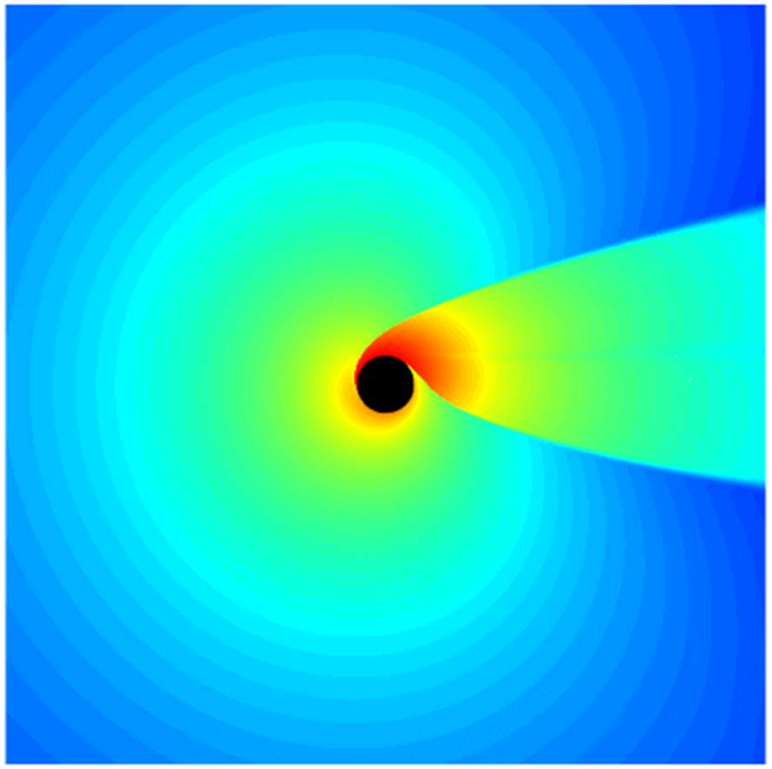
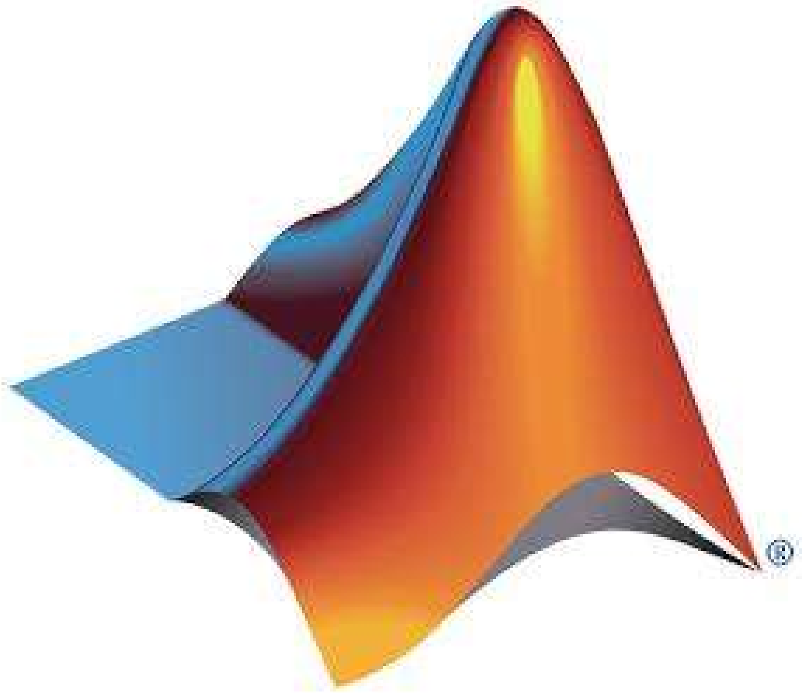
# 数学实验
# Mathematical Experiments





# 实验九：
# 数值积分实验
# Numerical Quadrature

## 实验背景

在工程问题和科学实验中，常常需要计算定积分.

例如

- 力学和电学中功和功率的计算
- 电流和电压平均值的计算
- 以及一些几何图形的面积
- 体积和弧长的计算等等.

另外，微分方程的求解也是以积分计算为基础的.

# 定积分的定义（一元函数为例）

设函数y=f(x)在[a, b]上有界，在区间[a, b]上任取n+1个分点，

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

把[a, b]分成n个小区间

$$\Delta_i = [x_{i-1}, x_i], i = 1,2, \cdots, n,$$

小区间$\Delta_i$的长度为$\Delta x_i = x_i - x_{i-1}$，取$\lambda = \max\{\Delta x_i\}\,(1 \le i \le n)$，

这些分点称为区间[a, b]的一个分割。在区间$\Delta_i$上任意取点$\xi_i \epsilon [x_{i-1}, x_i]$, 作函数值$f(\xi_i)$与区间长度$\Delta x_i$的乘积$f(\xi_i)\Delta x_i (i = 1,2, \ldots, n)$,

## 定积分的定义（一元函数为例）

并作和$S = \sum_{i=1}^{n} f(\xi_i)\Delta x_i$。若对任意的分割，当$\lambda \to 0$时，S极限存在，并称这个极限为函数f(x)在区间[a, b]上的定积分，记作$\int_a^b f(x)dx$，即

$$\int_a^b f(x)dx = \lim_{\lambda \to 0} \sum_{i=1}^{n} f(\xi_i)\Delta x_i$$

- 将[a, b]进行等分, 则$\Delta x_i = \frac{1}{n}(b-a), x_i = a + \frac{i}{n}(b-a), i = 1,2, \dots, n,$ 则

$$\int_a^b f(x)dx = \lim_{n \to \infty} \frac{1}{n}(b-a) \sum_{i=1}^{n} f(\xi_i)$$

# 精确求定积分的几种常用方法

(1)直接从定义出发，取近似和的极限.

这种方法原理简单，但计算量大.

然而数值积分的各种算法却是基于定积分定义建立起来的
.

(2)用不定积分计算定积分.不定积分是求导的逆运算，而定积分是考虑连续变量的求和(如计算曲边梯形的面积).表面上看是两个完全不同的概念，然而却通过牛顿-莱布尼兹公式联系在一起了，甚至两者的符号也是类似的，牛顿-莱布尼兹公式被称为微积分学基本定理.

微积分学基本定理(牛顿-莱布尼兹公式）如果$F(x)$是$f(x)$的原函数，即$F'(x)=f(x)$，则有
$\int_a^b f(x)dx = F(b) - F(a)$成立.

## 精确求定积分的几种常用方法

(3) 其它方法：

？ ？ ？

## 精确求定积分的几种常用方法

(3)其它方法:

？？？

化为多元函数问题、留数定理、符号计算、蒙特卡洛...

## 实验目的

我们总是期望求出定积分的精确值来，但在实践中很多定积分往往无法算出精确值（例如，椭圆积分），所以数值积分是非常有用的工具.

本次实验目的

(1)理解数值积分方法原理，并利用数值积分方法求解定积分。

(2)熟悉MATLAB中数值积分的内置函数及用法。

(3)使用MATLAB解决一些积分应用问题。

# 几何的角度看定积分

## What is Integration?

**Integration**

The process of measuring the area under a curve.

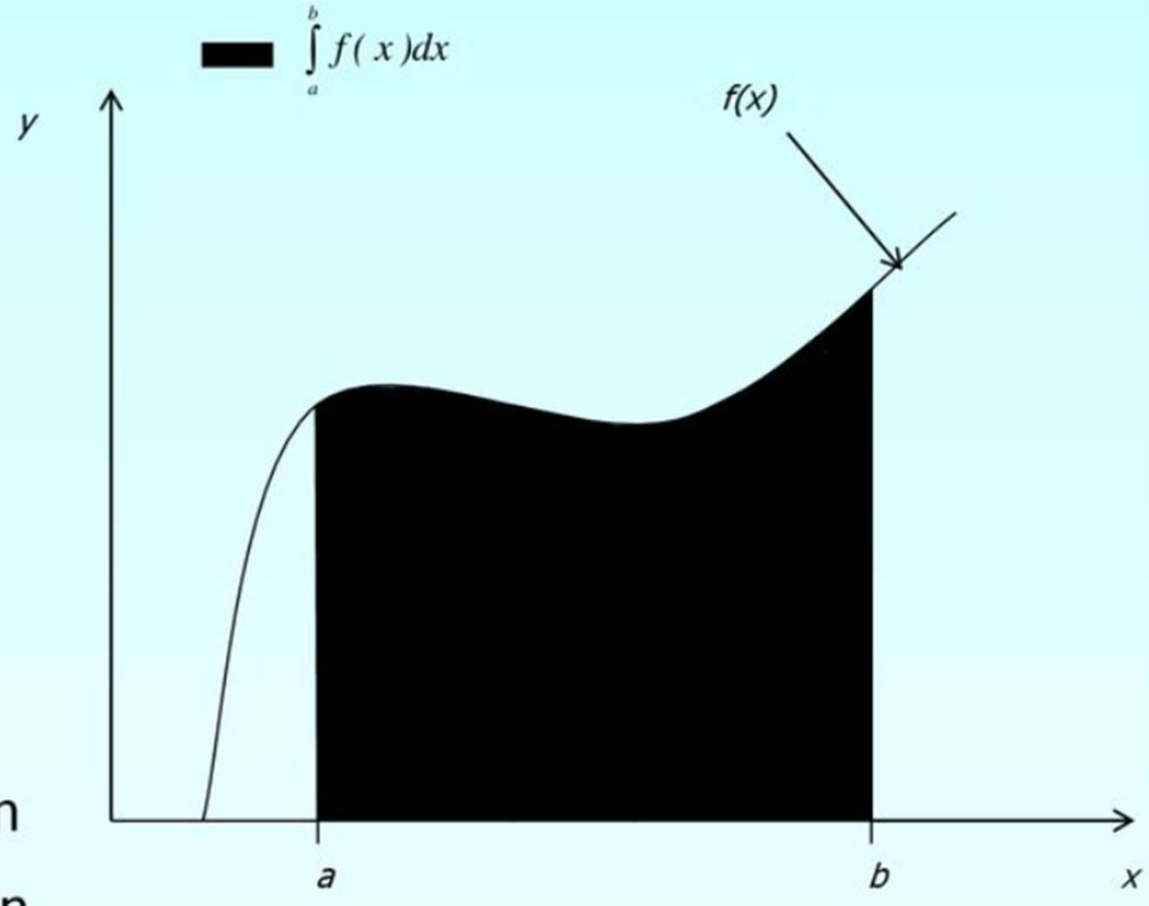
$$I = \int_a^b f(x)dx$$

Where:

