Computational Techniques and Numerical Calculations

European Master in Theoretical Chemistry and Computational Modelling

Elena Formoso
University of the Basque Country

November 22, 2022

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Schedule

- 22th November: **Introduction to Numerical Integration** (Elena Formoso)
- 24th November: Basic Concepts in Linear Algebra (Pere Alemany)
- 29th November: Introduction to Root-Finding and Optimization of Functions (Elena Formoso)
- 1st December: Linear Systems (Pere Alemany)
- 16th to 20th Jaunary (Intensive Course): practical exercises.

Deadlines

- 10th December: Numerical Integration homework (continuous evaluation, no marks)
- 17th December: Root-finding and function optimization homework (continuous evaluation, no marks)
- 10th March: Evaluation homeworks (gradable)

Evaluation

Evaluation: Percentages

- Local Part $\longrightarrow$Fortran: 30 %
- Numerical Integration + Root-Finding and Function's Optimization (Elena Formoso): 35 %
- Algebra (Pere Alemany): 35 %

Outline

1. Introduction
2. Numerical Integration Methods
   - Interpolation Methods I: The Newton-Cotes integration rules
   - Extrapolation Methods: Richardson extrapolation and Romberg integration
   - Interpolation Methods II: Gaussian Quadrature
   - Monte-Carlo Method

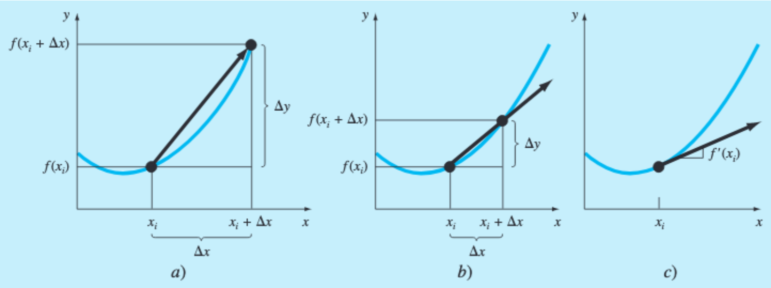
Outline

1. Introduction

2. Numerical Integration Methods
   - Interpolation Methods I: The Newton-Cotes integration rules
   - Extrapolation Methods: Richardson extrapolation and Romberg integration
   - Interpolation Methods II: Gaussian Quadrature
   - Monte-Carlo Method

Calculus

- Calculus is the mathematical study of change. The scientists Isaac Newton and Gottfried Wilhelm Leibniz are credited with the invention of it in XVII century.
- Calculus has two major branches
  - **Differential calculus** (rates of change and slopes of curves): $\frac{\Delta y}{\Delta x} = \frac{f(x_i+\Delta x)-f(x_i)}{\Delta x}$ if $\Delta x \to 0 \Rightarrow \frac{dy}{dx}$

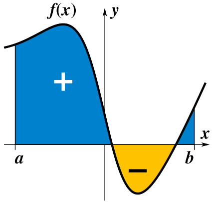


Calculus

- Calculus is the mathematical study of change. The scientists Isaac Newton and Gottfried Wilhelm Leibniz are credited with the invention of it in XVII century.
- Calculus has two major branches
  - **Differential calculus** (rates of change and slopes of curves): $\frac{\Delta y}{\Delta x} = \frac{f(x_i+\Delta x)-f(x_i)}{\Delta x}$ if $\Delta x \to 0 \Rightarrow \frac{dy}{dx}$
  - **Integral calculus**. Accumulation of quantities and the areas under and between curves.

Many scientific problems require solving **definite integrals**, usually assuming the Riemann integral, widely used by physicists and engineers.

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



We will deal with the mathematical problem of evaluating this definite integral

Analytical Integration

Fundamental theorem of calculus

Let f be continuous on $[a,b]$. If F is any antiderivative (indefinite integral) for f on $[a,b]$, then

$$F = \int f(x)dx \Longrightarrow \frac{dF}{dx} = f(x) \qquad \Longrightarrow \int_a^b f(x)\ dx = F(b) - F(a)$$

Example: Evaluate $\int_2^4 x^2 dx$

$$F = \int x^2 dx \Longrightarrow F?? \text{ so that } \frac{dF}{dx} = x^2 \Longrightarrow F = \frac{x^3}{3} + const.$$

$$\Longrightarrow \int_2^4 x^2 dx = F(4) - F(2) = \frac{4^3}{3} - \frac{2^3}{3} = 18.667$$

♣ The integration may lead to an algebraic function, fast and exact to evaluate.

But ...

Despite of having an "exact" expression one could:

- face numerical difficulties evaluating the antiderivative F as division by zero, etc...
- face an important number of computations: time-consuming processes.
- find that elementary functions, as log and *arctan*(*transcendental functions*), can only be evaluated to a certain degree of accuracy.
- Therefore, in many of these cases numerical integration can be more efficient.

Evaluate $\int_0^1 \frac{dx}{1+x^4}$

$$F = \int \frac{1}{1+x^4}dx = \frac{1}{4\sqrt{2}} \log \frac{x^2 + x\sqrt{2} + 1}{x^2 - x\sqrt{2} + 1} + \frac{1}{2\sqrt{2}} \arctan \frac{x\sqrt{2}}{1 - x^2} + const.$$

In addition ...

- For many functions the antiderivative F cannot even be expressed in terms of elementary functions, that is, F cannot be written as a combination of algebraic, exponential or logarithmic operations.

Examples: $\int e^{-x^2} dx$ and $\int \sin(\sin x) dx$

The indefinite integrals cannot be expressed as an elementary function. In these situations numerical integration is the only solution.

- Sometimes one has to integrate experimental data (discrete points). In such cases analytical integration is not possible and numerical integration is again the only solution.

Outline

1. Introduction

2. Numerical Integration Methods
    - Interpolation Methods I: The Newton-Cotes integration rules
    - Extrapolation Methods: Richardson extrapolation and Romberg integration
    - Interpolation Methods II: Gaussian Quadrature
    - Monte-Carlo Method

Classification:

Numerical integration methods can be classified in three groups:

1. Equally spaced points:
    - Interpolation: The Newton-Cotes quadrature formulas
    - Extrapolation: The Richardson and Romberg methods
2. Unequally spaced points:
    - Interpolation: Gaussian quadrature formulas
3. Selection of random points:
    - Monte-Carlo method

Objetives

1. **Understand** how to obtain different Newton-Cotes formulas
2. **Understand** the theory of Richardson extrapolation and how to apply Romberg integration
3. **Be able to choose** between different formulas for each problem
4. **Get used to use** different software to implement numerical methods in problem solving

Principle of approximation

It is fruitful method for obtaining quadrature formulas:

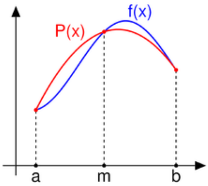
$$\boxed{\int_a^b f(x)dx \approx \int_a^b g(x)dx + Error}$$

the function $f(x)$ is <u>approximated</u> by another function $g(x)$, whose antiderivative can be expressed as a formula

♣ Interpolative methods (Newton-Cotes and Gaussian quadratures) are based on this principle.

