



Figure 4.2.1. Sequential calls to the routine `trapzd` incorporate the information from previous calls and evaluate the integrand only at those new points necessary to refine the grid. The bottom line shows the totality of function evaluations after the fourth call. The routine `qsimp`, by weighting the intermediate results, transforms the trapezoid rule into Simpson's rule with essentially no additional overhead.

There are also formulas of higher order for this situation, but we will refrain from giving them.

The *semi-open formulas* are just the obvious combinations of equations (4.1.11)–(4.1.14) with (4.1.15)–(4.1.18), respectively. At the closed end of the integration, use the weights from the former equations; at the open end use the weights from the latter equations. One example should give the idea, the formula with error term decreasing as $1/N^3$ which is closed on the right and open on the left:

$$\int_{x_1}^{x_N} f(x)dx = h \left[\frac{23}{12}f_2 + \frac{7}{12}f_3 + f_4 + f_5 + \dots + f_{N-2} + \frac{13}{12}f_{N-1} + \frac{5}{12}f_N \right] + O\left(\frac{1}{N^3}\right) \quad (4.1.20)$$

CITED REFERENCES AND FURTHER READING:

Abramowitz, M., and Stegun, I.A. 1964, *Handbook of Mathematical Functions*, Applied Mathematics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), §25.4. [1]

Isaacson, E., and Keller, H.B. 1966, *Analysis of Numerical Methods* (New York: Wiley), §7.1.

4.2 Elementary Algorithms

Our starting point is equation (4.1.11), the extended trapezoidal rule. There are two facts about the trapezoidal rule which make it the starting point for a variety of algorithms. One fact is rather obvious, while the second is rather “deep.”

The obvious fact is that, for a fixed function $f(x)$ to be integrated between fixed limits a and b , one can double the number of intervals in the extended trapezoidal rule without losing the benefit of previous work. The coarsest implementation of the trapezoidal rule is to average the function at its endpoints a and b . The first stage of refinement is to add to this average the value of the function at the halfway point. The second stage of refinement is to add the values at the 1/4 and 3/4 points. And so on (see Figure 4.2.1).

Without further ado we can write a routine with this kind of logic to it:

```

SUBROUTINE trapzd(func,a,b,s,n)
INTEGER n
REAL a,b,s,func
EXTERNAL func
  This routine computes the nth stage of refinement of an extended trapezoidal rule. func is
  input as the name of the function to be integrated between limits a and b, also input. When
  called with n=1, the routine returns as s the crudest estimate of  $\int_a^b f(x)dx$ . Subsequent
  calls with n=2,3,... (in that sequential order) will improve the accuracy of s by adding  $2^{n-2}$ 
  additional interior points. s should not be modified between sequential calls.
INTEGER it,j
REAL del,sum,tnm,x
if (n.eq.1) then
  s=0.5*(b-a)*(func(a)+func(b))
else
  it=2**(n-2)
  tnm=it
  del=(b-a)/tnm           This is the spacing of the points to be added.
  x=a+0.5*del
  sum=0.
  do 11 j=1,it
    sum=sum+func(x)
    x=x+del
  enddo 11
  s=0.5*(s+(b-a)*sum/tnm)  This replaces s by its refined value.
endif
return
END

```

The above routine (`trapzd`) is a workhorse that can be harnessed in several ways. The simplest and crudest is to integrate a function by the extended trapezoidal rule where you know in advance (we can't imagine how!) the number of steps you want. If you want $2^M + 1$, you can accomplish this by the fragment

```

do 11 j=1,m+1
  call trapzd(func,a,b,s,j)
enddo 11

```

with the answer returned as `s`.

Much better, of course, is to refine the trapezoidal rule until some specified degree of accuracy has been achieved:

```

SUBROUTINE qtrap(func,a,b,s)
INTEGER JMAX
REAL a,b,func,s,EPS
EXTERNAL func
PARAMETER (EPS=1.e-6, JMAX=20)
C  USES trapzd
  Returns as s the integral of the function func from a to b. The parameters EPS can be set
  to the desired fractional accuracy and JMAX so that 2 to the power JMAX-1 is the maximum
  allowed number of steps. Integration is performed by the trapezoidal rule.
INTEGER j
REAL olds
olds=-1.e30           Any number that is unlikely to be the average of the function
do 11 j=1,JMAX        at its endpoints will do here.
  call trapzd(func,a,b,s,j)
  if (j.gt.5) then    Avoid spurious early convergence.
    if (abs(s-olds).lt.EPS*abs(olds).or.
      (s.eq.0..and.olds.eq.0.)) return
  endif
  olds=s
enddo 11
pause 'too many steps in qtrap'
END

