Numerical integration
and the redemption of the trapezoidal rule

S. G. Johnson, MIT Applied Math, IAP Math Lecture Series 2011

Created January 2011; updated May 8, 2020

1 Numerical integration ("quadrature")

Freshman calculus revolves around differentiation and integration. Unfortunately, while you can almost always differentiate functions by hand (if the derivative exists at all), most functions cannot be integrated by hand in closed form [e.g. try integrating $\sin(x+\cos(x))$]. Instead, they must be integrated *approximately* on a computer, a process known as *numerical integration* or *quadrature*. (Historically, "quadrature" was a synonym for integration in general—literally, converting areas into equivalent squares—but in modern usage "quadrature" almost exclusively refers to computational algorithms.)

In particular, suppose we are computing the following definite integral, over an interval $[0,\pi]$ (chosen for convenience below):[1]

$$I = \int_0^\pi f(x)dx$$

for some given function $f(x)$ that is a "black box" (we may only know how to *evaluate it* given any x, not how to manipulate it symbolically). In numerical integration, we want to *approximate* this integral I by a *sum* I_N:

$$I \approx \sum_{n=0}^{N} w_n f(x_n) = I_N$$

for $N+1$ *quadrature points* $x_n \in [0,\pi]$ and corresponding *quadrature weights* w_n. The central problem in numerical integration is this:

- How can we choose the points x_n and the weights w_n so that the error $|I_N - I|$ **goes to zero as rapidly as possible** as we increase N?

This problem has a long history, dating back to ancient approximations for π by approximating the area or circumference of a circle with polygons, and involves beautiful and

[1] Note that there is no loss of generality in the choice of the $[0,\pi]$ interval: if we are computing some arbitrary integral $\int_a^b g(y)dy$, we can convert it back to the $[0,\pi]$ form by a change of variables: $\int_a^b g(y)dy = \frac{|b-a|}{\pi}\int_0^\pi g(\frac{x}{\pi}[b-a]+a)dx = \int_0^\pi f(x)dx$ for $f(x) = \frac{|b-a|}{\pi} g(\frac{x}{\pi}[b-a]+a)$.

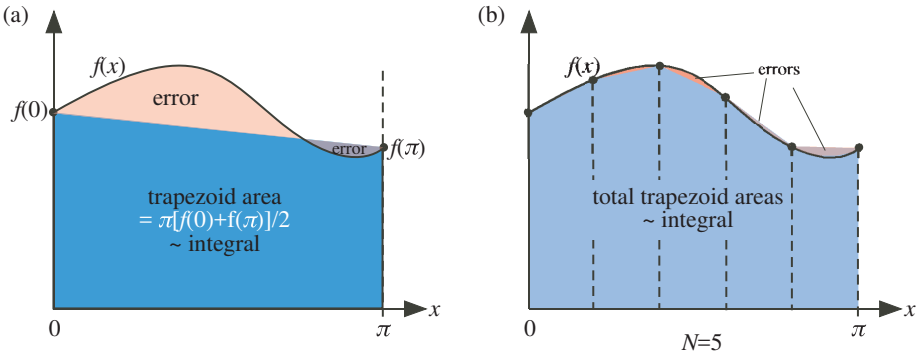

Figure 1: Illustration of (a) the trapezoidal rule and (b) the composite trapezoidal rule for integrating $f(x)$ on $[0,\pi]$. In each case, we approximate the area under $f(x)$ by the area of (a) one or (b) N trapezoids. That is, we evaluate $f(x)$ at $N+1$ points $x_n = n\pi/N$ for $n = 0, 1, \ldots, N$, connect the points by straight lines, and approximate the integral by the integral of this interpolated piecewise-linear function.

sometimes surprising mathematics—the exponential blowup of polynomial ("Newton–Cotes") approximations (Runge phenomena), orthogonal bases of polynomials (Gaussian quadrature, Chebyshev approximation, etc.), and deep forays into Fourier analysis (e.g., Clenshaw–Curtis quadrature). Quadrature is closely related to other important numerical algorithms such as numerical linear algebra (e.g. Lanczos iterations), approximation theory, and fast Fourier transform algorithms (FFTs, which themselves encompose a host of group theory, number theory, polynomial algebras, and other fascinating topics). For higher-dimensional numerical integration (*cubature*), the story becomes even more intricate, ranging from statistics (Monte–Carlo integration) and number theory (low-discrepancy sequences and quasi-Monte–Carlo methods) to fractals (sparse grids).

