Week 12: Wednesday, Apr 18

Adaptive error control

Last time, we discussed Simpson's rule for quadrature:

$$I[f] = \int_a^b f(x)\,dx \approx \frac{b-a}{6}\left(f(a) + 4f(c) + f(b)\right), \quad c \equiv \frac{a+b}{2}$$

Simpson's rule has a local error of $O(h^5)$ where h is the size of one panel[1], and the composite rule has an error of $O(h^4)$. Let $S(a,b)$ denote the Simpson's rule estimate for the panel from a to b. Then we know that

$$\begin{aligned} I[f] &= S(a,b) + Ch^5 + O(h^7) \\ I[f] &= S(a,c) + S(c,b) + 2C(h/2)^5 + O(h^7), \end{aligned}$$

Combining these estimates, we have that the error in the two-panel rule is approximately

$$E(a,b) = \frac{S(a,c) + S(c,b) - S(a,b)}{15}.$$

One way to take advantage of this error estimate is by using extrapolation to increase the degree of our method. If we wanted, we could keep continually uniformly subdivide the mesh on which we have sampled the function in order to get ever higher-degree quadrature rules; this is sometimes called *Romberg integration*. Another way to use the error estimate, though, is to *adaptively* refine our mesh. That is, we keep a running tally of the estimated error on each panel, and any panel that seems to contribute too much error gets subdivided. There are multiple ways in which to decide the order in which one should process the panels that need to be subdivided. The most elegant version might be a priority queue (subdivide the panel with the highest error estimate first), but there are also simpler recursive variants that try to keep subdividing until a leaf panel of size h has estimated error of at most ηh. MATLAB's `quad` uses an adaptive Simpson's rule, but I'm not sure which refinement strategy it uses.

Note that while in principle $E(a,b)$ is an error estimate for $S(a,c)+S(c,b)$, in practice we use this as an error estimate for the purposes of things like

[1]For this lecture, h is the size of a panel, and not $2h$.

adaptive refinement... but we also return the extrapolated estimate $S(a,c)+S(c,b)+E(a,b)$, even though $E(a,b)$ is not technically an error estimate for the extrapolated rule. We like to try to have our cake and eat it to.

Raising the degree

An interpolatory quadrature rule through n points has degree $n-1$, and so yields (total) error that decreases at least like $O(h^n)$, assuming that the function in question is sufficiently smooth. In some cases, though, we know that we get lucky and do even better. For example, the midpoint rule ($n=1$) has degree 2, and Simpson's rule ($n=3$) has degree 4. Why is this the true?

For convenience, let us consider a quadrature rule on $[-1,1]$. A quadrature rule with n points has degree $n+s$ for $s \geq 0$, that means it computes any polynomial of degree up to $n+s$ exactly. In particular, if $x_1,\ldots,x_n$ are the nodes, we can define the degree n polynomial $q(x)=(x-x_1)\ldots(x-x_n)$, and our rule should be able to integrate $q(x)x^j$ exactly for $0 \leq j \leq s$. But notice that $q(x)x^j$ is exactly zero at each of the quadrature nodes, so the quadrature rule returns exactly zero at each of these points. Therefore, the quadrature rule can have degree $n+s$ for $s \geq 0$ only if it satisfies the conditions

$$\int_{-1}^{1} q(x)x^j \, dx = 0, \quad 0 \leq j \leq s.$$

This says that with respect to the standard inner product for functions on $[-1,1]$, the polynomial q should be *orthogonal* to x^j for $0 \leq j \leq s$. Note that we must have $s < n$, since otherwise we would have that $\int_{-1}^{1} q(x)^2 \, dx$ was zero.

As it happens, the *Legendre polynomials* $P_k(x)$ satisfy the property that $P_0(x),\ldots,P_d(x)$ forms an orthogonal basis (with respect to the standard inner product on $[-1,1]$) for the degree d polynomials. The first few Legendre polynomials are

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ P_2(x) &= (3x^2-1)/2, \end{aligned}$$

and we can compute higher-order Legendre polynomials by a recurrence:

$$(k+1)P_{k+1}(x) = (2k+1)xP_k(x) - kP_{k-1}(x).$$

Interpolatory quadrature rules based on interpolation through the zeros of Legendre polynomials are *Gauss-Legendre* quadrature rules. The midpoint rule is the lowest-order such rule; the second rule is

$$\int_{-1}^{1} f(x)\,dx \approx f(-\sqrt{1/3}) + f(\sqrt{1/3}).$$

In general, n-point Gauss-Legendre quadrature rules have degree $2n-1$; the two-point Gauss-Legendre rule has degree 3, for example.

There are a few variants on the Gaussian integration theme. One involves constraining the nodes for computational convenience. For example, if we insist that the interval endpoints must be quadrature nodes, we arrive at the Gauss-Lobatto rules (degree $2n-3$). The Gauss-Kronrod rules involve a pair consisting of an n-point Gauss quadrature rule together with a $2n+1$ point rule that re-uses the Gauss quadrature nodes; these rules are popular for adaptive quadrature, since the Gauss rule and the Kronrod rule can be compared in order to get an error estimate.