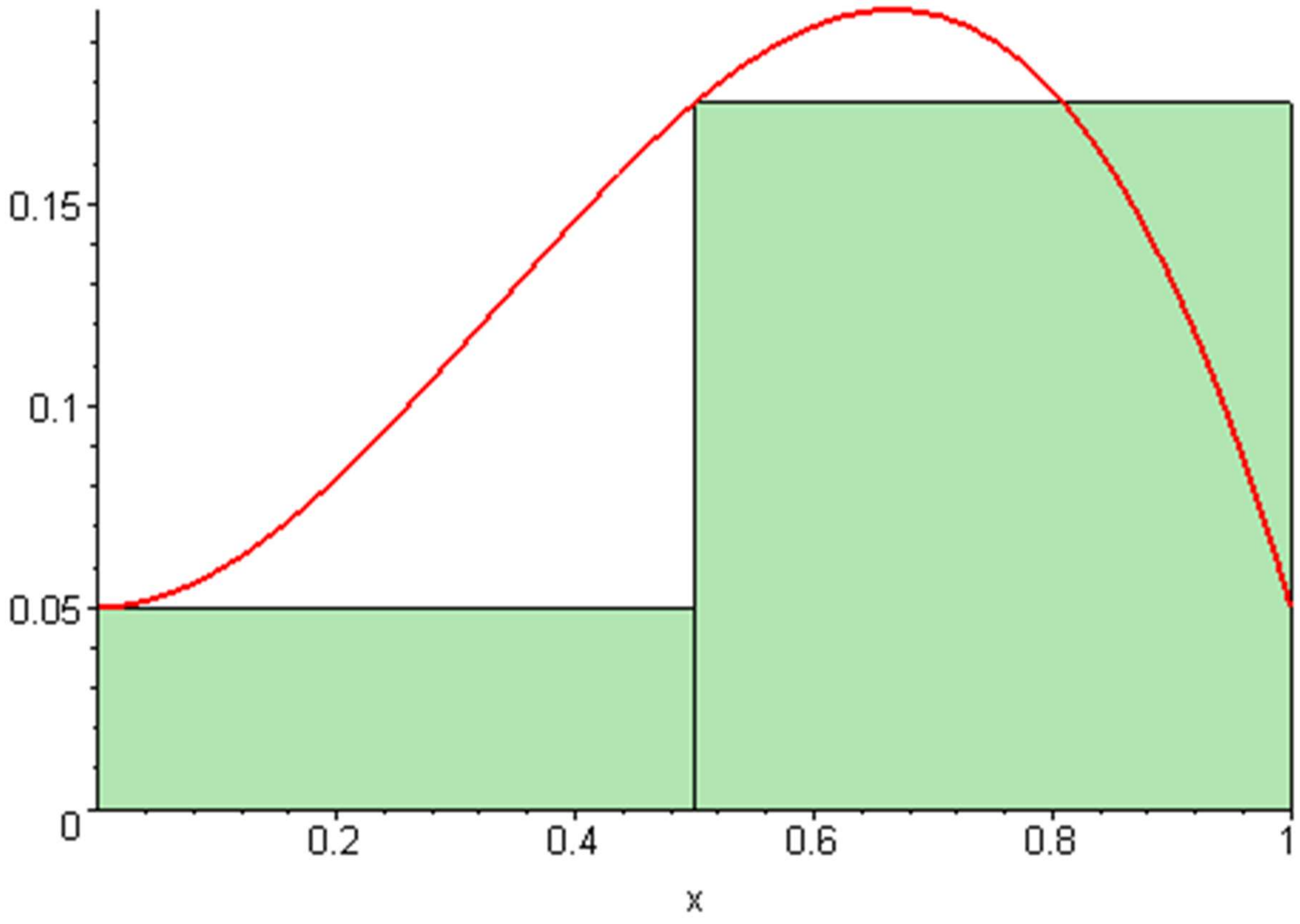
*f(x)* is the integrand

a= lower limit of integration

b= upper limit of integration



# 实验1：数值积分的基本算法简介



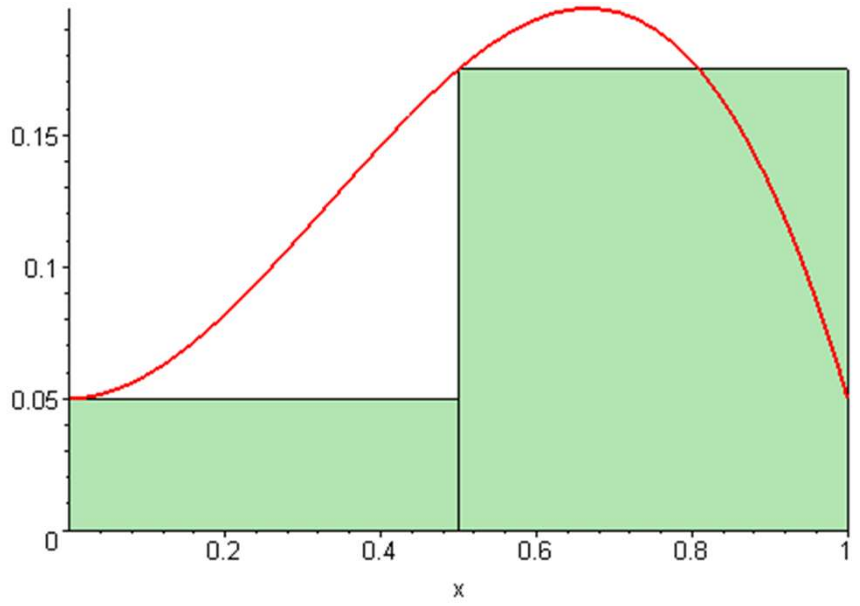
# 实验1：数值积分的基本算法简介

## 算法1：左矩形法

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

若取$\xi_i = x_{i-1}$，可得左矩形法

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_{i-1})\Delta x_i$$



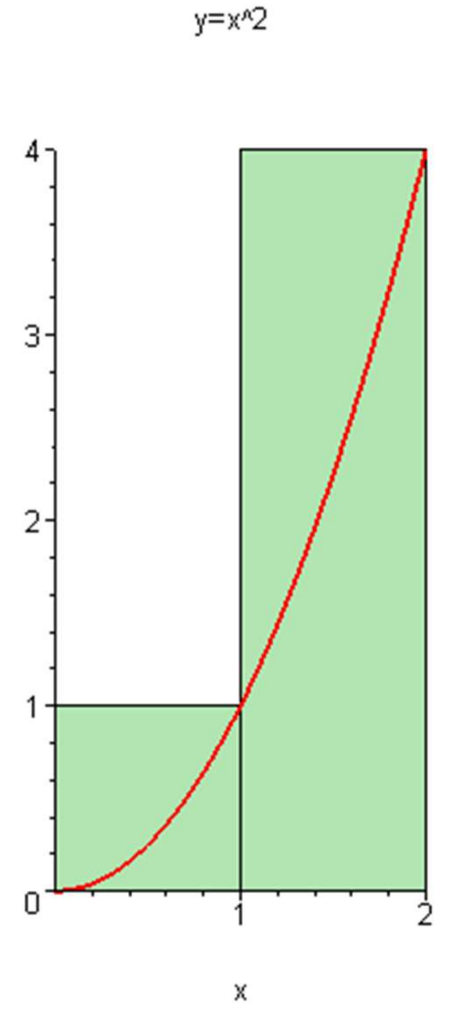
# 实验1：数值积分的基本算法简介

## 算法2：右矩形法

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

若取$\xi_i = x_i$，可得右矩形法

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_i)\Delta x_i$$



## 实验1：数值积分的基本算法简介

算法3：（复化的）中点法（Midpoint Rule）/中矩形法

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

若取$\xi_i = \frac{x_{i-1}+x_i}{2}$，可得

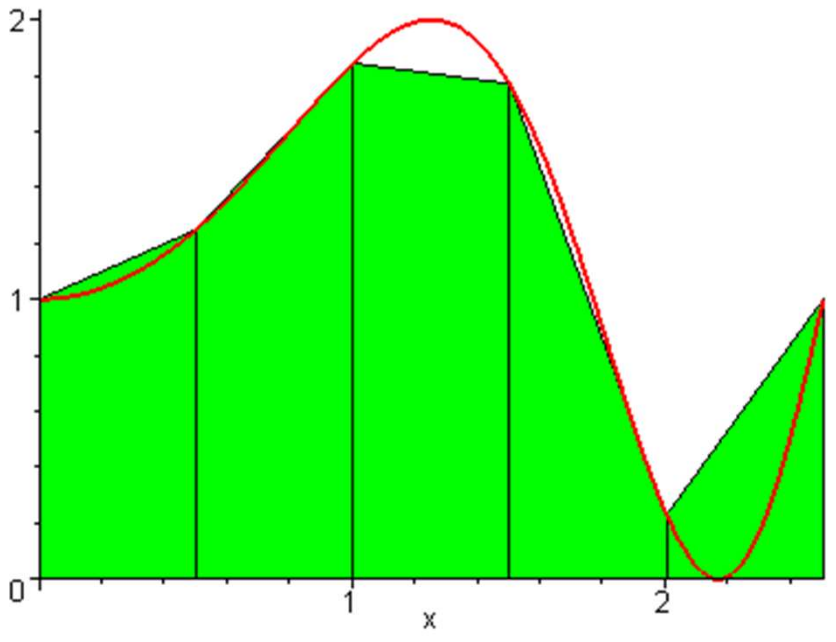
$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(\frac{x_{i-1}+x_i}{2})\Delta x_i$$

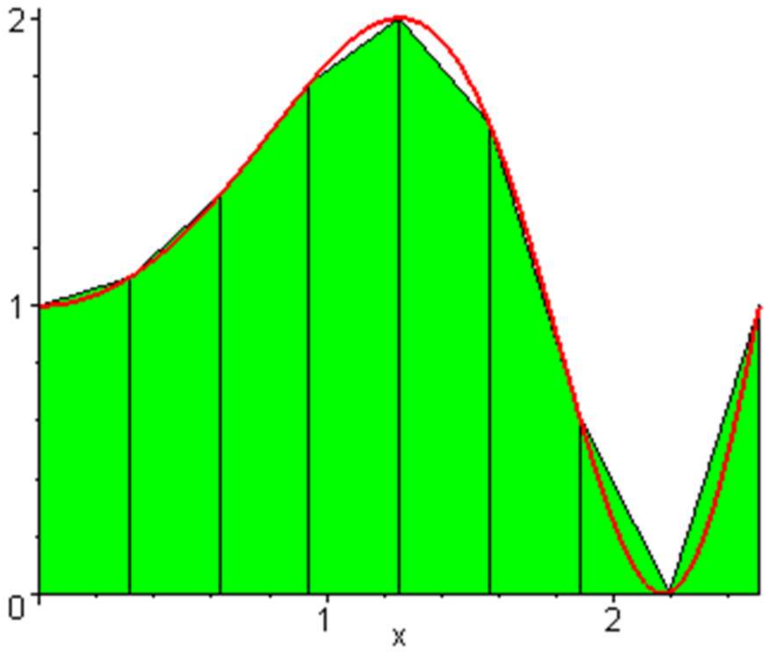
# 实验1：数值积分的基本算法简介

算法4：（复化的）梯形法

(Trapezoidal Rule)

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



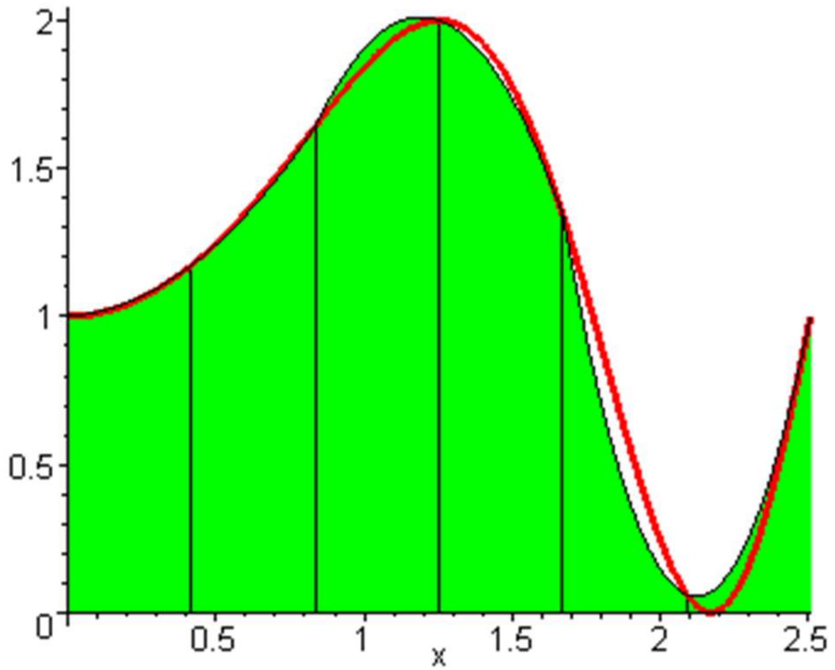


# 实验1：数值积分的基本算法简介

## 算法5：（复化的）辛普森法

(Simpson's Rule)

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



# 实验1：数值积分的基本算法简介

从多项式插值的角度重新审视数值积分公式

- 数值积分的本质就是用被积函数的点值的组合近似积分值
- 大部分数值积分公式对应着某种多项式插值公式
- 如何用尽可能少的点，获得尽可能高的精度？(Gaussian Quadrature Rule)

# 实验1：数值积分的基本算法简介

## 误差分析

以左矩形公式为例

$$\left|\int_a^b f(x)\,dx-(b-a)f(a)\right|\leq\frac{(b-a)^2}{2}\sup_{a\leq x\leq b}|f'(x)|,$$

- 从一般区间到小区间

## 实验1：数值积分的基本算法简介

- In order to characterize the accuracy of these rules, we need to determine the accuracy of polynomial interpolation:

**Theorem:** Let $f$ be $n+1$ times continuously differentiable. Let $\tilde{f}$ be the degree $n$ polynomial interpolation of $f$ at $x_0, \ldots, x_n$. Then the error of interpolation at $x$ can be written:

$$f(x) - \tilde{f}(x) = \frac{f^{(n+1)}(\eta)}{(n+1)!} \prod_j (x - x_j)$$

for some $\min(x_j) \leq \eta \leq \max(x_j)$.

