4. Romberg quadrature: accelerates convergence of Trapezoidal

It is based on a neat trick: Richardson extrapolation
which can applied to any algorithm with known error expansion;
i.e. if we know that error $= C_1 h + C_2 h^2 + \dots$ (with C_i independent of h)
Idea: compute with h and $\frac{h}{2}$, and form a linear combination of
the two results that kills the worst error term.

Romberg: on $T_N(f)$ = Trapezoidal for $I(f) = \int_a^b f(x)dx$ with N subintervals: $h = \frac{b-a}{N}$

error with N: $I - T_N = C_2 h^2 + C_4 h^4 + \dots$ (assuming smooth f)

error with $2N$: $I - T_{2N} = C_2 \frac{h^2}{4} + C_4 \frac{h^4}{16} + \dots$
$(h \to \frac{h}{2})$

Multiplying by 4 and subtracting from first one we kill h^2 term:

$$(I - T_N) - 4 \cdot (I - T_{2N}) = C_4 h^4 \left(1 - \frac{1}{4}\right) = O(h^4)$$

solve for I:

Romberg: $I = \frac{4T_{2N} - T_N}{3} + O(h^4)$ provides 4^{th} order approximation as good as Simpson!

Can repeat using T_{2N} and T_{4N} to get 6^{th} order!

Systematic formulas can be developed for consecutive Romberg terms.

Remarks: 1. The agreement in successive Romberg terms give good indication of accuracy

2. For specified accuracy, Romberg needs much smaller # of subintervals (less computation, so also reducing roundoff)

3. There is a very effective way of computing T_{2N} from T_N:

$$T_{2N} = \frac{1}{2} T_N + \frac{h}{2}\left[f\left(a + \frac{h}{2}\right) + f\left(a + \frac{3}{2}h\right) + \dots + f\left(a + \frac{2N-1}{2}h\right)\right]$$

which uses only N additional evaluations instead of $2N+1$ to get T_{2N}!

Example of Romberg for $I = \int_0^1 e^{-x^2}\,dx \approx 0.74682413$ to 8 decimals

In single precision:

N	Trapezoidal	$R = \frac{4T_{2N} - T_N}{3}$	Romberg again, $O(h^6)$
50	.74679947		
		.74682385	
100	.74681776		.74682356
		.74682357	
200	.74682212		

only up to 6 decimals correct can be obtained due to roundoff

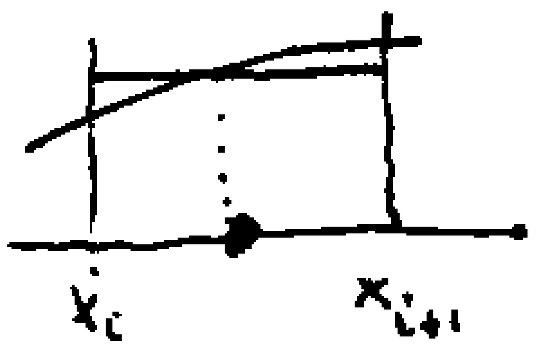
In double precision

N	T_N $O(h^2)$	$R^{(1)}$ $O(h^4)$	$R^{(2)}$ $O(h^6)$
50	.74679960		
		.74682413	
100	.74681800		.74682413
		.74682413	
200	.74682260		

Simpson with $N = 100$ also produces 8 correct digits

A′. Open Rules: avoid evaluation at end points

Midpoint Rule: version of Rectangle Rule that uses height at mid-point



$$\int_{x_i}^{x_{i+1}} f(x)\,dx \approx h_i \cdot f\left(\frac{x_i + x_{i+1}}{2}\right), \quad \text{local error} = -\frac{1}{24} f''(\xi_i)\cdot h_i^3$$

Composite: $\displaystyle\int_a^b f(x)\,dx \approx \sum_{i=0}^{N-1} h_i \cdot f_{i+\frac{1}{2}}$. error $= O(h^2)$: 2^{nd} order

Open rule from a closed rule: When $f(x)$ is known by a formula (so can evaluate it wherever we need to) we can construct an open rule from any closed rule for $I(f) = \int_a^b f(x)\,dx$:

Choose N and set $h = \frac{b-a}{N+2}$: nodes $x_{-1} = a$, $x_i = a + ih$ for $i = 0, \dots, N$, $x_{N+1} = b$

Apply closed Newton-Cotes rule on $[x_0, x_N] \subset [a, b]$ gives a corresponding open rule:

$$I(f) = \int_a^b f(x)\,dx \equiv \int_{x_{-1}}^{x_{N+1}} f(x)\,dx \approx \sum_{i=0}^{N} w_i f(x_i), \quad w_i = \int_a^b L_i(x)\,dx, \quad L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{x - x_j}{x_i - x_j}$$

$i = 0, \dots, N$

Precision of an $N+1$ point rule is $N+1$ if N = even, N if N = odd.

Roundoff error effect on Quadrature Rules: $|\text{roundoff error}| \lesssim (b-a)\cdot\varepsilon$

If $\tilde{f}_i = f_i + \varepsilon_i$, $\varepsilon = \max_i |\varepsilon_i|$, then $|\tilde{Q}_N - Q_N| \leq (b-a)\cdot\varepsilon$ is independent of N !!! (and of h)

Thus, quadrature is <u>well</u>-conditioned process, in contrast to num. differentiation

So we can use more points (bigger N, smaller h) to reduce discretization error $O(h^k)$ without getting penalized by roundoff growing!

B. Adaptive Quadrature: automatically choose nodes to meet a specified accuracy

Can be applied to any rule if the error can be estimated.

The method chooses nodes automatically to approximate the integral to within specified tolerance with "optimal efficiency" (if it works...) by placing more nodes where $f(x)$ varies rapidly, fewer where it does not. There are many versions of adaptive quadrature schemes...

Adaptive Simpson is one of the best

Idea: Apply Simpson with $N=2$ on $[a,b]$ to compute S_1
and also on $[a,m]$ and $[m,b]$, $m=\frac{a+b}{2}$ to compute S_2 $(N=4)$
If $|S_1 - S_2| < 16 \cdot TOL$ then the Richardson extrapolation value $R = \frac{16 S_2 - S_1}{15}$ is a good approximation for I.
else repeat on each of $[a,m]$ and $[m,b]$ with $TOL \to \frac{TOL}{2}$
Thus, some subintervals get refined to capture steep changes of $f(x)$.

Matlab's quad is adaptive Simpson: for $I(f)=\int_a^b F(x)\,dx$

$$\underbrace{[Q}_{\text{value}},\ \underbrace{nFeval}_{\#\text{ of F evaluations}}] = quad(\underbrace{F\_handle}_{@FCN},\ \underbrace{a,\ b}_{\text{interval}},\ \underbrace{TOL}_{\text{absolute Tolerance}})$$

quadgk : Gauss-Kronrod very high precision rule, adaptive

$$[Q,\ ERR] = quadgk\,(F\_handle,\ a,\ b,\ TOL)$$