CHƯƠNG 5: TÍNH GẦN ĐÚNG ĐẠO HÀM & TÍCH PHÂN

Tài liệu:

1. Giải Tích Số, Phạm Kỳ Anh
2. Elementary Numerical Analysis, Atkinson & Han
3. Numerical methods, Greenbaum & Chartier

Tác giả: TS. Hà Phi
Khoa Toán – Cơ - Tin học
ĐHKHTN, ĐHQGHN

PYTHON BUILT-IN FUNCTIONS: scipy.optimize.approx_fprime (1/2)

```
def approx_fprime(xk, f, epsilon=_epsilon, *args):
    """Finite difference approximation of the derivatives of a
    scalar or vector-valued function.

    If a function maps from :math:`R^n` to :math:`R^m`, its derivatives form
    an m-by-n matrix
    called the Jacobian, where an element :math:`(i, j)` is a partial
    derivative of f[i] with respect to ``xk[j]``.
```

Returns:	**jac** : *ndarray*	
		The partial derivatives of *f* to *xk*.

BÀI TOÁN 2. TÍNH GẦN ĐÚNG TÍCH PHÂN HỮU HẠN

$$I(f) := \int_a^b f(x)\,\mathrm{d}x$$

- Cách suy nghĩ thông thường: lập tổng Riemann (Darboux) rồi đi tìm $\lim_{n\to\infty} S_n(f; a, b)$

$$S_n(f;a,b) := \sum_{i=1}^{n} \Delta x_i f(\overline{x}_i), \quad \Delta x_i := x_i - x_{i-1}, \quad \overline{x}_i \in [x_{i-1}, x_i], \quad i = 1, 2, \ldots, n,$$

- Ý tưởng: đi tìm xấp xỉ tích phân dạng

$$\int_a^b f(x)dx \approx \sum_{k=0}^{n} f(x_k) w_k$$

- Công thức này rất có ý nghĩa khi mà f cho trước, n có thể cho trước (nhỏ hơn n trong tổng Darboux nhiều), và có thể hàm f chỉ biết giá trị tại các điểm x_k , k = 0,…,n.
- Ở đây chúng ta gọi w_k là các trọng số, x_k là các điểm nút.
- Công thức cầu phương có 2n+2 tham số bao gồm n+1 trọng số, n+1 nút.
- Việc chọn trọng số hay chọn nút sẽ dẫn đến 2 lớp phương pháp quan trọng: Newton-Cotes và Gauss

MỘT SỐ QUY TẮC CẦU PHƯƠNG ĐƠN GIẢN

- Các quy tắc 1 phía (n=0)

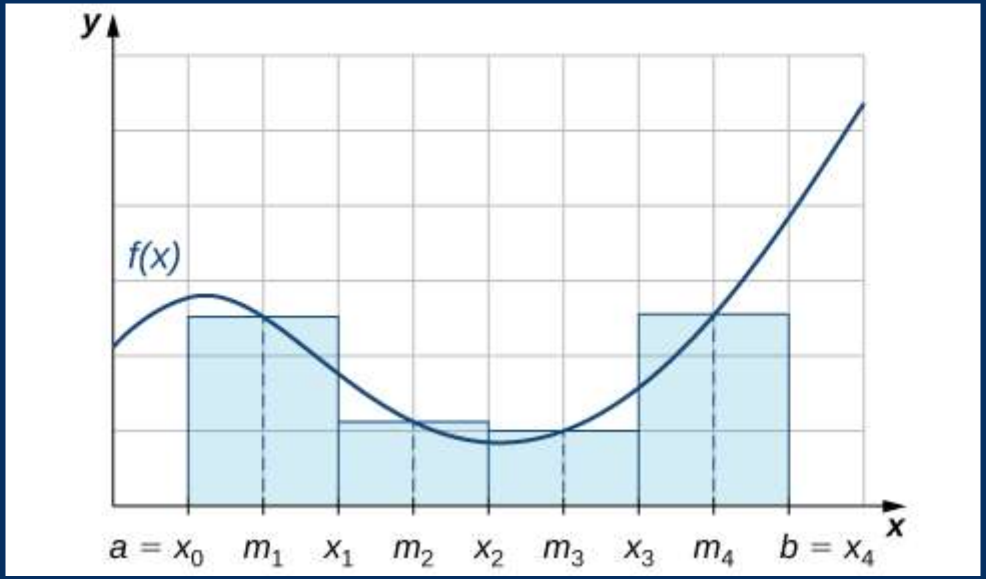
$$\int_a^b f(x)\,\mathrm{d}x \approx I_1(f) := (b-a)f(a), \text{ and } \int_a^b f(x)\,\mathrm{d}x \approx I_1(f) := (b-a)f(b)$$