*Proof:* Let $q(x) = \prod(x - x_j)$, and define $g$ by:

$$g(y) \equiv f(y) - \tilde{f}(y) - q(y)\frac{f(x) - \tilde{f}(x)}{q(x)}.$$

Note that $g$ is $n+1$ times continuously differentiable, and $g$ vanishes at $x$ and at $x_0, \ldots, x_n$. Therefore by Rolle's theorem $g'$ has at least $n+1$ zeros. Repeated application of Rolle's theorem gives that $g^{(n+1)}$ has at least one zero, which we denote $\eta$. Evaluating $g^{(n+1)}$ at $\eta$ gives the result.

## 实验1：数值积分的基本算法简介

Now we can derive the error estimate for the trapezoidal rule.

**Theorem:** Let $f$ be twice continuously differentiable. Then for some $a \le \eta \le b$, the error for the trapezoidal rule is:

$$\int_a^b f(x)dx - \frac{b-a}{2}(f(a)+f(b)) = -\frac{(b-a)^3}{12} f''(\eta).$$

*Proof:* First observe that if $\tilde{f}$ denotes the linear interpolation of $f$ on $[a,b]$, then the error can be represented as

$$\int_a^b f(x)dx - \frac{b-a}{2}(f(a)+f(b)) = \int_a^b \left(f(x)-\tilde{f}(x)\right)dx = \int_a^b (x-a)(x-b)\frac{f(x)-\tilde{f}(x)}{(x-a)(x-b)}dx.$$

The term $(x-a)(x-b)$ is non-positive, and the second factor is continuous. Therefore by the mean value theorem there exists $\eta$ such that the above becomes equal to

$$\frac{f(\eta)-\tilde{f}(\eta)}{(\eta-a)(\eta-b)} \int_a^b (x-a)(x-b)dx.$$

The second term is equal to $-(b-a)^3/6$, and the first term is equal to $f''(\theta)/2$ for some $\theta \in [a,b]$, which gives the result.

## 实验1：数值积分的基本算法简介

Now we can derive the error estimate for the trapezoidal rule.

**Theorem:** Let $f$ be twice continuously differentiable. Then for some $a \leq \eta \leq b$, the error for the trapezoidal rule is:

$$\int_a^b f(x)dx - \frac{b-a}{2}(f(a)+f(b)) = -\frac{(b-a)^3}{12} f''(\eta).$$

- Applying this result, if we use mesh $h$ then the error of the trapezoidal rule between successive abscissas decreases as $h^3$. Since there are $L/h$ intervals, the total error decreases as $h^2$, or as $1/m^2$ if there are $m$ abscissas. Thus in order to reduce the error by half, the number of quadrature points must be increased by approximately a factor of $\sqrt{2} \approx 1.4$.

# 实验1：数值积分的基本算法简介

- A direct extension of the theorem gives that the error of approximation for a degree $n$ interpolant $\tilde{f}$ is bounded in magnitude by

$$\max_{a\le u\le b}\left|\frac{f^{(n+1)}(u)}{(n+1)!}\right|(b-a)^{n+2}.$$

Applying this to Simpson's rule ($n = 2$) gives that the error for a single quadratic interpolating polynomial on $(a, b)$ decreases at order $(b-a)^4$. Thus the error for the

牛顿-柯特斯公式是一种插值型公式。假设$I = [a,b]$中取$x_k = a + k\dfrac{b-a}{n}$ ,可以写成:

$$\int_I f(t)dt \approx \mathbf{I}_{\mathbf{appr}}(x_1, x_2, \cdots, x_n) = (b-a)\sum_{k=0}^{n} C_k^{(n)} f(x_k)$$

其中的 $C_k^{(n)} = \dfrac{1}{n}\displaystyle\int_0^n \prod_{k\neq j}\frac{t-j}{k-j}dt = \frac{(-1)^{n-k}}{n\cdot k!(n-k)!}\int_0^n \prod_{k\neq j}(t-j)dt$

# 实验1：数值积分的基本算法简介

- A direct extension of the theorem gives that the error of approximation for a degree $n$ interpolant $\tilde{f}$ is bounded in magnitude by

$$\max_{a \le u \le b} \left| \frac{f^{(n+1)}(u)}{(n+1)!} \right| (b-a)^{n+2}.$$

  Applying this to Simpson's rule ($n = 2$) gives that the error for a single quadratic interpolating polynomial on $(a, b)$ decreases at order $(b-a)^4$. Thus the error for the interpolating spline with $m$ abscissas decreases at order $h^3$, or as $1/m^3$ for $m$ abscissas. Thus in order to reduce the error by half using Simpson's rule, the number of quadrature points must be increased by approximately a factor of $2^{1/3} \approx 1.26$.

- Surprisingly, Simpson's rule actually has error of magnitude $h^5 f^{(4)}(\eta)/90$ per interval, or $h^4 f^{(4)}(\eta)/90$ overall – an order better than calculated above.

  It is a general property that the order of the Newton-Cotes formula increases by two when moving from odd-order to even-order interpolations, and doesn't increase at all the other way around.

# 实验1：数值积分的基本算法简介

- 定义：（代数精度）如果一个求积公式对所有<=n次的多项式都是成立的，则称该求积公式的代数精度（至少）为n

# 实验1：数值积分的基本算法简介

- Integration rules can be characterized in terms of the highest degree polynomial for which the error is zero. The trapezoidal rule is exact for degree 1 polynomials and Simpson's rule is exact for degree 2 polynomials – these statements are true because the interpolating polynomial is exactly equal to the integrand, $f \equiv \tilde{f}$.

  In fact, Simpson's rule is exact for cubics. To see this it is adequate to check that it holds for $f(x) = x^3$ (since Simpson's rule is additive and we already know that it is exact for quadratics). By direct evaluation $\int_0^1 x^3 dx = 1/4$, while Simpson's rule gives $(1 \cdot 0 + 4/8 + 1 \cdot 1)/6 = 1/4$ (in applying Simpson's rule the abscissas are $0, 1/2, 1$ and the mesh is $h = 1/2$).

  The reason for this is that the difference $f - \tilde{f}$, while not identically zero, has zero integral. The quadratic interpolant through $(0, 0)$, $(1/2, 1/8)$, $(1, 1)$ is

$$\tilde{f}(x) = 3x^2/2 - x/2$$

  and the pointwise error of interpolation is

$$\tilde{f}(x) - f(x) = 3x^2/2 - x/2 - x^3 = -(x - 1/2)^3 + (x - 1/2)/4 + 1/2$$

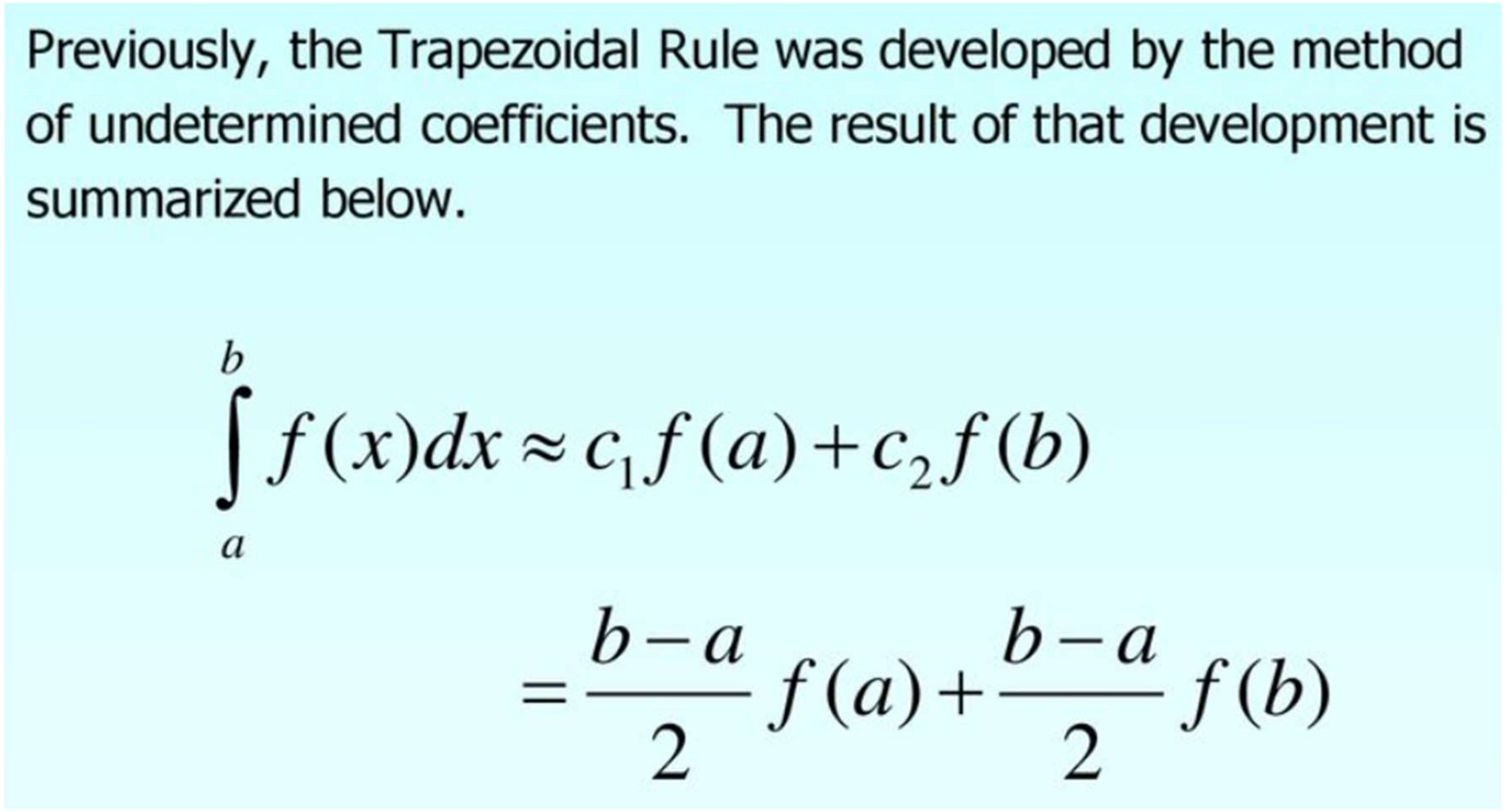
  which is an odd function with respect to $x = 1/2$, hence intergates to zero over any interval centered at $1/2$.

## 实验1：数值积分的基本算法简介

- 如何用尽可能少的点，获得尽可能高的精度？
- n个求积节点，最高的代数精度是多少？
- 高斯求积公式(Gaussian Quadrature Rule)

# 实验1：数值积分的基本算法简介

先从简单的特例出发，总结规律



Previously, the Trapezoidal Rule was developed by the method of undetermined coefficients. The result of that development is summarized below.

$$\int_a^b f(x)dx \approx c_1 f(a) + c_2 f(b)$$

$$= \frac{b-a}{2} f(a) + \frac{b-a}{2} f(b)$$

## 实验1：数值积分的基本算法简介

The two-point Gauss Quadrature Rule is an extension of the Trapezoidal Rule approximation where the arguments of the function are not predetermined as a and b but as unknowns $x_1$ and $x_2$. In the two-point Gauss Quadrature Rule, the integral is approximated as

$$I = \int_a^b f(x)dx \approx c_1 f(x_1) + c_2 f(x_2)$$

# 实验1：数值积分的基本算法简介

The four unknowns $x_1$, $x_2$, $c_1$ and $c_2$ are found by assuming that the formula gives exact results for integrating a general third order polynomial, $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$.

