附录 A 应用举例

A.1 应用: 定积分的数值计算

A.1.1 为什么要数值计算

考虑定积分

$$I = \int_a^b f(x)x. \tag{A.1}$$

在微积分中, 我们可以使用 Newton-Leibnitz 公式来计算, 即

$$\int_a^b f(x)x = F(b) - F(a),$$

其中 $F(x)$ 是被积函数 $f(x)$ 的一个原函数. 但是

- 在很多情况下, 被积函数的原函数很难求出, 或者原函数很复杂, 如 $f(x) = \dfrac{1}{1+x^6}$ 的原函数为

$$F(x) = \frac{1}{3}\arctan x + \frac{1}{6}\arctan\left(x - \frac{1}{x}\right) + \frac{1}{4\sqrt{3}}\ln\frac{x^2 + x\sqrt{3} + 1}{x^2 - x\sqrt{x} + 1} + C.$$

- 原函数无法用初等函数表示, 如

$$f(x) = \frac{\sin x}{x}, \quad f(x) = \mathrm{e}^{-x^2}, \quad f(x) = \sqrt{1 + k^2\sin^2 x}.$$

- 在某些实际应用中, 被积函数 $f(x)$ 的表达式是未知的, 只是通过实验或测量等手段给出了某些离散点上的值.

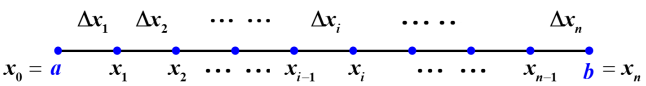
在这些情况下, 我们就需要考虑通过数值方法来计算定积分的近似值, 即**数值积分**.

A.1.2 定积分数值计算方法

假设定积分 $\int_a^b f(x)x$ 存在, 则根据定积分的定义, 我们有

$$\int_a^b f(x)x = \lim_{\Delta x\to 0}\sum_{i=1}^{n} f(\xi_i)\Delta x_i, \quad \xi_i \in [x_{i-1}, x_i], \quad \Delta x_i = x_i - x_{i-1}, \quad \Delta x = \max_{1\le i\le n}\Delta x_i,$$

其中 $a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$ 是 $[a, b]$ 的一个剖分, 即



为了方便计算, 我们不妨对 $[a, b]$ 进行 n 等分, 即取步长 $h = \frac{b-a}{n}$, 令

$$x_i = a + ih, \quad i = 0, 1, 2, \ldots, n.$$

于是有

$$\int_a^b f(x)x = \lim_{n\to\infty}\sum_{i=1}^{n} hf(\xi_i).$$

因此当 n 充分大时有

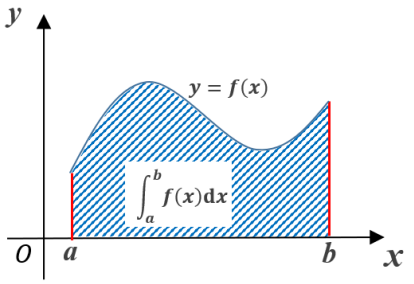
$$\int_a^b f(x)x \approx h\sum_{i=1}^{n} f(\xi_i). \tag{A.2}$$

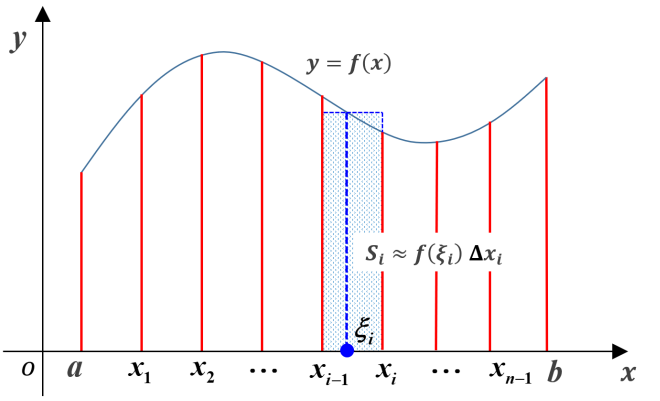
例 A.1 计算定积分 $\int_0^{\frac{\pi}{2}} \sin(x)\mathrm{d}x$ 的近似值.

思路: 将求积区间进行 n 等分, 然后在每个小区间中随机选取一个点 ξ_i, 不断增大 n, 直到计算结果满足精度要求. (ex03_Integral.cpp)

```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;

const double a = 0.0;
const double b = atan(1)*2; // b = pi/2

int main()
{
    long n=1; // 从一等分开始
    double x, h, S;

    srand(time(0));
    cout << fixed << setprecision(10);
    cout << left;
    for(int k=0; k<20; k++)
    {
        h = (b-a)/n;
        S = 0;
        for(int i=0; i<n; i++)
        {
            x = a + h*(i+double(rand())/RAND_MAX);
            S = S + sin(x);
        }
        cout << "n=" << setw(8) << n << " S=" << S*h << endl;
        n = 2*n; // 不断增大 n 的值
    }

    return 0;
}
```

由定积分的几何意义 (曲边梯形的面积, 下面的左图) 可知, 上述计算公式 A.2 是在每个小区间上用矩形的面积来近似曲边梯形的面积 (下面的右图) , 因此称为**矩形公式**.





$$S=\int_a^b f(x)\mathrm{d}x=\sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} f(x)\mathrm{d}x=\sum_{i=1}^{n} S_i\approx\sum_{i=1}^{n} f(\xi_i)\,\Delta x_i$$

为了提高计算精度, 可以对该方法进行改进, 构造具有更高精度的方法, 比如**梯形公式**:

$$\int_a^b f(x)x\approx\sum_{i=1}^{n}\frac{h\big(f(x_{i-1})+f(x_i)\big)}{2},$$

即用直边梯形的面积来近似曲边梯形的面积.

从另外一个角度看, 梯形法也可以看作是在每个小区间 $[x_{i-1},x_i]$ 内, 用经过端点 $(x_{i-1},f(x_{i-1}))$ 和 $(x_i,f(x_i))$ 的直线 $L_1(x)$ 来近似曲线 $f(x)$, 即

$$f(x)\approx L_1(x),\quad x\in[x_{i-1},x_i].$$

然后用 $\int_{x_{i-1}}^{x_i} L_1(x)x$ 近似 $S_i\triangleq\int_{x_{i-1}}^{x_i} f(x)x$. 这里用了函数逼近的思想, 即在小区间内用简单的函数来近似复杂的函数. 需要指出的是, 每个小区间内的 $L_1(x)$ 一般是不同的.

如果做进一步改进, 则可以在每个小区间 $[x_{i-1},x_i]$ 内用抛物线 $L_2(x)$ (即二次多项式) 来近似 $f(x)$, 然后用 $L_2(x)$ 在 $[x_{i-1},x_i]$ 上的定积分近似 S_i, 这就得到**抛物线公式**. 为了确定 $L_2(x)$, 我们须要求 $L_2(x)$ 经过三个点, 一般取左右端点 $(x_{i-1},f(x_{i-1}))$, $(x_i,f(x_i))$ 和中点 $(x_{i-1/2},f(x_{i-1/2}))$, 其中 $x_{i-1/2}=\frac{1}{2}(x_{i-1}+x_i)$. 经过计算, 我们可得**抛物线公式**如下:

$$\int_a^b f(x)x\approx\sum_{i=1}^{n}\frac{h}{6}\big(f(x_{i-1})+4f(x_{i-1/2})+f(x_i)\big).$$

本小节介绍了数值积分的几个简单计算方法, 更多计算方法可以参考**数值分析**方面的相关教材.