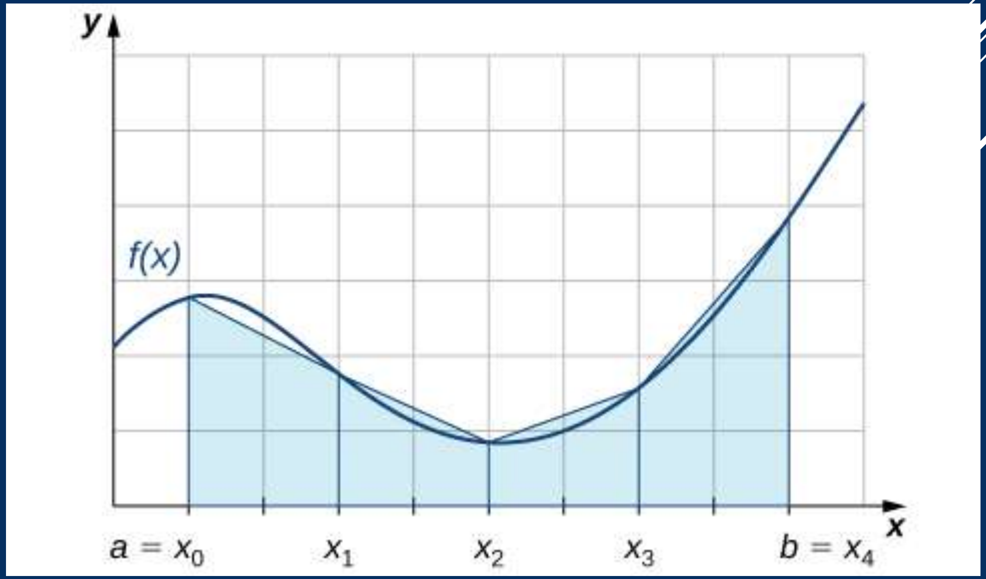
- Quy tắc trung điểm/hình chữ nhật (midpoint rule) (n=1), hình bên trái

$$\int_a^b f(x)\,\mathrm{d}x \approx I_1(f) := (b-a)f\left(\frac{a+b}{2}\right)$$

- Quy tắc hình thang (trapezoidal rule) (n=2), hình bên phải

$$\int_a^b f(x)\,\mathrm{d}x \approx I_2(f) := \frac{b-a}{2}[f(a)+f(b)].$$





CẤP CHÍNH XÁC CỦA MỘT SỐ QUY TẮC CẦU PHƯƠNG ĐƠN GIẢN

- Các quy tắc 1 phía (n=1) $\int_a^b f(x)\,\mathrm{d}x \approx I_1(f) := (b-a)f(a)$, and $\int_a^b f(x)\,\mathrm{d}x \approx I_1(f) := (b-a)f(b)$
- Quy tắc trung điểm/hình chữ nhật (midpoint rule) (n=1) $\int_a^b f(x)\,\mathrm{d}x \approx I_1(f) := (b-a)f\left(\frac{a+b}{2}\right)$
- Quy tắc hình thang (trapezoidal rule) (n=2) $\int_a^b f(x)\,\mathrm{d}x \approx I_2(f) := \frac{b-a}{2}[f(a)+f(b)].$

- Các quy tắc 1 phía chỉ chính xác cho các hàm hằng.
- Các quy tắc trung điểm và hình thang chính xác cho các hàm tuyến tính (y=ax+b).

CÁC PHƯƠNG PHÁP NEWTON-COTES

- Dựa trên nội suy Lagrange $f(x) \approx p_n(x) = \sum_{k=0}^{n} f(x_k)\ell_k(x)$ ta xấp xỉ $f = 1, x, x^2, \ldots, x^n$

  trong đó $w_k = \int_a^b \ell_k(x)dx$, các điểm nút x_k được lấy theo lưới đều.

- Việc tính toán trực tiếp các trọng số w_k (sử dụng công thức nội suy) rất không tốt.
- Phương pháp hệ số bất định: coi các w_k là các hệ số chưa biết.
- Chú ý: phương pháp Newton-Cotes chính xác với mọi đa thức bậc $\leq n$.
- Cho $f = 1,\ x,\ x^2, \ldots, x^n$ ta được hệ n+1 phương trình n+1 ẩn dạng Vandemonde. Giải hpt này

  ta tìm được các w_k , và có công thức xấp xỉ tích phân.

Các công thức Newton-Cotes cơ bản

Tên công thức	n	formula
Hình thang	1	$\frac{(b-a)}{2}[f(a)+f(b)]$
Simpson 1/3	2	$\frac{(b-a)}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$
Simpson 3/8	3	$\frac{(b-a)}{8}[f(a)+3f(a+h)+3f(b-h)+f(b)]$
Boole	4	$\frac{(b-a)}{90}\left[7f(a)+32f(a+h)+12f\left(\frac{a+b}{2}\right)+32f(b-h)+7f(b)\right]$

ĐÁNH GIÁ SAI SỐ

Dựa trên ước lượng sai số của phép nội suy Lagrange

Theorem

Given function f with $n+1$ continuous derivatives in the interval formed by $I = [min(\{x, x_0, \ldots, x_n\}), max(\{x, x_0, \ldots, x_n\})]$. If $p(x)$ is the unique interpolating polynomial of degree $\leqslant n$ with,

$$p(x_i) = f(x_i), \quad i = 0, 1, \ldots, n$$

then the error is computed by the formula,

$$p(x) - f(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\ldots(x-x_n), \quad \text{for some } \xi(x) \in I$$