Here, we will just dip our toes into the problem, beginning by analyzing one of the simplest—deceptively simple!—quadrature methods, the **trapezoidal rule**. Not only does a full analysis of the accuracy of this method lead us directly into the far-reaching topic of *Fourier series*, but we also find that a simple transformation turns the lowly trapezoidal rule from one of the crudest quadrature schemes into one of the best, **Clenshaw–Curtis quadrature**.

2 The trapezoidal rule

The trapezoidal rule, in its most basic form, connects the endpoints $(0, f(0))$ and $(\pi, f(\pi))$ by a straight line and approximates the area by the area of a trapezoid:

$$I \approx \pi \frac{f(0) + f(\pi)}{2},$$

as shown in figure 1(a). Of course this approximation is rather crude, so we refine it by increasing the number of trapezoids: by "trapezoidal rule" one usually means a

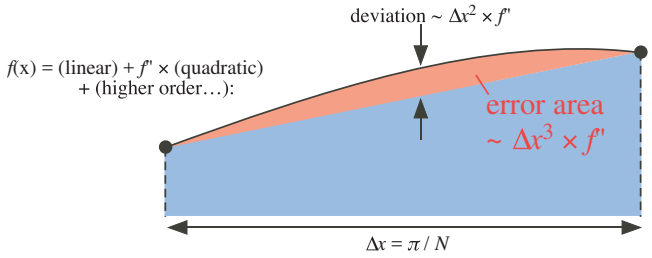

Figure 2: Schematic estimate of the local error of the trapezoidal rule. In a small interval $\Delta x = \pi/N$, the function $f(x)$ can be approximated by a Taylor expansion, and the lowest-order correction to the trapezoidal rule's linear approximation is represented by the quadratic term. Correspondingly, the maximum deviation of $f(x)$ from the trapezoid is $\sim \Delta x^2$ within the interval (multiplied by f'' at some point). The resulting error *area* (red shaded region) is therefore proportional to $\Delta x^2 \times \Delta x = \Delta x^3$.

composite trapezoidal rule: divide $[0,\pi]$ into N intervals and apply the trapezoidal rule to each one, as shown in figure 1(b). In the common case of *equal intervals* of width $\Delta x = \pi/N$, summing these trapezoid areas yields the following approximate integral, also called the *Euler–Maclaurin* formula:

$$I_N = \frac{\pi}{N}\left[\frac{f(0)+f(\pi)}{2} + \sum_{n=1}^{N-1} f(n\pi/N)\right].$$

Note that the $1/2$ factors cancelled except for the first and last points.

Clearly, as $N \to \infty$ we must have $I_N \to I$ (at least, for any Riemann-integrable function). The question now is, *how fast* does the error $|I - I_N|$ decrease with N?

2.1 A simple, pessimistic error estimate

A crude, but perhaps too pessimistic, upper bound on the error is as follows. The trapezoidal rule corresponds to approximating $f(x)$ by a straight line on each interval $\Delta x = \pi/N$. If we look at the Taylor expansion of $f(x)$, the lowest-order deviation from a straight line is the quadratic (f'') term, and this term means that f deviates from a straight line by at most $\sim \Delta x^2$ within the interval (multiplied by some coefficient proportional to f''), as depicted in figure 2. The corresponding error area is then proportional to $\Delta x^2 \times \Delta x = \Delta x^3 \sim 1/N^3$. This is the *local error* from a *single* interval. As there are N such intervals, the *total* error should be bounded above by $N \times 1/N^3 = 1/N^2$: **the error of the trapezoidal rule decreases at worst proportional to** $1/N^2$ (for continuous integrands).

