See Chapter 7 of Süli and Mayers.

**Motivation:** we’ve seen oscillations in polynomial interpolation—the Runge phenomenon—for high-degree polynomials.

**Idea:** split a required integration interval \([a, b] = [x_0, x_n]\) into \(n\) equal intervals \([x_{i-1}, x_i]\) for \(i = 1, \ldots, n\). Then use a composite rule:

\[
\int_a^b f(x) \, dx = \int_{x_0}^{x_n} f(x) \, dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x) \, dx
\]
in which each \(\int_{x_{i-1}}^{x_i} f(x) \, dx\) is approximated by quadrature.

Thus rather than increasing the degree of the polynomials to attain high accuracy, instead increase the number of intervals.

**Trapezium Rule:**

\[
\int_{x_{i-1}}^{x_i} f(x) \, dx = \frac{h}{2} [f(x_{i-1}) + f(x_i)] - \frac{h^3}{12} f''(\xi_i), \quad \text{for some } \xi_i \in (x_{i-1}, x_i).
\]

**Composite Trapezium Rule:**

\[
\int_{x_0}^{x_n} f(x) \, dx = \sum_{i=1}^{n} \left[ \frac{h}{2} [f(x_{i-1}) + f(x_i)] - \frac{h^3}{12} f''(\xi_i) \right] = \frac{h}{2} [f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)] + e_h^T,
\]

where \(\xi_i \in (x_{i-1}, x_i)\) and \(h = x_i - x_{i-1} = (x_n - x_0)/n = (b - a)/n\), and the error \(e_h^T\) is given by

\[
e_h^T = -\frac{h^3}{12} \sum_{i=1}^{n} f''(\xi_i) = -\frac{nh^3}{12} f''(\xi) = -(b - a) \frac{h^2}{12} f''(\xi)
\]

for some \(\xi \in (a, b)\), using the Intermediate-Value Theorem \(n\) times. Note that if we halve the stepsize \(h\) by introducing a new point halfway between each current pair \((x_{i-1}, x_i)\), the factor \(h^2\) in the error should decrease by four.

**Another composite rule:** if \([a, b] = [x_0, x_{2n}]\),

\[
\int_a^b f(x) \, dx = \int_{x_0}^{x_{2n}} f(x) \, dx = \sum_{i=1}^{n} \int_{x_{2i-2}}^{x_{2i}} f(x) \, dx
\]
in which each \(\int_{x_{2i-2}}^{x_{2i}} f(x) \, dx\) is approximated by quadrature.

**Simpson’s Rule:**

\[
\int_{x_{2i-2}}^{x_{2i}} f(x) \, dx = \frac{h}{3} [f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] - \frac{(2h)^5}{2880} f''''(\xi_i), \quad \text{for some } \xi_i \in (x_{2i-2}, x_{2i}).
\]
Composite Simpson’s Rule:

\[
\int_{x_0}^{x_{2n}} f(x) \, dx = \sum_{i=1}^{n} \left[ \frac{h}{3} [f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] - \frac{(2h)^5}{2880} f^{(m)}(\xi_i) \right]
\]

= \frac{h}{3} \left[ f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots \right]

+ 2f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_{2n}) + e_h^g

where \(\xi_i \in (x_{2i-2}, x_{2i})\) and \(h = x_i - x_{i-1} = (x_{2n} - x_0)/2n = (b - a)/2n\), and the error \(e_h^g\) is given by

\[
e_h^g = -\frac{(2h)^5}{2880} \sum_{i=1}^{n} f^{(m)}(\xi_i) = -\frac{n(2h)^5}{2880} f^{(m)}(\xi) = -(b-a) \frac{h^4}{180} f^{(m)}(\xi)
\]

for some \(\xi \in (a,b)\), using the Intermediate-Value Theorem \(n\) times. Note that if we halve the stepsize \(h\) by introducing a new point half way between each current pair \((x_{i-1}, x_i)\), the factor \(h^4\) in the error should decrease by sixteen (assuming \(f\) is smooth enough).

**Adaptive (or automatic) procedure:** if \(S_h\) is the value given by Simpson’s rule with a stepsize \(h\), then

\[
S_h - S_{\frac{1}{2}h} \approx -\frac{15}{16} e_h^g.
\]

This suggests that if we wish to compute \(\int_a^b f(x) \, dx\) with an absolute error \(\varepsilon\), we should compute the sequence \(S_h, S_{\frac{1}{2}h}, S_{\frac{1}{4}h}, \ldots\) and stop when the difference, in absolute value, between two consecutive values is smaller than \(\frac{16}{15}\varepsilon\). That will ensure that (approximately) \(|e_h^g| \leq \varepsilon\).

Often spatially-varying adaptivity is used in practice: refine only is regions where a local estimate is large.

**Comments:**

Sometimes much better accuracy may be obtained using the Trapezoidal Rule: for example, as might happen when computing Fourier coefficients, if \(f\) is periodic with period \(b - a\) so that \(f(a + x) = f(b + x)\) for all \(x\).

**Software:**

Octave and Matlab have various routines like `quad` and `quadgk`. Fortran package QUADPACK. Recently Matlab has been promoting a single front-end `integral`. Python has `scipy.integrate.quadrature` as well as QUADPACK interfaces. All of these use adaptivity.