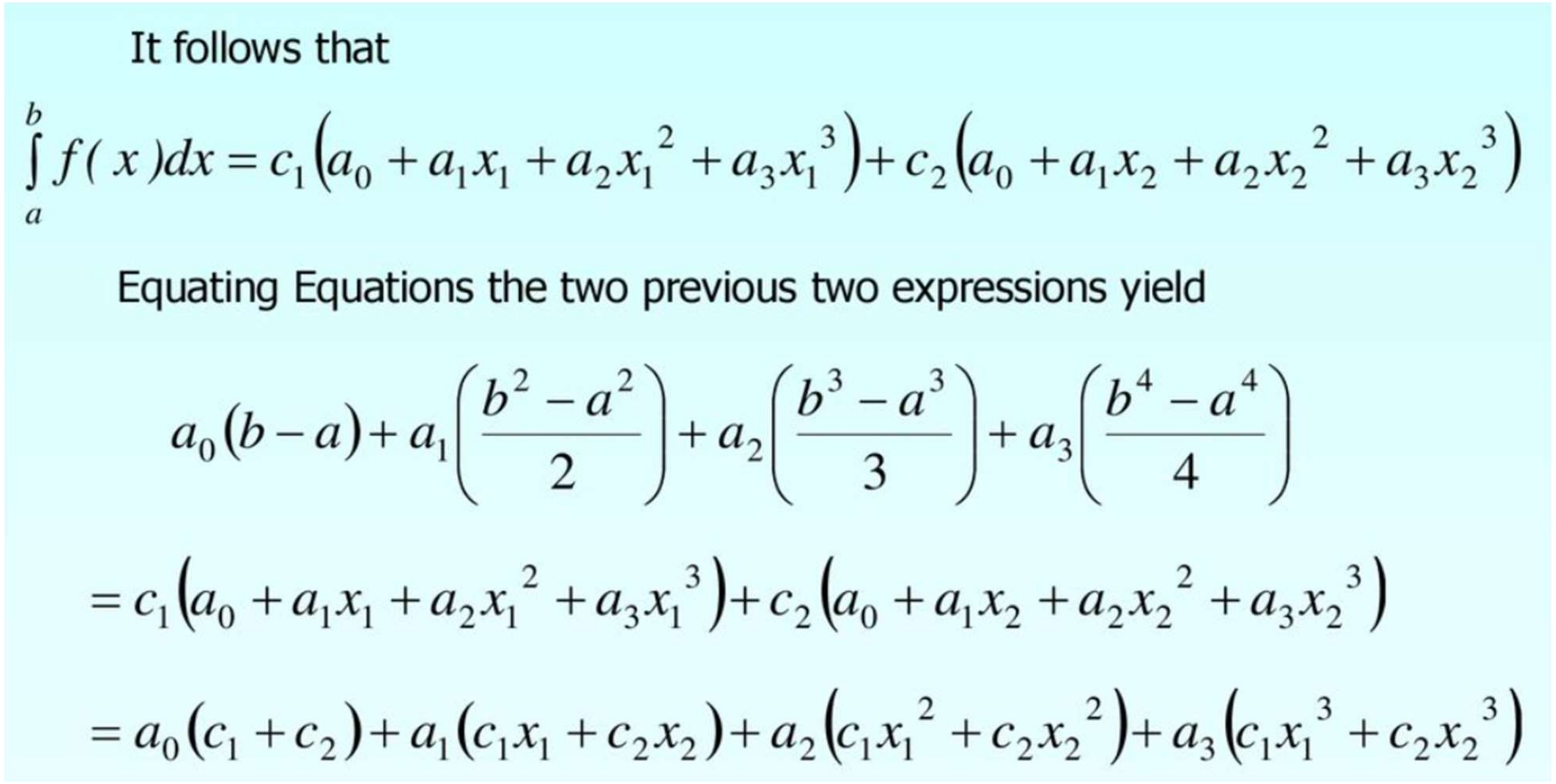
Hence

$$\int_a^b f(x)dx = \int_a^b \left(a_0 + a_1x + a_2x^2 + a_3x^3\right)dx$$

$$= \left[a_0x + a_1\frac{x^2}{2} + a_2\frac{x^3}{3} + a_3\frac{x^4}{4}\right]_a^b$$

$$= a_0(b-a) + a_1\left(\frac{b^2-a^2}{2}\right) + a_2\left(\frac{b^3-a^3}{3}\right) + a_3\left(\frac{b^4-a^4}{4}\right)$$

# 实验1：数值积分的基本算法简介



It follows that

$$\int_a^b f(x)dx = c_1\left(a_0 + a_1x_1 + a_2x_1^{\ 2} + a_3x_1^{\ 3}\right) + c_2\left(a_0 + a_1x_2 + a_2x_2^{\ 2} + a_3x_2^{\ 3}\right)$$

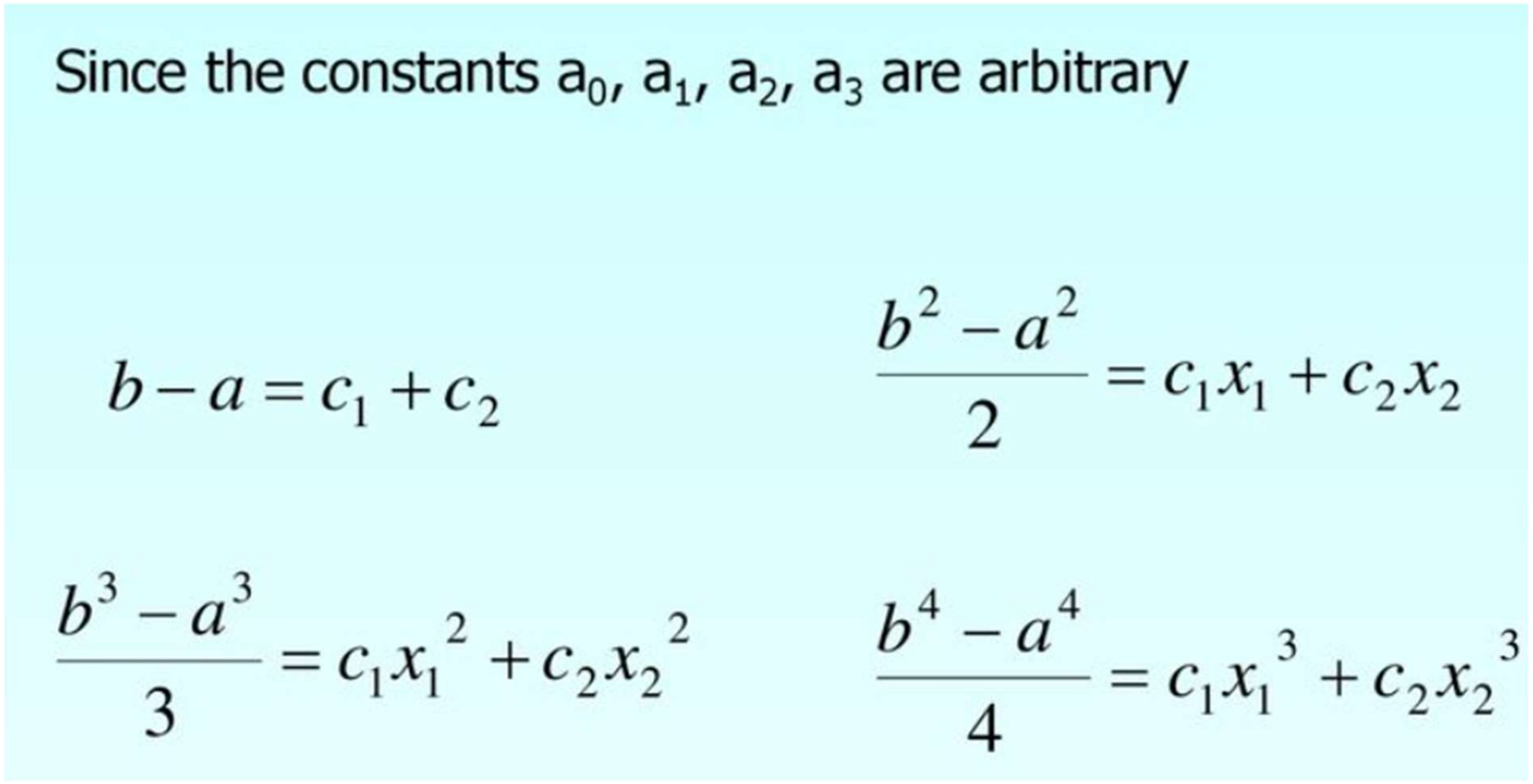
Equating Equations the two previous two expressions yield

$$a_0(b-a) + a_1\left(\frac{b^2-a^2}{2}\right) + a_2\left(\frac{b^3-a^3}{3}\right) + a_3\left(\frac{b^4-a^4}{4}\right)$$

$$= c_1\left(a_0 + a_1x_1 + a_2x_1^{\ 2} + a_3x_1^{\ 3}\right) + c_2\left(a_0 + a_1x_2 + a_2x_2^{\ 2} + a_3x_2^{\ 3}\right)$$

$$= a_0(c_1 + c_2) + a_1(c_1x_1 + c_2x_2) + a_2\left(c_1x_1^{\ 2} + c_2x_2^{\ 2}\right) + a_3\left(c_1x_1^{\ 3} + c_2x_2^{\ 3}\right)$$

