

Lecture 23: Interpolatory Quadrature

4. Quadrature.

The computation of continuous least squares approximations to $f \in C[a, b]$ required evaluations of the inner product $\langle f, \phi_j \rangle = \int_a^b f(x)\phi_j(x) dx$, where ϕ_j is a polynomial (a basis function for \mathcal{P}_n). Often such integrals may be difficult or impossible to evaluate exactly, so our next charge is to develop algorithms that approximate such integrals quickly and accurately. This field is known as *quadrature*, a name that suggests the approximation of the area under a curve by area of subtending quadrilaterals.[†]

4.1. Interpolatory Quadrature with Prescribed Nodes.

Given $f \in C[a, b]$, we seek approximations to the definite integral

$$\int_a^b f(x) dx.$$

If $p_n \in \mathcal{P}_n$ interpolates f at $n + 1$ points in $[a, b]$, then we might hope that

$$\int_a^b f(x) dx \approx \int_a^b p_n(x) dx.$$

This approach is known as *interpolatory quadrature*. Will such rules produce a reasonable estimate to the integral? Of course, that depends on properties of f and the interpolation points. When we interpolate at equally spaced points, the formulas that result are called *Newton–Cotes quadrature rules*.

You encountered the most basic method for approximating an integral when you learned calculus: the Riemann integral is motivated by approximating the area under a curve by the area of rectangles that touch that curve, which gives a rough estimate that becomes increasingly accurate as the width of those rectangles shrinks. This amounts to approximating the function f by a piecewise constant interpolant, and then computing the exact integral of the interpolant. When only one rectangle is used to approximate the entire integral, we have the most simple *Newton–Cotes* formula. This approach can be improved in two complementary ways: increasing the degree of the interpolating polynomial, and reducing the width of the subintervals over which each interpolating polynomial applies. The first approach leads to the *trapezoid rule* and *Simpson’s rule*; the second yields *composite rules*, where the integral over $[a, b]$ is split into the sum of integrals over subintervals. In many cases, the function f may be fairly regular over part of the domain $[a, b]$, but then have some region of rapid growth or oscillation. We ultimately seek quadrature software that automatically detects such regions, ensuring that sufficient function evaluations are performed there without requiring excessive effort on areas where f is well-behaved. Such *adaptive quadrature* procedures, discussed briefly at the end of these notes, are essential for practical problems.[‡]

4.1.1 Trapezoid Rule.

[†]The term *quadrature* is used to distinguish the numerical approximation of a definite integral from the numerical solution of an ordinary differential equation, which is often called *numerical integration*. Approximation of a double integral is sometimes called *cubature*.

[‡]For more details on the topic of interpolatory quadrature, see Süli and Mayers, Chapter 7, which has guided many aspects of our presentation here.

The trapezoid rule is a simple improvement over approximating the integral by the area of a single rectangle. A linear interpolant to f can be constructed, requiring evaluation of f at the interval end points $x = a$ and $x = b$,

$$p_1(x) = f(a) + \frac{f(b) - f(a)}{b - a} (x - a).$$

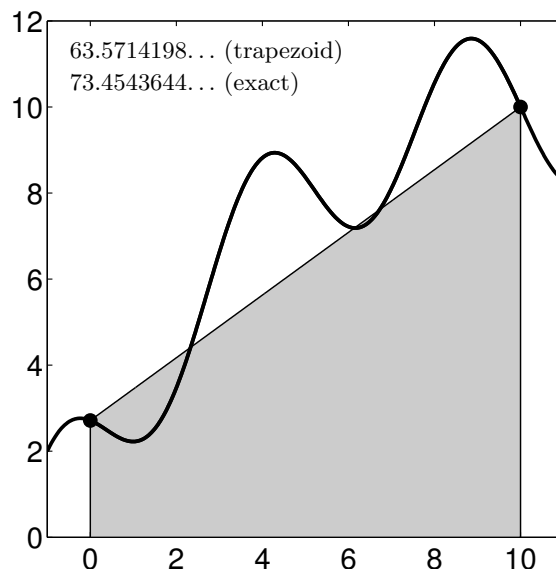
The integral of p_1 approximates the integral of f :

$$\begin{aligned} \int_a^b p_1(x) dx &= \int_a^b \left(f(a) + \frac{f(b) - f(a)}{b - a} (x - a) \right) dx \\ &= \left[f(a)x \right]_a^b + \left(\frac{f(b) - f(a)}{b - a} \right) \left[\frac{1}{2}(x - a)^2 \right]_a^b \\ &= \frac{b - a}{2} (f(a) + f(b)). \end{aligned}$$

In summary,

$$\textbf{Trapezoid Rule:} \quad \int_a^b f(x) dx \approx \frac{b - a}{2} (f(a) + f(b)).$$

The procedure behind the trapezoid rule is illustrated in the following picture, where the area approximating the integral is colored gray.



