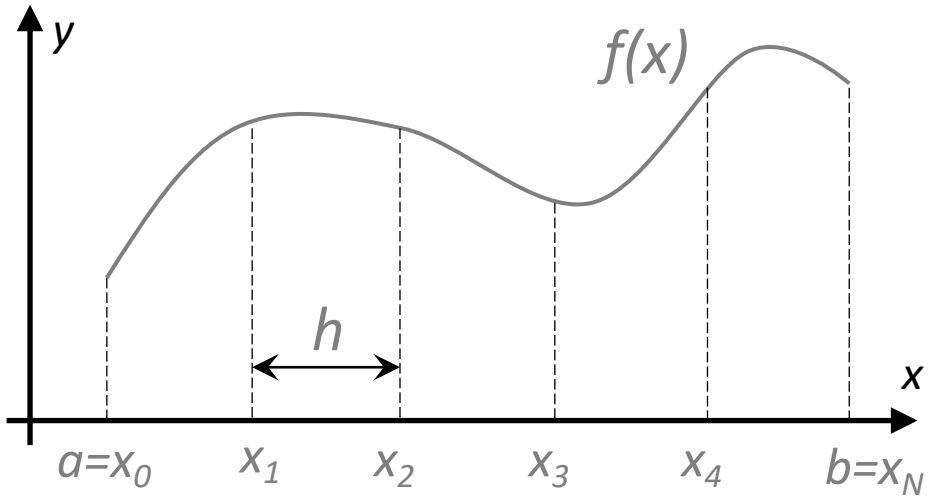
Ch. 03
Numerical Quadrature

Andrea Mignone
Physics Department, University of Torino
AA 2023-2024

Numerical Quadrature

- In numerical analysis "quadrature" refers to the computation of definite integrals.



- A traditional way to perform numerical integration is to take a piece of graph paper and count the number of boxes or *quadrilaterals* lying below a curve of the integrand. For this reason numerical integration is also called *numerical quadrature*

Numerical Quadrature

- The Riemann definitions of an integral is the limit of the sum over boxes as the width h of the box approaches zero:

$$\int_a^b f(x)\,dx = \lim_{h\to 0}\left[h\sum_{i=1}^{(b-a)/h} f(x_i)\right]$$

- The numerical integral of a function `f(x)` is approximated as the equivalent of a finite sum over boxes of height `f(x)` and width w_i:

$$\int_a^b f(x)\,dx \simeq \sum_{i=1}^{N} f(x_i)w_i$$

(`N = (b-a)/h`).

- This is similar to the Riemann definition except that there is no limit to an infinitesimal box size.

- The previous equation is the standard form for all integration algorithms. The function `f(x)` is evaluated at N points in the interval `[a,b]`, and the function values $f_i \equiv f(x_i)$ are summed with each term in the sum weighted by w_i.

Numerical Quadrature

$$\int_a^b f(x)\,dx \simeq \sum_{i=1}^{N} f(x_i) w_i$$

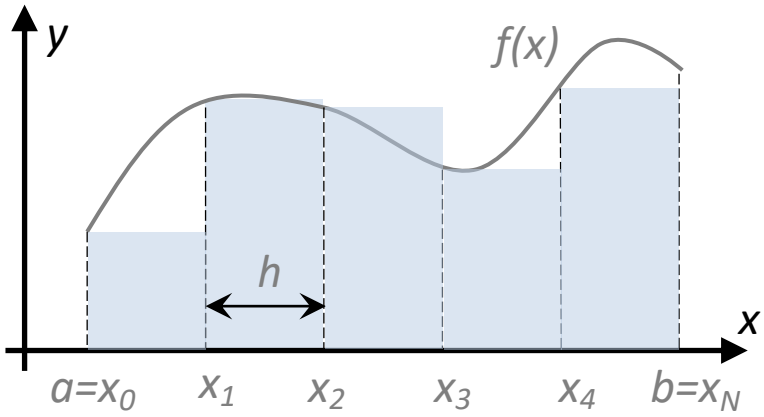
- While this sum gives the exact integral only when `N` → ∞, it may be exact for finite `N` if the integrand is a polynomial.
- Different integration algorithms amount to different ways of choosing the points and weights. Generally, the precision increases as `N` gets larger, with round-off error eventually limiting the increase.
- There's no universal "best" approximation: the computations depends on the specific behavior of `f(x)`.
- Singularities should be removed by hand before performing the actual computation.
- For integrands with slow (fast) variations in some regions, a change of variable that places less (more) points there is advisable.