# 实验1：数值积分的基本算法简介

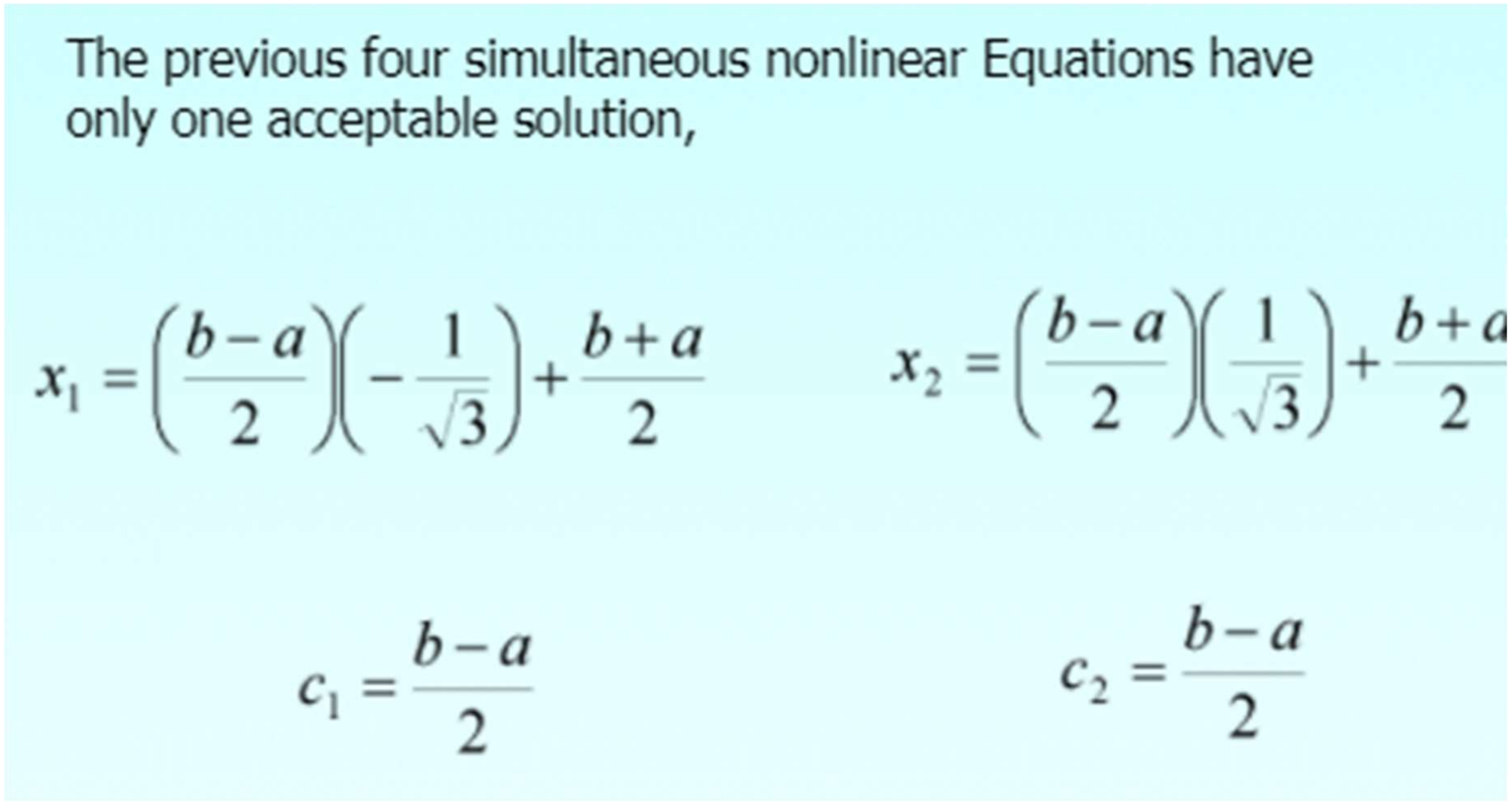


Since the constants $a_0$, $a_1$, $a_2$, $a_3$ are arbitrary

$$b-a=c_1+c_2 \qquad \frac{b^2-a^2}{2}=c_1x_1+c_2x_2$$

$$\frac{b^3-a^3}{3}=c_1{x_1}^2+c_2{x_2}^2 \qquad \frac{b^4-a^4}{4}=c_1{x_1}^3+c_2{x_2}^3$$

# 实验1：数值积分的基本算法简介

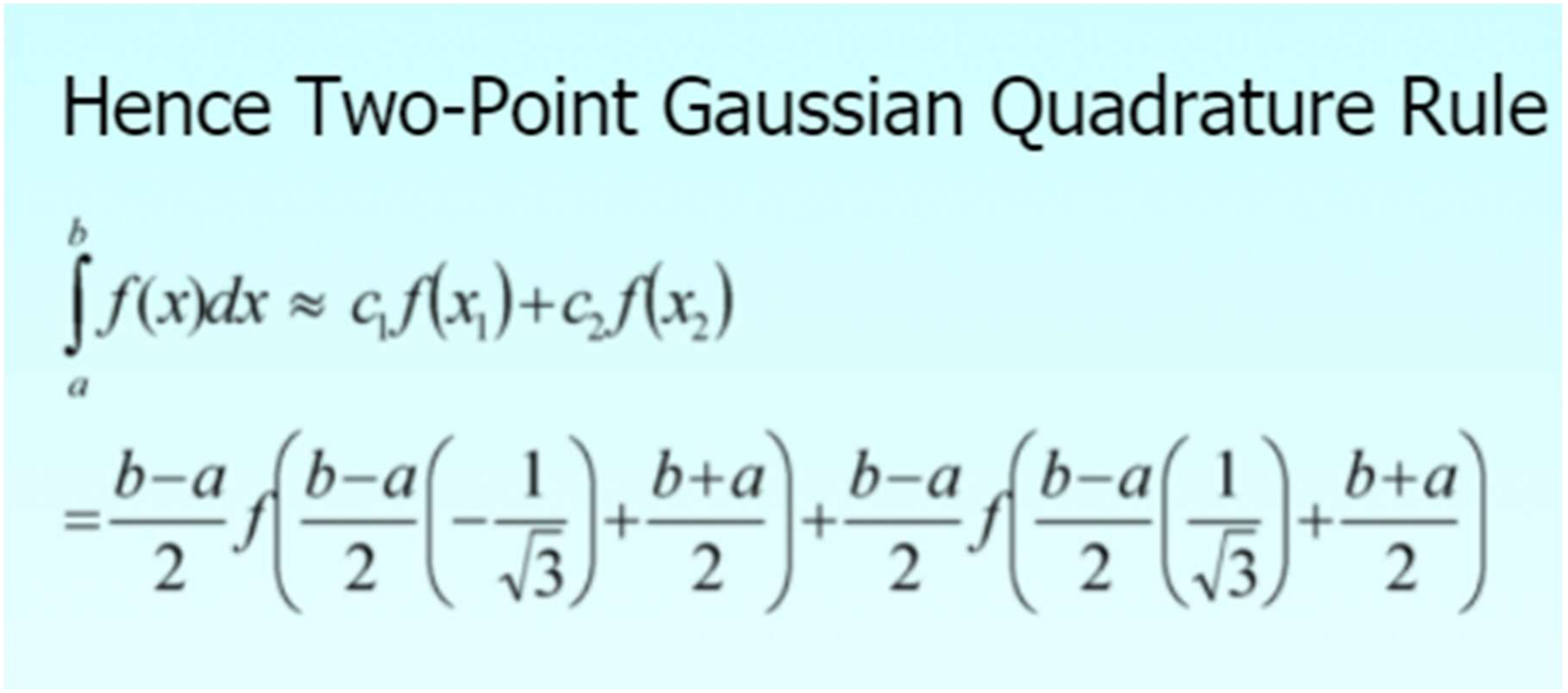


The previous four simultaneous nonlinear Equations have only one acceptable solution,

$$x_1 = \left(\frac{b-a}{2}\right)\left(-\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2} \qquad x_2 = \left(\frac{b-a}{2}\right)\left(\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2}$$

$$c_1 = \frac{b-a}{2} \qquad c_2 = \frac{b-a}{2}$$

# 实验1：数值积分的基本算法简介

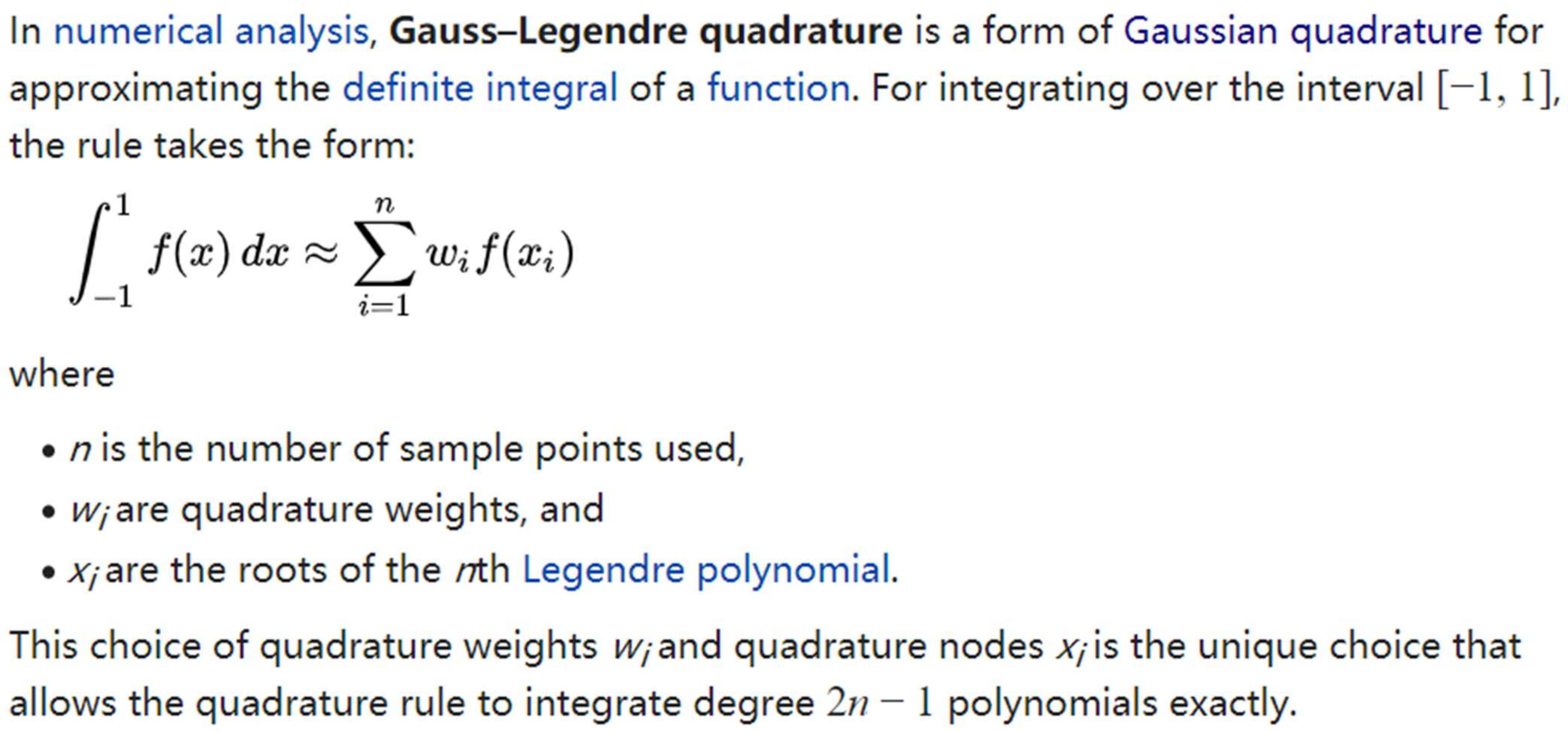


Hence Two-Point Gaussian Quadrature Rule

$$\int_a^b f(x)dx \approx c_1 f(x_1) + c_2 f(x_2)$$

$$= \frac{b-a}{2} f\left(\frac{b-a}{2}\left(-\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2}\right) + \frac{b-a}{2} f\left(\frac{b-a}{2}\left(\frac{1}{\sqrt{3}}\right) + \frac{b+a}{2}\right)$$

# 实验1：数值积分的基本算法简介



In numerical analysis, **Gauss–Legendre quadrature** is a form of Gaussian quadrature for approximating the definite integral of a function. For integrating over the interval $[-1, 1]$, the rule takes the form:

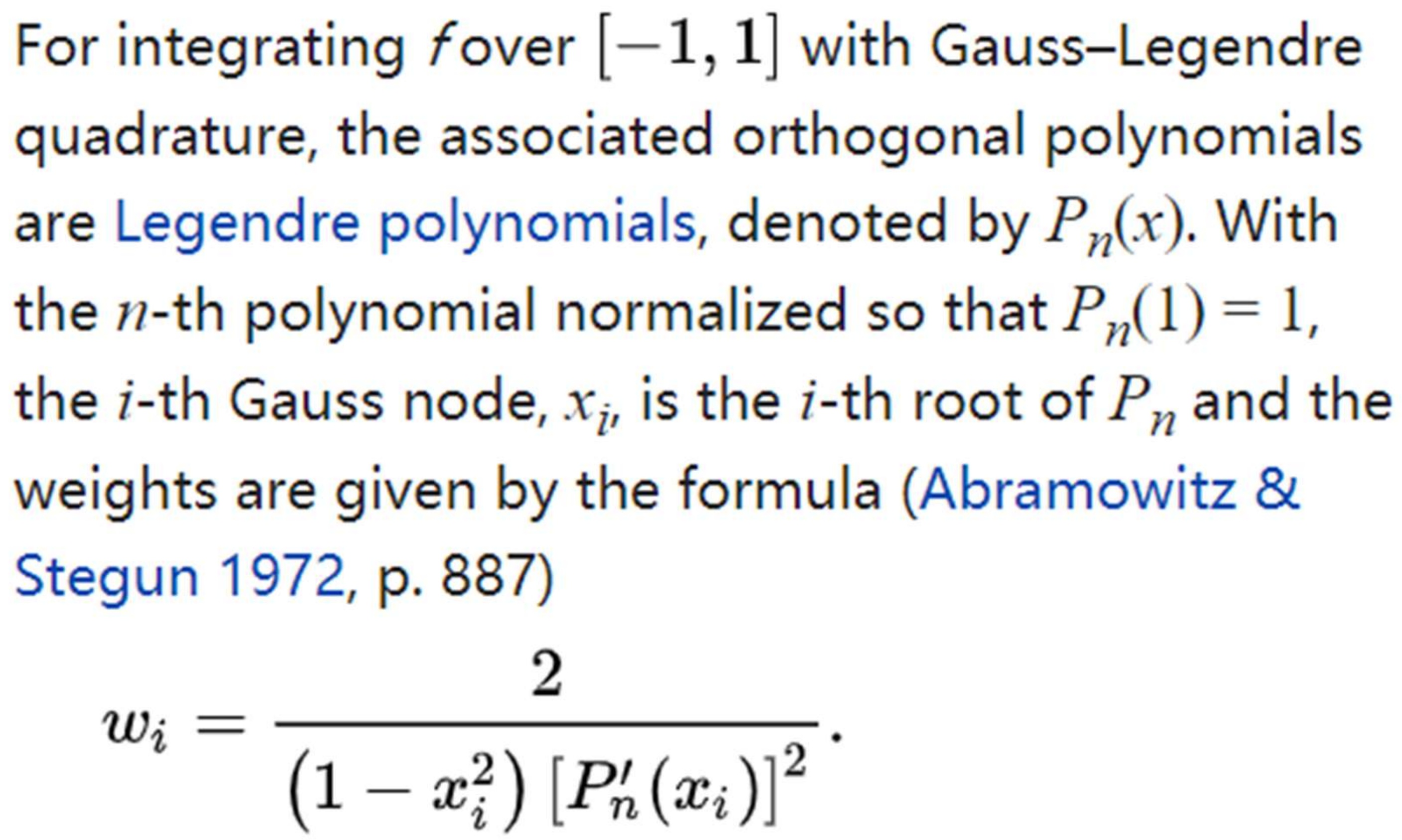
$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