High order vs adaptivity

The default MATLAB `quad` routine does not use Gauss-Kronrod quadrature rules (though `quadgk` does). Instead, it uses an adaptive Simpson's rule. There's a good reason for this: *high order convergence is only achieved for functions with lots of derivatives.* If a function has a discontinuity, it will be hard to get better than $O(h)$ global error; if there is a discontinuous derivative, it is hard to beat $O(h^2)$ error; and so on. Methods based on Gauss quadrature or Chebyshev interpolation (the Clenshaw-Curtis methods) converge ferociously quickly for smooth integrands. But simple adaptive methods based on Simpson's rule often converge fast enough for most purposes, while remaining remarkably robust to nonexistent – or just very large – higher order derivatives.

Special behaviors

Consider the integral

$$I(z) = \int_{-z}^{z} \frac{\cos(x)}{\sqrt{|x|}}\,dx$$

How could we compute $I(\pi/2)$ numerically?

First, note that the function is even, so we can compute

$$I(z) = 2 \int_0^z x^{-1/2} \cos(x)\, dx.$$

This function is now awkward because the integrand diverges at $x = 0$. We can deal with this in a few different ways:

1. **Subtracting off the singularity**: Write

   $$I(z) = 2 \left[\int_0^z x^{-1/2}\, dx + \int_0^z x^{-1/2} \left(\cos(x) - 1 \right)\, dx \right].$$

   The first term can be handled analytically:

   $$\int_0^z x^{-1/2}\, dx = 2\sqrt{z}.$$

   The second term has a removable singularity at the origin ($O(x^{3/2})$ as $x \to 0$), and we can treat the integrand as zero at that point.

2. **Integration by parts**: If we integrate by parts, we have

   $$\int_0^z x^{-1/2} \cos(x)\, dx = 2\sqrt{z}\cos(z) + \int_0^z 2\sqrt{x}\sin(x)\, dx$$

3. **Change of variables**: If we let $t^2 = x$, then we can use the change of variables formula to recast the integral with as

   $$I(z) = 2 \int_0^{z^2} t^{-1} \cos(t^2)(2t\, dt) = 4 \int_0^{z^2} \cos(t^2) dt$$

   As it happens, we could at this point declare victory, since this integral is closely related to the Fresnel integral

   $$C(x) = \int_0^x \cos(\pi t^2/2)\, dt,$$

   and there are libraries that will compute Fresnel integrals for you.

4. **Special quadrature rules**: The *Gauss-Jacobi* family of quadrature rules approximates integrals of the form

$$\int_{-1}^{1} f(x)(1-x)^{\alpha}(1+x)^{\beta}\,dx$$

If we set $\alpha = 0$, and $\beta = -1/2$, then this gives us a rule for computing something with an inverse-square-root singularity at $x = -1$. If we apply the change of variables

$$y = \frac{z}{2}(1+x),$$

we have

$$\int_{-1}^{1} f(y(x))(1+x)^{-1/2}\,dx = \left(\frac{2}{z}\right)^{1/2} \int_{0}^{z} y^{-1/2} f(y)\,dy,$$

so we have a quadrature rule that deals with integrals with this sort of singularity. Let $f(y) = \cos(y)$ and we're all set.

These are most ot the tricks I know for dealing with integrals with singularities or unbounded domains. Fortunately, these tricks tend to work well for a variety of problems.

Problems to ponder

1. How would you write an n-panel composite Simpson rule in MATLAB *without* any repeated function evaluations at the same point.

2. Show that running extrapolation on the trapezoidal rule results in Simpson's rule.

3. How might you numerically compute

$$\int_{0}^{\infty} \ln(x)\exp(-x)\,dx$$

to a relative error tolerance of around 10^{-6}, ideally without too many function evaluations?

4. Consider the integral $\int_0^z x^{-1/2} \cos(x)\, dx$ from our "subtracting the singularity" example. How many derivatives does the integrand have at zero? How does this compare to the original integral?

5. Adaptive quadrature methods can be tricked! How would you find a polynomial such that one-panel or two-panel Simpson quadrature on $[-1, 1]$ both return zero, even though the true integal is positive?

6. Suppose $f(x)$ is convex, i.e. $f''(x) \geq 0$ everywhere. Show that in this case, the composite midpoint rule underestimates the true integral.

7. Describe a strategy for devising Simpson-like rules for integrals of the form
$$\int_a^b x^\alpha f(x)\, dx$$
where $\alpha > -1$ is given and $f(x)$ is assumed to be smooth. Your rule should sample the function at a, b, and at some point in between – how would you choose the location of that point to get the highest possible order of accuracy?

   Note: it's fine to express the coefficients in this rule in terms of integrals that you know how to work out symbolically, even if you don't want to run through the algebra.