Rectangular Rule

- Let's divide the integration region [a,b] into N equally spaced intervals of length h:

$$h = \frac{b-a}{N},$$

$$x_i = a + ih \qquad (i = 0, ..., N-1)$$



- A simple way to approximate the integral within a single interval [x_i, x_i+h], is to assume to be piecewise constant, so that we have ***the rectangular rule:***

$$\int_{x_i}^{x_i+h} f(x)dx \approx f(x_i)h$$

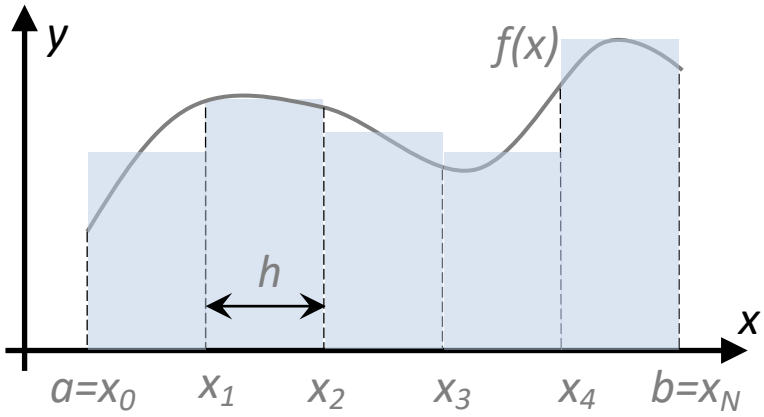
- The error can be found using Taylor expansion:

$$\int_{x_i}^{x_i+h} f(x)dx = \int_{x_i}^{x_i+h} \left[f(x_i) + (x-x_i)f'(x_i) + \frac{(x-x_i)^2}{2!}f''(x_i) + ... \right] dx$$

$$\approx f(x_i)h + \frac{h^2}{2}f'(x_i)$$

Midpoint Rule

- By looking at the error in the previous expressions, it is straightforward to realize that the linear term in the error cancels if we choose the interval midpoint (rather than the leftmost point):



$$\int_{x_i}^{x_i+h} f(x)dx \approx f\left(x_i + \frac{h}{2}\right) h$$

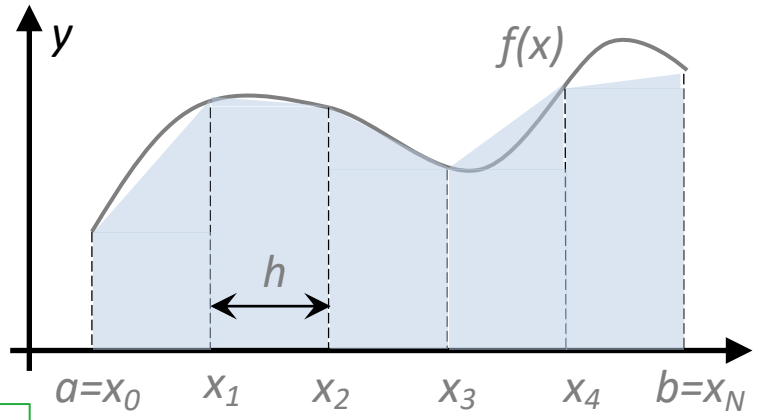
- The error is now $O(h^3)$:

$$\int_{x_i}^{x_i+h} f(x)dx = \int_{x_i}^{x_i+h} \left[f(x_{i+\frac{1}{2}}) + (x - x_{i+\frac{1}{2}})f'(x_{i+\frac{1}{2}}) + \frac{(x - x_{i+\frac{1}{2}})^2}{2!} f''(x_{i+\frac{1}{2}}) + \ldots\right] dx$$

$$\approx f(x_{i+\frac{1}{2}})h + \frac{h^3}{24} f''(x_{i+\frac{1}{2}})$$

Trapezoidal Rule

- The trapezoidal rule takes each integration interval `i` and constructs a trapezoid of width `h`.
- This approximates `f(x)` by a straight line in each interval `i` and uses the average height `(f_i+f_{i+1})/2`.
- The area of such trapezoid is



$$\int_{x_i}^{x_i+h} f(x)\,dx \simeq \frac{h(f_i+f_{i+1})}{2} = \frac{1}{2}hf_i + \frac{1}{2}hf_{i+1}$$

- The error can be shown to be $\epsilon = -\frac{h^3}{12}f''(\xi)\,, \qquad \xi \in [x_i, x_i+h]$

- Applying the trapezoidal rule to the entire region [a,b] we add contributions from each interval:

$$\int_a^b f(x)dx \approx \frac{h}{2}(f_0+f_1) + \frac{h}{2}(f_1+f_2) + \ldots = \frac{h}{2}f_0 + hf_1 + hf_2 + \ldots + \frac{h}{2}f_N$$

- In terms of our standard integration formula, the weights are $w_i = \left\{\frac{h}{2}, h, \ldots, h, \frac{h}{2}\right\}$

Simpson Rule

- If we approximate the function with a parabola we obtain a better approximation:

$$\int_{x_i}^{x_i+h} f(x)dx \approx \int_{x_i}^{x_i+h} (\alpha x^2+\beta x+\gamma)dx = \frac{\alpha x^3}{3} + \frac{\beta x^2}{2} + \gamma x \Big|_{x_i}^{x_i+h}$$

- In order to relate α, β and γ to the function, we consider an interval `[-1,1]` so that

$$f(-1) = \alpha - \beta + \gamma, \qquad f(0) = \gamma, \qquad f(1) = \alpha + \beta + \gamma,$$

$$\Rightarrow \quad \alpha = \frac{f(1)+f(-1)}{2} - f(0), \quad \beta = \frac{f(1)-f(-1)}{2}, \quad \gamma = f(0).$$

- In this way we can express the integral as the weighted sum over the values of the function at three points:

$$\int_{-1}^{1} (\alpha x^2 + \beta x + \gamma)\, dx = \frac{f(-1)}{3} + \frac{4f(0)}{3} + \frac{f(1)}{3}.$$

- The formula is actually correct for polynomials up to order 3.