```

Sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X)
 Copyright (C) 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software.
 Permission is granted for internet users to make one paper copy for their own personal use. Further reproduction, or any copying of machine-
 readable files (including this one) to any server computer, is strictly prohibited. To order Numerical Recipes books, diskettes, or CDROMs
 visit website <http://www.nr.com> or call 1-800-872-7423 (North America only), or send email to trade@cup.cam.ac.uk (outside North America).

Unsophisticated as it is, routine `qtrap` is in fact a fairly robust way of doing integrals of functions that are not very smooth. Increased sophistication will usually translate into a higher-order method whose efficiency will be greater only for sufficiently smooth integrands. `qtrap` is the method of choice, e.g., for an integrand which is a function of a variable that is linearly interpolated between measured data points. Be sure that you do not require too stringent an EPS, however: If `qtrap` takes too many steps in trying to achieve your required accuracy, accumulated roundoff errors may start increasing, and the routine may never converge. A value 10^{-6} is just on the edge of trouble for most 32-bit machines; it is achievable when the convergence is moderately rapid, but not otherwise.

We come now to the “deep” fact about the extended trapezoidal rule, equation (4.1.11). It is this: The error of the approximation, which begins with a term of order $1/N^2$, is in fact *entirely even* when expressed in powers of $1/N$. This follows directly from the *Euler-Maclaurin Summation Formula*,

$$\int_{x_1}^{x_N} f(x) dx = h \left[\frac{1}{2} f_1 + f_2 + f_3 + \cdots + f_{N-1} + \frac{1}{2} f_N \right] - \frac{B_2 h^2}{2!} (f'_N - f'_1) - \cdots - \frac{B_{2k} h^{2k}}{(2k)!} (f_N^{(2k-1)} - f_1^{(2k-1)}) - \cdots \quad (4.2.1)$$

Here B_{2k} is a *Bernoulli number*, defined by the generating function

$$\frac{t}{e^t - 1} = \sum_{n=0}^{\infty} B_n \frac{t^n}{n!} \quad (4.2.2)$$

with the first few even values (odd values vanish except for $B_1 = -1/2$)

$$\begin{aligned} B_0 &= 1 & B_2 &= \frac{1}{6} & B_4 &= -\frac{1}{30} & B_6 &= \frac{1}{42} \\ B_8 &= -\frac{1}{30} & B_{10} &= \frac{5}{66} & B_{12} &= -\frac{691}{2730} \end{aligned} \quad (4.2.3)$$

Equation (4.2.1) is not a convergent expansion, but rather only an asymptotic expansion whose error when truncated at any point is always less than twice the magnitude of the first neglected term. The reason that it is not convergent is that the Bernoulli numbers become very large, e.g.,

$$B_{50} = \frac{495057205241079648212477525}{66}$$

The key point is that only even powers of h occur in the error series of (4.2.1). This fact is not, in general, shared by the higher-order quadrature rules in §4.1. For example, equation (4.1.12) has an error series beginning with $O(1/N^3)$, but continuing with all subsequent powers of N : $1/N^4$, $1/N^5$, etc.

Suppose we evaluate (4.1.11) with N steps, getting a result S_N , and then again with $2N$ steps, getting a result S_{2N} . (This is done by any two consecutive calls of

trapzd.) The leading error term in the second evaluation will be 1/4 the size of the error in the first evaluation. Therefore the combination

