Polynomial Interpolating Quadrature

Atkinson Chapter 5, Stoer & Bulirsch Chapter 3, Dahlquist & Bjork Chapter 5

Sections marked with * are not on the exam

1 Quadrature. Will first consider 1D $f \in C[a,b]$, then singularities, then multivariate.

The method is to use a polynomial to approximate f, then to integrate the polynomial exactly. We know how to do minimax approximations, L^2 approximations, and interpolation. Minimax approximations are hard to compute; L^2 approximations don't lead to easy error analysis; we stick to 'interpolating' quadratures.

For an interpolating quadrature,

$$I[f] := \int_a^b f(x)\mathrm{d}x \approx \int_a^b p(x)\mathrm{d}x = \int \sum_{i=0}^{n} f(x_i)\ell_i(x)\mathrm{d}x = \sum_{i=0}^{n} w_i f(x_i) := I_n[f] \text{ where } w_i = \int_a^b \ell_i(x)\mathrm{d}x.$$

The above shows how to construct an interpolating quadrature, but it doesn't tell us how good the approximation is. An obvious approach is to use the interpolation error:

$$\int_a^b f(x)\mathrm{d}x - \int_a^b p(x)\mathrm{d}x = \int_a^b f(x) - p(x)\mathrm{d}x = \int_a^b \frac{\Psi(x)}{(n+1)!} f^{(n+1)}(\xi(x))\mathrm{d}x.$$

We already know that the interpolation error doesn't necessarily go to 0 as $n \to \infty$. This approach is therefore not widely used and piecewise polynomial interpolation is used instead. That being said, there are 2 categories of method where the function is approximated globally by a single polynomial that is then integrated exactly:

- Gaussian quadrature
- Clenshaw-Curtis and Fejér quadrature

Both of these methods converge because special nodes x_i are chosen carefully so that Runge phenomenon, i.e. non-convergence of the interpolant, is avoided. Interestingly, in both of these cases the convergence analysis is not based on direct analysis of the interpolation error. This is a general feature of the error analysis of interpolating quadratures: you *can* analyze the error using the interpolation error formula, but there are often better methods.

2 Interpolatory quadrature. Simple example: piecewise linear. Suppose you have $f_i = f(x_i)$ where $a = x_0 < x_1 < \cdots < x_n = b$, and you want $\int_a^b f(x)\mathrm{d}x$. First make a piecewise-linear interpolating polynomial. Define $x_{i+1} - x_i = h_i$ for notational convenience.

$$p(x) = \frac{1}{h_i}\left[f_{i+1}(x - x_i) + f_i(x_{i+1} - x)\right] \text{ for } x_i \le x \le x_{i+1}.$$

The interpolation error on the i^{th} interval is

$$f(x) - p(x) = \frac{(x - x_i)(x - x_{i+1})}{2} f''(\xi_i).$$

The integral of f is approximated by the integral of p. The integral over a single subinterval is

$$\int_{x_i}^{x_{i+1}} p(x)\mathrm{d}x = \frac{1}{2h_i}[f_{i+1}h_i^2 + f_i h_i^2] = \frac{f_{i+1} + f_i}{2} h_i$$

This is the area of a trapezoid (draw picture), so it's called the trapezoid rule. Adding up all the subintervals yields

$$I[f] \approx I[p] = \sum_{i=0}^{n-1} \frac{f_{i+1} + f_i}{2} h_i.$$

This is sometimes called the 'composite' trapezoid rule, or just the trapezoid rule. The error on a single interval is

$$\int_{x_i}^{x_{i+1}} \frac{(x-x_i)(x-x_{i+1})}{2} f''(\xi_i(x))\mathrm{d}x.$$

Now since $(x-x_i)(x-x_{i+1})$ is non-positive on the interval we can use the integral mean value theorem:

$$\int_{x_i}^{x_{i+1}} \frac{(x-x_i)(x-x_{i+1})}{2} f''(\xi_i(x))\mathrm{d}x = f''(\xi_i)\int_{x_i}^{x_{i+1}} \frac{(x-x_i)(x-x_{i+1})}{2}\mathrm{d}x = -f''(\xi_i)\frac{h_i^3}{12}.$$

If f'' is bounded, the error can be bounded as follows

$$|\int_{x_i}^{x_{i+1}} \frac{(x-x_i)(x-x_{i+1})}{2} f''(\xi_i(x))\mathrm{d}x| \le \frac{\max|f''|}{2}\int |(x-x_i)(x-x_{i+1})|\mathrm{d}x.$$

Now notice that $(x-x_i)(x-x_{i+1})$ is strictly negative over the integral, so we can just integrate it and then change sign after the fact. The answer is (easy, just some calc & algebra)

$$|\int_{x_i}^{x_{i+1}} \frac{(x-x_i)(x-x_{i+1})}{2} f''(\xi_i(x))\mathrm{d}x| \le \frac{\max|f''|}{12}h_i^3.$$

The total error is the sum of the errors on each subinterval, so, assuming f'' is bounded, the total error is bounded by

$$|\text{error}| \le \max|f''|\sum_{i=0}^{n-1}\frac{h_i^3}{12}.$$

If all the intervals have the same size $h_i = h$ then

$$|\text{error}| \le \max|f''|\frac{nh^3}{12} = \frac{(b-a)h^2}{12}\max|f''|.$$

To construct the quadrature rule you just use interpolation. To get an error estimate you use the interpolation error estimate, but then you do a little extra clever work when bounding the integral of the error. The method is 'second order' since the error is bounded by h^2 as $h \to 0$ (it's still second order for unequal h_i, as long as the largest one goes to zero).

3 Simplest example: piecewise constant. Suppose you have $a = x_0 < x_1 < \cdots < x_n = b$, but you instead have $f_i = f(x_{i+1/2})$ where $x_{i+1/2} = (x_{i+1} + x_i)/2$ is the midpoint of each subinterval. We can approximate f using a piecewise-constant 'polynomial' on each interval

$$p(x) = f_i \text{ for } x_i \le x < x_{i+1}.$$

The interpolation error on each subinterval is

$$f(x) - p(x) = (x - x_{i+1/2})f'(\xi_i(x)) \text{ for } x_i \le x < x_{i+1}.$$

The integration error on each subinterval is just the integral of the interpolation error. The obvious thing to do is to take absolute values, assume $|f'|$ is bounded, and then do the integral, which gives order h^2 error on each subinterval. Adding the errors up leads to an overall order h error.

This is technically correct: the error is bounded above by a constant times h as $h \to 0$. But a better bound is possible. Consider that the piecewise constant approximation is also a Taylor approximation, whose error formula can be expanded to

$$f(x) = f_i + (x - x_{i+1/2})f'(x_{i+1/2}) + \frac{(x-x_{i+1/2})^2}{2}f''(\xi_i(x))$$

so

$$f(x) - f_i = f(x) - p(x) = (x - x_{i+1/2})f'(x_{i+1/2}) + \frac{(x-x_{i+1/2})^2}{2}f''(\xi_i(x)) \text{ for } x_i \le x < x_{i+1}.$$

If we integrate this error, the first term drops out (integrates to 0), leaving

$$\int_{x_i}^{x_{i+1}} f(x) - p(x)\mathrm{d}x = \int_{x_i}^{x_{i+1}} \frac{(x - x_{i+1/2})^2}{2} f''(\xi_i(x))\mathrm{d}x.$$

As long as $|f''|$ is bounded on the subinterval, this can be bounded by

$$|\int_{x_i}^{x_{i+1}} f(x) - p(x)\mathrm{d}x| \leq \frac{\max|f''|h_i^3}{24}.$$

Supposing that $h = h_i$ and adding up the error bounds across all n subintervals we have that the error for the 'composite' midpoint rule is bounded by

$$\frac{(b-a)h^2}{24}\max|f''|.$$

This is a better bound than the trapezoid rule! Of course that doesn't mean that the midpoint rule error will always be smaller than the trapezoid rule error, just that we derived a better bound. The midpoint method is second order precisely because it uses the midpoint of the interval; any other piecewise-constant approximation will be just first order. This whole analysis shows that using the interpolation is just one way to analyze the quadrature error, and that it's often not the best way.

4 Simpson's rule. It uses a piecewise-quadratic approximation and equispaced points. The main point of today's lecture is to introduce a new way to compute the error without recourse to the standard interpolation error formula.