Extended Simpson rule

- Because three values of the function are needed, the integral should be evaluated over two adjacent intervals, (function eval. at the two endpoints and in the middle):

$$\int_{x_i-h}^{x_i+h} f(x)dx \approx \frac{h}{3}f_{i-1} + \frac{4h}{3}f_i + \frac{h}{3}f_{i+1}$$

- The error can be shown to be $\epsilon = -\frac{1}{90}\left(\frac{h}{2}\right)^5 f^4(\xi)\,, \quad \xi \in [x_i, x_i + h]$

- Simpson's rule must thus be carried out over pairs of intervals, which in turn requires that *the total number of intervals N be even* or that *the number of points (N+1) be odd*.
- If we apply the previous results to successive, non-overlapping *pairs* of intervals, we obtain (*extended Simpson rule*)

$$\int_a^b f(x)dx = h\left[\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \frac{4}{3}f_3 + \ldots + \frac{2}{3}f_{N-2} + \frac{4}{3}f_{N-1} + \frac{1}{3}f_N\right]$$

- According to our formulation, the weights[1] are: $w_i = \left\{\frac{h}{3}, \frac{4h}{3}, \frac{2h}{3}, \frac{4h}{3}, \ldots, \frac{4h}{3}, \frac{h}{3}\right\}$

[1] The 2/3, 4/3 alternation continues throughout the interior of the evaluation. Many people believe that the wobbling alternation somehow contains deep information about the integral of their function that is not apparent to mortal eyes. In fact, the alternation is an artifact of using the building block.

Practice Session #1

- `quadrature1.cpp`: given an interval `[a,b]` and a function `f(x)`, write a program to compute

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

- Divide the interval `[a,b]` into `N` equally spaced sub-interval separated by `N+1` points $\{x_0, x_1, x_2, \ldots x_N\}$. In each sub-interval apply the *rectangular*, *trapezoidal* and *Simpson* rules.

- Consider `f(x) = exp(-x)` and use `[a, b] = [0,1]`.

- Implement each quadrature rule using the same template, e.g.,

```
double QuadratureRule (double (*F)(double), double a, double b, int N)
{
  ...
  sum += ...   // do the sum
  return sum;
}
```

Practice Session #1

- The 1st function argument is a pointer to a function "*F". This argument should be passed from the caller using a valid function name:

```
sum = QuadratureRule (func, a, b, N); // This is how you call a quadrature rule
                                      // from your main().

...

double func(double x)      // This is the function you want to integrate
{
  return exp(-x);
}
```

- With 4 sub-intervals (N=4) your program output should give the following values:

```
Exact:       6.321205588286e-01

Rectangular: 7.144244988813e-01; (N = 4)

Trapezoidal: 6.354094290277e-01; (N = 4)

Simpson:     6.321341753205e-01; (N = 4)
```

Practice Session #1

- Next, iterate by doubling the value of intervals N (= 4, 8, 16, …) until convergence is achieved:

$$|I_N - I_{N/2}| < tol \qquad \text{where} \quad I_N = \sum_i^N w_i f(x_i)$$

where *tol* is a prescribed tolerance (e.g. 10^{-5}).

- The expected result is

```
Rectangular: 6.321302042719e-01; N = 32768

Trapezoidal: 6.321237739567e-01; N = 128

Simpson:     6.321206123892e-01; N = 16
```

Introducing Gaussian Quadrature

- Higher precision may be achieved if we relax the assumption of equally-spaced quadrature points.

- By choosing the x_n in some optimal sense we then have 2N parameters at our disposal in constructing the quadrature formula (2N = N abscissae + N weights).

- These can be chosen to satisfy

$$\int_{-1}^{1} x^p dx = \sum_{i=1}^{N} w_i x_i^p , \qquad \text{for} \quad p = 0, \ldots, 2N-1$$

- In other words, the quadrature formula using only N carefully chosen points can be made exact for polynomials up to degree 2N-1 or less.

- This is obviously more efficient than using equally-spaced abscissae.