- Trapezoid Rule:

$$\int_{x_0}^{x_1} f(x)\,dx \approx \int_{x_0}^{x_1} P_1(x)\,dx = \frac{1}{2}(f(x_0) + f(x_1))h$$

$$\int_{x_0}^{x_1} f(x)\,dx \approx \frac{1}{2}(f(x_0) + f(x_1))h \text{ , where } f(x) = 15\,x^2$$

Example

$$\int_1^2 15\,x^2 \approx \frac{1}{2}(15 * 1^2 + 15 * 2^2) * 1$$

$$= \frac{1}{2}(15 + 60) = 37.5$$

- Analytical answer is $\int_1^2 15\,x^2 = 5\,x^3\Big|_1^2 = 40 - 5 = 35.$

Newton-Cotes,Exact Error Bounds

The error,

$$error = \int_a^b f(x)dx - \text{ approximate formula}$$

for the various rules is given by the following table

(basic) Newton-Cotes rules:

name of formula	n	error
Trapezoid	1	$-\frac{(b-a)^3}{12}f^{(2)}(\xi)$
Simpson's $1/3$	2	$-\frac{(b-a)^5}{2880}f^{(4)}(\xi)$
Simpson's $3/8$	3	$-\frac{(b-a)^5}{6480}f^{(4)}(\xi)$
Boole's	4	$-\frac{(b-a)^7}{1935360}f^{(6)}(\xi)$

Trapezoid, Error Bound

For the Trapezoidal Rule we have,

$$\begin{aligned}
error &= \left| \int_a^b p_1(x) - f(x)dx \right| \\
&= \left| \int_a^b \frac{f^{(2)}(\xi(x))}{2!}(x-a)(x-b)dx \right| \\
&\leqslant \frac{M}{2}\int_a^b |(x-a)(x-b)|\, dx \text{ where } |f''(x)| \leqslant M \text{ for } x \in [a,b] \\
&= \frac{M}{12}(b-a)^3
\end{aligned}$$

If $b-a \ll 1$ we denote $h = b-a$ then our error bound is $O(h^3)$.
Note: If $f(x)$ is a linear function then $f''(x) = 0$ for all $x \in [a,b]$ and then $M = 0$ and our error bound is exact.
What if $h = b-a$ is large? Use a higher degree interpolating polynomial? Is there an alternative?

THỰC TẾ ỨNG DỤNG: CÁC CÔNG THỨC COMPOSITE

- Cần chia nhỏ đoạn [a,b] thành n đoạn đều nhau để xấp xỉ tích phân trên từng đoạn nhỏ.

$$a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b, \;\; x_i - x_{i-1} = h, \;\; h = \frac{b-a}{n}.$$

- Quy tắc trung điểm

$$\int_a^b f(x)dx \approx h\left[f\left(\frac{x_0+x_1}{2}\right) + f\left(\frac{x_1+x_2}{2}\right) + \ldots + f\left(\frac{x_{n-1}+x_n}{2}\right)\right]$$

Đánh giá sai số toàn phần $M\dfrac{(b-a)}{12}h^2$ trong đó $M = \sup\limits_{[a,b]} |f''(x)|$

- Quy tắc hình thang $\displaystyle\int_a^b f(x)dx \approx \frac{h}{2}\left[f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n)\right]$

Đánh giá sai số toàn phần $M\dfrac{(b-a)}{12}h^2$ trong đó $M = \sup\limits_{[a,b]} |f''(x)|$

- Quy tắc Simpson 1/3 $\displaystyle\int_a^b f(x)dx \approx \frac{h}{6}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 4f(x_{2n-1}) + f(x_{2n})\right]$

Đánh giá sai số toàn phần $M\dfrac{(b-a)}{180}h^4$ trong đó $M = \sup\limits_{[a,b]} |f^{(4)}(x)|$

Example

How many points should be used to ensure the composite Trapezoid rule is accurate to 10^{-6} for $\int_0^1 e^{-x^2}\,dx$? Need

$$\frac{|f''(\eta)|}{12}(b-a)h^2 \leqslant 10^{-6}$$

How big is $f''(x)$?

$$\begin{aligned} f(x) &= e^{-x^2} \\ f'(x) &= -2xe^{-x^2} \\ f''(x) &= -2e^{-x^2} + 4x^2e^{-x^2} \\ f'''(x) &= 12xe^{-x^2} - 8x^3e^{-x^2} \end{aligned}$$

So f''' is always positive for $x > 0$. So f'' is monotone increasing and thus $|f''|$ takes on a maximum at an endpoint: $|f''(0)| = 2$ and $|f''(1)| = \frac{2}{e}$. Then bound

$$\frac{(b-a)2h^2}{12} \leqslant 10^{-6}$$

Or

$$h^2 \leqslant 6 \times 10^{-6} \quad \Rightarrow \quad \sqrt{(1/6)}10^3 \leqslant n$$

or $n + 1 >= 410$.

