\[ E = \frac{f(2m)(\xi)}{(2m)!} \int_a^b \left( \prod_{i=0}^{m-1} (x-x_i) \right)^2 w(x) \, dx \] 

Where \( \xi \in (a,b) \).

\section*{4.5 Romberg Integration}

\textbf{Idea:} use Richardson's extrapolation to increase accuracy of composite trapezoidal rule.

\quad \Rightarrow \text{get high accuracy method from multiple applications of a low accuracy method.}

Recall composite trapezoidal rule:

\[ \int_a^b f(x) \, dx = \frac{b-a}{2} \left( f(a) + f(b) + 2 \sum_{j=1}^{n-1} f(x_j) \right) - \frac{(b-a)^3}{12} f''(\xi) \]

where \( \xi \in (a,b) \), \( h = \frac{b-a}{n} \) and \( x_j = a + jh \).

Romberg integration needs to evaluate composite trapezoidal rule

\begin{align*}
\text{for } n &= 2^k, \\
\text{for } k &= 0, 1, \ldots, M
\end{align*}

without superfluous function evaluations.

\[ x = \text{new function evaluations,} \]

\[ l = \text{function evaluation available from previous "level".} \]
Thus composite trapez Rule becomes with \( h_k = 2^{-k} \): 
\[
\int_a^b f(x) \, dx = \frac{h_k}{2} \left[ f(a) + f(b) + 2 \sum_{i=1}^{2^{k-1}} f(a + ih_k) \right] 
\]
\[
= R_{k,0} + \text{error} 
\]
where \( \Sigma \) is \( \Sigma (a_i, b) \).

So:
\[
R_{00} = \frac{h_0}{2} \left( f(a) + f(b) \right) = \frac{b-a}{2} \left( f(a) + f(b) \right)
\]
\[
R_{10} = \frac{h_1}{2} \left( f(a) + f(b) + 2f(a + h_1) \right)
\]
\[
= \frac{1}{2} R_{00} + R_1 f(a + h_1)
\]
\[
R_{20} = \frac{h_2}{2} R_{10} + R_2 \left( f(a + h_2) + f(a + 3h_2) \right)
\]
\[
R_{k0} = \frac{1}{2} R_0 + h_k \sum_{j=1}^{2^{k-1}} f(a + 2^k \cdot j - 1 \cdot h_k)
\]
Evaluation only at odd numbered nodes.

It can be shown that:

1. \[
\int_a^b f(x) \, dx = R_{k0} + K_1 \frac{h_k^2}{2} + K_2 \frac{h_k^4}{4} + K_3 \frac{h_k^6}{6} + \ldots
\]

2. \[
\int_a^b f(x) \, dx = R_{k+1,0} + K_1 R_{k+1} + K_2 \frac{h_{k+1}^4}{4} + K_3 \frac{h_{k+1}^6}{6} + \ldots 
\]
\[
= R_{k+1,0} + K_1 \frac{h_k^2}{4} + K_2 \frac{h_k^4}{16} + K_3 \frac{h_k^6}{64} + \ldots
\]
Do Richardson's extrapolation trick to cancel out leading term in error:

\[ 4x^2 - 1 : \]

\[ 3 \int_a^b f(x) \, dx = 4R_{k+1, 0} - R_k, 0 - \frac{3}{4} R_k^4 - \frac{15}{16} R_k^6 \]

\[ \int_a^b f(x) \, dx = 4 \frac{R_{k+1, 0} - R_k, 0 - \frac{1}{4} R_k^4 - \frac{5}{16} R_k^6}{4 - 1} \]

\[ = R_{k+1, 1} \approx O(h_k^4) \text{ approx} \]

If we keep doing same thing:

\[ R_{k+1, j} = \frac{4^j R_{k+1, j-1} - R_k, j-1}{4^j - 1} = O(h_k^{2(j+1)}) \]

Organizing work in a table we get: (for example)

\[ R_{10} \]
\[ R_{10} \rightarrow R_{11} \]
\[ R_{20} \rightarrow R_{21} \rightarrow R_{22} \rightarrow R_{23} \]
\[ R_{30} \rightarrow R_{31} \rightarrow R_{32} \rightarrow R_{33} \rightarrow R_{34} \rightarrow R_{43} \rightarrow R_{44} \]

\[ O(h_k^2) \quad O(h_k^4) \quad O(h_k^4) \quad O(h_k^8) \quad O(h_k^{10}) \]
§ 4.6 Adaptive quadrature methods

Using a uniform partition of an interval for a quadrature rule for approximating the integral of a function is not the most efficient way of doing things!