Unevenly spaced abscissae: An example

- As an example, let's look for a quadrature rule in the form

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

- We wish to make it exact for polynomials of degree ≤ 3.
- Because of the linearity of the quadrature, it suffices to make the rule exact for f(x) = 1, x, x^2 and x^3. Hence we obtain the following system of 4 equations

$$\int_{-1}^{1} x^p dx = w_0 x_0^p + w_1 x_1^p \qquad \Longrightarrow \qquad \begin{cases} w_0 + w_1 & = & 2 \\ w_0 x_0 + w_1 x_1 & = & 0 \\ w_0 x_0^2 + w_1 x_1^2 & = & 2/3 \\ w_0 x_0^3 + w_1 x_1^3 & = & 0 \end{cases}$$

- Solving for the weights and abscissae: $x_0 = -x_1 = \dfrac{1}{\sqrt{3}}, \quad w_0 = w_1 = 1$
- Our quadrature rule becomes $\displaystyle\int_{-1}^{1} f(x)dx \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$

The general case

- In the general case we have the system of 2N equations for p = 0, …, 2N-1

$$w_1 x_1^p + \ldots + w_N x_N^p = \int_{-1}^{1} x^p dx = \begin{cases} \dfrac{2}{p+1} & \text{for} \quad p = 0, 2, \ldots, 2N-2 \\ 0 & \text{for} \quad p = 1, 3, \ldots, 2N-1 \end{cases}$$

A1

- The system is nonlinear but it admits a solution and the resulting numerical integration rule is called Gaussian quadrature. It is not convenient, however, to solve the system in this way.
- ***A solution to the previous system*** can be expressed in terms of Legendre polynomials. In particular, any polynomial of degree 2N-1 (or less) can be written in the form

$$f(x) = Q(x)P_N(x) + R(x)$$

with Q and R polynomial of degree N-1 or less. The integral becomes:

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1} \left[Q(x)P_N(x) + R(x)\right]dx = \int_{-1}^{1} R(x)dx$$

where the second equality is a consequence of the orthogonality of P_N to all polynomial of degree N-1 or less.

Gauss-Legendre Quadrature

- Using our summation rule:

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{N} w_i f(x_i) = \sum_{i=1}^{N} w_i \Big[Q(x_i) P_N(x_i) + R(x_i) \Big]$$

- If we now take $\{x_i\}$ to be *the roots* of P_N, the we obtain *exactly*:

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{N} w_i f(x_i) = \sum_{i=1}^{N} w_i R(x_i)$$

- Thus the w_i satisfy the linear system A1 when the abscissae are the zeros of Legendre polynomials.

- It can be shown that the weights are related to the derivative of P_N at the corresponding zero:

$$w_i = \frac{2}{(1 - x_i^2)[P'_N(x_i)]^2}$$

Gauss-Legendre Quadrature: a note

- As a general rule, Gaussian quadrature is the method of choice when the integrand is known analytically, smooth or it can be made smooth enough by extracting from it a function that is the weight for a standard set of orthogonal polynomials.

- One has to evaluate weights and abscissae.

- If the integrand varies rapidly, we can repeat the basic Gaussian quadrature formula by applying it over several sub-intervals in the range of integration.

- Finally, if the integrand can be evaluated only at equally-spaced abscissae (for example when it is generated by integrating a differential equation), then Simpson (or higher) formula should be used.

Other Quadrature Rules

- Other type of orthogonal polynomials provide useful quadrature formulas when the integrand has a particular form.
- For instance, the Laguerre polynomials (which are orthogonal on the interval [0,∞] with weight function e^{-x}) lead to the Gauss-Laguerre quadrature formula:

$$\int_0^\infty e^{-x} f(x) dx \approx \sum_i w_i f(x_i)$$

- Here $\{x_i\}$ are the roots of the Laguerre polynomial of order N+1 and $\{w_i\}$ are related to the values at these points (see Abramowitz & Stegun, "Handbook of Mathematical Functions", section 25.4.29 and after).

- Likewise the Hermite polynomials provide Gauss-Hermite quadrature formulas for integrals in the form

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x) dx$$

- Other types of integral requires special care in choosing the polynomial. A list of them can be found in sect. 25.4 of Abramowitz & Stegun.

Change of Interval

- The previous considerations may be extended to any interval $[x_0, x_1]$ using a simple linear change of variable.

- In particular, with $x = \frac{x_1 - x_0}{2} t + \frac{x_0 + x_1}{2}$ we end up with

$$\int_{x_0}^{x_1} f(x)dx = \frac{x_1 - x_0}{2} \int_{-1}^{1} f\left(\frac{x_1 - x_0}{2} t + \frac{x_0 + x_1}{2}\right) dt$$