This is an upper bound on the error, because we have neglected the possibility that the errors from different intervals will be of opposite signs and mostly cancel if we are very lucky. At first glance, however, it seems unlikely that such cancellations will occur to such an extent that the error will decrease faster than $1/N^2$, and indeed this turns out to be the case—for *most* $f(x)$, the trapezoidal-rule error is exactly proportional to $1/N^2$

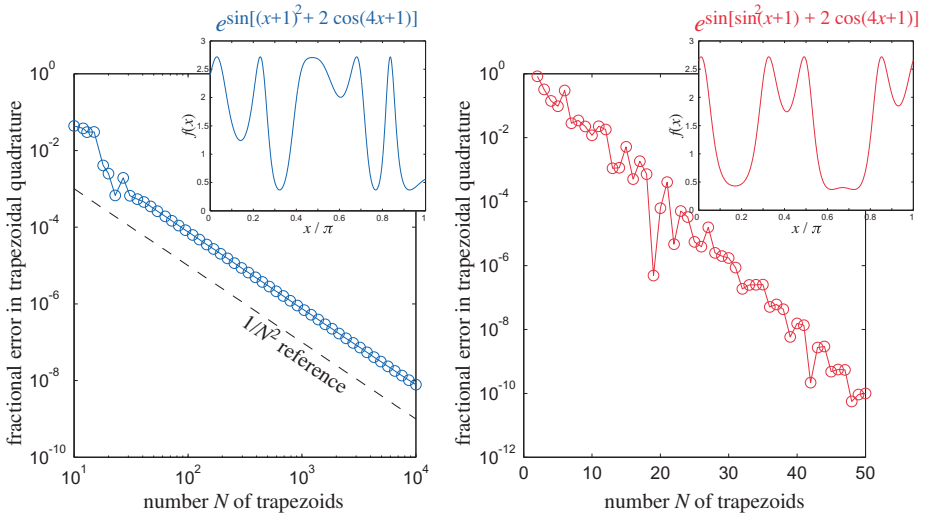

Figure 3: **Left:** Fractional error $|I_N - I|/|I|$ verus N for the trapezoidal rule I_N when integrating the function $e^{\sin[(x+1)^2+2\cos(4x+1)]}$ (inset). For reference, the dashed line shows $1/N^2$ dependence, demonstrating that the I_N errors indeed decrease asymptotically at this rate. **Right**: the same fractional error, but this time for the integrand $e^{\sin[\sin^2(x+1)+2\cos(4x+1)]}$ (inset). Note that this is a semi-log scale: the I_N errors appear to be decreasing at least *exponentially* fast with N—a miracle has occurred in the trapezoidal rule?

for large N. However, it turns out there are *some* functions that do much, much better; if we can understand these special cases, and in general understand the error more rigorously, we can try to rearrange the computation so that this improvement occurs almost always.

2.2 A numerical experiment and a Miracle

In figure 3(left) is plotted the fractional error $|I_N - I|/|I|$ in the trapezoidal-rule approximation versus N on a log–log scale for an arbitrarily chosen nasty-looking function $e^{\sin[(x+1)^2+2\cos(4x+1)]}$ shown in the inset. As expected from the crude analysis above, the errors decrease asymptotically at a rate that is almost exactly proportional to $1/N^2$ (an exact $1/N^2$ dependence is shown as a dashed line for reference). If we want eight decimal places of accuracy we need around 10^4 function evaluations, but this is not too bad on a computer (unless we need to evaluate millions of such integrals, or unless our integrand is a much nastier function like the output of a planetary climate simulation!).

For "fun," we try a slightly different integrand in the right panel of figure 3: $e^{\sin[\sin^2(x+1)+2\cos(4x+1)]}$ (inset). Again plotting error versus N, it seems again to be roughly a straight line (a little more wiggly than before)—but wait, this is no longer a log–log scale, this is a *log–linear* scale. A straight line on such a scale indicates an *exponential* dependence with N. How has the ordinary trapezoidal rule, piecewise-linear approximations, man-

aged to achieve apparently exponential accuracy (or better[2]) with N? Some miracle of cancellation seems to have occurred, but why? If we could somehow manage to obtain this miracle for more integrands, what a wonderful quadrature method we would have!

