Tema 2 - Operaciones matemáticas básicas

Integración numérica

M. en C. Gustavo Contreras Mayén

24 de septiembre de 2013

Contenido

Contenido

Contenido

Contenido

Problema inicial

Calcular

$$\int_a^b f(x)dx$$

donde $f(x)$ es una función dada.

Introducción

La integración numérica (también conocida como cuadratura) es un procedimiento con mayor precisión que la diferenciación numérica.

La cuadratura aproxima la integral definida

$$\int_a^b f(x)dx$$

mediante la suma

$$I = \sum_{i=0}^{n} A_i f(x_i)$$

donde las *abscisas nodales* x_i y los pesos A_i dependen de una regla en particular usada para la cuadratura.

Todas las reglas de cuadratura se dividen en dos grupos:

1. Fórmulas de Newton-Cotes.
2. Fórmulas de Cuadraturas Gaussianas.

Fórmulas de Newton-Cotes

Estas fórmulas se caracterizan por usar un espaciamiento uniforme y constante en las abscisas, aquí se consideran los métodos del trapecio y la regla de Simpson.

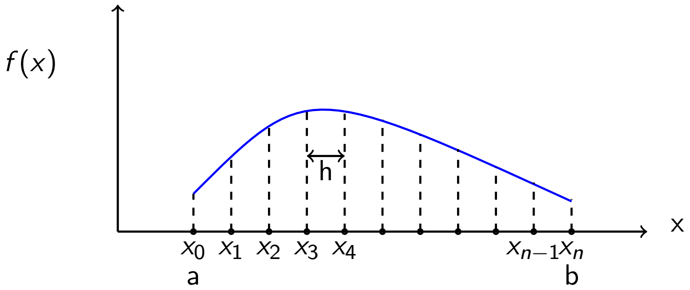
Son útiles si $f(x)$ se ha evaluado en intervalos iguales; dado que las fórmulas Newton-Cotes se basan en una interpolación local, se requiere de una porción para ajustarla al polinomio.

Consideremos la integral definida

$$\int_a^b f(x)dx$$

Dividimos el intervalo de integración $[a, b]$ en n intervalos de igual longitud $h = (b - a)/n$, y hacemos que las abscisas sean $x_0, x_1, \ldots, x_n$.

Aproximación polinomial de $f(x)$



Ahora aproximamos $f(x)$ con un polinomio de orden n que intersecta todos los nodos. La expresión para el polinomio de Lagrange es:

$$P_n(x) = \sum_{i=0}^{n} f(x_i)\mathcal{L}_i(x)$$

donde $\mathcal{L}_i(x)$ son las funciones definidas en el tema de interpolación.

Por tanto, un aproximación a la integral es

$$I = \int_a^b P_n(x)dx = \sum_{i=0}^{n} \left[f(x_i) \int_a^b \mathcal{L}_i(x)dx \right] = \sum_{i=0}^{n} A_i f(x_i)$$

donde

$$A_i = \int_a^b \mathcal{L}_i dx, \qquad i = 0, 1, \ldots, n$$

Las ecuaciones

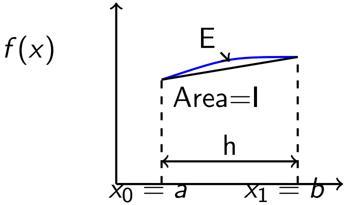
$$I = \int_a^b P_n(x)dx = \sum_{i=0}^{n} \left[f(x_i) \int_a^b \mathcal{L}_i(x)dx \right] = \sum_{i=0}^{n} A_i f(x_i)$$

se conocen como las fórmulas de Newton-Cotes.
Siendo los casos:

1. $n = 1$, Regla del trapecio.
2. $n = 2$, Regla de Simpson.
3. $n = 3$, Regla de Simpson de $3/8$.

La más importante es la regla del trapecio, ya que se puede combinar con la extrapolación de Richardson, en un algoritmo eficiente llamado: Integración de Romberg.

Regla del trapecio



Si $n = 1$ (un bloque), tenemos que
$l_0 = (x - x_1)/(x_0 - x_1) = (x - b)/h$ por tanto:

$$A_0 = \frac{1}{h}\int_a^b (x - b)dx = \frac{1}{2h}(b - a)^2 = \frac{h}{2}$$

Para $l_1 = (x - x_0)/(x_1 - x_0) = (x - a)/h$ tenemos

$$A_1 = \frac{1}{h} \int_a^b (x - a) dx = \frac{1}{2h} (b - a)^2 = \frac{h}{2}$$

Sustituyendo:

$$I = [f(a) + f(b)] \frac{h}{2}$$

Siendo la regla del trapecio. Representa el área del trapecio que se muestra en la figura anterior.