Error bound for the trapezoid rule. To derive an error bound, we simply integrate the interpolation error bound developed in Lecture 11. That bound ensured that for each $x \in [a, b]$, there exists some $\xi \in [a, b]$ such that

$$f(x) - p_1(x) = \frac{1}{2} f''(\xi)(x - a)(x - b).$$

Note that ξ will vary with x , which we emphasize by writing $\xi(x)$ below. Integrating, we obtain

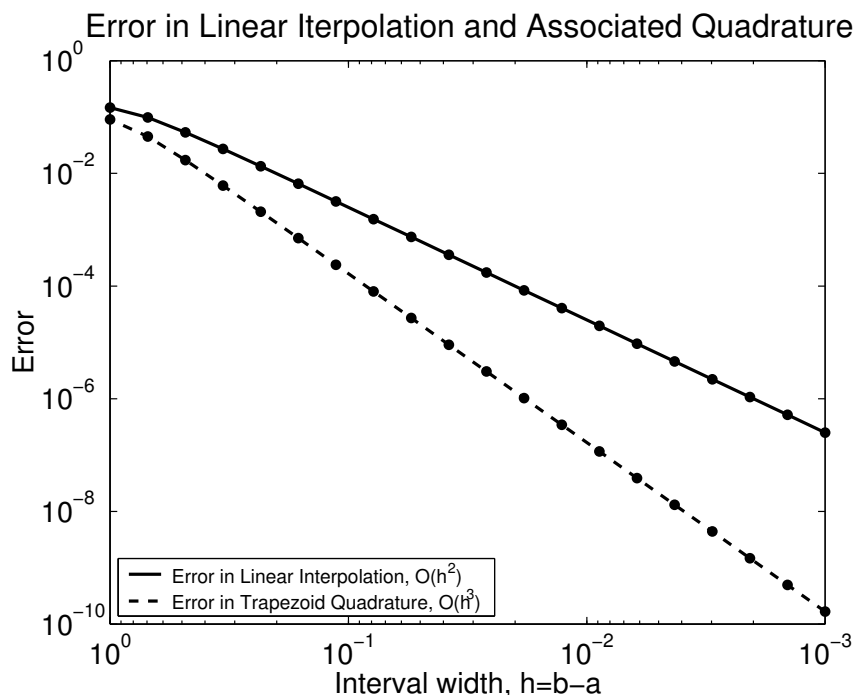
$$\int_a^b f(x) dx - \int_a^b p_1(x) dx = \int_a^b \frac{1}{2} f''(\xi(x))(x - a)(x - b) dx$$

$$\begin{aligned}
 &= \frac{1}{2} f''(\eta) \int_a^b (x-a)(x-b) dx \\
 &= \frac{1}{2} f''(\eta) \left(\frac{1}{6} a^3 - \frac{1}{2} a^2 b + \frac{1}{2} a b^2 - \frac{1}{6} b^3 \right) \\
 &= -\frac{1}{12} f''(\eta) (b-a)^3
 \end{aligned}$$

for some $\eta \in [a, b]$. The second step follows from the mean value theorem for integrals.[§]

(We shall develop a much more general theory from which we can derive this error bound, plus bounds for more complicated schemes, too, in Lecture 23b.)

Example: $f(x) = e^x(\cos x + \sin x)$. Here we demonstrate the difference between the error for linear interpolation of a function, $f(x) = e^x(\cos x + \sin x)$, between two points, $x_0 = 0$ and $x_1 = h$, and the trapezoid rule applied to the same interval. Our theory predicts that linear interpolation will have an $O(h^2)$ error as $h \rightarrow 0$, while the trapezoid rule has $O(h^3)$ error.



4.1.2 Simpson’s rule. We expect that better accuracy can be attained by replacing the trapezoid rule’s linear interpolant with a higher degree polynomial interpolant to f over $[a, b]$. This will increase the number of times we must evaluate f (often very costly), but hopefully will significantly decrease the error. Indeed it does – by an even greater margin than we might expect.