Approximation by using interpolating polynomials

Numerical approximation that relies on: $f(x)$ is approximated by an interpolating polynomial $P_{n-1}(x)$, which interpolates $f(x)$ at some n points $(x_1, x_2, \cdots, x_n)$, called interpolation abcissas in the [a, b] interval:



$$
\begin{aligned}
P_{n-1}(x_j) &= f(x_j); \quad j = 1, \cdots, n. \\
P_{n-1}(x) &= \sum_{i=1}^{n} a_i \cdot \underbrace{B_{n-1,i}(x)}_{\text{interp. basis}} \\
B_{n-1,i}(x) &= \begin{cases} (1, x, x^2, x^3, \cdots x^{n-1}) & \text{(algebraical)} \\ (1, \sin x, \cos x, \sin 2x, \cdots) & \text{(trigonometric)} \\ (1, e^{a_1 x}, e^{a_2 x}, \cdots) & \text{(exponential)} \end{cases}
\end{aligned}
$$

Approximation by means of polynomial interpolation is of great importance for the construction of quadrature formulas (numerical integration formulas).

Using algebraic polynomials: Lagrange interpolating formula.

Let's use $B = \{x^i; \quad i = 0, \cdots, n-1\}$, as the n interpolating polynomials

$$\boxed{P_{n-1}(x_k) = \sum_{i=1}^{n} a_i \cdot B_{n-1,i}(x_k) = \sum_{i=1}^{n} a_i . x_k^{i-1} = f(x_k); \quad k = 1, \ldots, n}$$

Or written in matrix form:

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \cdots \\ f(x_n) \end{bmatrix}$$

$$\Phi \cdot \mathbf{A} = \mathbf{F}$$

$$\mathbf{A} = \Phi^{-1} \cdot \mathbf{F} = \mathbf{D} \cdot \mathbf{F}$$

Generalizing:

$$P_{n-1}(x) = \sum_{i=1}^{n} a_i \cdot B_{n-1,i}(x) = \begin{bmatrix} 1 & x & \cdots & x^{n-1} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} = \mathbf{B} \cdot \mathbf{A} = \mathbf{B} \cdot \mathbf{D} \cdot \mathbf{F}$$

Approximation: Lagrange polynomials

Using algebraic polynomials: Lagrange interpolating formula.

$$\mathbf{B}\cdot\mathbf{D} = \begin{bmatrix} 1 & x & \cdots & x^{n-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix}^{-1} = \begin{bmatrix} L_1(x) & L_2(x) & \cdots & L_n(x) \end{bmatrix}$$

The polynomials $\{L_k(x); k = 1, ..., n\}$ are called Lagrangian polynomials.

$$\boxed{P_{n-1}(x) = \sum_{j=1}^{n} L_j(x).f(x_j)}$$

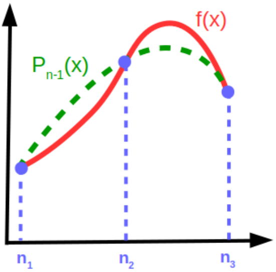
$$\int_a^b f(x)dx \approx \int_a^b P_{n-1}(x)dx = \underline{\underline{I_{rule}}} = \sum_{i=1}^{n} f(x_i) \underbrace{\int_a^b L_i(x)dx}_{\omega_i} = \boxed{\sum_{i=1}^{n} f(x_i)\cdot\omega_i}$$

ω_i are the <u>weights</u> of the function $P_{n-1}(x)$ at the n quadrature abscissas.

Summarizing

So far, what we have done:

$$\int_a^b f(x)dx \approx \int_a^b P_{n-1}(x)dx = \boxed{\sum_{i=1}^{n} f(x_i) \cdot \omega_i}$$



- We are going to use interpolation abcissas to create a new polynomial function.
- Note that each f(x_i) may have a different weight (ω_i).

Lagrange Interpolating Formula

The Lagrangian polynomials are given by:

$$L_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

The final interpolation formula is:

$$P_{n-1}(x) = \sum_{i=1}^{n} f(x_i) \cdot \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

The quadrature formula or integration rule is:

$$\boxed{I_{rule} = \int_a^b P_{n-1}(x)dx = \sum_{i=1}^{n} f(x_i) \underbrace{\int_a^b \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}\, dx}_{\omega_i}}$$

And the error of the integration is:

$$\boxed{E_{rule} = \int_a^b [f(x) - P(x)]dx = \int_a^b \frac{f^{(n)}(\xi)}{n!} \prod_{i=1}^{n}(x - x_i)dx}$$

The use of the Lagrangian functions leads the Newton-Cotes quadrature formula.

Outline

1. Introduction

2. Numerical Integration Methods
   - Interpolation Methods I: The Newton-Cotes integration rules
   - Extrapolation Methods: Richardson extrapolation and Romberg integration
   - Interpolation Methods II: Gaussian Quadrature
   - Monte-Carlo Method

Newton-Cotes interpolation method

The Newton-Cotes formulas are a very useful technique for numerical integration, based on equidistant interpolation abscissas in a [a,b] interval.

$$I_{rule} = \int_a^b P_{n-1}(x)dx = \sum_{i=1}^{n} f(x_i) \underbrace{\int_a^b \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}\, dx}_{\omega_i}$$

Depending on the value of n (number of interpolating points, abscissa points):

- $n = 1$ Rectangle rule
- $n = 2$ Trapezoids rule
- $n = 3$ Simpson rule
- $\vdots$

For each of these methods:

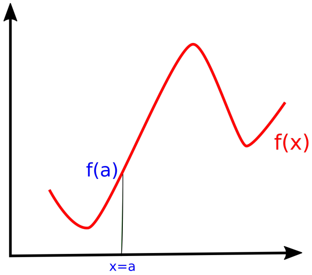
- Simple rule
- Composite rule

Newton-Cotes: **Rectangle rule**

- n = 1
- $x_1 = a$

$P_0(x) = \sum_{i=1}^{1} f(x_i) \cdot \prod_{j\neq i}^{1} \frac{x-x_j}{x_i-x_j} = f(a)$

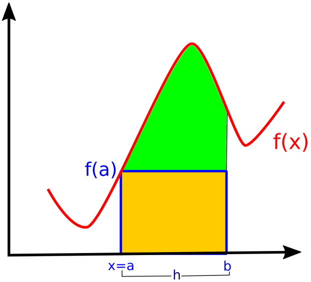
Simple rule:



Newton-Cotes: **Rectangle rule**

- n = 1
- $x_1 = a$

$P_0(x) = \sum_{i=1}^{1} f(x_i) \cdot \prod_{j\neq i}^{1} \frac{x-x_j}{x_i-x_j} = f(a)$



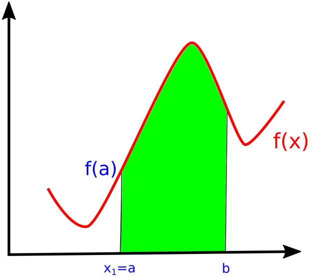
Simple rule:

$dx = b - a = h$

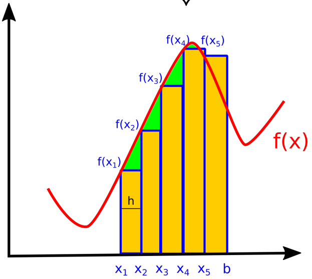
$$I_R = \int_a^b f(x)\,dx = \int_a^b P_{n-1}(x)\,dx = \int_a^b P_0(x)\,dx = \int_a^b f(a)\,dx = \boxed{f(a)h}$$

$$E_R = \frac{f^n(\xi)}{n!}\int_a^b (x-a)\,dx = \frac{f(\xi)}{2}h^2 = O(h^2); \quad \xi \in [a,b]$$

Composite rule:



$\Downarrow$



$h = (b-a)/N;$ $\quad N = \#$ of subintervals

$x_i = a + (i-1)h; \;\; i = 1, ..., N$

$$I_{CR} = \int_a^b f(x)\,dx = \boxed{h \sum_{i=1}^{N} f(x_i)}$$

$$E_{CR} \;\; = \;\; N \cdot \frac{f(\xi)}{2} h^2 = O(h^2); \quad \xi \in [a, b]$$

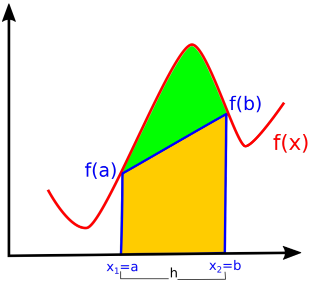
- $n = 2$
- $x_1 = a$; $x_2 = b$

$$P_1(x) = \sum_{i=1}^{2} f(x_i) \cdot \prod_{\substack{j=1 \\ j\neq i}}^{2} \frac{x - x_j}{x_i - x_j} = f(a)\cdot\frac{x-b}{a-b} + f(b)\cdot\frac{x-a}{b-a} = \frac{f(a)(b-x) + f(b)(x-a)}{b-a}$$

Simple rule:

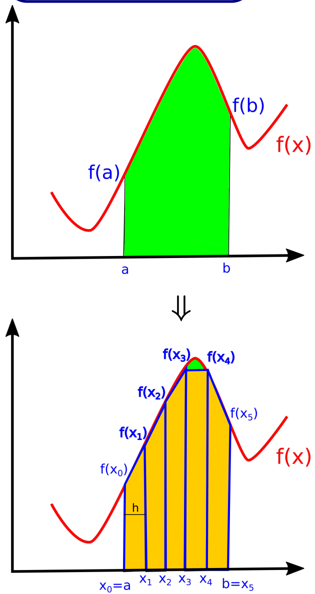
$$dx = b - a = h$$

$$I_T = \int_a^b P_1(x)\,dx = \boxed{h\left[\frac{f(a)+f(b)}{2}\right]}$$



$$E_T = \frac{f^n(\xi)}{n!}\int_a^b (x-a)(x-b)\,dx = -\frac{f^2(\xi)}{12}h^3 = O(h^3);\quad \xi\in[a,b]$$

Composite rule:



$h = (b-a)/N$; $N = \#$ of subintervals

$x_i = a + ih; \; i = 0, ..., N$

$$I_{CT} = \boxed{\frac{h}{2}\left[f(a) + f(b) + 2\cdot\sum_{i=1}^{N-1} f(x_i)\right]}$$

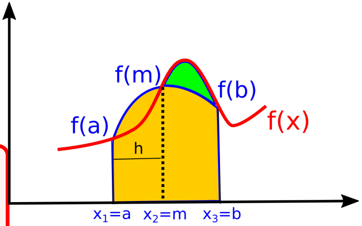
$$E_{CT} \;=\; -\,N\cdot\frac{f^2(\xi)}{12}h^3 = O(h^3), \quad \xi \in [a, b$$

- n = 3
- $x_1 = a$; $x_2 = \frac{a+b}{2}$; $x_3 = b$

$$P_2(x) = \sum_{i=1}^{3} f(x_i) \prod_{\substack{j=1 \\ j \neq i}}^{3} \frac{x-x_j}{x_i-x_j} = f(a) \cdot \frac{x-m}{a-m} \cdot \frac{x-b}{a-b} + f(m) \cdot \frac{x-a}{m-a} \cdot \frac{x-b}{m-b} + f(b) \cdot \frac{x-a}{b-a} \cdot \frac{x-m}{b-m}$$

Simple rule:

$$dx = h = (b-a)/2$$

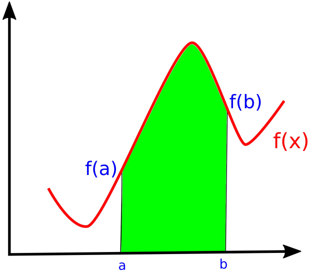


$$I_S = \int_a^b P_2(x)\, dx = \boxed{\frac{h}{3}\,[f(a) + 4.f(m) + f(b)]}$$

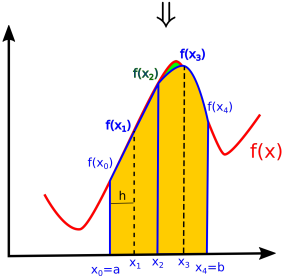
$$I_S = \boxed{\frac{(b-a)}{6}.\,[f(a) + 4.f(m) + f(b)]}$$

$$E_S = \frac{f^n(\xi)}{n!} \int_a^b (x-a)(x-m)(x-b)\, dx = -\frac{f^4(\xi)}{90} \cdot h^5 = -\frac{f^4(\xi)}{2880} \cdot (b-a)^5, \quad \xi \in [a, b]$$

Composite rule:



$$h = (b-a)/(2N); \quad N = \# \text{ of subintervals}$$
$$x_i = a + ih; \;\; i = 0, \ldots, 2N$$



$$I_{CS} = \boxed{\frac{h}{3}\left[f(a) + f(b) + 2\sum_{\substack{\text{even}\\ i=2}}^{2N-2} f(x_i) + 4\sum_{\substack{\text{odd}\\ i=1}}^{2N-1} f(x_i)\right]}$$

$$E_{CS} = -N \cdot \frac{f^4(\xi)}{90} \cdot h^5 = O(h^5); \quad \xi \in [a, b]$$