WLOG we'll consider a single subinterval where $x_0 = -h$, $x_1 = 0$, $x_2 = h$, and $f_i = f(x_i)$. The quadratic interpolating polynomial is

$$p(x) = f_0 + (x+h)f[-h, 0] + x(x+h)f[-h, 0, h].$$

The exact integral, after some simplification, is

$$\int p(x)\mathrm{d}x = \frac{h}{3}(f_0 + 4f_1 + f_2).$$

Just like trapezoid and midpoint, you can form a composite rule out of this. Atkinson derives an error bound directly from the interpolation error formula, but it requires some *ad hoc* tricks.[1]

5 There is another approach to estimating (bounding) the error that applies to any quadrature based on polynomial interpolation, including Hermite interpolation (S&B 3.2). It is formulated not in terms of interpolation, but just in terms of being able to integrate a polynomials exactly up to a certain degree. So we need some preliminaries. Note that if you have $n+1$ distinct interpolation points, and if $f(x)$ is a polynomial of degree $\leq n$ then the interpolating polynomial will recover $f(x)$ exactly. In which case the interpolation-based quadrature will be exact.

But there's more. What if $f(x)$ is a polynomial of degree $n+1$: $f(x) = a_{n+1}x^{n+1} + \ldots$? The interpolation error is

$$f(x) - p(x) = a_{n+1}(x - x_0)\cdots(x - x_n).$$

Assume that the points x_i are distributed symmetrically around the midpoint of the interval $(x_n + x_0)/2$. Then

- If n is even then the interpolation error is an odd-degree polynomial that is antisymmetric about the midpoint of the interval and must therefore integrate to 0.
- If n is odd then the interpolation error doesn't necessarily integrate to 0.

[1] ad hoc just means that it's not a systematic approach that can be applied in general.

So: An interpolating quadrature with $n+1$ nodes that are symmetrically placed around the center of the interval will integrate polynomials up to degree n exactly when n is odd, and up to degree $n+1$ exactly when n is even.

6 Let $Q[f]$ denote the quadrature rule. Define the error as (note sign convention)

$$R[f] = Q[f] - I[f].$$

(S&B 3.2.3) Suppose $R[p] = 0$ for every polynomial of degree $\leq m$. Then for all functions $f \in C^{m+1}[a,b]$

$$R[f] = \int_a^b f^{(m+1)}(t)K(t)\mathrm{d}t$$

where the Peano kernel is

$$K(t) = \frac{1}{m!}R[(x-t)_+^m].$$

Before proving the theorem, we'll see what it means and how it's useful. Apply to Simpson's rule. First note that $I_n[f] = Q[f]$ for Simpson's rule is

$$Q[f] = \frac{h}{3}(f(-h) + 4f(0) + f(h))$$

Simpson's rule integrates polynomials of degree $\leq 3 = m = n+1$ exactly (because of the quirk noted above), so the kernel is

$$K(t) = \frac{1}{3!}R[(x-t)_+^3] = \frac{1}{6}\left[Q[(x-t)_+^3] - \int_{-h}^{h}(x-t)_+^3\mathrm{d}x\right] = \frac{h}{18}((-h-t)_+^3 + 4(-t))_+^3 + (h-t)_+^3) - \frac{1}{6}\int_t^h (x-t)^3\mathrm{d}x.$$

Break it down:

- $(-h-t)_+^3 = 0$ for every $t \in [-h,h]$.
- $(-t)_+^3 = 0$ for $t > 0$ and $= -t^3$ for $t < 0$.
- $(h-t)_+^3 = (h-t)^3$ for every $t \in [-h,h]$.
- $\int_t^h (x-t)^3\mathrm{d}x = (h-t)^4/4$.

This implies

$$K(t) = \begin{cases} \frac{1}{72}(h-t)^3(h+3t) & \text{for } t \geq 0 \\ K(-t) & \text{for } t \leq 0 \end{cases}$$

Plot: Compact support.

We can now bound the error by

$$|R[f]| \leq \|f^{(4)}\|_\infty \int_{-h}^{h} |K(t)|\mathrm{d}t = \|f^{(4)}\|_\infty \frac{h^5}{90}.$$

Alternatively, since $K(t)$ has a single sign on the interval you can use the integral mean value theorem to write

$$R[f] = f^{(4)}(\xi)\int_{-h}^{h} K(t)\mathrm{d}t = f^{(4)}(\xi)\frac{h^5}{90}$$

for some ξ in the interval. If you form a composite Simpson's rule and add the error bounds you find that the method is 4th order.

Notice that midpoint was $n = 0$ and second order; trapezoid was $n = 1$ and second order; Simpson's is $n = 2$ and fourth order. This behavior turns out to be generic: a formula based on interpolation that uses

equispaced nodes and interpolating polynomials of degree $\leq n$ gives even-order global accuracy, either $n+2$ if n is even or $n+1$ if n is odd.

Formulas based on equispaced interpolation are called Newton-Cotes formulas. We already don't expect them to converge as $n \to \infty$ because of Runge phenomenon, but they can be useful in a piecewise interpolation context.

7* Proof of S&B 3.2.3. Note that the interpolation-based quadrature rule is linear. It is just the composition of 2 linear functions. First take the function and construct the approximating polynomial; this process is linear since the polynomial is constructed by solving a linear system where the RHS is just the function data. Next you exactly integrate the interpolating polynomial, and integration is linear.

Because the quadrature is linear, the error is linear too. Let $Q[f]$ denote the quadrature. The error is (note sign convention)

$$R[f] = Q[f] - I[f]$$

is a linear operator because it's a sum of linear operators.

Now consider the Taylor expansion with Peano remainder

$$f(x) = f(a) + f'(a)(x-a) + \ldots + \frac{f^{(m)}(a)}{m!}(x-a)^m + \frac{1}{m!}\int_a^x f^{(m+1)}(t)(x-t)^m \mathrm{d}t$$

This is something you should have seen elsewhere. You can look it up; I will not prove the Peano form of the Taylor remainder. But I will note that

$$\frac{1}{m!}\int_a^x f^{(m+1)}(t)(x-t)^m \mathrm{d}t = \frac{1}{m!}\int_a^b f^{(m+1)}(t)(x-t)_+^m \mathrm{d}t.$$

Apply R to the Taylor expansion of $f(x)$ and use linearity and the fact that the remainder is 0 for the polynomial part to get

$$R[f] = 0 + \frac{1}{m!} R\left[\int_a^b f^{(m+1)}(t)(x-t)_+^m \mathrm{d}t\right].$$

Now we need to be able to commute R with the integral, and then we'll have the desired result. S&B considers Hermite interpolation, which is why their discussion is more complicated. For our purpose we just need to show that

$$Q\left[\int_a^b f^{(m+1)}(t)(x-t)_+^m \mathrm{d}t\right] = \int_a^b f^{(m+1)}(t) Q[(x-t)_+^m] \mathrm{d}t$$

and

$$I\left[\int_a^b f^{(m+1)}(t)(x-t)_+^m \mathrm{d}t\right] = \int_a^b f^{(m+1)}(t) I[(x-t)_+^m] \mathrm{d}t.$$

All that $Q[\,]$ does is evaluate at particular values of x, multiply by constants, and add. This clearly commutes with the integration.

For the second part we just need to switch the order of integration

$$\int_a^b \int_a^b f^{(m+1)}(t)(x-t)_+^m \mathrm{d}t\mathrm{d}x = \int_a^b \int_a^b f^{(m+1)}(t)(x-t)_+^m \mathrm{d}x\mathrm{d}t$$

which is clearly valid because of continuity of the integrand over the 2D rectangle (Fubini). This proves that

$$R[f] = \frac{1}{m!}\int_a^b f^{(m+1)}(t) R[(x-t)_+^m] \mathrm{d}t.$$

8 We just saw that you can derive error formulas without making any reference to interpolation; instead you just require the quadrature formula to integrate polynomials exactly. So let's derive a 'new' kind of quadrature based on integrating polynomials exactly.