where

- $n$ is the number of sample points used,
- $w_i$ are quadrature weights, and
- $x_i$ are the roots of the $n$th Legendre polynomial.

This choice of quadrature weights $w_i$ and quadrature nodes $x_i$ is the unique choice that allows the quadrature rule to integrate degree $2n-1$ polynomials exactly.

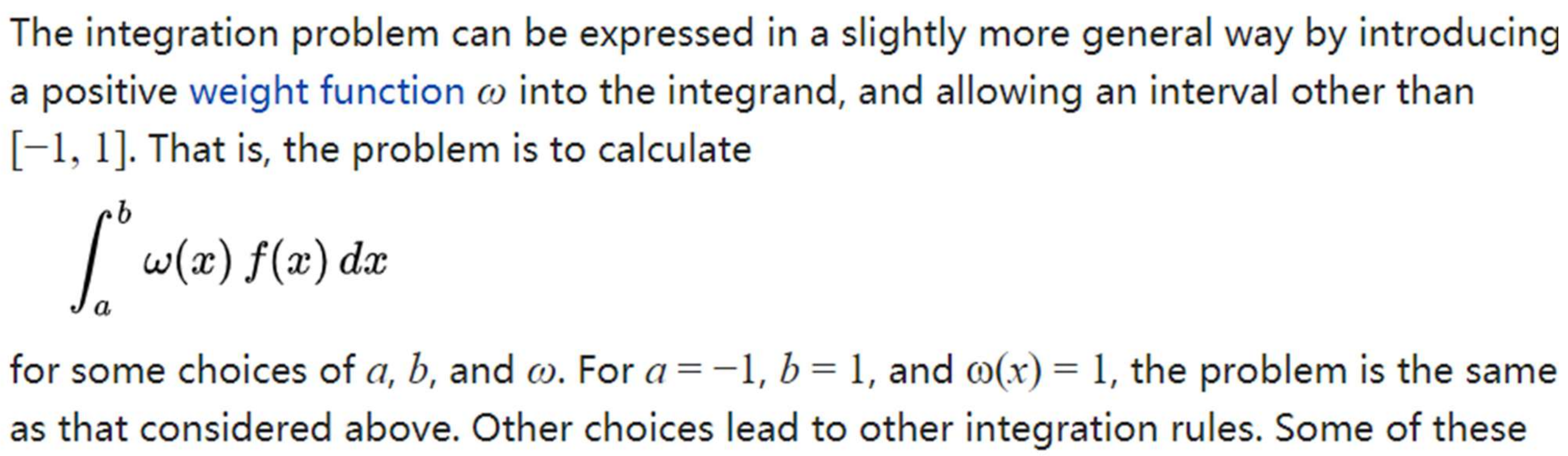
# 实验1：数值积分的基本算法简介



For integrating $f$ over $[-1, 1]$ with Gauss–Legendre quadrature, the associated orthogonal polynomials are Legendre polynomials, denoted by $P_n(x)$. With the $n$-th polynomial normalized so that $P_n(1) = 1$, the $i$-th Gauss node, $x_i$, is the $i$-th root of $P_n$ and the weights are given by the formula (Abramowitz & Stegun 1972, p. 887)

$$w_i = \frac{2}{\left(1 - x_i^2\right)\left[P'_n(x_i)\right]^2}.$$

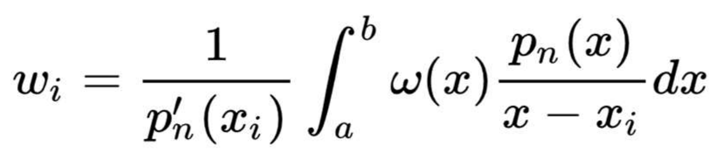
# 实验1：数值积分的基本算法简介



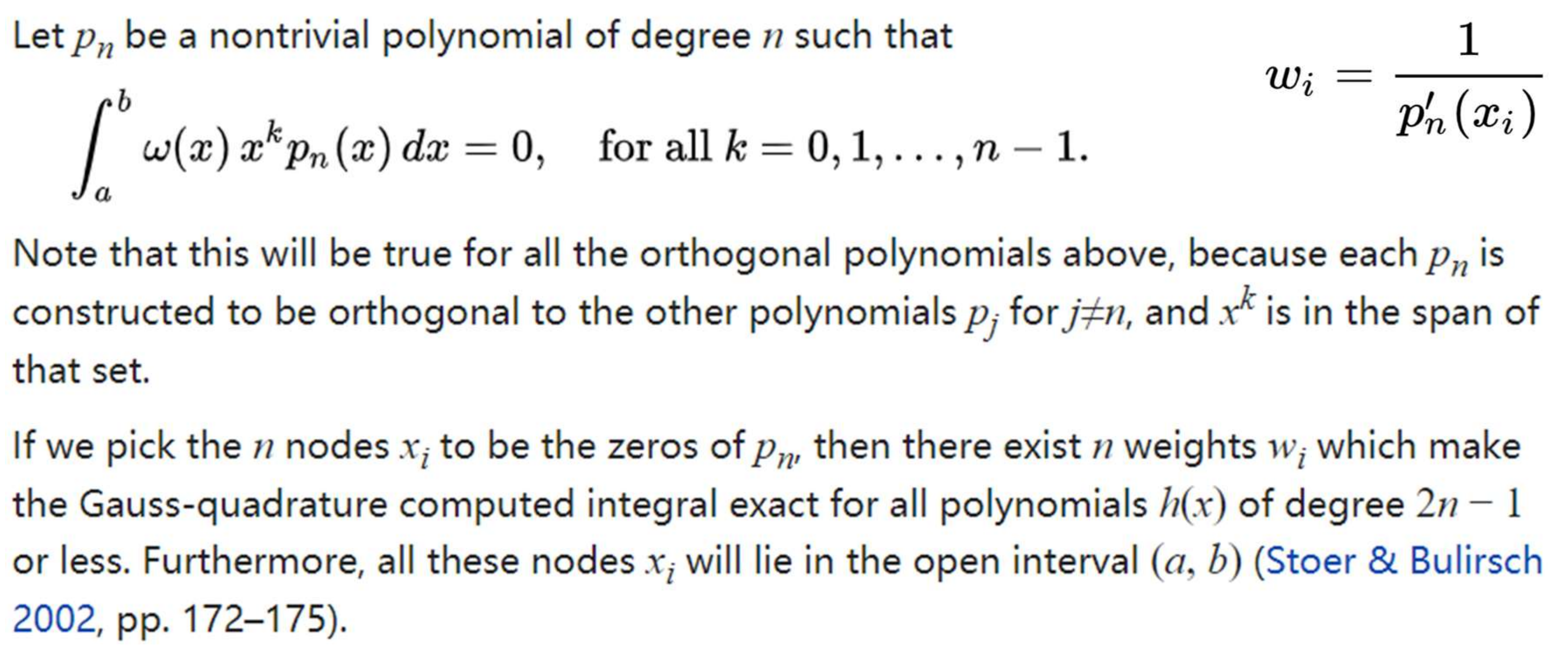
The integration problem can be expressed in a slightly more general way by introducing a positive weight function $\omega$ into the integrand, and allowing an interval other than $[-1, 1]$. That is, the problem is to calculate

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

for some choices of $a$, $b$, and $\omega$. For $a = -1$, $b = 1$, and $\omega(x) = 1$, the problem is the same as that considered above. Other choices lead to other integration rules. Some of these



$$w_i = \frac{1}{p'_n(x_i)} \int_a^b \omega(x) \frac{p_n(x)}{x - x_i} dx$$



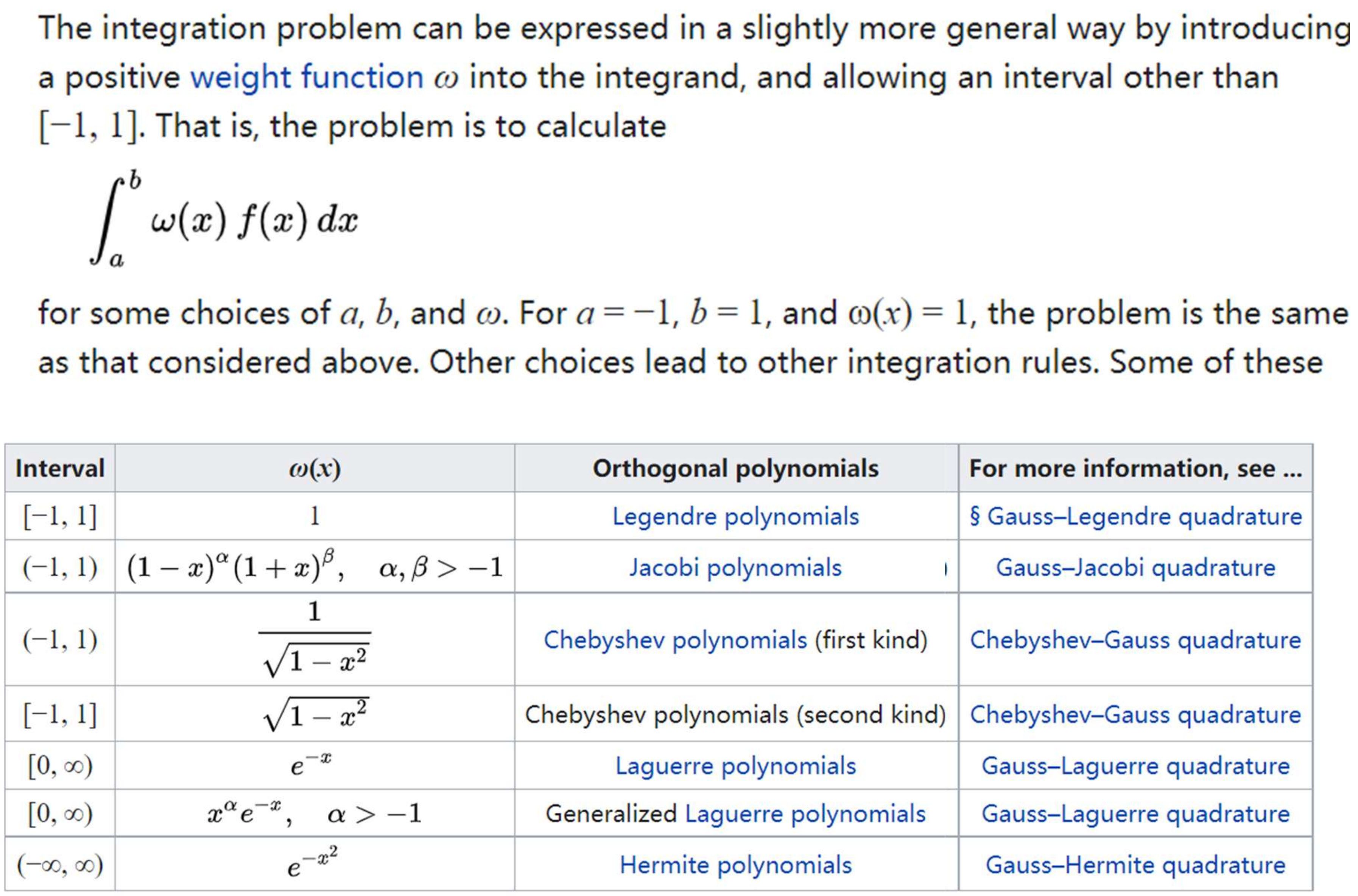
Let $p_n$ be a nontrivial polynomial of degree $n$ such that

$$\int_a^b \omega(x)\, x^k p_n(x)\, dx = 0, \quad \text{for all } k = 0, 1, \ldots, n-1.$$

Note that this will be true for all the orthogonal polynomials above, because each $p_n$ is constructed to be orthogonal to the other polynomials $p_j$ for $j \neq n$, and $x^k$ is in the span of that set.