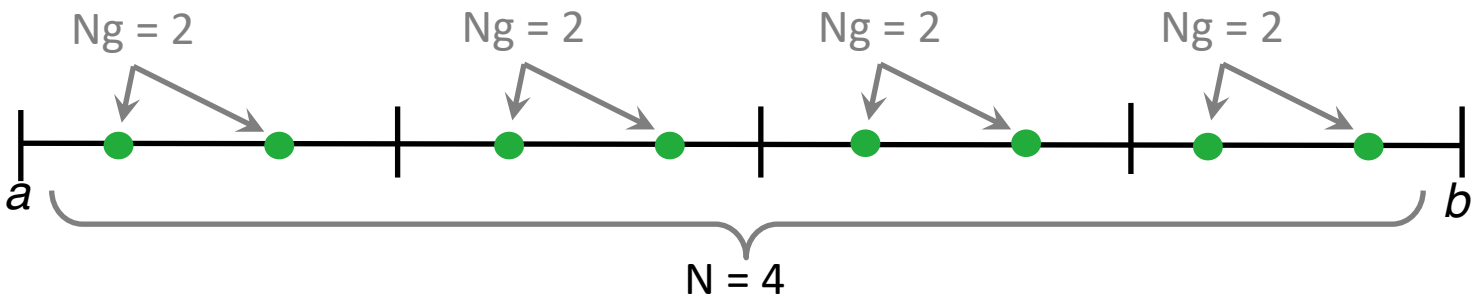
- Applying Gaussian quadrature,

$$\int_{x_0}^{x_1} f(x)dx = \frac{x_1 - x_0}{2} \sum_{i=1}^{N} w_i f\left(\frac{x_1 - x_0}{2} t_i + \frac{x_0 + x_1}{2}\right)$$

- Other change of variable that makes the integrand smoother may provide better accuracy.

Applying Gauss Quadrature over Sub-Intervals

- In general, we can still divide our integration domain [a,b] and into N equally spaced intervals and apply Gaussian quadrature to each interval separately; e.g.



- For this reason, it may be convenient to cast our Gaussian quadrature function in a slightly (more general way), by providing as input arguments both the *number of intervals N* and the *number of Gaussian points Ng*:

```
double GaussQuadratureRule (double (*F)(double), double a, double b, int N, int Ng)
{
  ...
  if      (Ng == 2) {w[] = ..., x[] = ...}
  else if (Ng == 3) {w[] = ..., x[] = ...}
  ...
  sum  = 0.0;
  for (i = 0; i < N; i++) { // Loop over intervals
    x0 = ...; x1 = ...      // Define left and right interval endpoints x0 & x1
    sumk = 0.0;             // Initialize sum for this interval
    for (k = 0; k < Ng; k++) sumk += ...  // Apply Gaussian rule to sub-interval
    sum  += sumk;       // Add partial sum to total integral
 }
  return sum;
}
```

Gauss-Legendre Quadrature

Number of points, n	Points, x_i	Weights, w_i
1	0	2
2	$\pm\sqrt{\frac{1}{3}}$	1
3	0	$\frac{8}{9}$
	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$
5	0	$\frac{128}{225}$
	$\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$	$\frac{322+13\sqrt{70}}{900}$
	$\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	$\frac{322-13\sqrt{70}}{900}$

Gauss-Legendre Quadrature

Table 25.4 ABSCISSAS AND WEIGHT FACTORS FOR GAUSSIAN INTEGRATION

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

Abscissas $= \pm x_i$ (Zeros of Legendre Polynomials)		Weight Factors $= w_i$	
$\pm x_i$	w_i	$\pm x_i$	w_i
$n = 2$		$n = 8$	
0.57735 02691 89626	1.00000 00000 00000	0.18343 46424 95650	0.36268 37833 78362
$n = 3$		0.52553 24099 16329	0.31370 66458 77887
0.00000 00000 00000	0.88888 88888 88889	0.79666 64774 13627	0.22238 10344 53374
0.77459 66692 41483	0.55555 55555 55556	0.96028 98564 97536	0.10122 85362 90376
$n = 4$		$n = 9$	
0.33998 10435 84856	0.65214 51548 62546	0.00000 00000 00000	0.33023 93550 01260
0.86113 63115 94053	0.34785 48451 37454	0.32425 34234 03809	0.31234 70770 40003
$n = 5$		0.61337 14327 00590	0.26061 06964 02935
0.00000 00000 00000	0.56888 88888 88889	0.83603 11073 26636	0.18064 81606 94857
0.53846 93101 05683	0.47862 86704 99366	0.96816 02395 07626	0.08127 43883 61574
0.90617 98459 38664	0.23692 68850 56189	$n = 10$	
$n = 6$		0.14887 43389 81631	0.29552 42247 14753
0.23861 91860 83197	0.46791 39345 72691	0.43339 53941 29247	0.26926 67193 09996
0.66120 93864 66265	0.36076 15730 48139	0.67940 95682 99024	0.21908 63625 15982
0.93246 95142 03152	0.17132 44923 79170	0.86506 33666 88985	0.14945 13491 50581
$n = 7$		0.97390 65285 17172	0.06667 13443 08688
0.00000 00000 00000	0.41795 91836 73469	$n = 12$	
0.40584 51513 77397	0.38183 00505 05119	0.12523 34085 11469	0.24914 70458 13403
0.74153 11855 99394	0.27970 53914 89277	0.36783 14989 98180	0.23349 25365 38355
0.94910 79123 42759	0.12948 49661 68870	0.58731 79542 86617	0.20316 74267 23066
		0.76990 26741 94305	0.16007 83285 43346
		0.90411 72563 70475	0.10693 93259 95318
		0.98156 06342 46719	0.04717 53363 86512

