(h)}{2^n - 1}. \quad (9)$$

Si se expresa el resultado en términos de una matriz, esta sería una matriz triangular inferior, de la forma:

$$\begin{array}{ccccccc}
 q_1(h) & & & & & & \\
 q_1(\frac{h}{2}) & q_2(h) & & & & & \\
 q_1(\frac{h}{4}) & q_2(\frac{h}{2}) & q_3(h) & & & & \\
 q_1(\frac{h}{8}) & q_2(\frac{h}{4}) & q_3(\frac{h}{2}) & q_4(h) & & & \\
 \vdots & \vdots & \vdots & \vdots & \ddots & & \\
 \uparrow & \uparrow & \uparrow & \uparrow & & & \\
 O(h) & O(h^2) & O(h^3) & O(h^4) & & &
 \end{array}$$

Algorithm 2 Extrapolación Richardson

Entrada: Mismo que algoritmo Newton Cotes

Salida: Resultado aproximado de la integral

- 1: Utilizar la función de Newton Cotes:
 - 2: **function** NEWTON-COTES(double a, double b, int n)

 - 3: Función para la extrapolación de Richardson:
 - 4: **function** EXTRARICHARDSON(int cantIntervalos, double a, double b, int n)
 - 5: int aprox = 4, // Hasta qué error
 - 6: Declarar matriz vacía de (4*2)(4*2): Laprox.
 - 7: **for** $i = 0; i < aprox * 2; i++$ **do**
 - 8: Laprox[i][0] = SolveIntegral(2^i , n, a, b)
 - 9: **for** $j = 1; j < aprox * 2; j++$ **do**
 - 10: **if** $i \geq j$ **then**
 - 11: $Laprox[i][j] = \frac{2^j Laprox[i][i-1] - Laprox[i-1][j-1]}{2^j - 1}$
 - 12: **return:** Matriz con resultados de aproximación.
-

2.3. Extrapolación de Romberg

La extrapolación de Romberg, es una extrapolación utilizando la regla del trapecio, también conocida como la regla compuesta del trapecio,

$$T(h) = \frac{h}{2} \left[f(a) + 2 \sum_{j=1}^{n-1} f(a + ij) + f(b) \right]. \quad (10)$$

Es un caso específico de la extrapolación de Richardson, y la fórmula cambia en términos de matrices:

$$Laprox(j, 0) = T(h/2^j), \quad (11)$$

$$Laprox(j, k) = \frac{4^k Laprox(j, k-1) - Laprox(j-1, k-1)}{4^k - 1}, \quad j \geq k > 0. \quad (12)$$

$$(13)$$

Algorithm 3 Extrapolación Romberg

Entrada: Mismo que algoritmo Newton Cotes para el caso de la regla del trapecio: $n = 1$.

Salida: Resultado aproximado de la integral

```
1: Utilizar la función de Newton Cotes:
2: function NEWTON-COTES(double a, double b, int n)

3: Función para la extrapolación de Richardson:
4: function EXTRARICHARDSON(int cantIntervalos, double a, double b, int n)
5:   int aprox = 4, // Hasta qué error
6:   Declarar matriz vacía de  $(4*2)(4*2)$ : Laprox.
7:   for  $i = 0; i < aprox * 2; i++$  do
8:     Laprox[i][0] = SolveIntegral( $2^i$ , n, a, b)
9:     for  $j = 1; j < aprox * 2; j++$  do
10:      if  $i \geq j$  then
11:        Laprox[i][j] =  $\frac{4^j \text{Laprox}[i][i-1] - \text{Laprox}[i-1][j-1]}{4^j - 1}$ 
12: return: Matriz con resultados de aproximación.
```

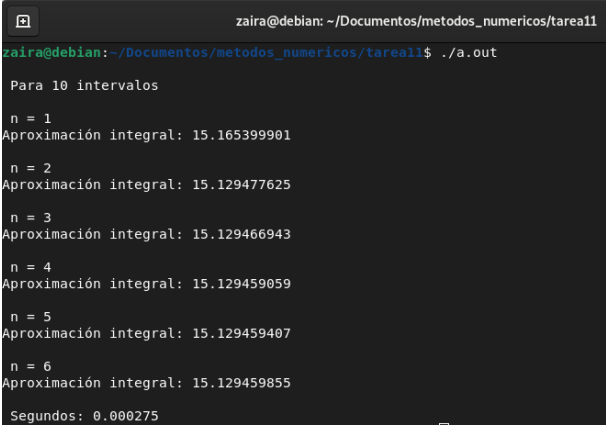
3. Resultados

3.1. Newton Cotes

Se calculó la integral:

$$\int_{-3}^3 \log(4x^2 + 4)dx = 6 \log(40) + 4 \arctan(3) - 12 \approx 15,129459814.$$