If we pick the $n$ nodes $x_i$ to be the zeros of $p_n$, then there exist $n$ weights $w_i$ which make the Gauss-quadrature computed integral exact for all polynomials $h(x)$ of degree $2n - 1$ or less. Furthermore, all these nodes $x_i$ will lie in the open interval $(a, b)$ (Stoer & Bulirsch 2002, pp. 172–175).

## 实验1：数值积分的基本算法简介



The integration problem can be expressed in a slightly more general way by introducing a positive weight function $\omega$ into the integrand, and allowing an interval other than $[-1, 1]$. That is, the problem is to calculate

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

for some choices of $a$, $b$, and $\omega$. For $a = -1$, $b = 1$, and $\omega(x) = 1$, the problem is the same as that considered above. Other choices lead to other integration rules. Some of these

| Interval | $\omega(x)$ | Orthogonal polynomials | For more information, see ... |
|---|---|---|---|
| $[-1, 1]$ | $1$ | Legendre polynomials | § Gauss–Legendre quadrature |
| $(-1, 1)$ | $(1-x)^\alpha (1+x)^\beta, \quad \alpha, \beta > -1$ | Jacobi polynomials | Gauss–Jacobi quadrature |
| $(-1, 1)$ | $\frac{1}{\sqrt{1-x^2}}$ | Chebyshev polynomials (first kind) | Chebyshev–Gauss quadrature |
| $[-1, 1]$ | $\sqrt{1-x^2}$ | Chebyshev polynomials (second kind) | Chebyshev–Gauss quadrature |
| $[0, \infty)$ | $e^{-x}$ | Laguerre polynomials | Gauss–Laguerre quadrature |
| $[0, \infty)$ | $x^\alpha e^{-x}, \quad \alpha > -1$ | Generalized Laguerre polynomials | Gauss–Laguerre quadrature |
| $(-\infty, \infty)$ | $e^{-x^2}$ | Hermite polynomials | Gauss–Hermite quadrature |

# 实验2：编程理解求积公式

例1

分别使用矩形法(三种形式)、梯形法、辛普生方法

编程计算$\int_0^1 \frac{1}{1+x^2} dx$, 比较与理论解间的误差。

解：理论解$\int_0^1 \frac{1}{1+x^2} dx = arc\tan \Big|_0^1 = \frac{\pi}{4}$，15位有效数字表示为

0.785398163397448。

讲解程序Exp9_2a.m

# 实验2：编程理解求积公式

下表给出了上面几种方法的数值求解结果，通过对比可以看出辛普森方法的数值计算精度明显优于另外几种方法。

| 数值方法 | | n=50 | n=100 | n=200 |
|---|---|---|---|---|
| 左矩形法 | 数值解 | 0.790381496730813 | 0.787893996730782 | 0.786647121730782 |
| | 误差 | 4.9833e -003 | 2. 4958e -003 | 1.2490e -003 |
| 右矩形法 | 数值解 | 0.780381496730813 | 0.782893996730782 | 0.784147121730782 |
| | 误差 | 5.0167e -003 | 2. 5042e -003 | 1.2510e -003 |
| 中矩形法 | 数值解 | 0.785406496730751 | 0.785400246730781 | 0.785398684230781 |
| | 误差 | 8.3333e -006 | 2.0833e -006 | 5.2083e -007 |
| 梯形法 | 数值解 | 0.785381496730813 | 0.785393996730783 | 0.785397121730781 |
| | 误差 | 1.6667e -005 | 4. 1667e -006 | 1.0417e -006 |
| 辛普森法 | 数值解 | 0.785398163397438 | 0.785398163397448 | 0.785398163397448 |
| | 误差 | 1.0103e-014 | 2. 2204e -016 | 1.1102e-016 |

# 实验2：编程理解求积公式

例2

使用辛普生方法计算以下问题(取n=200)：

(1) $\int_1^2 \frac{\sin x}{x} dx$；　　(2) $\frac{1}{\sqrt{2\pi}} \int_0^3 e^{-\frac{x^2}{2}} dx$。

解:

(1)在程序Exp9_2a.m中ff=1/(1+x^2)替换成sin(x)/x即可(命名Exp9_2b.m)；

(2)在程序Exp9_2a.m中ff=1/(1+x^2)替换成exp(-x^2/2)/sqrt(2*pi)即可(命名Exp9_2c.m)。

## 实验2：编程理解求积公式

具体结果如下：

```
>>format long
>>ss=Exp9_2b(5,1,2,200)
ss=
0.659329906435524
>>ss=Exp9_2c(5,0,3,200)
ss=
0.498650101966968
>>2*ss
ans=
0.997300203933936
```

# 实验2：编程理解求积公式

具体结果如下：

```
>>format long
>>ss=Exp9_2b(5,1,2,200)
ss=
0.659329906435524
>>ss=Exp9_2c(5,0,3,200)
ss=
0.498650101966968
>>2*ss
ans=
0.997300203933936
```

在概率论与数理统计中，经常使用正态分布来构造一些统计量，用于解决实际问题。

积分结果中 0.997300203933936 表示：若 $X \sim N(\mu, \sigma^2)$, 则 $P\{|X-\mu| < 3\sigma\} \approx 0.9973$, 即正态分布的$3\sigma$准则。

# 实验3：二重积分

## 1．二重积分定义

设函数z=f(x, y)是有界闭域D上的有界函数，将闭区域D任意分成n个小闭区域$\Delta\sigma_1, \Delta\sigma_2, \cdots, \Delta\sigma_n$，其中$\Delta\sigma_i$既表示第i个小闭区域，也表示它的面积，$||\Delta\sigma_i||$表示$\Delta\sigma_i$的直径，取$\lambda = \max\{||\Delta\sigma_i||\}(i = 1,2, \ldots, n)$。

在每个$\Delta\sigma_i$上任取一点$(\xi_i, \eta_i)$，作乘积$f(\xi_i, \eta_i)\Delta\sigma_i (i = 1,2, \ldots, n)$，并作和$S = \sum_{i-1}^{n} f(\xi_i, \eta_i)\Delta\sigma_i$，当$\lambda \to 0$时，S极限存在，并称这个极限为函数$z = f(x, y)$在有闭区域D上的定积分，记作$\iint_D f(x, y)d\sigma$，即

$$\iint_D f(x, y)d\sigma = \lim_{\lambda \to 0} \sum_{i-1}^{n} f(\xi_i, \eta_i)\Delta\sigma_i$$

## 实验3：二重积分

2. 单点柱体法

$$\iint_D f(x,y)d\sigma = \lim_{\lambda\to 0}\sum_{i-1}^{n} f(\xi_i,\eta_i)\Delta\sigma_i$$

由定积分的几何意义，上式表示以z=f(x，y)为顶且区域D为底的曲顶柱体的体积。在上式中，取$\Delta\sigma_i$为等分矩形域($\Delta\sigma_i = \Delta x\Delta y$)，并特殊地取$(\xi_i,\eta_i)$为区域$\Delta\sigma_i$的中心点$P_i$，则

$$\iint_D f(x,y)d\sigma = \lim_{\lambda\to 0}\sum_{i-1}^{n} f(P_i)\Delta x\Delta y$$

## 实验3：二重积分

例3　编写程序计算二重积分$\iint_D (xy+y^3)d\sigma$, 其中$D: 0 \le x \le 1, 0 \le y \le 1$。

解：对于此问题是可以理论求解的，具体过程如下：

$$\iint_D (xy+y^3)d\sigma = \int_0^1 dx \int_0^1 (xy+y^3)d\sigma = \int_0^1 x\frac{y^2}{2} + \frac{y^4}{4}\Big|_0^1 dx$$

$$= \int_0^1 \left(\frac{x}{2}+\frac{1}{4}\right) dx = \frac{1}{2}$$

现按照单点柱体法计算该问题

讲解程序Exp9_3a.m

执行计算及计算结果如下：

```
>>[ss, er]=Exp9_3a(0, 1, 0
, 1, 100, 100)
ss=
0.499987500000001
er=
1.249999999941576e -005
>>[ss, er]=Exp9_3a(0, 1, 0
, 1, 500, 500)
ss=
0.499999500000008
er=
4.999999921873055e-007
```

# 实验3：二重积分

上面的例子讨论了D为矩形域的二重积分计算问题，若积分区域为X型$(a \le x \le b, \varphi_1(x) \le y \le \varphi_2(x))$或Y型$(c \le x \le c, \psi_1(y) \le x \le \psi_2(y))$积分区域，则积分计算可以参照例4。

例4 编写程序计算二重积分$\iint_D xyd\sigma$, 其中D为$y = x$与$y = x^2$所围成区域。

解：该问题是有理论解的，具体求解过程如下：区域D可以表示成X型区域$:0 \le x \le 1, x^2 \le y \le x$，则

$$\iint_D (xyy^3)d\sigma = \int_0^1 dx \int_{x^2}^{x} xydy = \int_0^1 x\frac{y^2}{2}\bigg|_{x^2}^{x} dx = \int_0^1 (\frac{x^3}{2} - \frac{x^5}{2})\,dx = \frac{1}{24}$$

讲解程序Exp9_3b.m

理论计算与数值计算结果对比如下：

```
>>1/24
ans=
0.041666666666667
>>[ss,er]=Exp9_3b(0,1,500,0.1)
ss=
0.041702943291168
er=
8.706389880256960e-004
```

# 实验3：二重积分

例5 编写程序计算二重积分$\iint_D \sin\frac{3x+y}{5} e^{-x^2-xy}\, d\sigma$，其中D为$y = 2x$与$y = x^2$所围成区域。

解:

区域D可以表示成X型区域:$0 \le x \le 2, x^2 \le y \le 2x$, 编写程序

讲解程序Exp9_3c.m

执行ss=Exp9_3c(0,2,1000,0.1),可得ss=0.121683298582584,

## 实验4：Matlab内置求积函数（一元）

- 前面已经介绍了符号积分int函数的用法。
- 除int函数之外，MATLAB软件中还有一些数值积分函数，例如trapz、quad、quadl、quadgk等，下面详细介绍其用法。

### 1.trapz 函数

MATLAB 软件中，trapz函数表示使用梯形公式计算数值积分，调用格式：

trapz(Y)：表示使用梯形公式计算Y对其元素下标的数值积分。当Y为向量时，trapz(Y)返回Y的数值积分值，当Y为矩阵时，按矩阵Y的列返回一个以积分值为元素的行向量;

trapz(X,Y):表示使用梯形方法计算Y对X的积分值，X与Y是维数相同的向量。

## 实验4：Matlab内置求积函数（一元）

例6 使用梯形法计算积分$\int_0^{2\pi} e^{-0.5t}\sin(t+\frac{\pi}{4})\,dt$

```
format long
dx=pi/1000;
x=0:dx:2*pi;
y=exp(-0.5*x).*sin(x+pi/4);
z=trapz(y)*dx
z=
0.811859633628809
```

# 实验4：Matlab内置求积函数（一元）

## 2. quad 函数

quad函数使用辛普森方法计算数值积分，使用格式：

quad(fun, a, b, tol)

表示自适应递推辛普生方法计算数值积分, 其中fun为被积函数名，a、b为积分上下限，tol为精度，默认时，默认值为1.0e-6;

quad(fun, a, b, tol, trace)：fun、a、b、tol用法同上,

对于参数trace, 若取1表示用图形展示积分过程，若取0则无图形，默认时，不显示图形。

## 实验4：Matlab内置求积函数（一元）

例7 使用quad函数计算积分$\int_0^1 \frac{4}{1+x^2}dx$。

编制函数文件：

```
function y=ExpFun9_4a(x)
y=4./(x.^2+1);
```

在命令窗口中执行：

```
>>z=quad('ExpFun9_4a',0,1)
z=
3.141592682924567
```

# 实验4：Matlab内置求积函数（一元）

3. quadl 函数

quadl 函数采用递推自适应Lobatto法求数值积分，使用格式与quad 函数类似。相比quad函数，quadl函数适用于精度要求高、被积函数曲线比较平滑的数值积分。

例8 使用quadl函数计算积分$\int_0^2 \frac{1}{x^3-2x-c}dx\ (c=10)$.