Interval	$\omega(x)$	Orthogonal polynomials	A & S	For more information, see ...
[−1, 1]	1	Legendre polynomials	25.4.29	See Gauss–Legendre quadrature above
(−1, 1)	$(1-x)^\alpha(1+x)^\beta, \quad \alpha,\beta > -1$	Jacobi polynomials	25.4.33 ($\beta = 0$)	Gauss–Jacobi quadrature
(−1, 1)	$\frac{1}{\sqrt{1-x^2}}$	Chebyshev polynomials (first kind)	25.4.38	Chebyshev–Gauss quadrature
[−1, 1]	$\sqrt{1-x^2}$	Chebyshev polynomials (second kind)	25.4.40	Chebyshev–Gauss quadrature
[0, ∞)	e^{-x}	Laguerre polynomials	25.4.45	Gauss–Laguerre quadrature
[0, ∞)	$x^\alpha e^{-x}, \quad \alpha > -1$	Generalized Laguerre polynomials		Gauss–Laguerre quadrature
(−∞, ∞)	e^{-x^2}	Hermite polynomials	25.4.46	Gauss–Hermite quadrature

Practice Session #2

- Using Simpson rule with 2 intervals (3 points in total) and Gauss-Legendre (1 interval, 3 Gaussian points), compute the integral

$$\int_0^3 \sqrt{1+t}\, dt = 4.66666667$$

- The result you expect is

```
Simpson = 4.662277660168e+00
Gauss   = 4.666829051581e+00
```

- As you can see, both methods use the same number of function evaluations but Gauss is more accurate by one order of magnitude.

- Now, using the same code, test the two algorithms in computing the integral:

$$\int_{-1}^{5} f(x)\, dx \qquad \text{where} \quad f(x) = 1 - x + 2x^2 + \frac{x^3}{2} + \frac{x^4}{4} - \frac{x^5}{8}$$

again using 2 intervals for Simpson's rule and 1 interval for Gauss-Legendre with 3 Gaussian points. The exact result is –66/5. Which of the two is better ?

Indefinite Integrals

- In general, the indefinite integral of a function is $\int_a^x f(t)dt = F(x) - F(a)$ where F(x) is the *antiderivative*.

- However, for many functions, the antiderivative cannot be determined using elementary methods of calculus; e.g.

$$\int e^{-u^2}du, \qquad \int \frac{\sin x}{x}dx, \qquad \int \sqrt{1+x^4}dx, \qquad \ldots$$

Indefinite Integrals

- `integral_sin.cpp`: write a program to compute

$$\mathrm{Si}(x) = \int_0^x \frac{\sin t}{t} dt$$

at x = 0.8 using intervals h = 0.8, 0.4, 0.2, 0.1.

The correct value, to ten decimals, is

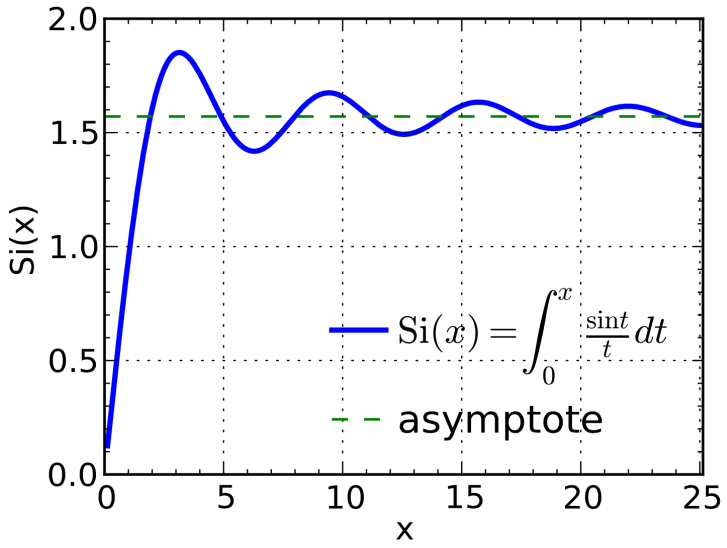
Si(0.8) = 0.77209 57855.

Evaluation of Si(0.8) using different methods:

```
[Gaussian n=1, ng=3] = 7.720958e-01
[Trapezoidal, n = 1] = 7.586780e-01
[Trapezoidal, n = 2] = 7.687574e-01
[Trapezoidal, n = 4] = 7.712622e-01
[Trapezoidal, n = 8] = 7.718874e-01
[Simpson, n = 2]     = 7.721171e-01
[Simpson, n = 4]     = 7.720971e-01
[Simpson, n = 8]     = 7.720959e-01
```

Next, using gnuplot, produce a plot like the one in the figure for 0 < x < 25.

Try to not sacrifice accuracy as x increases !



Multidimensional Integrals

- The quadrature rules discussed so far can be extended to compute integral in multiple spatial dimensions.