Y se obtuvieron los resultados:



```
zaira@debian: ~/Documentos/metodos_numericos/tarea11
zaira@debian:~/Documentos/metodos_numericos/tarea11$ ./a.out

Para 10 intervalos

n = 1
Aproximación integral: 15.165399901

n = 2
Aproximación integral: 15.129477625

n = 3
Aproximación integral: 15.129466943

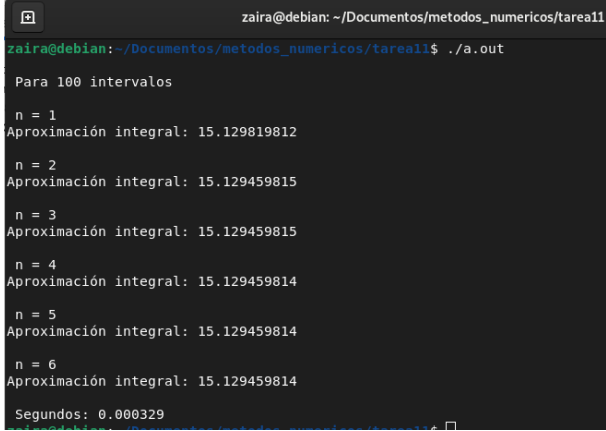
n = 4
Aproximación integral: 15.129459059

n = 5
Aproximación integral: 15.129459407

n = 6
Aproximación integral: 15.129459855

Segundos: 0.000275
```

Figura 1: Dividiendo cada intervalo en 10 partes.



```
zaira@debian: ~/Documentos/metodos_numericos/tarea11
zaira@debian:~/Documentos/metodos_numericos/tarea11$ ./a.out

Para 100 intervalos

n = 1
Aproximación integral: 15.129819812

n = 2
Aproximación integral: 15.129459815

n = 3
Aproximación integral: 15.129459815

n = 4
Aproximación integral: 15.129459814

n = 5
Aproximación integral: 15.129459814

n = 6
Aproximación integral: 15.129459814

Segundos: 0.000329
```

Figura 2: Dividiendo cada intervalo en 100 partes.

Así, conforme se aumente el grado del polinomio o aumentar la cantidad de subintervalos, el resultado será mejor.

3.2. Extrapolación de Richardson

Se utilizó la integral:

$$\int_0^{\pi} \sin(x)dx = 2,$$

y se encontró:

```
zaira@debian: ~/Documentos/metodos_numericos/tarea11
zaira@debian:~/Documentos/metodos_numericos/tarea11$ gcc richardson-extrapolation2.c -lm
zaira@debian:~/Documentos/metodos_numericos/tarea11$ ./a.out
0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
1.570796327 3.141592654 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
1.896118898 2.221441469 1.914724408 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
1.974231602 2.052344306 1.995978585 2.007586325 0.000000000 0.000000000 0.000000000 0.000000000
1.993570344 2.012909086 1.999764012 2.000304787 1.999819352 0.000000000 0.000000000 0.000000000
1.998393361 2.003216378 1.999985476 2.000017113 1.999997935 2.000003696 0.000000000 0.000000000
1.999598389 2.000803416 1.999999096 2.000001041 1.999999970 2.000000036 1.999999977 0.000000000
1.999899600 2.000200812 1.999999944 2.000000065 2.000000000 2.000000000 2.000000000 2.000000000
Segundos: 0.000198
```

Figura 3: Matriz de extrapolación de Richardson

3.3. Extrapolación de Romberg

Se utilizó la misma integral que el caso de la extrapolación de Richardson, y se encontró:

```
zaira@debian: ~/Documentos/metodos_numericos/tarea11
zaira@debian:~/Documentos/metodos_numericos/tarea11$ gcc rombert-method2.c -lm
zaira@debian:~/Documentos/metodos_numericos/tarea11$ ./a.out
0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
1.570796327 2.094395102 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
1.896118898 2.004559755 1.998570732 0.000000000 0.000000000 0.000000000 0.000000000 0.000000000
1.974231602 2.000269170 1.999983131 2.000005550 0.000000000 0.000000000 0.000000000 0.000000000
1.993570344 2.000016591 1.999999752 2.000000016 1.999999995 0.000000000 0.000000000 0.000000000
1.998393361 2.000001033 1.999999996 2.000000000 2.000000000 2.000000000 0.000000000 0.000000000
1.999598389 2.000000065 2.000000000 2.000000000 2.000000000 2.000000000 2.000000000 0.000000000
1.999899600 2.000000004 2.000000000 2.000000000 2.000000000 2.000000000 2.000000000 2.000000000
Segundos: 0.000185
```

Figura 4: Matriz de extrapolación de Richardson

4. Conclusiones

Como se observa, mientras más subintervalos se dividan al utilizar cualquier n de Newton Cotes, este se acercará más a la solución. En el caso de una división de 10 intervalos, los resultados numéricos tienen más diferencia pero a más puntos decimales. Asimismo, en el caso de la extrapolación de Romberg, esta convergió más rápido a la solución, cuando el método de Richardson lo hizo hasta la última extrapolación del error. Lo que es esperado porque uno tiene una potencia de 2 y el otro utiliza potencia de 4.

Referencias

- [1] R.L. Burden, J.D. Faires, and A.M. Burden. *Numerical Analysis*. Cengage Learning, 2015.