The integral of a polynomial of degree $\leq n$ is

$$\int_a^b a_0 + a_1 x + \ldots + a_n x^n \mathrm{d}x = a_0(b-a) + a_1 \frac{b^2 - a^2}{2} + \ldots + a_n \frac{b^{n+1} - a^{n+1}}{n+1}.$$

Suppose we have $n+1$ points and $n+1$ weights and apply our quadrature to the same polynomial

$$I_n[a_0 + a_1 x + \ldots + a_n x^n] = w_0(a_0 + a_1 x_0 + \ldots + a_n x_0^n) + \ldots + w_n(a_0 + a_1 x_n + \ldots + a_n x_n^n).$$

We want these expressions to be zero for any polynomial, so we need

$$\begin{aligned}
w_0 + w_1 + \ldots + w_n &= b - a \\
w_0 x_0 + w_1 x_1 + \ldots + w_n x_n &= \frac{b^2 - a^2}{2} \\
w_0 x_0^2 + w_1 x_1^2 + \ldots + w_n x_n^2 &= \frac{b^3 - a^3}{3} \\
&\vdots \\
w_0 x_0^n + w_1 x_1^n + \ldots + w_n x_n^n &= \frac{b^{n+1} - a^{n+1}}{n+1}
\end{aligned}$$

The coefficient matrix is the transpose of a Vandermonde matrix (which is sometimes called a Vandermonde matrix):

$$\mathbf{V}^T \boldsymbol{w} = \boldsymbol{b}, \quad \boldsymbol{b}_i = \int_a^b x^i \mathrm{d}x$$

so the solution is exists & is unique provided that the nodes are distinct. Overall, this approach gives a method for computing weights for general nodes and polynomial orders. Vandermonde matrices can be ill-conditioned, so in general don't use this for large systems (with some caveats; some node locations give OK conditioning).

Suppose $x_0 = a = -h$, $x_1 = 0$, and $x_2 = b = h$; then the solution for the weights is just the already-found Simpson's rule. In fact, this is not a 'new' kind of quadrature at all; the weights derived in this manner are exactly the same as the weights derived for interpolating quadrature. Recall that you can derive the weights from the Lagrange form of the interpolating polynomial:

$$\begin{aligned}
p(x) &= \sum_i f(x_i) \ell_i(x) \\
\int p(x) \mathrm{d}x &= \sum_i f(x_i) \int \ell_i(x) \mathrm{d}x \\
w_i &= \int \ell_i(x) \mathrm{d}x.
\end{aligned}$$

Also recall that the Lagrange polynomials are

$$\ell_i(x) = (\mathbf{V}^{-1})_{1i} + (\mathbf{V}^{-1})_{2i} x + \ldots + (\mathbf{V}^{-1})_{(n+1)i} x^n$$

where $\mathbf{V}$ is the Vandermonde matrix. So in the interpolating quadrature derivation of the weights we have

$$w_i = (\mathbf{V}^{-1})_{1i}(b-a) + (\mathbf{V}^{-1})_{2i} \frac{b^2 - a^2}{2} + \ldots + (\mathbf{V}^{-1})_{(n+1)i} \frac{b^{n+1} - a^{n+1}}{n+1}.$$

This is exactly the same as weights obtained via the solution of the transposed Vandermonde system derived using the condition that the quadrature exactly integrates polynomials (without reference to interpolation).

$$\boldsymbol{w} = \mathbf{V}^{-T} \boldsymbol{b}.$$

So we basically now have 2 ways to compute weights: integrate the Lagrange polynomials, or solve the transposed Vandermonde system.

Asymptotic Error Formulae & Extrapolation

1 Recall that the error for the (simple) trapezoid rule is

$$I[f] - I_1[f] = -\frac{h^3}{12} f''(\eta)$$

for some η in the interval. The error in the composite rule (with equal spacing, for simplicity) is thus

$$E_n[f] = -\frac{h^3}{12} \sum_{i=0}^{n} f''(\eta_i).$$

Notice the following curious fact

$$\lim_{n\to\infty} \frac{E_n[f]}{h^2} = -\lim_{n\to\infty} \frac{1}{12} \sum_{i=0}^{n} f''(\eta_i)h = -\frac{1}{12}\int_a^b f''(x)\mathrm{d}x = \frac{f'(a) - f'(b)}{12}.$$

(Riemann sum limits to an integral.) This gives us an *asymptotic* error formula

$$E_n[f] \sim \frac{h^2}{12}[f'(a) - f'(b)] \text{ as } n \to \infty.$$

If $f'(a) = f'(b) = 0$ or if $f'(a) = f'(b)$ the convergence is faster than h^2.

One way to use this is to define a 'corrected' (composite) trapezoid rule

$$\text{Corrected Trap Rule} = \text{Trap Rule} + \frac{h^2}{12}[f'(a) - f'(b)].$$

Naturally you can only do this if you know $f'(a)$ and $f'(b)$. This rule should be more accurate than the standard Trap Rule for most functions as long as n is large. Actually, as we'll see later, the Corrected Trap Rule is fourth order (globally), and you can keep correcting it.

2* The asymptotic error formula obtained above for the composite Trap rule has the form

$$I[f] - I_n[f] \sim ch^2$$

for a constant c independent of n. We will eventually derive formulas of the form

$$I[f] - I_n[f] \sim c_2h^2 + c_4h^4 + \ldots + c_{2n}h^{2n}.$$

It takes some effort though, so we'll start with background (S&B 3.3).

Consider that

$$\int_0^1 f(t)\mathrm{d}t = \int_0^1 \left(\frac{\mathrm{d}}{\mathrm{d}t}(t+c)\right) f(t)\mathrm{d}t = [(t+c)f(t)]_0^1 - \int_0^1 (t+c)f'(t)\mathrm{d}t.$$

We want to relate this to the Trapezoid Rule, which has the form

$$\int_0^1 f(t)\mathrm{d}t \approx \frac{1}{2}[f(0) + f(1)].$$

If we pick $c = -1/2$ in the first formula we get

$$\int_0^1 f(t)\mathrm{d}t = \frac{1}{2}[f(0) + f(1)] - \int_0^1 (t - \frac{1}{2})f'(t)\mathrm{d}t.$$

This gives us an error formula for the simple Trap Rule, though not a very useful one. Let's keep going.

$$\int_0^1 (t-\frac{1}{2})f'(t)\mathrm{d}t = \frac{1}{2}\left[\left(t^2-t+c\right)f'(t)\right]_0^1 - \frac{1}{2}\int_0^1 \left(t^2-t+c\right)f''(t)\mathrm{d}t.$$

If we choose c appropriately we'll get the asymptotic error formula for the Trapezoid Rule; viz. $c = 1/6$ implies

$$\int_0^1 f(t)\mathrm{d}t = \frac{1}{2}[f(0)+f(1)] - \frac{1}{12}\left[f'(1)-f'(0)\right] + \frac{1}{2}\int_0^1 \left(t^2-t+\frac{1}{6}\right)f''(t)\mathrm{d}t.$$

Here we recognize the asymptotic error formula for the Trap Rule with $h = 1$, and as a bonus we get an exact remainder term.

Let's keep going.

$$\int_0^1 \left(t^2-t+\frac{1}{6}\right)f''(t)\mathrm{d}t = \frac{1}{3}\left[\left(t^3-\frac{3}{2}t^2+\frac{1}{2}t+c\right)f''(t)\right]_0^1 - \frac{1}{3}\int_0^1 \left(t^3-\frac{3}{2}t^2+\frac{1}{2}t+c\right)f^{(3)}(t)\mathrm{d}t.$$

How should we choose c? It turns out to be very convenient to choose it so that the polynomial is 0 at both endpoints, i.e. $c = 0$, so

$$\int_0^1 \left(t^2-t+\frac{1}{6}\right)f''(t)\mathrm{d}t = -\frac{1}{3}\int_0^1 \left(t^3-\frac{3}{2}t^2+\frac{1}{2}t\right)f^{(3)}(t)\mathrm{d}t.$$

$$\int_0^1 \left(t^3-\frac{3}{2}t^2+\frac{1}{2}t\right)f^{(3)}(t)\mathrm{d}t = \frac{1}{4}\left[\left(t^4-2t^3+t+c\right)f^{(3)}(t)\right]_0^1 - \frac{1}{4}\int_0^1 \left(t^4-2t^3+t+c\right)f^{(4)}(t)\mathrm{d}t$$

Note that for any $c > 0$ the polynomial in the remainder term is positive on $[0,1]$, so we can use the integral mean value theorem to set

$$\int_0^1 \left(t^4-2t^3+t+c\right)f^{(4)}(t)\mathrm{d}t = f^{(4)}(\xi)\int_0^1 \left(t^4-2t^3+t+c\right)\mathrm{d}t = (\frac{1}{5}+c)f^{(4)}(\xi).$$

We should choose $c = 0$ which gives the best bound. Coincidentally this also sets the boundary terms to 0. If we put everything back together, we get an even better asymptotic estimate for the Trapezoid rule error; it would be order h^5 if we had used $[0,h]$ instead of $[0,1]$.