Đánh giá sai số cho công thức Newton-Cotes composite

The exact error,

$$error = \int_a^b f(x)\,dx - \text{ approximate formula}$$

for the various rules is given by the following table,

name of formula	error
Trapezoid	$-\frac{(b-a)h^2}{12} f''(\xi)$
Simpson's 1/3	$-\frac{(b-a)h^4}{180} f^{(4)}(\xi)$
Simpson's 3/8	$-\frac{(b-a)h^4}{80} f^{(4)}(\xi)$
Boole's	$-\frac{2(b-a)h^6}{945} f^{(6)}(\xi)$

where $h = \frac{(b-a)}{n}$ and n is the number of intervals of the partition of $[a, b]$.

BÀI TẬP

Bài 1

Hãy tính gần đúng tích phân

$$\int_0^{0.8} f(x)dx$$

với $f(x)$ lần lượt là $\sin x, \cos x, \exp(-x), \ln(1+x), 1/(1+x)$ bằng

a, công thức hình thang, $h = 0.2$ và $h = 0.1$. Hãy đánh giá sai số công thức. (các giá trị hàm số được qui tròn đến 6 chữ số sau dấu phẩy)

b, công thức Simpson, $h = 0.2$ và $h = 0.1$. Hãy đánh giá sai số công thức. (các giá trị hàm số được qui tròn đến 6 chữ số sau dấu phẩy)

c, hãy ước lượng độ lớn của h để sai số của CT hình thang và CT Simpson nhỏ hơn 10^{-3}, 10^{-6}, 10^{-8}.

Bài 2

Hãy xây dựng công thức Newton-Cotes để xấp xỉ $\int_0^1 f(x)dx$ sử dụng các nút $0, \frac{1}{3}, \frac{2}{3}$ và 1.

PYTHON BUILT-IN FUNCTIONS: scipy.integrate.newton_cotes (1/2)

`scipy.integrate.newton_cotes(rn, equal=0)` [source]

Return weights and error coefficient for Newton-Cotes integration.

Suppose we have (N+1) samples of f at the positions x_0, x_1, ..., x_N. Then an N-point Newton-Cotes formula for the integral between x_0 and x_N is:

$$\int_{x_0}^{x_N} f(x)dx = \Delta x \sum_{i=0}^{N} a_i f(x_i) + B_N (\Delta x)^{N+2} f^{N+1}(\xi)$$

where $\xi \in [x_0, x_N]$ and $\Delta x = \frac{x_N - x_0}{N}$ is the average samples spacing.

If the samples are equally-spaced and N is even, then the error term is $B_N (\Delta x)^{N+3} f^{N+2}(\xi)$.

Parameters: **rn** : *int*

The integer order for equally-spaced data or the relative positions of the samples with the first sample at 0 and the last at N, where N+1 is the length of *rn*. N is the order of the Newton-Cotes integration.

equal : *int, optional*

Set to 1 to enforce equally spaced data.

Returns: **an** : *ndarray*

1-D array of weights to apply to the function at the provided sample positions.

B : *float*

Error coefficient.

PYTHON BUILT-IN FUNCTIONS: scipy.integrate.newton_cotes (2/2)

Compute the integral of sin(x) in $[0, \pi]$:

```
>>> from scipy.integrate import newton_cotes
>>> def f(x):
...     return np.sin(x)
>>> a = 0
>>> b = np.pi
>>> exact = 2
>>> for N in [2, 4, 6, 8, 10]:
...     x = np.linspace(a, b, N + 1)
...     an, B = newton_cotes(N, 1)
...     dx = (b - a) / N
...     quad = dx * np.sum(an * f(x))
...     error = abs(quad - exact)
...     print('{:2d}  {:10.9f}  {:.5e}'.format(N, quad, error))
...
 2   2.094395102   9.43951e-02
 4   1.998570732   1.42927e-03
 6   2.000017814   1.78136e-05
 8   1.999999835   1.64725e-07
10   2.000000001   1.14677e-09
```

PYTHON BUILT-IN FUNCTIONS: scipy.integrate.trapezoid

`scipy.integrate.trapezoid(y, x=None, dx=1.0, axis=- 1)`

Integrate along the given axis using the composite trapezoidal rule.

If *x* is provided, the integration happens in sequence along its elements - they are not sorted.

Integrate *y* (*x*) along each 1d slice on the given axis, compute $\int y(x)dx$. When *x* is specified, this integrates along the parametric curve, computing $\int_t y(t)dt = \int_t y(t) \frac{dx}{dt}\Big|_{x=x(t)} dt$.

```
>>> np.trapz([1,2,3])
4.0
>>> np.trapz([1,2,3], x=[4,6,8])
8.0
>>> np.trapz([1,2,3], dx=2)
8.0
```

PYTHON BUILT-IN FUNCTIONS: scipy.integrate.simpson

```
def simpson(y, x=None, dx=1.0, axis=-1, even='avg'):
    """
    Integrate y(x) using samples along the given axis and the composite
    Simpson's rule. If x is None, spacing of dx is assumed.

    If there are an even number of samples, N, then there are an odd
    number of intervals (N-1), but Simpson's rule requires an even number
    of intervals. The parameter 'even' controls how this is handled.
```

```
>>> from scipy import integrate
>>> x = np.arange(0, 10)
>>> y = np.arange(0, 10)
```

```
>>> integrate.simpson(y, x)
40.5
```

```
>>> y = np.power(x, 3)
>>> integrate.simpson(y, x)
1642.5
>>> integrate.quad(lambda x: x**3, 0, 9)[0]
1640.25
```