Error en la regla del trapecio

El error viene dado por

$$E = \int_a^b f(x) dx - I$$

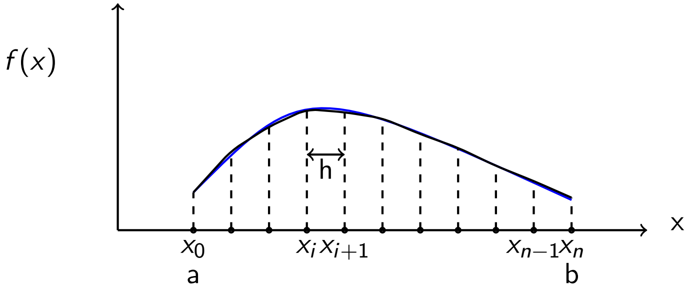
que es diferencia entre el área debajo de la curva de $f(x)$ y el la integral obtenida.

Integrando el error de interpolación:

$$
\begin{aligned}
E =& \frac{1}{2!} \int_a^b (x - x_0)(x - x_1) f''(\xi) dx \\
=& \frac{1}{2} f''(\xi) \int_a^b (x - a)(x - b) dx = \\
=& -\frac{1}{12}(b - a)^3 f''(\xi) \\
=& -\frac{h^3}{12} f''(\xi)
\end{aligned}
$$

Regla extendida del trapecio

En la práctica la regla del trapecio se usa con una división en el dominio. La siguiente figura muestra la región $[a, b]$ dividida en n bloques, de longitud h.



La función $f(x)$ se integrará con una aproximación lineal en cada panel. De la regla del trapecio, tenemos una que para el i-ésimo panel:

$$I_i = [f(x_i) + f(x_{i+1})]\frac{h}{2}$$

y como el área total, representada por la integral:

$$I \simeq \sum_{i=0}^{n-1}[f(x_0)+2f(x_1)+2f(x_2)+\ldots+2f(x_{n-1})+f(x_n)]\frac{h}{2}$$

que es la regla del extendida del trapecio.

Regla recursiva del trapecio

Sea I_k la integral evaluada con la regla compuesta del trapecio, usando 2^{k-1} bloques. Con la notación $H = b - a$,de la regla compuesta del trapecio, para $k = 1, 2, 3$

$$k = 1 \text{ (1 bloque) } :$$

$$I_1 = [f(a) + f(b)]\frac{H}{2}$$

$k = 2$ (2 bloques) :

$$\begin{aligned} I_2 &= \left[f(a) + 2f\left(a + \frac{H}{2}\right) + f(b) \right] \frac{H}{4} \\ &= \frac{1}{2} I_1 + f\left(a + \frac{H}{2}\right) \frac{H}{2} \end{aligned}$$

$$
\begin{aligned}
k =& 3 \text{ (4 bloques)} \quad : \\
I_3 =& \left[f(a) + 2f\left(a + \frac{H}{4}\right) + 2f\left(a + \frac{H}{2}\right) + \right. \\
& \left. + 2f\left(a + \frac{3H}{4}\right) + f(b) \right] \frac{H}{8} \\
=& \frac{1}{2} I_2 \left[f\left(a + \frac{H}{4}\right) + f\left(a + \frac{3H}{4}\right) \right] \frac{H}{4}
\end{aligned}
$$

Regla recursiva del trapecio

Para un $k > 1$ arbitrario, tenemos

$$I_k = \frac{1}{2} I_{k-1} + \frac{H}{2^{k-1}} \sum_{i=1}^{2^{k-2}} f\left[a + \frac{(2i-1)H}{2^{k-1}}\right], \qquad k = 2, 3, \ldots$$

Otra forma de la misma ecuación es:

$$I(h) = \frac{1}{2} I(2h) + h \sum f(x_{\text{nuevo}})$$

Ejercicio

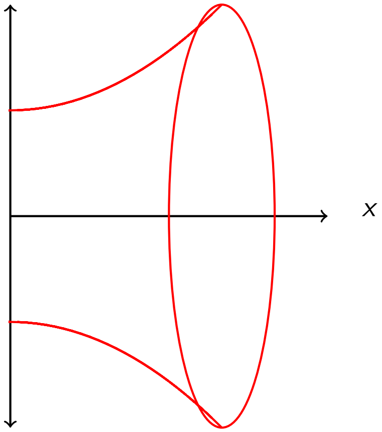
El cuerpo de revolución que se muestra en la figura, se obtiene al girar la curva dada por

$$y = 1 + \left(\frac{x}{2}\right)^2, \quad 0 \leq x \leq 2$$

en torno al eje x. Calcula el volumen, usando la regla extendida del trapecio con $N = 2, 4, 8, 16, 32, 64, 128$.