- However, integrals of functions of several variables, over regions with dimension greater than one, are *not easy*.

- The approach is to express the multiple integral as repeated one-dimensional integrals by applying to Fubini's theorem.

- This approach requires the function evaluations to grow exponentially as the number of dimensions increases. Three methods are known to overcome this so-called *curse of dimensionality*.

- There are two reasons for this:
  - the number of function evaluations to sample an N-dimensional space increases as the Nth power of the number needed to do a one-dimensional integral: if you need 30 function evaluations in 1D, then you will likely need $\approx 3 \times 10^4$ evaluations in 3D.
  - the region of integration in N-dim is defined by an N – 1 dimensional boundary which may be complicated: it need not be convex or simply connected, for example.

Multidimensional Quadrature: Rules

- Generally speaking, if the boundary is complicated but the integrand is *not* strongly peaked in very small regions, and relatively low accuracy is tolerable, then *Monte Carlo integration* is the best approach.

- If the boundary is simple and the function is very smooth, breaking up the problem into repeated *one-dimensional integrals* or *multidimensional Gaussian quadratures*, will be effective and relatively fast. If you need high accuracy, these approaches are in any case the *only* ones available to you (*Monte Carlo methods* are asymptotically slow to converge).

- For low accuracy, use repeated *one-dimensional integration* or *multidimensional Gaussian quadratures* when the integrand is slowly varying and smooth in the region of integration, *Monte Carlo* when the integrand is oscillatory or discontinuous, but not strongly peaked in small regions.

- If the integrand *is* strongly peaked in small regions, and you know where those regions are, break the integral up into several regions so that the integrand is smooth in each, and do each separately.