Simpson’s rule follows from using a quadratic approximation that interpolates f at a , b , and the midpoint $(a + b)/2$. If we use the Newton form of the interpolant with $x_0 = a$, $x_1 = b$, and

[§]The mean value theorem for integrals states that if $h, g \in C[a, b]$ and h does not change sign on $[a, b]$, then there exists some $\eta \in [a, b]$ such that $\int_a^b g(t)h(t) dt = g(\eta) \int_a^b h(t) dt$. The requirement that h not change sign is essential. For example, if $g(t) = h(t) = t$ then $\int_{-1}^1 g(t)h(t) dt = \int_{-1}^1 t^2 dt = 2/3$, yet $\int_{-1}^1 h(t) dt = \int_{-1}^1 t dt = 0$, so for all $\eta \in [-1, 1]$, $g(\eta) \int_{-1}^1 h(t) dt = 0 \neq \int_{-1}^1 g(t)h(t) dt = 2/3$.

$x_2 = (a + b)/2$, we obtain

$$p_2(x) = f(a) + \frac{f(b) - f(a)}{b - a} (x - a) + \frac{2f(a) - 4f(\frac{1}{2}(a + b)) + 2f(b)}{(b - a)^2} (x - a)(x - b).$$

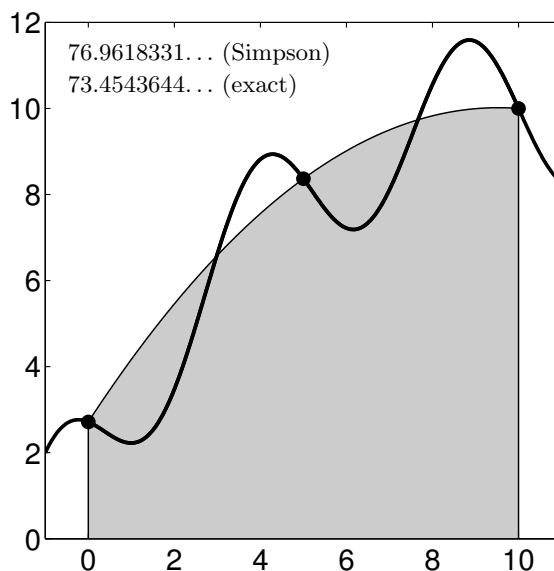
Simpson's rule then approximates the integral of f with the integral of p_2 :

$$\begin{aligned} \int_a^b p_2(x) dx &= \int_a^b p_1(x) dx + \frac{2f(a) - 4f(\frac{1}{2}(a + b)) + 2f(b)}{(b - a)^2} \int_a^b (x - a)(x - b) dx \\ &= \frac{b - a}{2} (f(a) + f(b)) + \frac{2f(a) - 4f(\frac{1}{2}(a + b)) + 2f(b)}{(b - a)^2} \frac{(b - a)^3}{6} \\ &= \frac{b - a}{6} \left(f(a) + 4f(\frac{1}{2}(a + b)) + f(b) \right), \end{aligned}$$

where we have used the fact that the first two terms of p_2 are identical to the linear approximation p_1 used above for the trapezoid rule. In summary:

Simpson's Rule:
$$\int_a^b f(x) dx \approx \frac{b - a}{6} \left(f(a) + 4f(\frac{1}{2}(a + b)) + f(b) \right).$$

The picture below shows an application of Simpson's rule on $[a, b] = [0, 10]$.



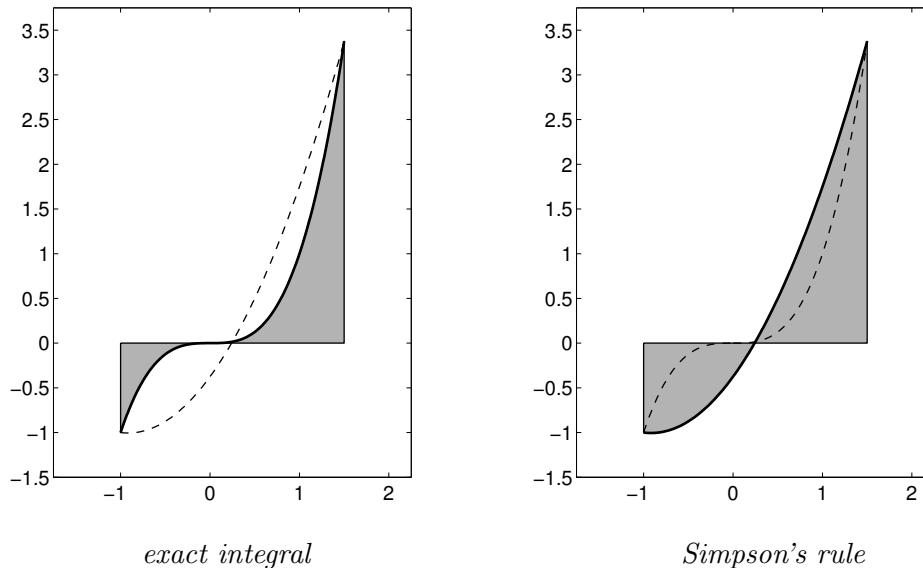
Error bound for Simpson's rule. Simpson's rule enjoys a remarkable feature: though it only approximates f by a quadratic, *it integrates any cubic polynomial exactly!* One can verify this by directly applying Simpson's rule to a generic cubic polynomial.[¶] In Lecture 23b, we shall derive the tools to compute an error bound for Simpson's rule:

$$\int_a^b f(x) dx - \int_a^b p_2(x) dx = -\frac{1}{90} \frac{(b - a)^5}{2^5} f^{(4)}(\eta)$$

[¶]It turns out that Newton–Cotes formulas based on approximating f by an even-degree polynomial always exactly integrate polynomials one degree higher.

for some $\eta \in [a, b]$.

We emphasize that although Simpson's rule is exact for cubics, the interpolating polynomial we integrate really is quadratic. Though this should be clear from the discussion above, you might find it helpful to see this graphically. Both plots below show a cubic function f (solid line) and its quadratic interpolant (dashed line). On the left, the area under f is colored gray – its area is the integral we wish to compute. On the right, the area under the interpolant is colored gray. Accounting area below the x axis as negative, both integrals give an identical value. It is remarkable that this is the case for *all* cubics.



4.1.3 Clenshaw–Curtis quadrature. To get faster convergence for a fixed number of function evaluations, one might wish to increase the degree of the approximating polynomial still further, then integrate that high-degree polynomial. As we learned in our study of polynomial interpolation, the success of such an approach depends significantly on the choice of the interpolation points. For example, we would not expect to get an accurate answer by integrating a high degree polynomial that interpolates Runge's function $f(x) = (x^2 + 1)^{-1}$ over uniformly spaced points on $[-5, 5]$.

One expects to get better results by integrating the interpolant to f at Chebyshev points. This procedure is known as *Clenshaw–Curtis quadrature*. The formulas get a bit intricate, but the results are fantastic if f is smooth (e.g., analytic in a region of the complex plane containing $[a, b]$).^{||}

4.1.4 Composite rules. As an alternative to integrating a high-degree polynomial, one can pursue a simpler approach that is often very effective: Break the interval $[a, b]$ into subintervals, and apply the trapezoid rule or Simpson's rule on each subinterval. Applying the trapezoid rule gives

$$\int_a^b f(x) dx = \sum_{j=1}^n \int_{x_{j-1}}^{x_j} f(x) dx \approx \sum_{j=1}^n \frac{(x_j - x_{j-1})}{2} (f(x_{j-1}) + f(x_j)).$$

The standard implementation assumes that f is evaluated at uniformly spaced points between a

^{||}See L. N. Trefethen, 'Is Gauss Quadrature Better than Clenshaw–Curtis?', *SIAM Review* 50 (2008) 67–87.

and b , $x_j = a + jh$ for $j = 0, \dots, n$ and $h = (b - a)/n$, giving the following famous formulation:

$$\textbf{Composite Trapezoid:} \quad \int_a^b f(x) \, dx \approx \frac{h}{2} \left(f(a) + 2 \sum_{j=1}^{n-1} f(a + jh) + f(b) \right).$$

(Of course, one can readily adjust this rule to cope with irregularly spaced points.) The error in the composite trapezoid rule can be derived by summing up the error in each application of the trapezoid rule:

$$\begin{aligned} \int_a^b f(x) \, dx - \frac{h}{2} \left(f(a) + 2 \sum_{j=1}^{n-1} f(a + jh) + f(b) \right) &= \sum_{j=1}^n \left(-\frac{1}{12} f''(\eta_j) (x_j - x_{j-1})^3 \right) \\ &= -\frac{h^3}{12} \sum_{j=1}^n f''(\eta_j) \end{aligned}$$

for $\eta_j \in [x_{j-1}, x_j]$. We can simplify these f'' terms by noting that $\frac{1}{n} (\sum_{j=1}^n f''(\eta_j))$ is the average of n values of f'' evaluated at points in the interval $[a, b]$. Naturally, this average cannot exceed the maximum or minimum value that f'' assumes on $[a, b]$, so there exist points $\xi_1, \xi_2 \in [a, b]$ such that