El valor exacto es $I = 11.7286$. Evalúa el error para cada N.

$$y = 1 + \left(\frac{x}{2}\right)^2$$



Resolviendo el problema

Hay que definir inicialmente la función que queremos integrar, por tanto

$$I = \int_{a}^{b} f(x) dx$$

donde

$$f(x) = \pi \left(1 + \left(\frac{x}{2} \right)^{2} \right)^{2}$$

```
def trapecios(f,a,b,n):
    h = (b-a)/float(n)
    x = a
    suma = 0
    for i in range(1,n):
        x = x + h
        suma = suma + funcion(x)
    return (h/2.)*(funcion(a) + funcion(b) + 2*
        suma)
```

i	Integral	Error
2	12.762720155	8.81708094e-02
4	11.989593838	2.22527700e-02
8	11.794011288	5.57707550e-03
16	11.744971839	1.39589033e-03
32	11.732702989	3.49827695e-04
64	11.729635215	8.82641387e-05
128	11.728868236	2.28702562e-05

Librería Scipy

SciPy es un conjunto de algoritmos matemáticos y funciones de conveniencia construidos en la extensión NumPy para Python.

Se agrega un poder significativo a la sesión interactiva de Python mediante la presentación del usuario a comandos de alto nivel y clases para la manipulación y visualización de datos.

Organización de Scipy

SciPy está organizada en sub-paquetes que cubren diferentes áreas de computación científica. Estos se resumen en la siguiente tabla:

Subpaquete	Descripción
cluster	Algortimos para clusters
constants	Constantes físicas y matemáticas
fftpack	Rutinas para la Transformada Rápida de Fourier
integrate	Integración y EDO
interpolate	Interpolación y uso de splines
io	Rutinas de entrada y salida

Subpaquete	Descripción
linalg	Algebra lineal
ndimage	Procesamiento N-dimensional de imagenes
odr	Regresión de distancias ortogonales
optimize	Optimizació y rutinas para encontrar raíces
signal	Procesamiento de señales
sparse	Matrices sparse y rutinas asociadas
spatial	Estructura de datos espaciales
special	Funciones especiales
stats	Distribuciones estadísticas
weave	Integración con C/C++

Integración (`scipy.integrate`)

El subpaquete `scipy.integrate` proporciona varias técnicas de integración.

quad	Integración en general.
dblquad	Integración doble en general.
tplquad	Integración triple en general.
fixed-quad	Integración de f(x) usando cuadraturas gaussianas de orden n.
quadrature	Integra con tolerancia dada usando cuadratura gaussiana.
romberg	Integra una función mediante la integración de Romberg.

Ejemplo del uso de `integrate.quad`

```
>>> from scipy import integrate
>>> x2 = lambda x: x**2
>>> integrate.quad(x2,0.,4.)
(21.333333333333336, 2.3684757858670001e-13)
```

El operador `lambda` sirve para crear funciones anónimas en línea. Al ser funciones anónimas, es decir, sin nombre, éstas no podrán ser referenciadas más tarde.

Las funciones `lambda` se construyen mediante el operador `lambda`, los parámetros de la función separados por comas (atención, SIN paréntesis), dos puntos (:) y el código de la función.

El problema del volumen con `integrate.quad`

Para comparar el resultado que nos da el módulo `scipy.integrate.quad`, veamos cómo implementar la solución del problema del sólido de revolución.

```
from numpy import pi
from scipy.integrate import quad

def f(x):
    return pi*(1+(x/2)**2)**2

print quad(f,0,2)
```

El resultado que nos devuelve es: $(11.728612573401893, 1.3021375725889e-13)$.

El valor posterior al resultado de la integral es el error asociado al algoritmo que usa `integrate.quad`, para que no lo reporte en el resultado, basta con indicar que queremos sólo el primer elemento de la lista:

```
print quad(f,0,2)[0]
```