A clue to the miracle is that the integrand in figure 3(right) does have one special property that is obvious with a little thought: it is periodic [$f(x+\pi) = f(x)$]. Why should periodicity help so much, though? That is probably not clear yet, but it points us towards an analytical technique where periodicity is central: Fourier analysis.

2.3 Error analysis by the Fourier cosine series

A beautiful, powerful, and far-reaching way to rigorously analyze the approximation errors in the trapezoidal rule is to use Fourier analysis, which relies on an amazing fact: any "reasonable" function can be expressed as an infinite series of sines and/or cosines, and sines/cosines are much easier to analyze than arbitrary functions. In particular, for this problem it turns out to be convenient to use a Fourier cosine series: write $f(x)$ as

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(kx)$$

for coefficients a_k given by:

$$a_k = \frac{2}{\pi} \int_0^{\pi} f(x) \cos(kx)\, dx.$$

For any $f(x)$ that is continuous and non-singular on $[0, \pi]$, this series converges to $f(x)$ at every x, and $a_k \to 0$ for $k \to \infty$. It turns out that how *fast* the Fourier series converges (how fast $a_k \to 0$) is determined entirely by the *smoothness* of $f(x)$ and by its *endpoint* derivatives at $x = 0, \pi$. Moreover, we will be able to describe the convergence rate of the trapezoidal rule entirely in terms of the convergence rate of the Fourier cosine series!

2.3.1 The exact integral I

Plugging the cosine series into the exact integral I, we obtain a simple result:

$$\begin{aligned} I &= \int_0^{\pi} f(x)dx = \frac{a_0\pi}{2} + \sum_{k=1}^{\infty} a_k \cancel{\int_0^{\pi} \cos(kx)dx} \\ &= \frac{a_0\pi}{2}, \end{aligned}$$

because all of the $k > 0$ terms integrate to zero.

[2] In fact, the rate of decrease here may in fact be faster than exponential, a consequence of the niceness of this integrand everywhere in the complex x plane, but it is hard to tell the difference without plotting over a much larger range of errors (impossible here because of the finite precision of computer arithmetic).

2.3.2 The trapezoidal rule I_N

Plugging the cosine series into the trapezoidal rule I_N, the constant a_0 term just gets multiplied by N (it appears in $N+1$ terms, two of which have coefficient $1/2$), while the other terms give a nasty-looking sum:

$$\begin{aligned} I_N &= \frac{\pi}{N}\left[\frac{a_0}{2}N + \sum_{k=1}^{\infty} a_k \left(\frac{\cos(0k\pi/N) + \cos(Nk\pi/N)}{2} + \sum_{n=1}^{N-1} \cos(nk\pi/N) \right)\right] \\ &= I + \frac{\pi}{N}\sum_{k=1}^{\infty} a_k S_k, \end{aligned}$$

where the a_0 term is just the exact integral I and S_k is the sum:

$$S_k = \frac{\cos\left(0\frac{k\pi}{N}\right) + \cos\left(N\frac{k\pi}{N}\right)}{2} + \sum_{n=1}^{N-1} \cos\left(n\frac{k\pi}{N}\right).$$

Amazingly (you will prove this for homework, below), this sum S_k is almost always zero. The only time it is nonzero is when $k = 2mN$ where m is any integer (i.e., when k is an *even* multiple of N), in which case:

$$\begin{aligned} S_{2mN} &= \frac{\cos(0) + \cos(2mN\pi)}{2} + \sum_{n=1}^{N-1} \cos\left(n\frac{2m\cancel{N}\pi}{\cancel{N}}\right) \\ &= \frac{1+1}{2} + \sum_{n=1}^{N-1} 1 = N, \end{aligned}$$

using the fact that $\cos(2\pi\ell) = 1$ for any integer ℓ. This N factor cancels with π/N in I_N Therefore, the nasty I_N sum reduces to the remarkably simple expression:

$$I_N = I + \pi \sum_{m=1}^{\infty} a_{2mN}.$$

2.3.3 Convergence rate

The error in the trapezoidal rule is therefore exactly:

$$I_N - I = \pi \sum_{m=1}^{\infty} a_{2mN}.$$

As N increases, this depends on a_k for larger and larger k, but $a_k \to 0$ as $k \to \infty$, and so the error goes to zero. **The convergence rate of the Fourier cosine series determines the convergence rate of the trapezoidal rule!** The faster $a_k \to 0$ with increasing k, the faster we will have $I_N - I \to 0$ with N.