Obviously we can do better if we adapt the number of points to put more points where they are needed the most ⇒ principle behind adaptive methods.

key ingredient: we need a way that tells us how good our quadrature is working. We could do a number of things: for example using more modes to estimate error term or using a more accurate method. Here we choose to refine the number of points to estimate error.

On some interval \([a, b]\):

\[
\int_a^b f(x) \, dx = \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right) - \frac{b-a}{20} f^{(4)}(\xi)
\]

for some \(\xi \in (a, b)\) \(\equiv S(a,b)\) where \(h = \frac{b-a}{2}\).

Of course we could estimate error if we knew \(f^{(4)}\) but this is asking too much from end user!

Instead we apply Simpson’s rule again on two subintervals of \([a, b]\):

\[
\int_a^b f(x) \, dx = S\left(a, \frac{a+b}{2}\right) + S\left(\frac{a+b}{2}, b\right) - \frac{h^5}{2880} f^{(10)}(\xi_1) - \frac{h^5}{2880} f^{(10)}(\xi_2)
\]

\[
= \quad \quad \quad - \frac{h^5}{90}\left(\frac{1}{16}\right) f^{(4)}(\xi_3)
\]

where \(\xi_1, \xi_2 \in \left(\frac{a+b}{2}, b\right)\), and \(\xi_3 \in (a, b)\).
Now assume \( f^{(4)} \) does not vary too much on \((a,b)\) so:

\[
f^{(4)}(\xi) \approx f^{(4)}(\xi)
\]

so to estimate the error we simply do \((**\rightarrow *)\) to get:

\[
\frac{b^5}{30} f^{(4)}(\xi) \approx -\frac{16}{15} \left[ Sc(a, a^{\frac{1}{2}}) + Sc(b, \frac{b^{\frac{1}{2}}}{2}) - Sc(a, b) \right]
\]

Plugging into \((**\rightarrow *)\) and bounding cut-off portion error:

\[
\left| \int_a^b f(x)dx - Sc(a, a^{\frac{1}{2}}) - Sc(b, \frac{b^{\frac{1}{2}}}{2}) \right| \leq \frac{1}{15} \left| Sc(a, b) - Sc(a, a^{\frac{1}{2}}) - Sc(b, \frac{b^{\frac{1}{2}}}{2}) \right|
\]

So if we want \((**\rightarrow *)\) to be accurate to within \(E\) we need:

\[
\left| Sc(a, b) - Sc(a, a^{\frac{1}{2}}) - Sc(b, \frac{b^{\frac{1}{2}}}{2}) \right| < 15E
\]

To be conservative (and account for some changes in \(f^{(4)}\)) we can require 155 bound.

Algorithm: the book implements a recursive procedure using a for loop. This is like writing a tree traversal algorithm with for loops! Very complicated! => use recursion: i.e. a function that calls itself.

⚠️ Using recursion is probably less efficient than running code in the book, but it's much easier to understand!

⚠️ Also some programming languages are limited in # of recursion levels that are allowed e.g. if you run a C program you may get "stack overflow" or other messages of the kind.

⚠️ It may be good to impose a limit on # of recursions in "industrial" code as infinite recursions can be nastier than infinite loops!
function \( r = \text{asimpson}(a, b, f, \varepsilon) \)

\[
S = \frac{b-a}{6} \left( f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right)
\]

\[
S_1 = \frac{b-a}{12} \left( f(a) + 4f\left(a + \frac{b-a}{4}\right) + f\left(a + \frac{3(b-a)}{4}\right) \right)
\]

\[
S_2 = \frac{b-a}{12} \left( f\left(\frac{a+b}{2}\right) + 4f\left(\frac{a+b}{2} - \frac{3(b-a)}{4}\right) + f\left(b\right) \right)
\]

If \( |S - S_1 - S_2| < \varepsilon \)

\[ r = S_1 + S_2 \quad \therefore \text{we are happy with approx.} \]

else

\[ r = \text{asimpson}\left(a, \frac{a+b}{2}, f, \varepsilon/2\right) + \text{asimpson}\left(\frac{a+b}{2}, b, f, \varepsilon/2\right) \]

- recurse all of adaptive Simpson code for left and right halves of intervals.
- Requiring that error be \( \varepsilon/2 \) ensures the total error from left + right subintervals will not exceed \( \varepsilon \) when summed up.
- instead of accumulating only integrals we can also accumulate e.g. points of discretization.