11. Adaptive Quadrature

Previously, we considered a simple iterative algorithm based on interval doubling for computing an approximation to the value of an integral. In such a method, based on the composite trapezoidal rule, for example, a simple stopping criterion might be to quit when two successive approximations agree to a given tolerance. The problem with such a procedure is that at each iteration, all the subintervals are cut in half without taking into account that we may already have a very good approximation on some subintervals. Thus, the procedure is inefficient since we are doing function evaluations that do not lead to improved accuracy.

For example, if we consider the composite trapezoidal rule on N not necessarily equally spaced subintervals, we have:

$$I(f) = \int_a^b f(x)\,dx = \sum_{i=1}^N I_i(f) = \sum_{i=1}^N \int_{x_{i-1}}^{x_i} f(x)\,dx = \sum_{i=1}^N (x_i - x_{i-1})[f(x_{i-1} + f(x_i)]/2 \\ - \sum_{i=1}^N f''(\eta_i)(x_i - x_{i-1})^3/12, \quad \eta_i \in (x_{i-1}, x_i).$$

Hence the contribution to the error formula from the ith subinterval, $f''(\eta_i)(x_i - x_{i-1})^3/12$, depends on both the size of the subinterval and the size of f'' on the subinterval. Where f'' is large, we expect to have to take smaller subintervals to control the error.

11.1. **Estimating the error on a subinterval.**

Since we don't know the size of f'' on the subintervals, we need some way to estimate the error on the subinterval. There are several ways this can be done. One is to make use of the interval doubling strategy on a subinterval.

Recall that the approximation of $I_i(f) = \int_{x_{i-1}}^{x-i} f(x)\,dx$ given by the trapezoidal rule using only one subinterval is

$$I_i(f) = (x_i - x_{i-1})[f(x_{i-1}) + f(x_i)]/2 - f''(\eta_i)(x_i - x_{i-1})^3/12.$$

If we break up the subinterval (x_{i-1}, x_i) into two equal sized subintervals and use the trapezoidal rule on each, then we get the formula

$$\begin{aligned} I_i(f) &= (x_{i-1/2} - x_{i-1})[f(x_{i-1} + f(x_{i-1/2})]/2 + (x_i - x_{i-1/2})[f(x_{i-1/2} + f(x_i)]/2 \\ &\quad - f''(\eta_i^1)(x_{i-1/2} - x_{i-1})^3/12 - f''(\eta_i^2)(x_i - x_{i-1/2})^3/12 \\ &= (x_i - x_{i-1})[f(x_{i-1}) + 2f(x_{i-1/2}) + f(x_i)]/4 \\ &\quad - f''(\xi_i)(x_i - x_{i-1})^3/48. \end{aligned}$$

Now, if f'' does not change much on the subinterval (x_{i-1}, x_i), we might assume that $f''(\eta_i) \approx f''(\xi_i)$. For example, if f''' is continuous on (x_{i-1}, x_i), then by the Mean Value Theorem,

$$|f''(\eta_i) - f''(\xi_i)| = |f'''(\delta_i)(\eta_i - \xi_i)| \le M_3|\eta_i - \xi_i| \le M_3|x_i - x_{i-1}|.$$

Setting $\xi_i = \eta_i$, we get

$$\begin{aligned} &f''(\eta_i)(x_i - x_{i-1})^3/12 - f''(\eta_i)(x_i - x_{i-1})^3/48 \\ &\qquad \approx (x_i - x_{i-1})[f(x_{i-1}) + f(x_i)]/2 - (x_i - x_{i-1})[f(x_{i-1}) + 2f(x_{i-1/2}) + f(x_i)]/4. \end{aligned}$$

Hence,

$$f''(\eta_i)(x_i - x_{i-1})^3/16 \approx (x_i - x_{i-1})[f(x_{i-1}) - 2f(x_{i-1/2}) + f(x_i)]/4.$$

Thus an estimate for the error (using the trapezoidal rule on one subinterval) is given by:

$$I_i(f) - (x_i - x_{i-1})[f(x_{i-1}) + f(x_i)]/2 \approx -(1/3)(x_i - x_{i-1})[f(x_{i-1}) - 2f(x_{i-1/2}) + f(x_i)].$$