Fortunately, there is a long history of analysis of the convergence rate of the Fourier series which we can now bring to bear on the trapezoidal rule, and very general theorems are known. In the attached handout on cosine series, I go through one of the simplest analysis of convergence rates, which only requires repeated integration by parts on the a_ℓ formula.

In particular, for an arbitrary continuous $f(x)$ that has no particularly special behavior at the endpoints $x = 0, \pi$, the coefficients a_k approach an $a_k \approx \#/k^2$ dependence for large k, with some coefficient #. This means that, for large N, the trapezoidal-rule error approaches:

$$I_N - I \approx \pi \sum_{m=1}^{\infty} \frac{\#}{(2mN)^2} = \frac{\pi\#}{4N^2} \sum_{m=1}^{\infty} \frac{1}{m^2} \sim \frac{1}{N^2},$$

precisely the $1/N^2$ dependence we predicted from a crude analysis above. (In fact, the sum of $1/m^2$, the famous "Basel problem" solved by Euler, is exactly $\pi^2/6$, so the error approaches $\pi^3\#/24N^2$.)

However, if f' vanishes at the endpoints, then $a_k \sim 1/k^4$ and hence $I_N - I \sim 1/N^4$. And if f''' vanishes at the endpoints, then $a_k \sim 1/k^6$ and $I_N - I \sim 1/N^6$. In fact, we don't actually care what a_k does for odd k, only for even k, and for even k it turns out to be enough for f' etcetera to be periodic, i.e. $f'(0) = f'(\pi)$ and so on, to obtain this faster convergence. And if *all* of the odd derivatives of f are periodic at the endpoints, e.g. if $f(x)$ itself is *periodic* (and smooth, i.e. infinitely differentiable) *or* if $f(x)$ is *even* around both boundaries (which makes all the odd derivatives vanish), then $I_N - I$ **vanishes faster than any power of** $1/N$ (typically exponentially fast).[3]

3 Clenshaw–Curtis quadrature

Even if $f(x)$ is a nice, smooth function inside the integration interval, it is often not even or periodic at its endpoints, a fact which is responsible for reducing the error of the trapezoidal rule to the $\sim 1/N^2$ worst case. However, there is a *simple change of variables* which allows us to obtain the miraculous exponential convergence *all the time* (at least for smooth functions), and this technique is called **Clenshaw–Curtis quadrature**.

For this section, it is more convenient to start with an integral over $[-1, 1]$ rather than $[0, \pi]$, and then make a change of variables $x = \cos\theta$ to transform the former into the latter:

$$I = \int_{-1}^{1} f(x)dx = \int_{0}^{\pi} f(\cos\theta)\sin\theta\, d\theta.$$

By construction, $f(\cos\theta) = f(\cos[-\theta])$ and $f(\cos[\pi - \theta]) = f(\cos[\pi + \theta])$ so $f(\cos\theta)$ is an *even function* around both $x = 0$ and $x = \pi$, and hence *all of its odd-order derivatives with* θ *vanish at the endpoints*. This is precisely one of the conditions under which the Fourier cosine series converges super-fast, and hence the trapezoidal rule converges super-fast for $f(\cos\theta)$. Unfortunately, this is spoiled in I thanks to our multiplication with $\sin\theta$, which is odd. We fix this problem by an additional trick: we *first* expand $f(\cos\theta)$ in a cosine series, and *then* do the I integrals analytically term-by-term:

$$f(\cos\theta) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k\theta),$$

[3] A more detailed analysis of exactly how fast the convergence is for periodic/even functions requires a fair amount of complex analysis (contour integration and so on). For example, if $f(x)$ is an "analytic" function in a finite-width strip around the real-x axis, then one can show by contour integration that the error converges at least exponentially fast. See the references at the end.

$$a_k = \frac{2}{\pi}\int_0^\pi f(\cos\theta)\cos(k\theta)\,d\theta,$$

$$I = \int_0^\pi \left[\frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k\theta)\right] \sin\theta\, d\theta = a_0 + \sum_{k=1}^{\infty} \frac{2a_{2k}}{1-(2k)^2}.$$