$$f''(\xi_1) \leq \frac{1}{n} \sum_{j=1}^n f''(\eta_j) \leq f''(\xi_2).$$

Thus the intermediate value theorem guarantees the existence of some $\eta \in [a, b]$ such that

$$f''(\eta) = \frac{1}{n} \sum_{j=1}^n f''(\eta_j).$$

The composite trapezoid error bound thus simplifies to

$$\int_a^b f(x) \, dx - \frac{h}{2} \left(f(a) + 2 \sum_{j=1}^{n-1} f(a + jh) + f(b) \right) = -\frac{h^2}{12} (b - a) f''(\eta).$$

This error analysis has an important consequence: *the error for the composite trapezoid rule is only $O(h^2)$, not the $O(h^3)$ we saw for the usual trapezoid rule (in which case $b - a = h$ since $n = 1$).*

Similar analysis can be performed to derive the composite Simpson's rule. We now must ensure that n is even, since each interval on which we apply the standard Simpson's rule has width $2h$. Simple algebra leads to the formula

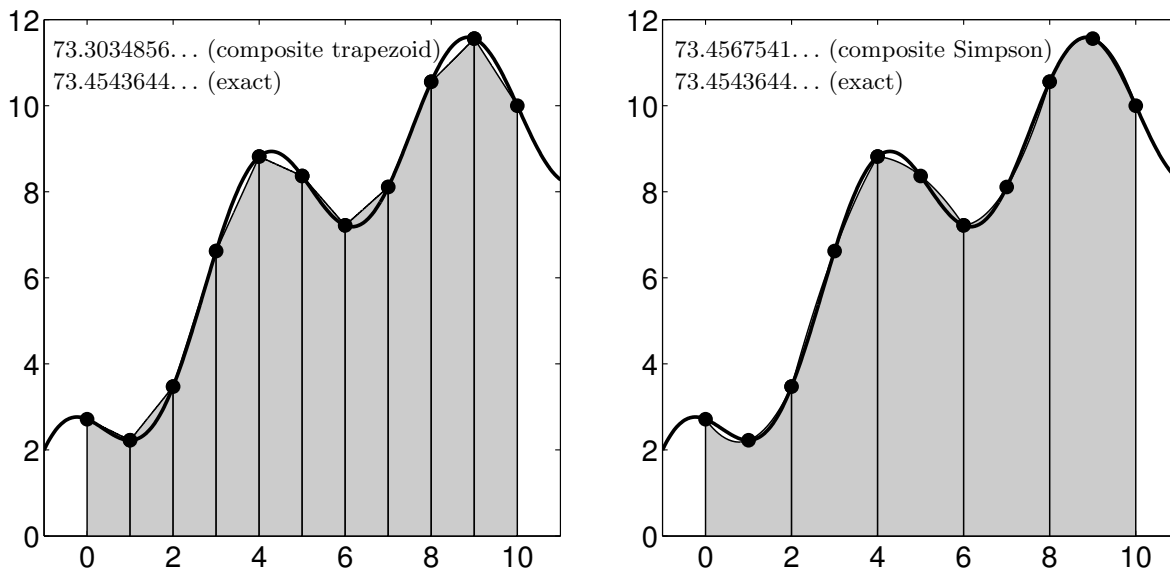
$$\textbf{Composite Simpson:} \quad \int_a^b f(x) \, dx \approx \frac{h}{3} \left(f(a) + 4 \sum_{j=1}^{n/2} f(a + (2j-1)h) + 2 \sum_{j=1}^{n/2-1} f(a + 2jh) + f(b) \right).$$

Derivation of the error formula for the composite Simpson's rule follows the same strategy as the analysis of the composite trapezoid rule. One obtains

$$\int_a^b f(x) \, dx - \frac{h}{3} \left(f(a) + 4 \sum_{j=1}^{n/2} f(a + (2j-1)h) + 2 \sum_{j=1}^{n/2-1} f(a + 2jh) + f(b) \right) = -\frac{h^4}{180} (b - a) f^{(4)}(\eta)$$

for some $\eta \in [a, b]$.

The illustrations below compare the composite trapezoid and Simpson's rules for the same number of function evaluations. One can see that Simpson's rule, in this typical case, gives the better accuracy.