3 Now let's generalize the previous approach. We have

$$\int_0^1 f(t)\mathrm{d}t = B_1(t)f(t)|_0^1 - \int_0^1 B_1(t)f'(t)\mathrm{d}t$$

$$\int_0^1 B_1(t)f'(t)\mathrm{d}t = \frac{1}{2}B_2(t)f'(t)|_0^1 - \frac{1}{2}\int_0^1 B_2(t)f''(t)\mathrm{d}t$$

$$\vdots$$

$$\int_0^1 B_{k-1}(t)f^{(k-1)}(t)\mathrm{d}t = \frac{1}{k}B_k(t)f^{(k-1)}(t)|_0^1 - \frac{1}{k}\int_0^1 B_k(t)f^{(k)}(t)\mathrm{d}t.$$

The $B_k(t)$ are polynomials that satisfy $B'_{k+1}(t) = (k+1)B_k(t)$ and $B_1 = t - 1/2$. There are a bunch of unknown constants of integration which are completely arbitrary as far as the above sequence of integrations-by-parts is concerned. We will make some very special choices that lead to convenient formulas. Specifically, we want

$$B_{2k+1}(0) = B_{2k+1}(1) = 0 \text{ for } k > 1.$$

This will guarantee that the boundary terms at odd orders will be 0, leaving

$$\int_0^1 f(t)\mathrm{d}t = \frac{1}{2}[f(1)+f(0)] + \text{ even-order boundary terms } + \text{ the last integral }.$$

It also specifies the $B_k(t)$ uniquely. If you have $B_k(t)$ where k is odd, then first you integrate to get $B_{k+1}(t)$ with a free constant, then you integrate again to get B_{k+2} with two free constants. You then set those 2 constants by the conditions above, which gives you exact expressions for $B_{k+1}(t)$ and $B_{k+2}(t)$. These polynomials are the 'Bernoulli polynomials,' and the numbers $B_k = B_k(0)$ are the 'Bernoulli numbers.'

Collecting all the notation together we find

$$\int_0^1 f(t)\mathrm{d}t = \frac{1}{2}[f(0)+f(1)]+\sum_{l=1}^{m}\frac{B_{2l}}{(2l)!}[f^{(2l-1)}(0)-f^{(2l-1)}(1)]+\frac{1}{(2m+2)!}\int_0^1 (B_{2m+2}(t)-B_{2m+2})f^{(2m+2)}(t)\mathrm{d}t.$$

Nota bene This is a version of the Euler-MacLaurin formula. If you're taking the prelim, you would do well to memorize the formula (5.4.9) from Atkinson that corresponds to the Euler-MacLaurin formula for the composite rule and for an integral over $[a, b]$ rather than $[0, 1]$.

It just so happens (as a result of the specific choice of constants of integration; proof in S&B 3.3) that the kernel $B_{2m+2}(t) - B_{2m+2}$ is sign-definite on $[0, 1]$. So, by the integral mean value theorem

$$\int_0^1 f(t)\mathrm{d}t = \frac{1}{2}[f(0) + f(1)] + \sum_{l=1}^{m}\frac{B_{2l}}{(2l)!}[f^{(2l-1)}(0) - f^{(2l-1)}(1)] + \frac{f^{(2m+2)}(\xi)}{(2m+2)!}\int_0^1 (B_{2m+2}(t) - B_{2m+2})\mathrm{d}t.$$

When you rescale the interval from $[0, 1]$ to $[0, h]$, you get an asymptotic formula for the simple (not composite) trapezoid rule error

$$\int_0^h f(t)\mathrm{d}t - \frac{h}{2}[f(0) + f(h)] = c_2h^2 + c_4h^4 + \ldots + c_{2m}h^{2m} + c_{2m+2}(h)h^{2m+2}.$$

The $c_{2m+2}(h)$ is related to the term $f^{(2m+2)}(\xi)$ and the h^{2m+2} comes from the integral of the kernel, which we didn't prove. In the last term $c_{2m+2}(h)$ is bounded as $h \to 0$.

4. Comments:

- Briefly note that if you form a composite rule and add all the asymptotic error estimates, the terms correspoding to interior nodes will cancel except in the last term, leaving an asymptotic error formula of the form (see comment above; 5.4.9 in Atkinson)

$$\int_a^b f(t)\mathrm{d}t = I_n[f] + C_2h^2 + C_4h^4 + \ldots + C_{2m}h^{2m} + C_{2m+2}(h)h^{2m+1}.$$

  (The coefficients c_{2p} in the formula for the simple rule are different from the coefficients C_{2p} in the composite rule, but both are independent of h.)

- If $f \in C^\infty[a, b]$ and all its derivatives go to zero at the endpoints, then the trapezoid rule will converge faster than any power of h.

- If f is periodic and C^∞ then the trapezoid rule will converge faster than any power of h.

5 The composite trapezoid rule has an asymptotic error formula of the form

$$\text{error} = C_2h^2 + C_4h^4 + \ldots + C_{2m}h^{2m} + C_{2m+2}(h)h^{2m+1}$$

as long as the integrand has a sufficient number of continuous derivatives. Lots of other quadrature rules have asymptotic error formulas that look like this, **but not all**. For example, applying the trapezoid rule to infinitely-smooth periodic functions implies that all the constants c_p are zero. Gaussian Quadrature does not have this kind of asymptotic error formula.

But suppose we have such a formula. We can use it to 'extrapolate' the quadrature. The simplest extrapolation (Aitken's) requires

$$E_n[f] \sim \frac{c}{n^p}$$

(e.g. Trap rule has $p = 2$, Simpson's rule has a higher-order asymptotic error formula, and the corrected trap rule has $p = 4$.) You might not know whether there is an asymptotic error formula, but you can check empirically. If there is an asymptotic error formula then

$$R_{4n} := \frac{I_{2n} - I_n}{I_{4n} - I_{2n}} = \frac{I - I_n - (I - I_{2n})}{I - I_{2n} - (I - I_{4n})} \sim \frac{c/n^p - c/(2n)^p}{c/(2n)^p - c/(4n)^p} = 2^p.$$

So we can estimate $p \approx \log_2(R_{4n})$. If we compute $\log_2(R_{4n})$ for several different n and find the same (or similar) result, then we can assume that there's an asymptotic error formula; otherwise not.

Now, assuming we have an asymptotic error behavior, consider

$$\frac{I - I_n}{I - I_{2n}} \sim 2^p \sim \frac{I - I_{2n}}{I - I_{4n}}$$

Setting left equal to right and solving for I yields

$$I \approx A_{4n} = I_{4n} - \frac{(I_{4n} - I_{2n})^2}{I_{4n} - 2I_{2n} + I_n}.$$

Notice that you're correcting the most-accurate estimate I_{4n}. This is Aitken's extrapolation; it's similar to Aitken's extrapolation for fixed points.

The above derivation doesn't show what the error of the extrapolated quadrature method is. Let's now assume that the original formula had an asymptotic error formula of the form

$$E_n[f] \sim c_0 n^{-p} + c_1 n^{-q}$$

with $q > p$ (this is still consistent with the original formula, BTW). Plug in $I_n = I - E_n$ to the expression $I - A_{4n}$ and simplify. You'll find that the new quadrature has asymptotic error formula

$$I - A_{4n} \sim \frac{c_1 4^{-q} \left(2^p - 2^q\right)^2}{\left(2^p - 1\right)^2} n^{-q} \text{ as } n \to \infty.$$

As expected, we've corrected the leading-order term in the asymptotic error expansion, so the next one now becomes the leading-order term.