PYTHON BUILT-IN FUNCTIONS: scipy.integrate.quad (1/2)

```
def quad(func, a, b, args=(), full_output=0, epsabs=1.49e-8, epsrel=1.49e-8,
         limit=50, points=None, weight=None, wvar=None, wopts=None, maxp1=50,
         limlst=50):
    """
    Compute a definite integral.

    Integrate func from `a` to `b` (possibly infinite interval) using a
    technique from the Fortran library QUADPACK.
```

PYTHON BUILT-IN FUNCTIONS: scipy.integrate.quad (2/2)

If the function to integrate takes additional parameters, they can be provided in the *args* argument. Suppose that the following integral shall be calculated:

$$I(a,b) = \int_0^1 ax^2 + b\,dx.$$

This integral can be evaluated by using the following code:

```
>>> from scipy.integrate import quad
>>> def integrand(x, a, b):
...     return a*x**2 + b
...
>>> a = 2
>>> b = 1
>>> I = quad(integrand, 0, 1, args=(a,b))
>>> I
(1.6666666666666667, 1.8503717077085944e-14)
```

```
>>> from scipy.integrate import quad
>>> def integrand(t, n, x):
...     return np.exp(-x*t) / t**n
...
```

```
>>> def expint(n, x):
...     return quad(integrand, 1, np.inf, args=(n, x))[0]
...
```

CÁC PHƯƠNG PHÁP CẦU PHƯƠNG GAUSS

- Ý tưởng: đi tìm xấp xỉ tích phân dạng $\int_a^b f(x)dx \approx \sum_{k=0}^{n} f(x_k)w_k$
- Ở đây chúng ta gọi w_k là các trọng số, x_k là các điểm nút.
- Công thức cầu phương có 2n+2 tham số bao gồm n+1 trọng số, n+1 nút.
- Như vậy có 2n+2 ẩn số, ta cần 2n+2 phương trình => ta chọn các đa thức có bậc $\leq 2n+1$
- Thay lần lượt $f(x) = 1,\ x,\ x^2, \ldots, x^{2n+1}$ vào ta được hệ phương trình.
- Câu hỏi: Vậy phương pháp Gauss khác Newton-Cotes ở chỗ nào?

➢ Trả lời: 1. Newton-Cotes lưới các nút là đều, hệ phương trình là tuyến tính

2. Gauss lưới các nút là không đều, hệ phương trình phi tuyến

3. Gauss có nhiều tham số để lựa chọn hơn, nên cấp chính xác tốt hơn.

Ví dụ: Cầu phương Gauss với n = 1. Liệu có tốt hơn quy tắc hình thang?

Again, we are considering $[a, b] = [-1, 1]$ for simplicity:

$$\int_{-1}^{1} f(x)\, dx \approx w_0 f(x_0) + w_1 f(x_1)$$

Goal: find w_0, w_1, x_0, x_1 so that the approximation is exact up to cubics. So try any cubic:

$$f(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$$

This implies that:

$$\begin{aligned}
\int_{-1}^{1} f(x)\, dx &= \int_{-1}^{1} \left(c_0 + c_1 x + c_2 x^2 + c_3 x^3\right)\, dx \\
&= w_0 \left(c_0 + c_1 x_0 + c_2 x_0^2 + c_3 x_0^3\right) + \\
&\quad\; w_1 \left(c_0 + c_1 x_1 + c_2 x_1^2 + c_3 x_1^3\right)
\end{aligned}$$

Ví dụ: Cầu phương Gauss với n = 1. Liệu có tốt hơn quy tắc hình thang?

$$\begin{aligned}\int_{-1}^{1} f(x)\,dx &= \int_{-1}^{1}\left(c_0 + c_1x + c_2x^2 + c_3x^3\right)\,dx \\ &= w_0\left(c_0 + c_1x_0 + c_2x_0^2 + c_3x_0^3\right) + \\ &\quad\; w_1\left(c_0 + c_1x_1 + c_2x_1^2 + c_3x_1^3\right)\end{aligned}$$

Rearrange into constant, linear, quadratic, and cubic terms:

$$c_0\left(w_0 + w_1 - \int_{-1}^{1} dx\right) + c_1\left(w_0x_0 + w_1x_1 - \int_{-1}^{1} x\,dx\right) +$$
$$c_2\left(w_0x_0^2 + w_1x_1^2 - \int_{-1}^{1} x^2\,dx\right) + c_3\left(w_0x_0^3 + w_1x_1^3 - \int_{-1}^{1} x^3\,dx\right) = 0$$

Since c_0, c_1, c_2 and c_3 are arbitrary, then their coefficients must all be zero.

Ví dụ: Cầu phương Gauss với n = 1. Liệu có tốt hơn quy tắc hình thang?