An estimate for the error (using the trapezoidal rule on two subintervals) is given by:

$$\begin{aligned} &I_i(f) - (x_i - x_{i-1})[f(x_{i-1}) + 2f(x_{i-1/2}) + f(x_i)]/4 \\ &\qquad \approx -(1/12)(x_i - x_{i-1})[f(x_{i-1}) - 2f(x_{i-1/2}) + f(x_i)]. \end{aligned}$$

Another way of obtaining such estimates is to approximate $f''(x)$ on the interval $[x_{i-1}, x_i]$ by the second difference

$$\frac{f(x_{i-1}) - 2f(x_{i-1/2}) + f(x_i)}{[(x_i - x_{i-1})/2]^2}.$$

A second approach is to approximate the integral on a subinterval by two different formulas, say $A_i^1(f)$ and $A_i^2(f)$, with different orders of accuracy, e.g.,

$$|I_i(f) - A_i^1(f)| = O(h^r), \qquad |I_i(f) - A_i^2(f)| = O(h^s),$$

where we assume $s > r$. Then

$$|I_i(f) - A_i^1(f)| \le |I_i(f) - A_i^2(f)| + |A_i^2(f) - A_i^1(f)| \le |A_i^2(f) - A_i^1(f)| + O(h^s).$$

Since $|I_i(f) - A_i^1(f)| = O(h^r)$, $|A_i^2(f) - A_i^1(f)|$ gives an estimate for the dominant part of the error. For example, we could approximate $I_i(f)$ by the trapezoidal rule using two subintervals, or using the same function evaluations, by Simpson's rule

$$I_i(f) \approx (x_i - x_{i-1})[f(x_{i-1}) + 4f(x_{i-1/2}) + f(x_i)]/6.$$

Then, an estimate for the error using the trapezoidal rule on two subintervals is given by:

$$\begin{aligned} &|(x_i - x_{i-1})[f(x_{i-1}) + 4f(x_{i-1/2}) + f(x_i)]/6 - (x_i - x_{i-1})[f(x_{i-1}) + 2f(x_{i-1/2}) + f(x_i)]/4| \\ &\qquad = (x_i - x_{i-1})|f(x_{i-1}) - 2f(x_{i-1/2}) + f(x_i)|/12. \end{aligned}$$

Of course, we expect that the use of Simpson's rule gives better accuracy, so if we accept the approximation, we should use the approximation given by Simpson's rule.

These approaches are easily generalized to higher order closed Newton-Cotes formulas using equally spaced points.

11.2. **An adaptive algorithm.** In an adaptive algorithm for numerical integration, we use our estimates for the local error on each subinterval to decide whether we have a sufficiently accurate approximation or whether we need to decrease the size of some subintervals to increase the accuracy of the approximation. We now discuss how this can be done. First, we need to supply an error tolerance. This might be an absolute error tolerance, i.e., $|E| \leq \epsilon_a$ or a relative error tolerance, $|E| \leq \epsilon_r I(f)$. Many codes ask for both, and quit when $|E| \leq \epsilon_a + \epsilon_r A(f)$, where $A(f)$ denotes the current approximation to $I(f)$.

Suppose we are trying to make the total error $\leq \epsilon$. One way to proceed (simpler to program) is to allow an error of $\epsilon h_i/(b-a)$ on a subinterval of length h_i, where $b-a$ denotes the total length of the interval of integration. Then, if we had say N subintervals, i.e., $\sum_{i=1}^N h_i = b-a$, the total error would be

$$\sum_{i=1}^{N} \epsilon h_i/(b-a) = \epsilon.$$

If on any subinterval, our estimate of the local error exceeds this threshold, then we cut the subinterval size in half and apply our method on each piece with the aim of satisfying the above criterion. Note that if we cut the interval size in half, then we are allowing an error of only $\epsilon h_i/(2[b-a])$ on each piece. However, even using the trapezoidal rule, our local error estimate is of order h_i^3, so cutting h_i in half, should reduce the error by a factor of 8. This is why we expect that halving the subinterval size will reduce the total error. This way of proceeding assumes we want to equidistribute the error in the sense that we are trying to make the error proportional to the subinterval size. However, the goal is to make the total error less than ϵ, so we may be doing unnecessary work. Another possibility is the following. Suppose we have on each interval $I_i = [x_{i-1}, x_i]$ an estimate of the error given by ϵ_i. Then the total error is given by $\sum_i \epsilon_i$. We now simply pick the largest ϵ_i and divide the corresponding interval I_i in half.