6 Richardson Extrapolation. We are dealing with Trap Rule, so we have the following

$$I[f] = I_n[f] + C_2 h^2 + C_4 h^4 + \ldots + C_{2m+1}(h) h^{2m+1}$$
$$I[f] = I_{2n}[f] + C_2 \frac{h^2}{4} + C_4 \frac{h^4}{2^4} + \ldots + C_{2m+1}(h/2) \frac{h^{2m+1}}{2^{2m+1}}$$

where h is the spacing for the rule with $n+1$ points ($I_n[f]$). Notice that we can eliminate the leading-order error term by multiplying the bottom line by 4, then subtracting from the top:

$$3I[f] = I_n[f] - 4I_{2n}[f] + \frac{3}{4} C_4 h^4 + \ldots + \text{error}.$$

We can write this as

$$I[f] = I_{2n}^{(1)}[f] + d_4 h^4 + d_6 h^6 + \ldots + \text{error, where } I_{2n}^{(1)}[f] := \frac{I_n[f] - 4I_{2n}[f]}{3}.$$

This is called a 'Richardson extrapolation.' It's interesting to note that this is actually just Simpson's rule. We can see immediately that it's fourth order.

We can do the same thing again:

$$I[f] = I_{2n}^{(1)}[f] + d_4 h^4 + d_6 h^6 + \ldots + \text{error}$$
$$I[f] = I_{4n}^{(1)}[f] + d_4 \frac{h^4}{2^4} + d_6 \frac{h^6}{2^6} + \ldots + \text{different error}$$

If we multiply the bottom line by 2^4 and subtract from the top line we have

$$I[f] = I_{4n}^{(2)}[f] + e_6 h^6 + e_8 h^8 + \dots \text{ where } I_{4n}^{(2)}[f] = \frac{I_{2n}^{(1)}[f] - 2^4 I_{4n}^{(1)}[f]}{1 - 2^4}.$$

Actually this is just Boole's rule, and we can see immediately that it's sixth-order.

Clearly the process can continue, and in general it does not produce more Newton-Cotes rules. We can write the following formula

$$I_n^{(k)}[f] = \frac{4^k I_n^{(k-1)}[f] - I_{n/2}^{(k-1)}[f]}{4^k - 1}$$

where n is even. Assuming the function is smooth enough, the error has asymtotic order h^{2k+2}.

7 Romberg integration.

$$\begin{array}{ccccccc}
I_1^{(0)} & & & & & & \\
I_2^{(0)} & I_2^{(1)} & & & & & \\
I_4^{(0)} & I_4^{(1)} & I_4^{(2)} & & & & \\
I_8^{(0)} & I_8^{(1)} & I_8^{(2)} & I_8^{(3)} & & & \\
I_{16}^{(0)} & I_{16}^{(1)} & I_{16}^{(2)} & I_{16}^{(3)} & I_{16}^{(4)} & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots &
\end{array}$$

The number of points $n+1$ increases downwards, the order of Richardson extrapolation increases rightwards. Romberg integration uses the diagonal of the above table.

$$J_k[f] = I_{2^k}^{(k)}.$$

Since this is just a Richardson extrapolation, the error is known and has asymptotic order h^{2k+2}. Previously we considered k to be fixed while we decrease h, but here h is decreasing as k is increasing $h = (b-a)/(2^k)$. To put things on an even footing, note that $k = \log_2(b-a) - \log_2(h)$, so the error is asymptotically of the form

$$h^{2\log_2((b-a)/h)+2}$$

as $h \to 0$. The error is thus going to 0 faster than any power of h.

8 Summary. The Euler-MacLaurin (and similar) asymptotic error formula leads to

- Improved-rate quadratures like corrected-Trapezoid.
- Improved-rate quadratures like Aitken extrapolation and/or Richardson extrapolation.
- Quadratures that converge faster-than-algebraically for functions that are infinitely smooth: (i) Romberg, and (ii) Trapezoid for periodic functions.

Gaussian Quadrature & Chebyshev Methods

1 We've looked at choosing the weights w_i so that we achieve the highest possible order of integration. We also saw that it's sometimes possible to choose the *node* locations to achieve order higher than n, e.g. midpoint rule has $n = 0$ but integrates polynomials up to degree 1 exactly; similarly for Simpson's rule $n = 2$ but it integrates polynomials of degree 3 exactly. Recall that the Newton-Cotes (equispaced) rules always have even global order of accuracy: midpoint is second order, trapezoid is second order, Simpson's is fourth order, the next one is fourth order, etc.

The general idea now is to choose both nodes $x_i$and weights w_i to achieve as high-order accuracy as possible. We now have $2(n+1)$ degree of freedom, so we 'should' be able to integrate polynomials of degree $\leq 2n+1$ exactly. You can enforce this by requiring $\sum_i w_i (x_i)^k = \int_a^b x^k \mathrm{d}x$ for all $k \leq 2n+1$, but this is a

nonlinear system! Generally we don't attack the nonlinear system directly; there are better ways that we will discuss.

But first, let's briefly expand our range of integrals to something like this

$$\int_a^b f(x)w(x)\mathrm{d}x$$

where $w(x)$ is a weight function with the usual properties (non-negative and $\int_a^b x^k w(x)\mathrm{d}x < \infty$ for all $k \in \mathbb{Z}^+$). One reason for looking at this is to allow singularities; e.g. if you want to compute

$$\int_{-1}^1 \frac{e^x}{\sqrt{1-x^2}}\mathrm{d}x$$

then technically you can't use any of the rules we've studied so far because we always assume that f is continuous. Instead you can just let $f(x) = e^x$ and $w(x) = (1-x^2)^{-1/2}$. We'll return to this idea later. **Beware: We didn't derive a Peano kernel error formula for weighted integrals.** In the meantime,

> (D&B Theorem 5.1.3) Let $\sum_{i=0}^n w_i f(x_i)$ be a quadrature that integrates polynomials of degree $\leq n$ exactly, and define $\psi(x) = (x-x_0)\cdots(x-x_n)$. Then the interpolatory quadrature rule integrates all polynomials of degree $\leq n+1+k$ if and only if
>
> $$\int_a^b w(x)p(x)\psi(x)\mathrm{d}x = 0$$
>
> for all polynomials p of degree $\leq k$.

Proof:

- First prove that if the rule integrates polynomials with degree $\leq n+1+k$ exactly, then the integral above is zero. Note that $\psi(x)p(x)$ is a polynomial of degree $\leq n+1+k$, so it will be integrated exactly. Then plug in the quadrature formula

  $$\int_a^b \psi(x)p(x)w(x)\mathrm{d}x = \sum_{i=0}^n w_i\psi(x_i)p(x_i) = 0$$

  last equality is because $\psi(x_i) = 0$.

- Now prove that the integral implies exact quadrature. Let

  $$p(x) = q(x)\psi(x) + r(x)$$

  where $q(x)$ has degree $\leq k$ and $r(x)$ has degree $\leq n$.

  $$\int_a^b p(x)w(x)\mathrm{d}x = \int_a^b q(x)\psi(x)w(x)\mathrm{d}x + \int_a^b r(x)w(x)\mathrm{d}x.$$

  By assumption, the first integral on the RHS is 0. Also by assumption, the second integral can be computed exactly using the quadrature

  $$\int_a^b p(x)w(x)\mathrm{d}x = \int_a^b r(x)w(x)\mathrm{d}x = \sum_i w_i r(x_i).$$

  Notice that $p(x_i) = r(x_i)$, so

  $$\int_a^b p(x)w(x)\mathrm{d}x = \sum_i w_i r(x_i) = \sum_i w_i p(x_i).$$

QED.

The idea is to pick the nodes x_i so that the polynomial $\psi(x)$ is orthogonal to all polynomials of degree $\leq k$. E.g. let $w(x) = 1$, $[a, b] = [-1, 1]$. Suppose we have 3 nodes ($n = 2$) and want to be able to integrate quintics ($k = 2$) exactly. We need to choose the nodes so that $\psi(x)$ is a cubic that is orthogonal to all quadratics. So $\psi(x) \propto$ the third-order Legendre polynomial, which we already know has exactly 3 simple roots in $[-1, 1]$. Once we know the roots/nodes, we can compute the weights using the standard algorithms.

Note that $\psi(x)$ has degree $n + 1$, so by choosing $\psi(x)$ proportional to the $n + 1$-st orthogonal polynomial, it will be orthogonal to all polynomials of degree $\leq n$, and will have exactly $n + 1$ simple roots in the interval $[a, b]$. This shows that you can choose the nodes first and then find the weights so that the quadrature rule integrates polynomials of order $2n + 1$ exactly, just as expected. This is called Gaussian quadrature. Sometimes for $w(x) = 1$ it is called Gauss-Legendre; for other weight functions it is sometimes called Gauss-Christoffel quadrature.