Higher order Newton-Cotes rules

For $n = 4, 5, 6, \cdots$; $f_i \equiv f(x_i)$; $i = 1, ..., n$

$h = b - a$

$\xi \in [a, b]$

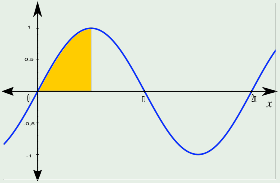
$$
\begin{aligned}
I_1 = P_0 &= h \cdot (f_1) \\
I_2 = P_1 &= \frac{h}{2} \cdot (f_1 + f_2) \\
I_3 = P_2 &= \frac{h}{6} \cdot (f_1 + 4f_2 + f_3) \\
I_4 = P_3 &= \frac{h}{8} \cdot (f_1 + 3f_2 + 3f_3 + f_4) \\
I_5 = P_4 &= \frac{h}{90} \cdot (7f_1 + 32f_2 + 12f_3 + 32f_4 + 7f_5) \\
&\vdots
\end{aligned}
$$

$$
\begin{aligned}
E_R &= \frac{h^2}{2} \cdot f(\xi) \\
E_T &= -\frac{h^3}{12} \cdot f^2(\xi) \\
E_S &= \frac{h^5}{90} \cdot f^4(\xi) \\
E_4 &\propto h^5 \cdot f^4(\xi) \\
E_5 &\propto h^7 \cdot f^6(\xi) \\
&\vdots
\end{aligned}
$$

Newton-Cotes example

Example:

$$I = \int_0^{\pi/2} sin(x)dx = 1.0$$



Rule	Simple	Composite			
		N=2	N=10	N=20	N=100
Rectangle	0.00000	0.55536	0.91940	0.96021	0.99212

Newton-Cotes example

Example:

$$I = \int_0^{\pi/2} sin(x)dx = 1.00$$

Rule	Simple	Composite			
		N=2	N=10	N=20	N=100
Rectangle	0.00000	0.55536	0.91940	0.96021	0.99212
Trapezoid	0.78540	0.94806	0.99794	0.99948	0.99998

Newton-Cotes example

Example:

$$I = \int_0^{\pi/2} sin(x)dx = 1.00$$

Rule	Simple	Composite			
		N=2	N=10	N=20	N=100
Rectangle	0.00000	0.55536	0.91940	0.96021	0.99212
Trapezoid	0.78540	0.94806	0.99794	0.99948	0.99998
Simpson's	1.00228	1.00228	1.00000	1.00000	1.00000

Outline

1. Introduction

2. Numerical Integration Methods
   - Interpolation Methods I: The Newton-Cotes integration rules
   - Extrapolation Methods: Richardson extrapolation and Romberg integration
   - Interpolation Methods II: Gaussian Quadrature
   - Monte-Carlo Method

Richardson extrapolation method

- It is a remarkable, fundamental but often overlooked tool for speeding up the convergence of a sequence.
- It is based on the extrapolation of two function values calculated at $\lim_{h\to 0} R(h)$ and $\lim_{h\to 0} R(\frac{h}{2})$ to improve the result.
- It eliminates errors of the form $E(h) = Ch^n$

Richardson extrapolation method

1. Approximate G function by $R(h_1)$, where $h_1 = x_a - x_0$

$$G = R(h_1) + Ch_1^n \qquad \text{Error} = O(h_1^n)$$

$R(h_1)$ can be expanded in a Taylor series:

$$R(h) = R(x_0) + c_1 h + c_2 h^2 + c_3 h^3 + \cdots \qquad \text{Error} = O(h)$$

$$\text{so, } \lim_{h \to 0} R(h) = R(x_0)$$

2. Similarly, approximate G by $R(h_2)$, where $h_2 = \frac{h_1}{2} = x_b - x_0$

$$G = R(h_2) + Ch_2^n = R(h_2) + \tfrac{1}{2} Ch_1^n \qquad \text{Error} = O(h_1^n)$$

$$R(h_2) = R(h_1/2) \quad = \quad R(x_0) + c_1 \frac{1}{2} h + c_2 \frac{1}{4} h^2 + c_3 \frac{1}{8} h^3 + \cdots \qquad Error = O(h)$$

3. **Extrapolation**. G may be more accurate than R with small h: less rounding errors and/or less number of calculations.

$$\begin{cases} \left(G = R(h_2) + Ch_2^n\right) 2^n \\ -\ G = R(h_1) + Ch_1^n \end{cases}$$

Richardson extrapolation method

$$\boxed{G(h) = \frac{2^n R(h/2) - R(h)}{2^n - 1}}$$

n	Rule	Function	Error
1	$G(h) = 2R(h/2) - R(h)$	$G(h) = R(x_0) - c_2 \frac{1}{2} h^2 - c_3 \frac{3}{4} h^3 + \dots$	$O(h^2)$
2	$G(h) = \frac{4R(h/2) - R(h)}{3}$	$G(h) = R(x_0) + c_3 \frac{3}{8} h^3 + \dots$	$O(h^3)$

$\dots$

So, no need of new calculations to improve substantially the final result.

Romberg integration algorithm

Romberg integration:

- **Iterative Richardson extrapolation applied on the Composite Trapezoidal Rule function**
- Provides two mechanism for improving the accuracy
    - Reduce h parameter
    - Apply Richardson extrapolation
- $I_{CT} = \int_a^b f(x)dx \approx \frac{h_k}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{N-1} f(x_i)\right]$
- $h_k = \frac{(b-a)}{N} = \frac{(b-a)}{2^{k-1}};\ k = 1, 2, 3...$
- For each k value, Richardson extrapolation can be applied $k-1$ times to previously computed approximation
- Both mechanism are applied simultaneously

$$
\begin{array}{|lllll}
R(1,1) & & & & \\
R(2,1) & R(2,2) & & & \\
R(3,1) & R(3,2) & R(3,3) & & \\
R(4,1) & R(4,2) & R(4,3) & R(4,4) & \\
\cdots & \cdots & \cdots & \cdots & \ddots \\
\uparrow & \uparrow & \uparrow & \uparrow & \\
O(h^2) & O(h^4) & O(h^6) & O(h^8) &
\end{array}
$$

- Table elements: $R(k, j)$
- **k**: how many different h are computed initially. Related to the number of trapezoids (subintervals)