This implies:

$$w_0 + w_1 = \int_{-1}^{1} dx = 2 \qquad w_0 x_0 + w_1 x_1 = \int_{-1}^{1} x\, dx = 0$$

$$w_0 x_0^2 + w_1 x_1^2 = \int_{-1}^{1} x^2\, dx = \frac{2}{3} \qquad w_0 x_0^3 + w_1 x_1^3 = \int_{-1}^{1} x^3\, dx = 0$$

Some algebra leads to:

$$w_0 = 1 \quad w_1 = 1 \quad x_0 = -\frac{\sqrt{3}}{3} \quad x_1 = \frac{\sqrt{3}}{3}$$

Therefore:

$$\int_{-1}^{1} f(x)\, dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$$

Trường hợp đoạn [a.b] tổng quát

$$\int_{-1}^{1} f(x)\,dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$$

$$\int_{a}^{b} f(x)\,dx \approx ?$$

- integrating over $[a, b]$ instead of $[-1, 1]$ needs a transformation: a change of variables
- want $t = c_1 x + c_0$ with $t = -1$ at $x = a$ and $t = 1$ at $x = b$
- let $t = \frac{2}{b-a}x - \frac{b+a}{b-a}$
- (verify)
- let $x = \frac{b-a}{2}t + \frac{b+a}{2}$
- then $dx = \frac{b-a}{2}dt$

Trường hợp đoạn [a.b] tổng quát

$$\int_a^b f(x)\,dx \approx ?$$

- let $x = \frac{b-a}{2}t + \frac{b+a}{2}$
- then $dx = \frac{b-a}{2}dt$

$$\int_a^b f(x)\,dx = \int_{-1}^{1} f\left(\frac{(b-a)t+b+a}{2}\right)\frac{b-a}{2}\,dt$$

- now use the quadrature formula over $[-1, 1]$
- note: using two points, $n = 1$, gave us exact integration for polynomials of degree less 2*1+1 = 3 and less.

Ý tưởng đột phá của Gauss: sử dụng các đa thức trực giao.

Karl Friedrich Gauss proved the following result:
Let $q(x)$ be a nontrivial polynomial of degree $n+1$ such that

$$\int_a^b x^k q(x)dx = 0 \qquad (0 \leqslant k \leqslant n)$$

and let $x_0, x_1, \ldots, x_n$ be the zeros of $q(x)$. If $\ell_i(x)$ is the i-th Lagrange basis function based on the nodes $x_0, x_1, \ldots, x_n$ then,

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} A_i f(x_i), \text{ where } A_i = \int_a^b \ell_i(x)dx$$

will be exact for all polynomials of degree at most $2n+1$. (Wow!)

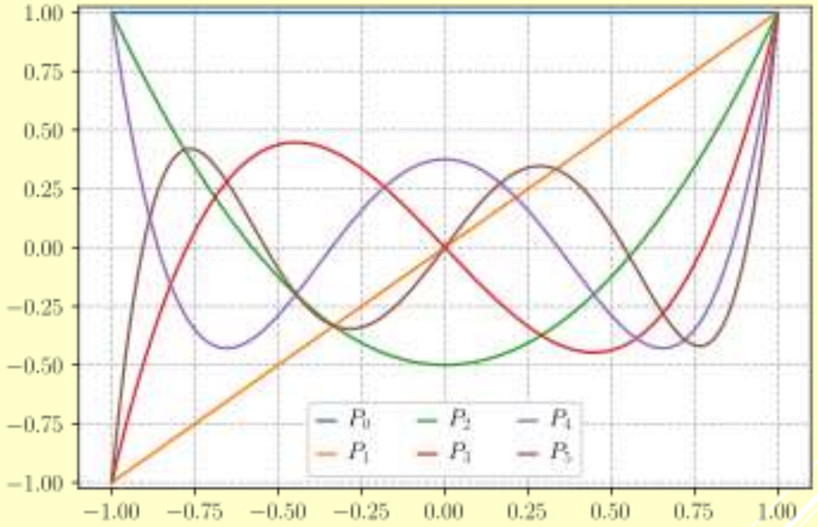
ĐA THỨC TRỰC GIAO

Orthogonality of Functions

Two functions $g(x)$ and $h(x)$ are *orthogonal* on $[-1, 1]$ if

$$\int_{-1}^{1} g(x)h(x)\,dx = 0$$

- so the nodes we're using are roots of orthogonal polynomials
- these are the *Legendre* Polynomials



$$\phi_0 = 1$$

$$\phi_1 = x$$

$$\phi_2 = \frac{3x^2 - 1}{2}$$

$$\phi_3 = \frac{5x^3 - 3x}{2}$$

In general:

$$\phi_n(x) = \frac{2n-1}{n} x\phi_{n-1}(x) - \frac{n-1}{n}\phi_{n-2}(x)$$

Ý tưởng đột phá của Gauss: sử dụng các đa thức trực giao.

Theorem

Suppose that $x_0, x_1, \ldots, x_n$ are roots of the nth Legendre polynomial $\phi_{n+1}(x)$ and that for each $i = 0, 1, \ldots, n$ the numbers w_i are defined by

$$w_i = \int_{-1}^{1} \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} dx = \int_{-1}^{1} \ell_i(x)\, dx$$

Then

$$\int_{-1}^{1} f(x) dx = \sum_{i=0}^{n} w_i f(x_i),$$

where $f(x)$ is any polynomial of degree less or equal to $2n + 1$.