For small to moderate n you can just look the nodes and weights up. For larger n there are purpose-built algorithms that will accurately and rapidly compute them for you.

2 Now that we've constructed Gaussian quadrature, let's analyze the error. The most basic convergence result (without a *rate* of convergence) is based on the fact that the weights are positive. We already know that the weights are the integrals of the Lagrange polynomials

$$w_i = \int_a^b w(x)\ell_i(x)\mathrm{d}x.$$

Notice that ℓ_i is a polynomial of degree n, so ℓ_i^2 is a polynomial of degree $2n$, and will be integrated exactly. So

$$0 < \int_a^b \ell_j(x)^2 w(x)\mathrm{d}x = \sum_i w_i \ell_j(x_i)^2 = w_j.$$

Now that we know Gaussian quadrature weights are positive, we need to know why that matters.

The quadratures derived so far are linear operators. Suppose you have a method $I_n[f]$ that gives you an exact answer for all polynomials up to degree n, i.e.

$$I_n[p] = I[p] \text{ for any polynomial } p \text{ of degree } \leq n.$$

Well, we know that *there is* a polynomial of degree $\leq n$ that optimally approximates f in the L^∞ norm. Denote this by p_*. Then

$$I[f] - I_n[f] = I[f] - I[p_*] + I_n[p_*] - I_n[f]$$

i.e. we can relate the error in integrating f to the error in approximating f by p_*. Now one part is easy

$$|I[f] - I[p_*]| = |I[f - p_*]| \leq (b - a)\rho_n(f).$$

The Weierstrass approximation theorem says that this will go to 0 as $n \to \infty$ for any continuous f. What about the other piece? All our intepolation-based quadratures are linear operators. If they're bounded then

$$|I_n[f] - I_n[p_*]| = |I_n[f - p_*]| \leq \|I_n\|\rho_n(f).$$

We now derive a bound on the operator. (There is a simpler way to do this that doesn't require you to think about operator norms.)

$$\|I_n\|_\infty = \max_{\|f\|_\infty = 1} |I_n[f]| = \max |\sum_i w_i f(x_i)| \leq \max_i |f(x_i)| \sum_i |w_i| \leq \sum_i |w_i|.$$

If we can now show that there is an f with $\|f\|_\infty = 1$ that achieves this upper bound then we'll have shown that this is actually the norm of the operator. Let integrable f be any function s.t. $f(x_i) = \text{sign} w_i$, then

$$|I_n[f]| = \sum_i |w_i|.$$

Now the fact that the weights have to integrate the polynomial 1 exactly imply that $\sum_i w_i = b - a$ for any n. The weights clearly depend on n and on the location of the nodes. Consider what happens as you increase n. If you choose the nodes in such a way that the weights remain positive, then $\|I_n\|_\infty = (b-a)$ for any n, and by the arguments above the quadrature will converge to the true integral. Otherwise it's possible that the weights can grow causing $\|I_n\|_\infty$ to grow and possibly causing the quadrature to diverge. This is another explanation of why the equispaced Newton-Cotes formulas are not reliable: the weights are not positive.

This leads to a very nice conclusion for Gaussian quadrature: it will converge as $n \to \infty$ for any $f \in C([a,b])$.

3 The above analysis tells us that GQ converges for lots of functions, but it doesn't tell us anything about the size of the error for fixed n. The Peano kernel gives one way of deriving an error formula for Gaussian quadrature (Gauss-Legendre, anways; we didn't derive the Peano form for weighted integrals). The following is another way. The idea is to show that Hermite quadrature at the Gauss-Christoffel nodes is the same as Gaussian quadrature, and then use the Hermite error formula.

Consider a global Hermite interpolating polynomial

$$p(x) = \sum_{i=0}^{n} f(x_i) h_i(x) + \sum_{i=0}^{n} f'(x_i) \tilde{h}_i(x).$$

The quadrature rule is

$$\int_a^b w(x) f(x) \mathrm{d}x \approx \sum_i f(x_i) w_i + \sum_i f(x_i) \tilde{w}_i$$

where

$$w_i = \int_a^b w(x) h_i(x) \mathrm{d}x, \quad \tilde{w}_i = \int_a^b w(x) \tilde{h}_i(x) \mathrm{d}x.$$

We need to show that this is the same as Gaussian quadrature when the nodes are the roots of the $n+1$-st degree orthogonal polynomial. We do this by showing that $\tilde{w}_i = 0$ and $w_i = \int w(x)(\ell_i(x))^2 \mathrm{d}x$, which are the same as the Gaussian quadrature weights.

- First $\tilde{w}_i = 0$.

  $$\tilde{h}_i(x) = (x - x_i)(\ell_i(x))^2 = \frac{[(x-x_0)\cdots(x-x_n)][(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_i)]}{(x_i-x_0)^2\cdots(x-x_{i-1})^2(x-x_{i+1})^2\cdots(x-x_i)^2]}$$

  $\tilde{h}_i$ is $\psi(x)$ times a polynomial of degree $\leq n-1$. By definition

  $$\tilde{w}_i = \int_a^b w(x) \tilde{h}_i(x) \mathrm{d}x = \int_a^b w(x) \psi(x) \text{poly of lower degree } \mathrm{d}x = 0.$$

  The last equality is *only* true because we've chosen the nodes to be the roots of the $n+1$-st degree orthogonal polynomial: the $\tilde{w}_i$ in a Hermite quadrature rule are not zero for general node locations, only for these special ones.

- Now show that the weights are $w_i = \int w(x)(\ell_i(x))^2 \mathrm{d}x$, which is the same as the GQ weights.

  $$h_i(x) = [1 - 2\ell_i'(x_i)(x - x_i)](\ell_i(x))^2$$

  $$w_i = \int_a^b w(x) h_i(x) \mathrm{d}x = \int_a^b w(x)(\ell_i(x))^2 \mathrm{d}x - 2\ell_i'(x_i) \int_a^b w(x)(x - x_i)(\ell_i(x))^2 \mathrm{d}x$$

  The last integral is just $\tilde{w}_i$, so it's zero by the argument above.

We've shown that Hermite quadrature at the GQ nodes is the same as Gaussian quadrature, so we can use the Hermite error formula to derive an error formula for GQ. The Hermite interpolation error is (assuming sufficient smoothness)

$$f(x) - p(x) = \frac{(x-x_0)^2 \cdots (x-x_n)^2}{(2n+2)!} f^{(2n+2)}(\xi(x)).$$

The quadrature error can be analyzed by integrating the interpolation error.

$$\int f(x)\mathrm{d}x - \sum_i w_i f(x_i) = \int \frac{(x-x_0)^2 \cdots (x-x_n)^2}{(2n+2)!} f^{(2n+2)}(\xi(x))\mathrm{d}x = f^{(2n+2)}(\xi) \int \frac{(x-x_0)^2 \cdots (x-x_n)^2}{(2n+2)!}\mathrm{d}x$$

where the second equality is a mean value theorem. In practice this isn't terribly useful, so we quote a more-useful but harder-to-prove theorem from ATAP Chapter 19 (Theorem 19.4):

> If f and its derivatives through $p-1$ are absolutely continuous on $[a,b]$ and $f^{(p)}$ is of bounded variation (with total variation V) then the $n+1$ point GQ applied to f satisfies
>
> $$|I[f] - I_n[f]| \leq \frac{32}{15} \frac{V}{\pi p (n-2p-1)^{2p+1}}$$
>
> for $n > 2p+1$. (Integral is assumed to be over $[-1,1]$.)

This implies that GQ converges quickly as $n \to \infty$ even when f doesn't have an infinite number of derivatives (i.e. when we can't use the interpolation error formula). You can also prove that convergence is faster than algebraic for certain analytic functions f (ATAP Theorem 19.3).

4 Sometimes you want to pre-specify a few node locations, and then let the rest be free. E.g. Gauss-Lobatto includes both endpoints and Gauss-Radau includes the left endpoint. Gauss-Kronrod nests the nodes so that when you increase n you get to re-use old function values (cf Gauss-Legendre, where the all the nodes & weights change when you increase n). Let $x_0, \ldots, x_p$ be the fixed nodes and $z_0, \ldots, z_q$ be the free nodes, with weights w_i and v_i, respectively. The quadrature rule will have the form

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=0}^{p} w_i f(x_i) + \sum_{i=0}^{q} v_i f(z_i).$$

(You can put a weight function in if you want.) We have $p+1+2(q+1)$ degrees of freedom (nodes & weights), so we could hope to integrate all polynomials of degree $\leq n = p+2q+2$ exactly.