Now, we have performed the I integral analytically, but how do we perform the integrals to obtain the Fourier coeffients a_k? We do these numerically by the trapezoidal rule! That is:

$$a_k \approx \frac{2}{\pi} \times \frac{\pi}{N}\left[\frac{f(\cos 0) + f(\cos\pi)}{2} + \sum_{n=1}^{N-1} f\left(\cos\frac{n\pi}{N}\right)\cos\left(\frac{kn\pi}{N}\right)\right].$$

But how is this an improvement? We have just swapped one trapezoidal rule summation for a zillion of them! Three points save us:

- The a_ℓ integrand, $f(\cos\theta)\cos(\ell\theta)$, is even around its endpoints and so the trapezoidal rule for a_k converges super-fast (usually exponentially fast). So, we only need a small N.
- $f(\cos\theta)$ is even around its endpoints, so the coefficents a_k go to zero super-fast (usually exponentially fast). So, we only need a small number of a_k coeffients. In fact, the convergence rate for a_k is about the same as the convergence rate of the trapezoidal rule, so we usually only compute N coefficients a_k.
- When computing a_k, we always evaluate $f(x)$ at the same points $f(\cos\frac{n\pi}{N})$ independent of k. So, we only need to evaluate $f(x)$ at $N+1$ points *once.*

There are actually additional nice properties. The trapezoidal rule for a_k turns out to be a very special summation called a *discrete cosine transform*, allowing us to factorize the sum in a way that shares computations for different k via a "fast Fourier transform" (FFT). All in all, it turns out that one can accurately compute the integral in $\sim N\log N$ operations.

Moreover, one can *precompute* the weights of a quadrature rule for a given N. If we denote the vector of $(a_0, a_2, \ldots, a_N)$ values as $\vec{a}$, then $I_N = \vec{d}^T\vec{a}$ where $\vec{d}$ is the vector of $1/(1-4k^2)$ coefficients of our truncated I sum above. Let $\vec{f}$ denote the vector of $f(\cos\frac{n\pi}{N}) = f(x_n)$ values, in which case the a_k computation above can be written as a matrix–vector product $\vec{a} = C\vec{f}$, where C is a discrete-cosine transform matrix with entries $\sim\cos(\frac{kn\pi}{N})$. It then follows that $I_N = \vec{d}^T C\vec{f} = \vec{w}^T\vec{f} = \sum w_n f(x_n)$, where $\vec{w} = C^T\vec{d}$ is a precomputed vector of quadrature weights (and, in fact, C^T is another discrete cosine transform, so again one can use FFT algorithms to compute $\vec{w}$ in $O(N\log N)$ operations. (Moreover, if we exploit the fact that only *even*-index a_{2k} coefficients are needed for I_N, the problem simplifies a bit further: a single weight is used for the symmetric combination $f(\cos\frac{n\pi}{N}) + f(\cos\frac{(N-n)\pi}{N}) = f(x_n) + f(-x_n)$, and the size of C is halved.)

The key idea was that, by changing variables $x = \cos\theta$ and doing a cosine-series expansion of $f(\cos\theta)$, we can transform *any* integral into integrals purely of even functions, allowing us to **apply the miraculous convergence of the trapezoidal rule for such functions to *any integral*.** (The only thing that will spoil this is if f has other discontinuities or singularities within the integration interval.)

3.1 Chebyshev polynomials

I would be remiss not to mention a connection between the above analysis and another beautiful branch of mathematics: *Chebyshev approximation*. Look back at the cosine series for $f(\cos\theta)$ but write it in terms of the original variable $x = \cos\theta$ via $\theta = \cos^{-1} x$:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k \cos^{-1} x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k T_k(x),$$

which is expanding $f(x)$ in the nasty-looking functions $T_k(x) = \cos(k\cos^{-1} x)$. What a horrible way to disguise a cosine series! But then a miracle occurs: the functions $T_k(x)$ are not nasty at all, they are actually **just polynomials** called **Chebyshev polynomials**:

$$T_0(x) = \cos(0\cos^{-1} x) = 1,$$

$$T_1 = \cos(\cos^{-1} x) = x,$$

$$T_2 = \cos(2\cos^{-1} x) = 2\cos^2(\cos^{-1} x) - 1 = 2x^2 - 1,$$

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x),$$

where the T_2 and T_{k+1} expressions use angle-addition identities.