- $R_{1,1} \longrightarrow k = 1 \Longrightarrow N = 2^{(1-1)} = 1$ interval

$h_1 = (b - a)$

$$
\boxed{R_{1,1} = \frac{h_1}{2}\left[f(a) + f(b)\right]}
$$

Romberg integration algorithm

- $R_{2,1} \longrightarrow k = 2 \Longrightarrow N = 2^{(2-1)} = 2$ subintervals

$h_2 = \frac{(b-a)}{2} = \frac{h_1}{2}$

$$I_{CT} = \int_a^b f(x)dx \approx \frac{h_k}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{N-1} f(x_i)\right]$$

$$R_{2,1} = \frac{h_2}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{2-1} f(a + ih_2)\right]$$

$$\boxed{R_{2,1} = \frac{h_2}{2}\left[f(a) + f(b) + 2f(a + h_2)\right]}$$

- $R_{3,1} \longrightarrow k = 3 \Longrightarrow N = 2^{(3-1)} = 4$ subintervals

$h_3 = \frac{(b-a)}{4} = \frac{h_1}{4} = \frac{h_2}{2}$

$$R_{3,1} = \frac{h_3}{2}\left[f(a) + f(b) + 2\sum_{i=1}^{4-1} f(a + ih_3)\right]$$

$$\boxed{R_{3,1} = \frac{h_3}{2}\left[f(a) + f(b) + 2\left[f(a + h_3) + f(a + 2h_3) + f(a + 3h_3)\right]\right]}$$

Romberg integration algorithm

$$R_{2,1} = \frac{h_2}{2}\left[f(a) + f(b) + 2f(a+h_2)\right] =$$
$$\frac{1}{2}\left[\frac{h_1}{2}\left(f(a) + f(b)\right) + 2\frac{h_1}{2}f(a+h_2)\right] = \frac{1}{2}\left[R_{1,1} + h_1 f(a+h_2)\right]$$

$$R_{3,1} =$$
$$\frac{h_3}{2}\left[f(a) + f(b) + 2\left[f(a+h_3) + f(a+2h_3) + f(a+3h_3)\right]\right] =$$
$$\frac{1}{2}\left[\frac{h_2}{2}\left(f(a) + f(b) + 2\left[f(a+h_3) + f(a+2\frac{h_2}{2}) + f(a+3h_3)\right]\right)\right] =$$
$$\frac{1}{2}\left[\frac{h_2}{2}\left(f(a) + f(b) + 2f(a+h_2)\right) + h_2\left(f(a+h_3) + f(a+3h_3)\right)\right] =$$
$$\frac{1}{2}\left[R_{2,1} + h_2\left(f(a+h_3) + f(a+3h_3)\right)\right]$$

$$\boxed{R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k)\right] \; ; k > 1 = j}$$

Romberg integration algorithm

$$\begin{array}{l|lllll} & \boxed{R(1,1)} & & & & \\ & R(2,1) & R(2,2) & & & \\ & R(3,1) & R(3,2) & R(3,3) & & \\ & R(4,1) & R(4,2) & R(4,3) & R(4,4) & \\ & \cdots & \cdots & \cdots & \cdots & \ddots \end{array}$$

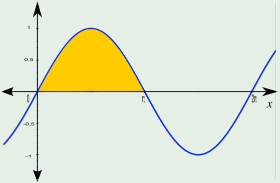
$$R_{1,1} = \frac{h_1}{2}\left[f(a) + f(b)\right]$$

$$R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k)\right] \ ; k > 1 = j$$

$$R_{k,j} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1} \qquad ; k \geq j > 1$$

Example

$I = \int_0^\pi sin(x)dx = 2.0$



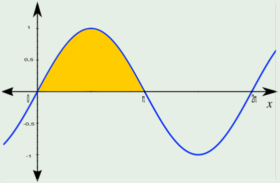
We start with $h_1 = (b - a) = \pi$

$$R_{1,1} = \frac{h_1}{2}\left[f(a) + f(b)\right] = \frac{\pi}{2}(0 + 0) = 0$$

	$R_{k,1}$
$k = 1$	0.00000000

Example

$I = \int_0^{\pi} sin(x)dx = 2.0$



We start with $h_1 = (b-a) = \pi$

$$
\begin{aligned}
h_k &= (b-a)/2^{k-1} \Longrightarrow h_2 = \pi/2 \\
R_{k,1} &= \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a+(2i-1)h_k)\right] \qquad ; k > 1 = j \\
R_{2,1} &= \frac{1}{2}\left[R_{1,1} + h_1 f(a+h_2)\right] = \frac{1}{2}\left[R_{1,1} + \pi\, sin\left(\frac{\pi}{2}\right)\right]
\end{aligned}
$$

	$R_{k,1}$
$k = 1$	0.00000000
$k = 2$	1.57079633

Example

$$I = \int_0^{\pi} sin(x)dx = 2.0$$

We start with $h_1 = (b-a) = \pi$

$$\begin{aligned} h_k &= (b-a)/2^{k-1} \Longrightarrow h_2 = \pi/2 \\ R_{k,j} &= R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1}-1} \qquad ; k \geq j > 1 \\ R_{2,2} &= R_{2,1} + \frac{R_{2,1} - R_{1,1}}{4^{2-1}-1} = R_{2,1} + \frac{R_{2,1} - R_{1,1}}{3} \end{aligned}$$

	$R_{k,1}$	$R_{k,2}$
$k=1$	0.00000000	
$k=2$	1.57079633	2.09439510

Example:

$$I = \int_0^{\pi} sin(x)dx = 2.0$$

We start with $h_1 = (b-a) = \pi$

$$h_k = (b-a)/2^{k-1} \Longrightarrow h_3 = \pi/4$$

$$R_{k,1} = \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a+(2i-1)h_k)\right] \qquad ; k > 1$$

$$R_{3,1} = \frac{1}{2}\left[R_{2,1} + h_2\left(f(a+h_3) + f(a+3h_3)\right)\right] = \frac{1}{2}\left[R_{2,1} + \frac{\pi}{2}\left(\sin\left(\frac{\pi}{4}\right) + \sin\left(\frac{3\pi}{4}\right)\right)\right]$$

	$R_{k,1}$	$R_{k,2}$
$k=1$	0.00000000	
$k=2$	1.57079633	2.09439510
$k=3$	1.89611890	
	$\vdots$	$\vdots$
	$O(h^2)$	$O(h^4)$

Example:

$$I = \int_0^{\pi} sin(x)dx = 2.0$$

We start with $h_1 = (b-a) = \pi$

$$\begin{aligned} h_k &= (b-a)/2^{k-1} \Longrightarrow h_3 = \pi/4 \\ R_{k,j} &= R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1}-1} \qquad ; k \geq j > 1 \\ R_{3,2} &= R_{3,1} + \frac{R_{3,1} - R_{2,1}}{4^{2-1}-1} = R_{3,1} + \frac{R_{3,1} - R_{2,1}}{3} \end{aligned}$$