Reduction to 1D integrals

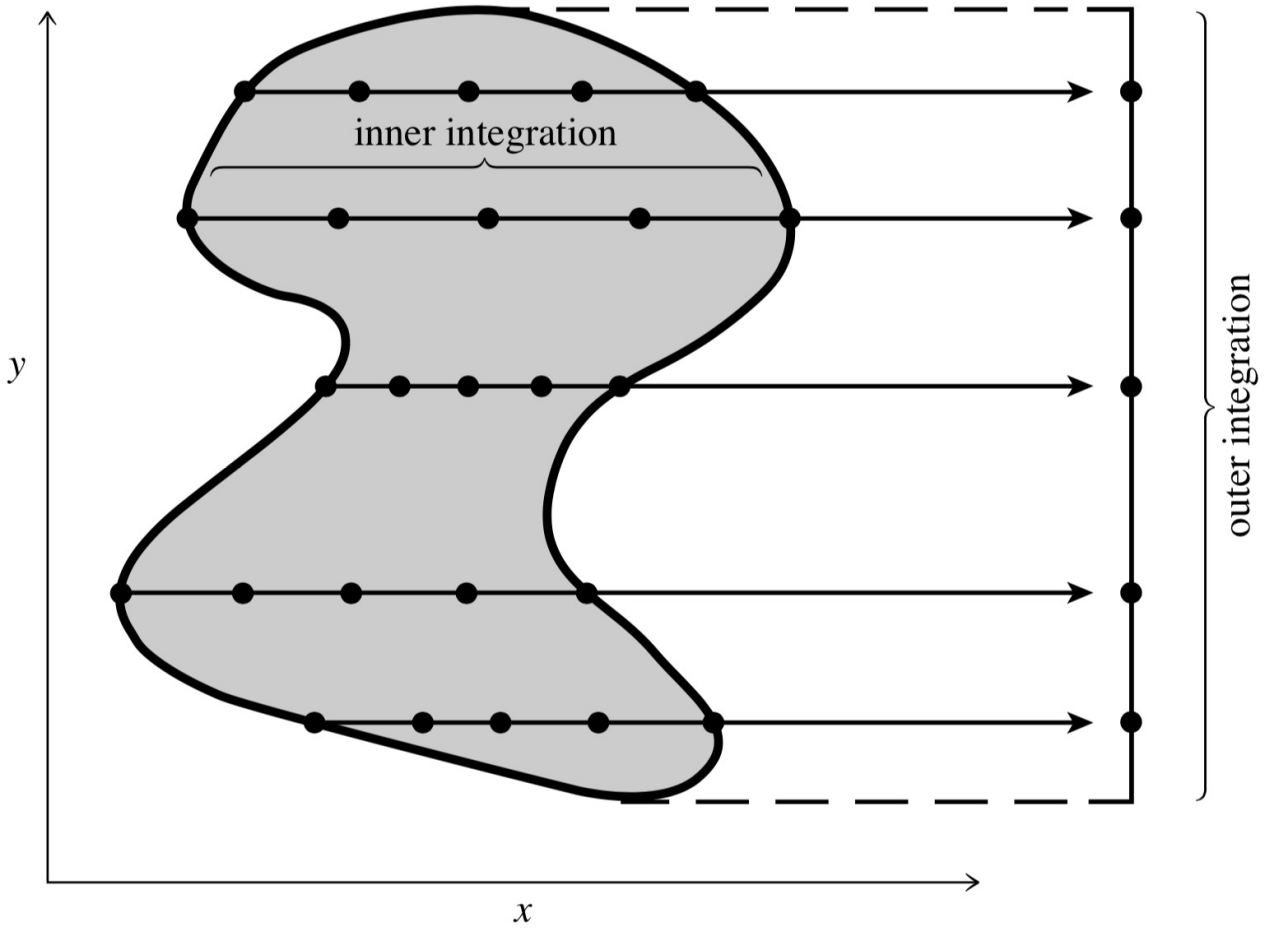


Figure 4.6.1. Function evaluations for a two-dimensional integral over an irregular region, shown schematically. The outer integration routine, in y, requests values of the inner, x, integral at locations along the y axis of its own choosing. The inner integration routine then evaluates the function at x locations suitable to *it*. This is more accurate in general than, e.g., evaluating the function on a Cartesian mesh of points.

Practice Session #3

- `multid_quadrature.cpp`: use Gaussian integration to compute the integral of a 2D function on the square `[-1,1]`^{2}. An integral in the 2D plane may be written as:

$$\int\int f(x,y)dxdy = \int dy \left(\int f(x,y)dx \right) = \int dyG(y)$$

- This form allows you to re-use your previously 1D quadrature rules:

$$\int G(y)dy = \sum_j w_j G(y_j) = \sum_j w_j \left(\sum_i w_i f(x_i, y_j) \right)$$

- This suggests to choose between:
  1. A genuine multiD integrator (suggested)
  2. Re-use our 1D integrators and define y_j externally using, e.g., a global variable approach. In this approach we will need:

     - A function `Func2D = f(x,y):` the actual 2D function;
     - A Function `FFunc1D = f(x,y_j):` used to perform 1D integration in x;
     - A Function `GGunc1D = G(y) = \int f(x,y) dx:` used to perform 1D integration in y;

Practice Session #3

- Here's a pseudo-code for a 2D integrator:

```
double Gaussian2DQuad (double (*Func)(double, double),
                       double  xa, double xb,
                       double ya, double yb,
                       int n, int ngauss)
{
  ...

  dx = ...;
  dy = ...;

  sum = 0.0;
  for (j = ...){    // Loop on sub-intervals in y
  for (i = ...){    // Loop on sub-intervals in x
    xc = ... // Center of interval (x)
    yc = ... // Center of interval (y)

    sumy = 0.0;
    for (jk = 0; jk < ngauss; jk++){
      sumx = 0.0;                        // Initialize sum for this interval
      for (ik = 0; ik < ngauss; ik++) { // Apply Gaussian rule to sub-interval
        sumx += wg[ik]*Func(xc + 0.5*dx*xg[ik], yc + 0.5*dy*xg[jk]);
      }
      sumx *= 0.5*dx;
      sumy += wg[jk]*sumx;
    }
    sumy *= 0.5*dy;
    sum += sumy;
  }}
}
```

Practice Session #3

- Here's a pseudo-code that enables us to employ our 1D implementation:

```
// Global variables are defined prior to any function can be seen everywhere throughout the file.
// A good common practice is to name them differently. Here I use "g_<name>"
static double g_ycoord;  // = fixed value of y for which we integrate along x
static int    g_nint;    // = number of intervals in quadrature function

int main()
{
  double ybeg, yend;   // Define here your domain boundary in y
  sum = GaussianQuad (GFunc1D, ybeg, yend, g_nint, 3);  //  Integrate in the y direction
  ...
}

double GFunc1D(double y)    // Return the integral G(y) = \int f(x,y) dx for a specified value of y.
{
  double sum_x, xbeg, xend; // Define here your domain boundary in x

  g_ycoord = y;             // This is the right place to change the global "y" coordinate
  sum_x = GaussianQuad (FFunc1D, xbeg, xend, g_nint, 3);  //  Integrate in the x-direction for fixed y
  return sum_x;
}

double FFunc1D(double x)    // Return the integrand f(x,y) for a specified value of y ( = g_ycoord).
{
  return Func2D (x, g_ycoord);
}

double Func2D(double x, double y) // This is the actual 2D function f(x,y).
{
  return f(x,y);
}
```

Practice Session #3

- Test your program on the function

$$f(x,y) = x^4y^2 + 2y^2x^2 - yx^2 + 2$$

for which the integral on $-1 \le x,y \le 1$ evaluates to $412/45$ (≈ 9.15556). Since the degree is 4, a Gaussian quadrature rule with $N_{gauss} \ge 3$ should compute the integral exactly.

- Next consider the unit disk again on the domain $[-1,1]^2$:

$$f(x,y) = \begin{cases} 1 & \text{if} \quad \sqrt{x^2+y^2} \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Using $N_{gauss} = 4$, how many intervals must be used to obtain π with an absolute accuracy of 10^{-5} ? Is the error uniformly decreasing ?