$$S = \frac{4}{3}S_{2N} - \frac{1}{3}S_N \quad (4.2.4)$$

will cancel out the leading order error term. But there *is* no error term of order $1/N^3$, by (4.2.1). The surviving error is of order $1/N^4$, the same as Simpson's rule. In fact, it should not take long for you to see that (4.2.4) is *exactly* Simpson's rule (4.1.13), alternating 2/3's, 4/3's, and all. This is the preferred method for evaluating that rule, and we can write it as a routine exactly analogous to qtrap above:

```

SUBROUTINE qsimp(func,a,b,s)
INTEGER JMAX
REAL a,b,func,s,EPS
EXTERNAL func
PARAMETER (EPS=1.e-6, JMAX=20)
C USES trapzd
  Returns as s the integral of the function func from a to b. The parameters EPS can be set
  to the desired fractional accuracy and JMAX so that 2 to the power JMAX-1 is the maximum
  allowed number of steps. Integration is performed by Simpson's rule.
INTEGER j
REAL os,ost,st
ost=-1.e30
os= -1.e30
do || j=1,JMAX
  call trapzd(func,a,b,st,j)
  s=(4.*st-ost)/3.          Compare equation (4.2.4), above.
  if (j.gt.5) then         Avoid spurious early convergence.
    if (abs(s-os).lt.EPS*abs(os).or.
*      (s.eq.0..and.os.eq.0.)) return
  endif
  os=s
  ost=st
enddo ||
pause 'too many steps in qsimp'
END

```

The routine qsimp will in general be more efficient than qtrap (i.e., require fewer function evaluations) when the function to be integrated has a finite 4th derivative (i.e., a continuous 3rd derivative). The combination of qsimp and its necessary workhorse trapzd is a good one for light-duty work.

CITED REFERENCES AND FURTHER READING:

- Stoer, J., and Bulirsch, R. 1980, *Introduction to Numerical Analysis* (New York: Springer-Verlag), §3.3.
- Dahlquist, G., and Bjorck, A. 1974, *Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall), §§7.4.1–7.4.2.
- Forsythe, G.E., Malcolm, M.A., and Moler, C.B. 1977, *Computer Methods for Mathematical Computations* (Englewood Cliffs, NJ: Prentice-Hall), §5.3.

4.3 Romberg Integration

We can view Romberg's method as the natural generalization of the routine `qsimp` in the last section to integration schemes that are of higher order than Simpson's rule. The basic idea is to use the results from k successive refinements of the extended trapezoidal rule (implemented in `trapzd`) to remove all terms in the error series up to but not including $O(1/N^{2k})$. The routine `qsimp` is the case of $k = 2$. This is one example of a very general idea that goes by the name of *Richardson's deferred approach to the limit*: Perform some numerical algorithm for various values of a parameter h , and then extrapolate the result to the continuum limit $h = 0$.

Equation (4.2.4), which subtracts off the leading error term, is a special case of polynomial extrapolation. In the more general Romberg case, we can use Neville's algorithm (see §3.1) to extrapolate the successive refinements to zero stepsize. Neville's algorithm can in fact be coded very concisely within a Romberg integration routine. For clarity of the program, however, it seems better to do the extrapolation by subroutine call to `polint`, already given in §3.1.

```

SUBROUTINE qromb(func,a,b,ss)
INTEGER JMAX,JMAXP,K,KM
REAL a,b,func,ss,EPS
EXTERNAL func
PARAMETER (EPS=1.e-6, JMAX=20, JMAXP=JMAX+1, K=5, KM=K-1)
C USES polint, trapzd
  Returns as ss the integral of the function func from a to b. Integration is performed by
  Romberg's method of order 2K, where, e.g., K=2 is Simpson's rule.
  Parameters: EPS is the fractional accuracy desired, as determined by the extrapolation
  error estimate; JMAX limits the total number of steps; K is the number of points used in
  the extrapolation.
INTEGER j
REAL dss,h(JMAXP),s(JMAXP)           These store the successive trapezoidal approximations
h(1)=1.                               and their relative stepsizes.
do j=1,JMAX
  call trapzd(func,a,b,s(j),j)
  if (j.ge.K) then
    call polint(h(j-KM),s(j-KM),K,0.,ss,dss)
    if (abs(dss).le.EPS*abs(ss)) return
  endif
  s(j+1)=s(j)
  h(j+1)=0.25*h(j)                   This is a key step: The factor is 0.25 even though
  enddo j                               the stepsize is decreased by only 0.5. This makes
  pause 'too many steps in qromb'      the extrapolation a polynomial in  $h^2$  as allowed
END                                       by equation (4.2.1), not just a polynomial in  $h$ .

```

The routine `qromb`, along with its required `trapzd` and `polint`, is quite powerful for sufficiently smooth (e.g., analytic) integrands, integrated over intervals which contain no singularities, and where the endpoints are also nonsingular. `qromb`, in such circumstances, takes many, *many* fewer function evaluations than either of the routines in §4.2. For example, the integral

$$\int_0^2 x^4 \log(x + \sqrt{x^2 + 1}) dx$$