	$R_{k,1}$	$R_{k,2}$
$k = 1$	0.00000000	
$k = 2$	1.57079633	2.09439510
$k = 3$	1.89611890	2.00455975

Example:

$$I = \int_0^{\pi} sin(x)dx = 2.0$$

We start with $h_1 = (b-a) = \pi$

$$\begin{aligned} h_k &= (b-a)/2^{k-1} \Longrightarrow h_3 = \pi/4 \\ R_{k,j} &= R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1}-1} \qquad ; k \geq j > 1 \\ R_{3,3} &= R_{3,2} + \frac{R_{3,2} - R_{2,2}}{4^{3-1}-1} = R_{3,2} + \frac{R_{3,2} - R_{2,2}}{15} \end{aligned}$$

	$R_{k,1}$	$R_{k,2}$	$R_{k,3}$
$k=1$	0.00000000		
$k=2$	1.57079633	2.09439510	
$k=3$	1.89611890	2.00455975	1.99857073
	$\vdots$	$\vdots$	$\vdots$
	$O(h^2)$	$O(h^4)$	$O(h^6)$

Example:

$$I = \int_0^{\pi} sin(x)dx = 2.0$$

We start with $h_1 = \pi$

$$\begin{aligned} R_{1,1} &= \frac{h_1}{2}[f(a) + f(b)] \\ R_{k,1} &= \frac{1}{2}\left[R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a + (2i-1)h_k)\right] \qquad ; k > 1 \\ R_{k,j} &= R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1} \qquad ; k \geq j > 1 \\ h_k &= (b-a)/2^{k-1} \end{aligned}$$

	$R_{k,1}$	$R_{k,2}$	$R_{k,3}$	$R_{k,4}$	$R_{k,5}$
$k = 1$	0.00000000				
$k = 2$	1.57079633	2.09439510			
$k = 3$	1.89611890	2.00455975	1.99857073		
$k = 4$	1.97423160	2.00026917	1.99998313	2.00000555	
$k = 5$	1.99357034	2.00001659	1.99999975	2.00000002	1.99999999
$k = 6$	1.99839336	2.00000103	1.99999999	2.00000000	1.99999999

Outline

1. Introduction

2. Numerical Integration Methods
   - Interpolation Methods I: The Newton-Cotes integration rules
   - Extrapolation Methods: Richardson extrapolation and Romberg integration
   - Interpolation Methods II: Gaussian Quadrature
   - Monte-Carlo Method

Gaussian Quadrature

Newton-Cotes

- Abscissa points are equally spaced.
- The n abscissa points $(x_1, x_2, \cdots, x_n)$ are fixed in advance fot the $[a, b]$ interval.
- The integration rule is exact for polynomials of degree up to $n-1$.

$$\int_a^b f(x)dx \approx \int_a^b P_{\mathbf{n-1}} dx = \sum_{i=1}^{n} f(x_i) \cdot \omega_i \qquad i = 1, \cdots, n \text{ being equally spaced.}$$

- The maximum degree of accuracy for this case is *n-1*.

Gaussian Quadrature

- Seeks to obtain the best numerical integral by picking optimal abscissas x_i at which to evaluate the function $f(x)$. Unequally spaced points.
- The integration rule is exact for polynomials of degree up to $2n-1$.

$$\int_a^b f(x)dx \approx \int_a^b P_{\mathbf{2n-1}} dx = \sum_{i=1}^{n} f(x_i) \cdot \omega_i \qquad i = 1, \cdots, n \text{ are appropriately chosen}$$

- The maximum degree of accuracy is in this case *2n-1*.

Gaussian Quadrature

Fundamental theorem of Gaussian quadrature

The optimal abscissas of the n-point Gaussian quadrature formulas are precisely the roots of the orthogonal polynomial for the same interval and weighing function.

The solution relates to the orthogonal polynomials generated by the weight function W(x).

$$\int_a^b f(x)dx \approx \int_c^d W(t)P_{2n-1}(t)dt = \sum_{i=1}^{n} P(t_i)\cdot\omega_i \qquad \left\{ \begin{array}{l} P_{2n-1} \text{ is an orthogonal polynomial} \\ t_i \text{ are NOT equally spaced} \\ t_i \text{ are the zero's of } P_{2n-1} \end{array} \right.$$

$P_{2n-1}(t)$ optimized for a specic range (t∈[c,d]).
Abscissas, t_i, and weights, ω_i, must be determined.

The computation of Gaussian quadrature rules involves:

1. change of the variable if necessary.
2. the generation of the orthogonal polynomials
3. the computation of the zeros of the orthogonal polynomials
4. the computation of the associated weights

Gauss Quadrature methods

These are the *weight functions*, the *intervals*, and the *recurrence relations* for the most commonly used orthogonal polynomials:

- Gauss-Legendre

  $W(x)=1$

  $-1<x<1$

  $(j+1)P_{j+1}=(2j+1)xP_j-jP_{j-1}$

- Gauss-Chebyshev

  $W(x)=(1-x^2)^{-1/2}$

  $-1<x<1$

  $T_{j+1}=2xT_j-T_{j-1}$

- Gauss-Laguerre

  $W(x)=x^{\alpha}e^{-x}$

  $0<x<\infty$

  $(j+1)L^{\alpha}_{j+1}=(-x+2j+\alpha+1)L^{\alpha}_j-(j+\alpha)L^{\alpha}_{j-1}$

- Gauss-Jacobi

  $W(x)=(1-x)^{\alpha}(1+x)^{\beta}$

  $-1<x<1$

  $c_jP^{(\alpha,\beta)}_{j+1}=(d_j+e_jx)P^{(\alpha,\beta)}_j-f_jP^{(\alpha,\beta)}_{j-1}$

- Gauss-Hermite

  $W(x)=e^{-x^2}$

  $-\infty<x<\infty$

  $H_{j+1}=2xH_j-2jH_{j-1}$

Gauss Quadrature methods

These are the *weight functions*, the *intervals*, and the *recurrence relations* for the most commonly used orthogonal polynomials:

- Gauss-Legendre

  $W(x)=1$

  $\boxed{-1<x<1}$

  $(j+1)P_{j+1}=(2j+1)xP_j-jP_{j-1}$

- Gauss-Chebyshev

  $W(x)=(1-x^2)^{-1/2}$

  $-1<x<1$

  $T_{j+1}=2xT_j-T_{j-1}$

- Gauss-Laguerre

  $W(x)=x^{\alpha}e^{-x}$

  $0<x<\infty$

  $(j+1)L_{j+1}^{\alpha}=(-x+2j+\alpha+1)L_j^{\alpha}-(j+\alpha)L_{j-1}^{\alpha}$

- Gauss-Jacobi

  $W(x)=(1-x)^{\alpha}(1+x)^{\beta}$

  $-1<x<1$

  $c_jP_{j+1}^{(\alpha,\beta)}=(d_j+e_jx)P_j^{(\alpha,\beta)}-f_jP_{j-1}^{(\alpha,\beta)}$