Chú ý: dùng trọng số và node theo bảng – đừng tính lại làm gì

Number of points, n	Points, x_i		Weights, w_i	
1	0		2	
2	$\pm\frac{1}{\sqrt{3}}$	±0.57735...	1	
3	0		$\frac{8}{9}$	0.888889...
	$\pm\sqrt{\frac{3}{5}}$	±0.774597...	$\frac{5}{9}$	0.555556...
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.339981...	$\frac{18+\sqrt{30}}{36}$	0.652145...
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	±0.861136...	$\frac{18-\sqrt{30}}{36}$	0.347855...
5	0		$\frac{128}{225}$	0.568889...
	$\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$	±0.538469...	$\frac{322+13\sqrt{70}}{900}$	0.478629...
	$\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	±0.90618...	$\frac{322-13\sqrt{70}}{900}$	0.236927...

Ví dụ

Approximate $\int_1^{1.5} x^2 \ln x \, dx = 0.192259357732796$ using Gaussian quadrature with $n = 1$.

<u>SOLUTION</u> As derived earlier we want to use $\int_{-1}^{1} f(x)\, dx \approx f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$

From earlier we know that we are interested in

$$\int_1^{1.5} f(x)\, dx = \int_{-1}^{1} f\left(\frac{(1.5-1)t + (1.5+1)}{2}\right) \frac{1.5-1}{2}\, dt$$

Therefore, we are looking for the integral of

$$\frac{1}{4}\int_{-1}^{1} f\left(\frac{x+5}{4}\right) dx = \frac{1}{4}\int_{-1}^{1} \left(\frac{x+5}{4}\right)^2 \ln\left(\frac{x+5}{4}\right) dx$$

Using Gaussian quadrature, our numerical integration becomes:

$$\frac{1}{4}\left[\left(\frac{-\frac{\sqrt{3}}{3}+5}{4}\right)^2 \ln\left(\frac{-\frac{\sqrt{3}}{3}+5}{4}\right) + \left(\frac{\frac{\sqrt{3}}{3}+5}{4}\right)^2 \ln\left(\frac{\frac{\sqrt{3}}{3}+5}{4}\right)\right] = 0.1922687$$

BÀI TẬP

Bài 1

Xét công thức xấp xỉ

$$\int_{-1}^{1} f(x)dx \approx f(\alpha) + f(-\alpha).$$

a, Xác định α sao cho công thức là chính xác với mọi đa thức bậc bằng hoặc nhỏ hơn 1.

b, Xác định α sao cho công thức là chính xác với mọi đa thức bậc bằng hoặc nhỏ hơn 3.

c, Xác định α sao cho công thức là chính xác với mọi đa thức bậc bằng hoặc nhỏ hơn 4.

Bài 2

a, Hãy xác định công thức xấp xỉ

$$\int_{0}^{1} f(x)dx \approx A_0 f(x_0) + A_1 f(x_1),$$

sao cho công thức là chính xác với mọi đa thức bậc bằng hoặc nhỏ hơn 3.

b, Hãy xác định công thức xấp xỉ

$$\int_{0}^{1} x f(x)dx \approx A_0 f(x_0) + A_1 f(x_1),$$

sao cho công thức là chính xác với mọi đa thức bậc bằng hoặc nhỏ hơn 3.

PYTHON BUILT-IN FUNCTIONS: scipy.integrate.fixed_quad

```
def fixed_quad(func, a, b, args=(), n=5):
    """

    Compute a definite integral using fixed-order Gaussian quadrature.


    Integrate `func` from `a` to `b` using Gaussian quadrature of
    order `n`.
```

```
>>> from scipy import integrate
>>> f = lambda x: x**8
>>> integrate.fixed_quad(f, 0.0, 1.0, n=4)
(0.1110884353741496, None)
>>> integrate.fixed_quad(f, 0.0, 1.0, n=5)
(0.1111111111111102, None)
>>> print(1/9.0)  # analytical result
0.1111111111111111
```

NHỮNG GÌ CHƯA ĐỀ CẬP TRONG SÁCH (MÀ CÓ THỂ GẶP TRONG THỰC TẾ)

- Cầu phương tích phân bội
- Phương pháp Monte-Carlo tính tích phân bội

TÍCH PHÂN BỘI 2: scipy.integrate.dblquad

```
def dblquad(func, a, b, gfun, hfun, args=(), epsabs=1.49e-8, epsrel=1.49e-8):
    """

    Compute a double integral.


    Return the double (definite) integral of ``func(y, x)`` from ``x = a..b``
    and ``y = gfun(x)..hfun(x)``.
```

As example for non-constant limits consider the integral

$$I = \int_{y=0}^{1/2} \int_{x=0}^{1-2y} xy \, dx \, dy = \frac{1}{96}.$$

This integral can be evaluated using the expression below (Note the use of the non-constant lambda functions for the upper limit of the inner integral):

```
>>> from scipy.integrate import dblquad
>>> area = dblquad(lambda x, y: x*y, 0, 0.5, lambda x: 0, lambda x: 1-2*x)
>>> area
(0.010416666666666668, 1.1564823173178715e-16)
```