Next we present a MATLAB script implementing the composite trapezoid rule; Simpson's rule is a straightforward modification that is left as an exercise. To get the standard (not composite) trapezoid rule, call `trapezoid` with `N=1`.

```
function intf = trapezoid(f, a, b, N)
% Composite trapezoid rule to approximate the integral of f from a to b.
% Uses N+1 function evaluations (N>1; h=(b-a)/N).

h = (b-a)/N; intf = 0;
intf = (feval(f,a)+feval(f,b))/2;
for j=1:N-1
    intf = intf + feval(f,a+j*h);
end
intf = intf*h;
```

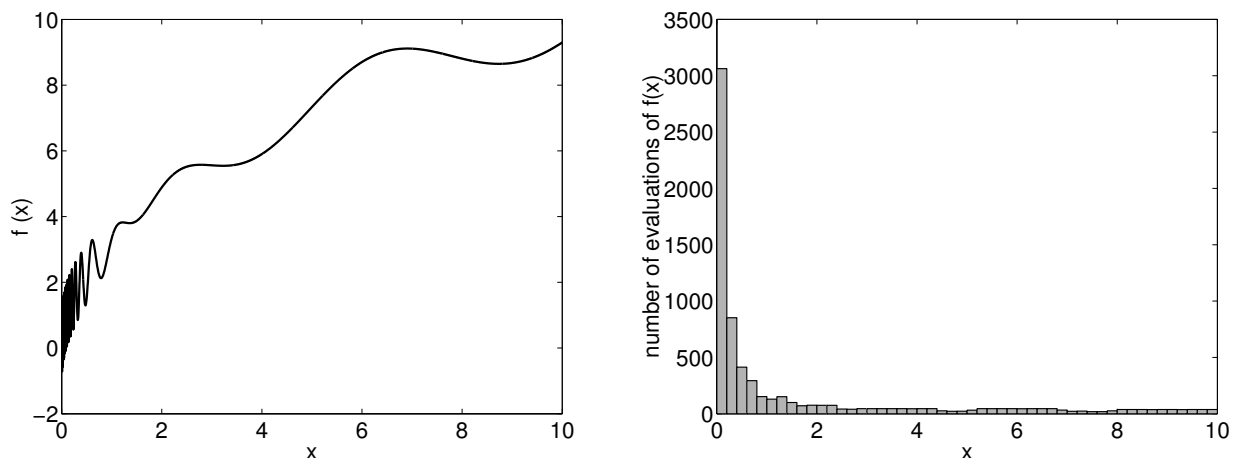
Pause a moment for reflection. Suppose you are willing to evaluate f a fixed number of times. How can you get the most bang for your buck? If f is smooth, a rule based on a high-order interpolant (such as Clenshaw–Curtis quadrature, or the Gaussian quadrature rules we will present in a few lectures) are likely to give the best result. If f is not smooth (e.g., with kinks, discontinuous derivatives, etc.), then a robust composite rule would be a good option. (A famous special case: If the function f is sufficiently smooth and is periodic with period $b - a$, then the trapezoid rule converges *exponentially*.)

Adaptive Quadrature. If f is continuous, we can attain arbitrarily high accuracy with composite rules by taking the spacing between function evaluations, h , to be sufficiently small. This might

be necessary to resolve regions of rapid growth or oscillation in f . If such regions only make up a small proportion of the domain $[a, b]$, then uniformly reducing h over the entire interval will be unnecessarily expensive. One wants to concentrate function evaluations in the region where the function is the most ornery. Robust quadrature software adjusts the value of h locally to handle such regions. To learn more about such techniques, which are not foolproof, see W. Gander and W. Gautschi, “Adaptive quadrature—revisited,” *BIT* 40 (2000) 84–101.**

MATLAB’s quadrature routines. The MATLAB quadrature routine `quad` implements an adaptive composite Simpson’s rule. A different quadrature routine, `quadl`, uses Gaussian quadrature, which we shall talk about a few classes from now.

The following illustrations show MATLAB’s adaptive quadrature rule at work. On the left, we have a function that varies smoothly over most of the domain, but oscillates wildly over 10% the region of interest. The plot on the right is a histogram of the number of function evaluations used by MATLAB’s `quad`. Clearly, this routine uses many more function evaluations in the region where the function oscillates most rapidly (`quad` identified this region itself; the user only supplies the region of integration $[a, b]$).



**This paper criticizes the routines `quad` and `quad8` that were included in MATLAB version 5. In light of this analysis MATLAB improved its software, essentially incorporating the two routines suggested in this paper in version 6 as the routines `quad` and `quadl`.