Now let $\phi_0(x), \ldots, \phi_n(x)$ be basis functions for the space of polynomials with degree $\leq n$ (e.g. monomials). The nonlinear system for weights and nodes is

$$\sum_{i=0}^{p} w_i \phi_k(x_i) + \sum_{i=0}^{q} v_i \phi_k(z_i) = \int_a^b \phi_k(x)\mathrm{d}x, \quad k = 0, \ldots, n.$$

where the RHS is known. As with GQ this is not the best way to solve the problem.

There is a fairly straighforward algorithm in D&B to solve the general problem as posed here. Instead of going over it (not prelim material) we will discuss the special case of Gauss-Lobatto quadrature with $w(x) = 1$, i.e. we will set $x_0 = -1$ and $x_1 = 1$ and try to choose nodes $x_1, \ldots, x_{n-1}$ and weights $w_0, \ldots, w_n$ to achieve the highest polynomial exactness possible. The method is based on the theorem we proved earlier:

> (D&B Theorem 5.1.3) Let $\sum_{i=0}^{n} w_i f(x_i)$ be a quadrature that integrates polynomials of degree $\leq n$ exactly, and define $\psi(x) = (x-x_0)\cdots(x-x_n)$. Then the interpolatory quadrature rule integrates all polynomials of degree $\leq n+1+k$ if and only if
>
> $$\int_a^b w(x)p(x)\psi(x)\mathrm{d}x = 0$$
>
> for all polynomials p of degree $\leq k$.

We developed GQ by making $\psi(x)$ proportional to an orthogonal polynomial, which makes it satisfy the condition of the theorem.

Now we want to use the weight function $w(x) = 1$, and we have pre-specified the nodes x_0 and x_n. So we want to choose the nodes $x_1, \ldots, x_{n-1}$ so that

$$\int_a^b (x - x_0)(x - x_1)p(x)(x - x_1)\cdots(x - x_{n-1})\mathrm{d}x = 0$$

for all polynomials p of degree $\leq k$ for k as large as possible. This is the subtle part: Notice that $\tilde{\psi}(x) = (x - x_1)\cdots(x - x_{n-1})$ is a polynomial of degree $n - 1$ and that $\tilde{w}(x) = (x - x_0)(x - x_n) = (x - a)(x - b)$ is the weight function for the $(1, 1)$ Jacobi polynomials. If we chose $\tilde{\psi}(x) = (x - x_1)\cdots(x - x_{n-1})$ proportional to the $n - 1$-degree $(1, 1)$ Jacobi polynomial then the integral

$$\int_a^b p(x)\psi(x)\mathrm{d}x = \int_a^b \tilde{w}(x)p(x)\tilde{\psi}(x)\mathrm{d}x = 0$$

for all polynomials p of degree $\leq n - 2$. The theorem then guarantees that the Gauss-Lobatto quadrature will be exact for polynomials up to degree $2n - 1$. The same kind of argument can be applied to Gauss-Radau quadrature, but with a $(1, 0)$ or $(0, 1)$ Jacobi polynomial (depending on which endpoint you use). **Summary:** Gauss-Legendre-Lobatto quadrature nodes are roots of a $(1, 1)$ Jacobi polynomial; it integrates polynomials of degree $\leq 2n - 1$ exactly.

5* Recall that if we interpolate at the roots of a Chebyshev polynomial, then the node polynomial is proportional to that Chebyshev polynomial, and we have the following bound on the interpolation error

$$\|f(x) - p(x)\|_\infty \leq \frac{\|f^{(n+1)}\|_\infty}{2^n(n + 1)!}.$$

In fact, we stated in the section on interpolation (without proof and without details) that the interpolation error goes to zero for a large class of functions when interpolating at the Chebyshev nodes.

If we use an interpolatory quadrature where the nodes are the roots of a Chebyshev polynomial, then the quadrature error is (as usual) just the integral of the interpolation error, which is bounded by

$$\left|\int_a^b f(x)\mathrm{d}x - I_n[f]\right| \leq (b - a)\frac{\|f^{(n+1)}\|_\infty}{2^n(n + 1)!}.$$

That's really small!

- Fejér's 'first' rule is exactly as above: quadrature nodes are the roots of a Chebyshev polynomial.
- Fejér's 'second' rule uses the critical points of a Chebyshev polynomial (i.e. the roots of a Chebyshev polynomial of the second kind) as nodes.
- The Clenshaw-Curtis quadrature uses the same nodes as Fejér's second rule plus the endpoints, i.e. the extrema of the Chebyshev polynomial on $[-1, 1]$.

In every case there are analytical expressions for the weights, all of which are positive. There are also efficient implementations based on the FFT. Clenshaw-Curtis and Fejér's second rule have the benefit that you can re-use points if you move from $n + 1$ to $2n + 1$ points.

The fact that the weights are positive implies that the quadrature converges for any continuous integrand. This is another example of the gap between quadrature error and interpolation error: interpolation at the Chebyshev points *is not* guaranteed to converge for any continuous f, but the quadrature *is* guaranteed to converge.

The *rate* of convergence depends on how many continuous derivatives the function has. The following theorem shows a surprising property of Clenshaw-Curtis quadrature:

(ATAP Theorem 19.5) If f and its derivatives through $p-1$ are absolutely continuous on $[a,b]$ and $f^{(p)}$ is of bounded variation (with total variation V) then the $n+1$ point Clenshaw-Curtis quadrature applied to f satisfies

$$|I[f] - I_n[f]| \leq \frac{32}{15} \frac{V}{\pi p (n-2p-1)^{2p+1}}$$

for $n > N$ where N is a threshold that depends on p but not f. (Integral is assumed to be over $[-1,1]$.)

As with the GQ result, this can't be based on interpolation error because it only assumes f has a finite number of derivatives, while the interpolation error assumes that f has $n+1$ derivatives. More surprisingly: Clenshaw-Curtis has the *same* bound on the rate of convergence as GQ despite the fact that it only integrates polynomials up to degree n exactly.

Miscellaneous

1 Singular integrals: either $f(x)$ (or a derivative) has a singularity on the boundary of $[a,b]$, or the interval is infinite.

- If the integrand is piecewise-smooth, just split the integral into subintervals where the integrand is smooth on each subinterval
- Change variables to an integral without a singularity
- Put the singularity in the weight function
- Change variables to a problem with a finite interval

Example: Change from an integrand that is only $C[0,1]$ to one that is $C^\infty[0,1]$.

$$\int_0^1 \sqrt{x} \mathrm{d}x = 2 \int_0^1 t^2 \mathrm{d}t$$

Example: Change from a singular integrand to a smooth one.

$$\int_0^1 \frac{e^x}{\sqrt{x}} \mathrm{d}x = 2 \int_0^1 e^{t^2} \mathrm{d}t$$

Example: Put the singularity in the weight function.

$$\int_0^1 \frac{e^x}{\sqrt{x}} \mathrm{d}x, \;\; w(x) = x^{-1/2}.$$

Let's find a Gauss-Christoffel quadrature with this weight function. We need to compute the orthogonal polynomials; just use Gram-Schmidt

$$\phi_0(x) = 1, \;\; \phi_1(x) = x - \frac{\langle 1, x \rangle}{\|1\|_w^2} 1 = x - \frac{1}{3}, \;\; \phi_2(x) = x^2 - \frac{\langle \phi_1, x^2 \rangle}{\|\phi_1\|_w^2} \phi_1(x) - \frac{\langle 1, x^2 \rangle}{\|1\|_w^2} 1 = x^2 - \frac{6}{7}\left(x - \frac{1}{3}\right) - \frac{1}{5}$$

$$\phi_3(x) = \ldots = x^3 - \frac{15}{11}\phi_2(x) - \frac{5}{7}\left(x - \frac{1}{3}\right) - \frac{1}{7}$$

The roots of ϕ_3 are available in closed form because it's cubic;

$$x_0 \approx 0.0569391, \;\; x_1 \approx 0.437198, \;\; x_2 \approx 0.869499$$

We can find the weights by integrating the Lagrange polynomials

$$w_0 = \int_0^1 w(x)\ell_0(x)\mathrm{d}x \approx 0.935828, \; w_1 = \int_0^1 w(x)\ell_1(x)\mathrm{d}x \approx 0.721523, \; w_2 = \int_0^1 w(x)\ell_2(x)\mathrm{d}x \approx 0.342649$$

Notice that the sum of the weights equals $\int_0^1 w(x)\mathrm{d}x = 2$. The answer is about 2.9 and the error is about 1.6×10^{-6}. That's pretty good for a 3-point quadrature, but then again we're essentially approximating e^x by a quadratic on $[0,1]$, which is pretty accurate.