TÍCH PHÂN BỘI 3: scipy.integrate.tplquad

```
def tplquad(func, a, b, gfun, hfun, qfun, rfun, args=(), epsabs=1.49e-8,
            epsrel=1.49e-8):
    """
    Compute a triple (definite) integral.

    Return the triple integral of ``func(z, y, x)`` from ``x = a..b``,
    ``y = gfun(x)..hfun(x)``, and ``z = qfun(x,y)..rfun(x,y)``.
```

Compute the triple integral of `x * y * z`, over `x` ranging from 1 to 2, `y` ranging from 2 to 3, `z` ranging from 0 to 1. That is, $\int_{x=1}^{x=2} \int_{y=2}^{y=3} \int_{z=0}^{z=1} xyz \, dz \, dy \, dx$.

```
>>> from scipy import integrate
>>> f = lambda z, y, x: x*y*z
>>> integrate.tplquad(f, 1, 2, 2, 3, 0, 1)
(1.8749999999999998, 3.3246447942574074e-14)
```

Calculate $\int_{x=0}^{x=1} \int_{y=0}^{y=1-2x} \int_{z=0}^{z=1-x-2y} xyz \, dz \, dy \, dx$. Note: *qfun*/*rfun* takes arguments in the order (x, y), even though `f` takes arguments in the order (z, y, x).

```
>>> f = lambda z, y, x: x*y*z
>>> integrate.tplquad(f, 0, 1, 0, lambda x: 1-2*x, 0, lambda x, y: 1-x-2*y)
(0.05416666666666668, 2.1774196738157757e-14)
```

TỔNG KẾT MODULE

Integration (`scipy.integrate`)

The `scipy.integrate` sub-package provides several integration techniques including an ordinary differential equation integrator. An overview of the module is provided by the help command:

```
>>> help(integrate)
 Methods for Integrating Functions given function object.

   quad          -- General purpose integration.
   dblquad       -- General purpose double integration.
   tplquad       -- General purpose triple integration.
   fixed_quad    -- Integrate func(x) using Gaussian quadrature of order n.
   quadrature    -- Integrate with given tolerance using Gaussian quadrature.
   romberg       -- Integrate func using Romberg integration.

 Methods for Integrating Functions given fixed samples.

   trapezoid            -- Use trapezoidal rule to compute integral.
   cumulative_trapezoid -- Use trapezoidal rule to cumulatively compute integral.
   simpson              -- Use Simpson's rule to compute integral from samples.
   romb                 -- Use Romberg Integration to compute integral from
                        -- (2**k + 1) evenly-spaced samples.

   See the special module's orthogonal polynomials (special) for Gaussian
      quadrature roots and weights for other weighting factors and regions.

 Interface to numerical integrators of ODE systems.

   odeint        -- General integration of ordinary differential equations.
   ode           -- Integrate ODE using VODE and ZVODE routines.
```

MONTE CARLO INTEGRATION

We compute the integral of $f : \mathbb{R}^d \to \mathbb{R}, \quad d \geqslant 1$ by generating n random points in $\Omega \subset \mathbb{R}^d$ and use the approximation,

$$\int\int \cdots \int_{\Omega} f(x_1, x_2, \ldots, x_d) \; dx_1 dx_2 \ldots dx_d \approx volume(\Omega) * \frac{\sum_{i=1}^{n} f(\mathbf{z_i})}{n}$$

where $\mathbf{z_i}$ are randomly chosen values from $\mathbb{R}^d$. We can also use this technique to compute volumes (areas) in $\mathbb{R}^d$. Define the characteristic function χ_Ω of a region Ω as,

$$\begin{aligned} \chi_\Omega(x) &= 1 \text{ if } x \in \Omega \\ &= 0 \text{ if } x \notin \Omega \end{aligned}$$

then for a rectangular region that bounds Ω we have,

$$volume(\Omega) = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_d}^{b_d} \chi_\Omega(x) \; dx_1 dx_2 \ldots dx_d \approx \prod_{i=1}^{n} (b_i - a_i) * \frac{\sum_{i=1}^{n} \chi(\mathbf{z_i})}{n}$$

MONTE CARLO INTEGRATION ERROR

The error in computing the integral of $f : \mathbb{R}^d \to \mathbb{R}, \quad d \geqslant 1$ by generating n random points in $\mathbb{R}^d$ and using the Monte Carlo Method is,

$$O(\frac{1}{\sqrt{n}}) = \left| \int \int \ldots \int_\Omega f(x_1, x_2, \ldots, x_d) \;\; dx_1 dx_2 \ldots dx_d - volume(\Omega) * \frac{\sum_{i=1}^n f(\mathbf{z_i})}{n} \right|$$

where $\mathbf{z_i}$ are randomly chosen values from $\Omega \subset \mathbb{R}^d$. Thus, to increase the accuracy of your approximation by one decimal digit using a Monte Carlo method you must increase the number of sample points by a factor of 100.

- Two requirements for MC:
  - knowing which probability distributions are needed
  - generating sufficient random numbers
- The probability distribution depends on the problem (theoretical or empirical evidence)
- The probability distribution can be approximated well by simulating a large number of trials