编制函数文件：

```
function y=ExpFun9_4b(x,c)
y=1./(x.^3-2*x-c);
```

在命令窗口中输入：

```
>>z=quadl('ExpFun9_4b',0,2,[],[],10)
z=
-0.204337044828626
```

# 实验4：Matlab内置求积函数（一元）

## 4.quadgk 函数

quadgk函数采用自适应Gauss-Kronrod方法进行数值积分计算，适用于高精度和震荡数值积分，支持无穷区间，并且能够处理端点包含奇点的情况，同时还支持沿着不连续函数积分，复数域线性路径的围道积分法。

使用格式：

[q,errbnd]=quadgk(fun,a,b,param1,val1,param2,val2,...)

fun表示被积函数，a、b为积分限，param、val为函数的其他控制参数，

主要有'AbsTol '(绝对误差)、'RelTol '(相对误差)、'Waypoints '(路径点)、'MaxlntervalCount '(最大积分区间数)；

输出项q、errbnd分别表示积分值与误差。

## 实验4：Matlab内置求积函数（一元）

### 4.quadgk 函数

使用该函数进行具体计算时，注意以下几点：

(1)积分限[a,b]可以是无穷区间，但必须快速衰减。

(2)被积函数在端点可以有奇点，如果区间内部有奇点，将以奇点区间划分成多个，也就是说奇点只能出现在端点上。

(3)被积函数可以剧烈震荡。

(4)可以计算不连续积分，此时需要用到'Waypoints ‘参数，'Waypoints'中的点必须严格单调。

(5)可以计算围道积分，此时需要用到'Waypoints'参数，并且为复数，各点之间使用直线连接。

# 实验4：Matlab内置求积函数（一元）

例9 计算积分$\int_0^{+\infty} e^{-x^2} ln^2 x dx$。

编制函数文件：

```
function f=ExpFun9_4c(x)
f=exp(-x.^2).*(log(x)).^2;
```

在命令窗口中执行：

```
>>f=quadgk(@ ExpFun9_4c,0,inf)
f=
1.947522220295560
```

对于被积函数中含有参数情形，该函数仍能计算，例如对例8，只需执行：

```
>>q=quadgk(@(x)ExpFun9_4b(x,10),0,2)
q=
-0.204337044825130
```

# 实验4：Matlab内置求积函数（一元）

例10 计算震荡积分$\int_0^{+\infty} e^{-x^2} sinxdx.$

编制函数文件：

```
function f=ExpFun9_4d(x)
f=exp(-x.^2).*sin(x);
```

命令窗口中执行：

```
>>[q,er]=quadgk(@ExpFun9_4d,0,inf,'RelTol',1e-
8,'AbsTol',1e-10)
q=
0.424436383502022
er=
2.883391888748205e-009
```

# 实验4：Matlab内置求积函数（一元）

例11 计算积分$\int_L \frac{1}{2z-1}dz$，其中路径L为-2-i→2-i→2+i-→-2+i→-2-i.

编制函数文件：

```
function f=ExpFun9_4e(z)
f=1./(2*z-1);
```

命令窗口中执行：

```
>>L=[-2-i,2-i,2+i,-2+i,-2-i];
>>[q,er]= quadgk(@ExpFun9_4e,-2-i,-2-
i,'Waypoints',L)
q=
0.000000000000000+3.141592653589794i
er=
2.447982782903024e-009
```

# 实验5：Matlab内置求积函数（多元）

## 1. int 函数

对于二元函数的符号积分，可以先转化成逐次积分形式，利用int函数进行求解。(前面已经介绍了使用int函数使用方法，不再重复)。

## 2. dblquad 函数

dblquad 函数用于矩形区域的数值积分，调用格式：

dblquad(fun,xmin,xmax,ymin,ymax,tol,method)

表示函数fun在xmin≤x≤xmax， ymin≤y≤ymax范围内求解定积分；tol为误差，默认时，默认值为1.0e-6;

method指求数值积分方法，例如quadl函数等，默认时，默认quad函数。

## 实验5：Matlab内置求积函数（多元）

例12 使用dblquad函数计算积分$\iint_D xe^y + y\sin x d\sigma$，其中$D: 0 \le x \le \pi, 0 \le y \le 1$。

编制函数文件：

```
function f=ExpFun9_4f(x,y)
f=x.*exp(y)+y.*sin(x);
```

命令窗口中执行：

```
>>q=dblquad(@ExpFun9_4f,0,pi,0,1,1e-8)
q=
9.479380948204494
>>q=
dblquad(@ExpFun9_4f,0,pi,0,1,1e-8,@quadl)
q=
9.479380946608845
```

# 实验5：Matlab内置求积函数（多元）

## 3.triplequad 函数

MATLAB软件中，triplequad函数用于长方体区域三重积分的数值计算，使用格式：

triplequad(fun,xmin,xmax,ymin,ymax,zmin,zmax,tol,methd)

表示函数fun在xmin≤x≤xmax,ymin≤y≤ymax,zmin≤z≤zmax,范围内求解定积分；tol、method的意义同上。

## 实验5：Matlab内置求积函数（多元）

例13 计算三重积分$\iiint_D y\sin x + z\cos x\, dv$, 其中 $\Omega$: $0 \le x \le \pi, 0 \le y \le 1, -1 \le z \le 1$。

编制函数文件：

```
function f=ExpFun9_4g(x,y,z)
f=y.*sin(x)+z.*cos(x);
```

命令窗口中执行：

```
>>q=triplequad(@ExpFun9_4g,0,pi,0,1,-1,1,1e-10)
q=
1.99999999999748
```

## 实验6：应用性实验1

### 一、机器转售的最佳时机

人们使用机器从事生产是为获取更大的利润，通常是把购买的机器使用一段时间后转售出去，再买更好的机器。那么，一台机器使用多少时间再转售出去才能获得最大的利润，是使用机器者最想知道的。

现有一种机器，由于折旧等因素其转售价格R(t)服从函数关系$R(t)=\frac{3A}{4}e^{-\frac{t}{96}}$(元)，其中t是时间，单位是周，A是机器的最初价格。此外，还知道任何时间t，机器开动就能产生$p=\frac{A}{4}e^{-\frac{t}{48}}$的利润。

问：该机器使用了多长时间后转售出去，能使总利润最大？最大利润是多少？机器卖了多少钱？

# 实验6：应用性实验1

## 一、机器转售的最佳时机

设机器总共使用了x周，总收入为S(x)。由于总收入=机器转售收入+机器创造利润收入

因而$S(x)=\frac{3A}{4}e^{-\frac{x}{96}}+\int_0^x \frac{A}{4}e^{-\frac{t}{48}}dt, 0<x<+\infty$

令$S'(x)=0$, 得$-\frac{1}{96}\frac{3A}{4}e^{-\frac{x}{96}}+\frac{A}{4}e^{-\frac{x}{48}}=0$

求得驻点$x_0$，即为总收入函数最大的点，然后代入可以得到总利润$S(x_0)$。使用MATLAB求解上述问题，过程如下：

```
>>syms x A
>>ff=-1/96*(3*A/4)*exp(-x/A6)+A/4*exp(-x/48);
>>xx=solve(ff,x)
xx=
480*log(2)
```

# 实验6：应用性实验1

## 一、机器转售的最佳时机

使用MATLAB求解上述问题，过程如下：

```
>>syms x A
>>ff=-1/96*(3*A/4)*exp(-x/96)+A/4*exp(-x/48);
>>xx=solve(ff,x)
xx=
480*log(2)
>>s1=3*A/4*exp(-xx/96)
s1=
3/128*A
>>s2=int(A/4*exp(-x/48),x,0,xx);
>>ss=s1+s2
ss=
3075/256*A
```

由计算结果知求得唯一驻点480ln2，即在使用时间 $x = 480ln2 \approx 32.7106 \approx 333$周时获得最大收入为$S(x) = \frac{3075}{256}A\ (12.0117A)$，最大利润为11.0117A，机器卖了 $\frac{3}{128}A\ (0.02334A)$。

# 实验7：应用性实验2

## 二、人造卫星轨道的长度

　　人造地球卫星的轨道可视为平面上的椭圆。我国第一颗人造地球卫星近地点距地球表面439km，远地点距地球表面2384km，地球半径为6371km，求该卫星的轨道长度。

# 实验7：应用性实验2

## 二、人造卫星轨道的长度

人造地球卫星的轨道可视为平面上的椭圆。我国第一颗人造地球卫星近地点距地球表面439km，远地点距地球表面2384km，地球半径为6371km，求该卫星的轨道长度。

解：设卫星轨道椭圆的参数方程为

$$\begin{cases} x = a\cos t \\ y = b\sin t \end{cases} \quad 0 \le t \le 2\pi$$

式中：a、b分别表示椭圆的长、短半轴，且

a=6371+2384=8755，b=6371+439=6810。

由对弧长的曲线积分知(使用对称性)

$$L = 4\int_0^{\frac{\pi}{2}} \sqrt{x'^2(t) + y'^2(t)}dt$$

$$= 4\int_0^{\frac{\pi}{2}} \sqrt{a^2 sin^2 t + b^2 cos^2 t}dt$$

# 实验7：应用性实验2

## 二、人造卫星轨道的长度

编制函数文件：

```
function f=ExpFun9_7a(t)
a=8755;b=6810;
f=4*sqrt(a^2*sin(t).^2+b^2*cos(t).^2);
```

在命令窗口中执行：

```
>>q=quadl('ExpFun9_7a',0,pi/2,1e-6)
```

我国第一颗人造卫星的轨道长度为49089.965km。

# 实验8：用数值积分计算$\pi$

半径为1的圆称为单位圆，它的面积等于$\pi$. 只要算出单位圆的面积，就算出了$\pi$.

以单位圆的圆心为原点建立直角坐标系，则单位圆在第一象限内的部分G是一个扇形，由曲线$y=\sqrt{1-x^2}(x\in[0,1])$及两条坐标轴围成，它的面积$S=\frac{\pi}{4}$算出了$S$的近似值，它的4倍就是$\pi$的近似值.

扇形面积S实际上就是定积分

$$\int_0^1 \sqrt{1-x^2}\,dx=\frac{\pi}{4}.$$

与$\pi$有关的定积分很多，比如$\frac{1}{1+x^2}$的定积分（本次实验的例1）

$$\int_0^1 \frac{1}{1+x^2}dx=\frac{\pi}{4}$$

就比$\sqrt{1-x^2}$的定积分更容易计算，更适合于用来计算$\pi$.