Example: Change an infinite interval to a finite one. (Could use GQ on this one too.)

$$\int_{-\infty}^{\infty} \frac{e^{-x^2}}{1+x^2}\mathrm{d}x = \int_{-\pi/2}^{\pi/2} e^{1-\sec^2(\theta)}\mathrm{d}\theta$$

True value ≈ 1.34329, Trap rule with $n = 4, 8, 16$ points:

$$1.5007, 1.33928, 1.34323$$

Trap rule with 32 points has error -3.7×10^{-9}. This is an example showing that the Trap Rule is extremely accurate for functions that are C^∞ with all derivatives equal to zero at the endpoints.

2 We have seen that the trapezoid rule will converge faster-than-algebraically for infinitely-smooth periodic functions. Recall that in Fourier L^2 approximation theory the Fourier coefficients are given by[2]

$$a_j = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(jx)\mathrm{d}x,\ j \geq 1$$

$$b_j = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(jx)\mathrm{d}x,\ j \geq 1$$

More generally, the coefficients of the complex Fourier series are

$$c_j = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-\mathrm{i}jx}\mathrm{d}x.$$

It's certainly convenient if you can evaluate these integrals analytically, but often you can't. Enter the Trapezoid Rule. If you use the trapezoid rule to compute Fourier coefficients the results won't be perfect, but the errors will converge (as $h \to 0$) extremely fast for smooth functions f.

Applied to the complex Fourier integral above,

$$c_j \approx \frac{1}{n}\sum_{k=0}^{n-1} f(x_k)e^{-\mathrm{i}jx_k},\ \ x_k = k\frac{2\pi}{n}.$$

Note that there's no halving of the endpoints because the function is periodic. Note what happens if you attempt to compute c_{j+n}

$$c_{j+n} \approx \frac{1}{n}\sum_{k=0}^{n-1} f(x_k)e^{-\mathrm{i}(j+n)x_k},\ \ (j+n)x_k = 2\pi\frac{jk}{n} + 2\pi k.$$

$$e^{-\mathrm{i}(j+n)x_k} = e^{-\mathrm{i}jx_k}e^{-2\pi k\mathrm{i}} = e^{-\mathrm{i}jx_k}.$$

This implies that the computed coefficients c_j and c_{j+n} are the same. This is called 'aliasing'. We can therefore only compute n distinct Fourier coefficients using n equispaced points, and we compute Fourier coefficients for values of $j = 0, \ldots, n-1$.

If we want to compute *all* the Fourier coefficients from $j = 0, \ldots, n-1$ we can write them as

$$\boldsymbol{c} = \mathbf{F}\boldsymbol{f}$$

[2]Our analysis in the Approximation notes was for $[-\pi, \pi)$; it has been changed here to $[0, 2\pi)$ to make the connection to DFT cleaner.

where the elements of $\mathbf{F}$ are $\mathbf{F}_{j,k} = e^{-2\pi \mathrm{i} jk/n}/n$. This matrix is the same as the DFT matrix we saw in the section on equispaced trigonometric interpolation (trig interpolation section 3 from the notes). It is unitary up to a constant scaling factor, and there is a fast algorithm (costing $\mathcal{O}(n \log(n))$ rather than $\mathcal{O}(n^2)$ operations) called the FFT to apply it to vectors.

Suppose that $f(x)$ has the following exact Fourier representation

$$f(x) = \sum_{m=-\infty}^{\infty} c_m e^{\mathrm{i}mx}.$$

Applying the Trapezoid Rule to this we see that

$$\text{according to the Trap Rule } c_j \approx \frac{1}{n} \sum_{k=1}^{n} \sum_{m=-\infty}^{\infty} c_m e^{\mathrm{i}(m-j)x_k} = \frac{1}{n} \sum_{m=-\infty}^{\infty} c_m \sum_{k=1}^{n} e^{\mathrm{i}(m-j)x_k}.$$

Look at the inner sum:

$$\sum_{k=1}^{n} e^{\mathrm{i}(m-j)x_k} = \sum_{k=1}^{n} e^{\mathrm{i}(m-j)(k\frac{2\pi}{n})} = \sum_{k=1}^{n} \left(e^{(m-j)\frac{2\pi}{n}\mathrm{i}}\right)^k.$$

This is a geometric sum with the value n whenever $m-j$ is divisible by n, and zero otherwise. Returning to the overall sum

$$\text{according to the Trap Rule } c_j \approx \sum_{q=-\infty}^{\infty} c_{j+qn}.$$

So the computed coefficient is the sum of coefficients

$$\ldots, c_{j-n}, c_j, c_{j+n}, \ldots$$

The error between the Fourier coefficient computed using the Trap Rule and the true Fourier coefficient is the sum of all the true coefficients in the aliasing set.

3* Adaptive quadrature. Integrate(f(x),a,b,tol)

- Compute $I_n[f]$
- Estimate the error
- If the error is small enough, return $I_n[f]$
- Else return Integrate(f(x),a,(a+b)/2,tol') + Integrate(f(x),(a+b)/2,b,tol')

Draw Picture

The absolute value of the total error is $\leq$ the sum of the absolute errors on each subinterval. The practical approach is to estimate the error by computing two different quadratures on each interval, then subtracting them. For example,

$$\text{error} \approx |I_n[f] - I_{2n}[f]|.$$

The main questions are (i) what's the underlying quadrature I_n, and how do you estimate the local error, and (ii) what is tol'? There are lots of ways to do this. We will not go into detail on any methods. Basically you just need be aware that adaptive methods exist.

4* Multivariate. The simplest way to approach a multivariate integral is to reduce it to a sequence of univariate integrals. E.g.

$$\int_a^b \int_c^d f(x,y)\mathrm{d}y\mathrm{d}x = \int_a^b F(x)\mathrm{d}x \text{ where } F(x) = \int_c^d f(x,y)\mathrm{d}y.$$

The difficulty here is that you're not able to evaluate $F(x)$ exactly. To develop an error bound for the whole integral you couldn't use our standard estimates because our standard estimates assume that you're evaluating the integrand exactly.

Our quadratures are all of the form

$$I_n[f] = \sum_i w_i f(x_i).$$

If you can't evaluate the integrand exactly, then you have instead

$$\sum_i w_i (f(x_i) + \epsilon_i) = \sum_i w_i f(x_i) + \sum_i w_i \epsilon_i.$$

Now the usual error estimates apply to the first term, and we can bound the second term by

$$|\sum_i w_i \epsilon_i| \leq \sum_i |w_i|\,|\epsilon_i|.$$

The errors ϵ_i are bounded using the standard estimates for the inner integral, so overall you get a bounded error.

An alternative is to develop quadrature based on multivariate interpolation. Our only multivariate interpolation strategy (in this class) is to use a tensor product grid. In 2D the $n \times n$ interpolation problem reduces to a sequence of n one-dimensional interpolation problems. Then, once you have the interpolating polynomial you can compute its integral which gives you the quadrature. The main difficulty is that a tensor product grid requires a rectangular domain, so if you're integrating over a circle or triangle or something you can't exactly fill it with rectangles. (Draw picture.) To deal with this, suppose that Ω is your domain of integration, and that you fill it as much as possible with rectangles so that the quadrature domain is $\Omega_h \subset \Omega$. Under appropriate assumptions on Ω (e.g. compactness), we have that $\Omega_h \to \Omega$ as you refine the grid. Now consider the integral

$$\int_\Omega f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \int_{\Omega_h} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int_{\Omega \setminus \Omega_h} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}.$$

Assuming f is continuous on the compact set Ω then the second integral will go to 0 as $h \to 0$ (i.e. as the grid is refined). Conversely, if we use a convergent quadrature on each of the rectangles in Ω_h, then the quadrature will converge to the first integral as the grid is refined.

There are many other multidimensional quadrature rules, e.g. you can use triangles rather than rectangles as your basic grid. You can also use Monte-Carlo estimates for high-dimensional problems.