- Gauss-Hermite

  $W(x)=e^{-x^2}$

  $-\infty<x<\infty$

  $H_{j+1}=2xH_j-2jH_{j-1}$

Gauss Quadrature methods

These are the *weight functions*, the *intervals*, and the *recurrence relations* for the most commonly used orthogonal polynomials:

- Gauss-Legendre

  $W(x) = 1$

  $\boxed{-1 < x < 1}$

  $(j+1)P_{j+1} = (2j+1)xP_j - jP_{j-1}$

- Gauss-Jacobi

  $W(x) = (1-x)^{\alpha}(1+x)^{\beta}$

  $-1 < x < 1$

  $c_j P_{j+1}^{(\alpha,\beta)} = (d_j + e_j x) P_j^{(\alpha,\beta)} - f_j P_{j-1}^{(\alpha,\beta)}$

- Gauss-Hermite

  $W(x) = e^{-x^2}$

  $-\infty < x < \infty$

  $H_{j+1} = 2xH_j - 2jH_{j-1}$

The computation of Gaussian quadrature rules involves:

1. change of the variable if necessary.
2. the generation of the orthogonal polynomials
3. the computation of the zeros of the orthogonal polynomials
4. the computation of the associated weights

Gauss Quadrature methods

Change of the variable: $\int_a^b f(x)dx \to \int_{-1}^{1} g(t)\,dt$

$$\mathbf{x = c + m\ t \longrightarrow dx = m\ dt}$$

$$\left.\begin{array}{l} x = a \longrightarrow t = -1 \Longrightarrow a = c - m \\ x = b \longrightarrow t = +1 \Longrightarrow b = c + m \end{array}\right\} c = \frac{a+b}{2} \quad \text{and} \quad m = \frac{b-a}{2}$$

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

$$I = m \int_{-1}^{1} g(t)\ dt = m \sum_{i=1}^{n} \omega_i\ g(t_i) \quad t_i \in [-1, 1]$$

Gaussian Quadrature

Gauss-Legendre zeros and weights for n points quadrature formula

n	t_i values	weights (ω_i)
1	0.0	2.0
2	$\pm\sqrt{\frac{1}{3}}$	1.0
3	0.0	0.88888889
	$\pm$ 0.77459667	0.55555555
4	$\pm$ 0.33998104	0.65214515
	$\pm$ 0.86113631	0.34785485
5	0.0	0.56888889
	$\pm$ 0.53846931	0.47862867
	$\pm$ 0.90617985	0.23692689
6	$\pm$ 0.23861918	0.46791393
	$\pm$ 0.66120939	0.36076157
	$\pm$ 0.93246951	0.17132449

n	t_i values	weights (ω_i)
7	0.0	0.41795918
	$\pm$ 0.40584515	0.38183005
	$\pm$ 0.74153119	0.27970539
	$\pm$ 0.94910791	0.12948497
8	$\pm$ 0.18343464	0.36268378
	$\pm$ 0.52553241	0.31370665
	$\pm$ 0.79666648	0.22238103
	$\pm$ 0.96028986	0.10122854
10	$\pm$ 0.14887434	0.29552422
	$\pm$ 0.43339539	0.26926672
	$\pm$ 0.67940957	0.21908636
	$\pm$ 0.86506337	0.14945135
	$\pm$ 0.97390653	0.06667134

Gauss Quadrature methods

Example: Calculate $I = \int_0^{\pi/2} \sin x \; dx = 1.0$ using a Gauss-Legendre quadrature

- Variable change: $c = \frac{a+b}{2} = \frac{\pi}{4}$ and $m = \frac{b-a}{2} = \frac{\pi}{4}$
- Computation of roots and weights for n=2 (check the table):

$$\omega_1 = \omega_2 = 1.0$$

$$t_1 = +\sqrt{\tfrac{1}{3}} \text{ and } t_2 = -\sqrt{\tfrac{1}{3}}.$$

$$I = m \sum_{i=1}^{2} \omega_i \; g(t_i) =$$

$$= m \left[\omega_1 \sin\left(\frac{a+b}{2} + \frac{b-a}{2} \; t_1 \right) + \omega_2 \; \sin\left(\frac{a+b}{2} + \frac{b-a}{2} \; t_2 \right) \right] =$$

$$= \frac{\pi}{4} \left[1.0 \; \sin\left(\frac{\pi}{4} \left(1 + \sqrt{\frac{1}{3}} \right) \right) + 1.0 \; \sin\left(\frac{\pi}{4} \left(1 - \sqrt{\frac{1}{3}} \right) \right) \right] = \underline{\mathbf{0.9984726}}$$

Gaussian Quadrature

Example $I = \int_0^{\pi/2} \sin x \, dx = 1.0$

Table: Comparison between a Gauss-Legendre and a Simpson result.

N	Gauss-Legendre	Simpson's rule
2	0.9984726135	1.0022798775
4	0.9999999770	1.0001345845
6	0.9999999904	1.0000263122
8	1.0000000001	1.0000082955
10	0.9999999902	1.0000033922

The Gauss-Legendre results are several orders of magnitude more accurate that those of the Simpson's rule for the same number of function evaluations.

Gaussian Quadrature

Now, we will cover:

- How roots (t_i) and weights (ω_i) shown in previous Table are determined.
- This part is not needed for writing the Fortran program. I will provide you a subroutine to calculate them.
- But good to know the origin of these values.

GQ: n=1. Determination of nodes and weights

How can we determine the nodes (t_i) and the weights (ω_i)?

For $n = 1$ point, we have $2n$ unknowns values: t_1 and ω_1.
We want to obtain these unknown values so that the quadrature rule is exact for all polynomials of degree up to $2n - 1 = 1$.
It should be exact for polynomials of degree 0 and 1.

$$\mathbf{I} = \boldsymbol{\omega_1}\ \mathbf{g(t_1)}$$

$$\left.\begin{array}{ll} g(t) = 1; & \int_{-1}^{+1} 1dt \rightarrow [t]_{-1}^{1} = 2 \\ g(t) = t; & \int_{-1}^{1} tdx \rightarrow \left[\frac{t^2}{2}\right]_{-1}^{1} = 0 \end{array}\right\} \begin{array}{l} \Longrightarrow \mathbf{2} = \boldsymbol{\omega_1} \\ \Longrightarrow 0 = t_1\ \omega_1 \Longrightarrow \mathbf{0} = \mathbf{t_1} \end{array}$$

$$\boxed{I = \int_{-1}^{1} g(t)\ dt = \sum_{i=1}^{1} \omega_1 g(t_1) = 2\ g(0)}$$

GQ: n=2. Determination of nodes and weights

For $n = 2$ points, we have $2n$ unknowns: t_1, ω_1, t_2 and ω_2.
We want to find the values of these unknowns so that the quadrature rule is exact for all polynomials of degree up to $2n - 1 = 3$.