An adaptive code based on Simpson's rule is used in Matlab. As in the trapezoidal rule discussion above, rather than compare Simpson's rule based on one and two subintervals, the code compares Simpson's rule based on two subintervals with a higher order formula obtained by extrapolation of Simpson's rule (i.e., as done in Romberg integration). Note that the extrapolated formula does not require any new function evaluations.

11.3. **Adaptive Gauss codes.** Just as we have developed composite Newton-Cotes quadrature formulas, we can also develop composite Gauss quadrature formulas. If we use these in an adaptive way, then we need a means of adding new quadrature points to produce a higher order formula that can be used to estimate the local errors on each subinterval. In addition, we do not want the calculation of this higher order formula to be too expensive. This is a problem for Gauss quadrature formulas, since the points used for lower order formulas are not a subset of the points used for higher order formulas. This has led to the development of quadrature formulas based on the following idea. Suppose we have a Gaussian quadrature formula of the form

$$\int_a^b w(x)f(x)\,dx = \sum_{j=0}^{n} H_j f(x_j)$$

and we want to produce a new formula (using $x_0, \ldots, x_n$ and m new abscissas $x_{n+1}, \ldots, x_{n+m}$) of the form

$$\int_a^b w(x)f(x)\,dx = \sum_{j=0}^{n+m} K_j f(x_j)$$

that is exact for polynomials of as high a degree as possible. Since there are $2m+n+1$ free parameters (m abscissas and $n+m+1$ weights), we hope to produce a formula that is exact for polynomials of degree $\leq 2m+n$. In general, it is not clear that this can be done with points $x_{n+1}, \ldots, x_{n+m} \in (a,b)$. Kronrod studied the case when $m = n+2$ and produced successful formulas pairing a Gauss rule with a higher order Konrod formula that reused the points from the Gauss rule.

Example: The 3-point Gauss rule on [-1,1] ($n = 2$) is given by

$$I(f) \approx (8/9)f(0) + (5/9)[f(-\sqrt{.6}) + f(\sqrt{.6})].$$

The Konrod rule adds 4 points to produce a rule of the form

$$I(f) \approx \alpha_0 f(0) + \alpha_1[f(-\sqrt{.6}) + f(\sqrt{.6})] + \alpha_2[f(-\beta_1) + f(\beta_1)] + \alpha_3[f(-\beta_2) + f(\beta_2)]$$

with β_1^2 and β_2^2 the smallest and largest roots, respectively, of $x^2 - (10/9)x + (155/891) = 0$. As before, the weights $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ are determined by integrating the Lagrange polynomial of degree 6 interpolating $f(x)$ at the nodes $(0, \pm\sqrt{.6}, \pm\beta_1, \pm\beta_2)$.

For the case $w(x) \equiv 1$, Patterson produced a sequence of formulas starting from a given Gauss-Legendre rule $I_{n_0}(f) = \sum_{i=0}^{n_0} H_j f(x_j)$ and adding exactly $n+2$ points at each step, where n denotes the number of points used in the current formula. All the new abscissas lie in (a,b) and all the weights are positive. For example, choosing $n_0 = 2$, i.e., a 3-point Gauss rule, we get the formulas $I_2, I_6, I_{14}, I_{30}, I_{62}, \ldots$. Thus, the points for each successive higher order formula include all the points from the previous formula.

11.4. Gauss-Radau and Gauss-Lobatto integration.

In some applications, it is important to use one or both of the end points of the interval of integration as quadrature points. If we require one of the end points to be a quadrature point and choose the remaining points and weights to make the formula exact for polynomials as high a degree as possible, we get a Radau quadrature formula. Analogous formulas in which both end points are chosen as quadrature points are called Lobatto quadrature formulas.

Gauss-Lobatto integration rules are reasonable choices for composite formulas, since the endpoints are used on two adjoining subintervals. An effective adaptive scheme has been written based on a Gauss-Lobatto formula and a Konrod-type formula to estimate the error. See W. Gander and W. Gautschi, Adaptive Quadrature - Revisited, BIT Vol. 40, No. 1, March 2000, pp. 84–101.