It should be exact for polynomials of degree 0, 1, 2 and 3.

$$\mathbf{I = \omega_1\ g(t_1) + \omega_2\ g(t_2)}$$

$$\left.\begin{array}{lll}
g(t) = 1; & \int_{-1}^{+1} 1dt & \Longrightarrow 2 = \omega_1 + \omega_2 \\
g(t) = t; & \int_{-1}^{+1} tdt & \Longrightarrow \left[\frac{t^2}{2}\right]_{-1}^{+1} = 0 = t_1\ \omega_1 + t_2\ \omega_2 \\
g(t) = t^2; & \int_{-1}^{+1} t^2 dt & \Longrightarrow \left[\frac{t^3}{3}\right]_{-1}^{+1} = \frac{2}{3} = t_1^2\ \omega_1 + t_2^2\ \omega_2 \\
g(t) = t^3; & \int_{-1}^{+1} t^3 dt & \Longrightarrow \left[\frac{t^4}{4}\right]_{-1}^{+1} = 0 = t_1^3\ \omega_1 + t_2^3\ \omega_2
\end{array}\right\}
\begin{array}{l}
\Longrightarrow \mathbf{1 = \omega_1 = \omega_2} \\
\Longrightarrow \mathbf{t_1 = -\frac{1}{\sqrt{3}}} \\
\Longrightarrow \mathbf{t_2 = +\frac{1}{\sqrt{3}}}
\end{array}$$

$$\boxed{I = \sum_{i=1}^{2} \omega(t_i) g(t_i) = g\left(-\frac{1}{\sqrt{3}}\right) + g\left(+\frac{1}{\sqrt{3}}\right)}$$

GQ: n > 2. Determination of nodes and weights

How can we determine the nodes (t_i) and the weights (ω_i)?

For n ($n > 2$) points, we have $2n$ unknowns: t_1, ω_1, t_2, ω_2, $\cdots$, t_n, ω_n.
We want to find the values of these unknowns so that the quadrature rule is exact for all polynomial of degree up to $2n - 1$.

It should be exact for polynomials of degree 0 to $2n - 1$.

$$\mathbf{I = \sum_{i=1}^{n} \omega_i \, g(t_i)}$$

$$\left.\begin{aligned}
&g(t) = 1; \quad \int_{-1}^{+1} 1dt \quad \Longrightarrow 2 = \omega_1 + \omega_2 + \cdots + \omega_n \\
&g(t) = t; \quad \int_{-1}^{+1} tdt \quad \Longrightarrow 0 = t_1\ \omega_1 + t_2\ \omega_2 + \cdots + t_n\ \omega_n \\
&g(t) = t^2; \quad \int_{-1}^{+1} t^2 dt \quad \Longrightarrow \tfrac{2}{3} = t_1^2\ \omega_1 + t_2^2\ \omega_2 + \cdots + t_n^2\ \omega_n \\
&\vdots \\
&g(t) = t^{2n-1}; \quad \int_{-1}^{+1} t^{2n-1} dt \quad \Longrightarrow 0 = t_1^3\ \omega_1 + t_2^3\ \omega_2 + \cdots + t_n^3\ \omega_n
\end{aligned}\right\}$$

One should solve this set of equations.

Outline

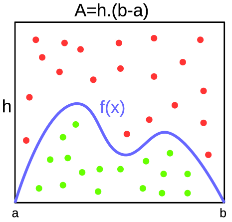
Monte-Carlo method: 1-dimension

Monte-Carlo method is based on random selection of points.

"Hit and miss method"

- $Area = h \cdot (b - a)$ such that $f(x)$ is within the rectangle.
- Generate N pairs of points (x_i, y_i)
- Check if $y_i \leq f(x_i)$
- Estimate the area:

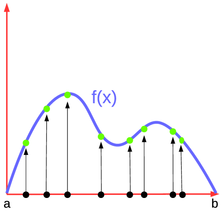
$$\mathbf{I_n = A \cdot \frac{N_{inside}}{N}}$$



"Sample mean method"

- Generate N x_i points randomly
- Obtain $f(x_i)$
- Estimate the area:

$$I_n = (b-a) < f >= (b-a) \cdot \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$



Monte-Carlo: Multi-dimensional Integration

Higher-dimensional Integrals

Many problems in physics involve averaging over many variables. Monte Carlo methods become specially attractive when dealing with integration in higher dimensions. For example,

$$I = \int_{a_1}^{a_2}\int_{b_1}^{b_2}\int_{c_1}^{c_2} f(x,y,z)dx\ dy\ dz \approx (a_2-a_1)(b_2-b_1)(c_2-c_1)\frac{1}{N}\sum_{i=1}^{N} f(x_i,y_i,z_i)$$

where (x_i, y_i, z_i) is a random sequence of points. One needs $3N$ random numbers to generate N random points.

In MC methods:

- Error <u>always</u> decreases as $1/\sqrt{N}$ independently of the number of dimensions.

In standard numerical integration methods:

- Error decreases as $n^{-a/d}$. $d =$(number of dimensions)

Therefore MC is more efficient for higher dimensions.

References

- J. D. Hoffman
  Numerical Methods for Engineers and Scientist (McGraw-Hill, New York, 1992)
- A. R. Krommer and C. W. Ueberhuber
  Numerical Integration on Advanced Computer Systems (Springer-Verlag Berlin, Heidelberg, 1994)
- P. J. Davis and P. Rabinowitz
  Methods of Numerical Integration (second edition, Academic Press, Inc., London, 1984)
- S. C. Chapra and R. C. Canale
  Numerical Methods for engineers (fifth edition, The McGraw-Hill Companies, Inc., México, 2006)
- R. M. Corless and N. Fillion
  Graduate Introduction to Numerical Methods (1st edition, Springer, New York, 2013)
- S. A. Teukolsky, W. T. Vetterling and B. P. Flannery
  Numerical Recipes in Fortran 77 (second edition, Univ. Press, Cambridge, 2003)

How to write the programs

A good code:

- It must provide the right answer ... and more:
  - It must be compiled without errors.
  - It must be well-structured and properly explained with comments. Add comments!
  - It produces a concise but understandable output.
  - It contains functions and subroutines:
    - Use functions to calculate the values of a f(x) function.

All these factors will be evaluated.