So, expanding $f(\cos\theta)$ in a cosine series is exactly equivalent to expanding $f(x)$ in a series of polynomials. But they are special, wonderful, beautiful polynomials that have all the amazing properties of a cosine series. The coefficients a_k of these polynomials converge exponentially fast for smooth $f(x)$. These polynomials $T_k(x)$ stay in between -1 and 1 for x in $[-1,1]$, and all of their k roots are in the interval $[-1,1]$. We can compute the coefficients a_k quickly by FFT algorithms. And, of course, once we have expanded $f(x)$ in polynomials, almost anything that we might want to do (integration, differentiation, finding roots, etcetera) becomes easy.

4 Further reading

Much more information about the convergence rates of Fourier series and related numerical methods can be found e.g. in the book *Chebyshev and Fourier Spectral Methods* by John P. Boyd (available online at `http://tinyurl.com/24mepgc`). A wonderful practical realization of Clenshaw–Curtis quadrature and a host of related numerical techniques can be found in the *chebfun* package by Trefethen et al., which available as free software for Matlab along with a series of very readable review-style publications at `http://www2.maths.ox.ac.uk/chebfun/`; further developments of these ideas can be found in the Julia *ApproxFun* package (`https://github.com/JuliaApproximation/ApproxFun.jl`).

Many numerical-analysis textbooks instead analyze the error of the trapezoidal rule via something called Bernoulli polynomials; this approach can be found (with references) in the Wikipedia article on the Euler–Maclaurin formula or in (e.g.) *An Introduction to Numerical Analysis* by Süli and Mayers or *An Introduction to Numerical Analysis* by Stoer and Bulirsch. However, I find it much nicer to relate the problem to

Fourier series, which are not only much more widely known and more widely applicable, but also point the way towards more accurate methods such as Clenshaw–Curtis. I've also put some notes and links on Clenshaw–Curtis techniques up on Wikipedia ("Clenshaw–Curtis quadrature," *Wikipedia*, accurate as of 5 January 2011).

Homework problems

1. In lecture, I claimed that the sum

$$S_k = \frac{\cos\left(0\frac{k\pi}{N}\right) + \cos\left(N\frac{k\pi}{N}\right)}{2} + \sum_{n=1}^{N-1} \cos\left(n\frac{k\pi}{N}\right)$$

   is equal to zero unless k is an even multiple of N (i.e. $k = 2mN$ for some integer m). Prove this. *Hint:* recall the identity $\cos(A)\cos(B) = \frac{\cos(A+B)+\cos(A-B)}{2}$ and show that $S_k \cos(k\pi/N) = S_k$. It follows that either $S_k = 0$ or $\cos(k\pi/N) = 1$, and so...

2. The trapezoidal rule is based on linear interpolation of pairs of $f(x)$ points. If instead we employ *quadratic* interpolation of *triples* of points, we get *Simpson's rule* for integration. Dividing $[0,\pi]$ again into equal intervals and summing the Simpson rule for each interval, we obtain the **composite Simpson rule** $\tilde{I}_N$, which for even N is:

$$I \approx \tilde{I}_N = \frac{2\pi}{3N}\left[\frac{f(0)+f(\pi)}{2} + \sum_{\substack{0<n<N \\ n \text{ even}}} f(n\pi/N) + 2\sum_{\substack{0<n<N \\ n \text{ odd}}} f(n\pi/N)\right]$$

$$= \frac{2\pi}{3N}\left[\frac{N}{\pi}I_N + \sum_{\substack{0<n<N \\ n \text{ odd}}} f(n\pi/N)\right]$$

   where I_N is the trapezoidal rule from above. (Google will turn up multiple derivations of the composite-Simpson formula if you are curious.)

   In this problem, you will employ the techniques from lecture to analyze the accuracy (convergence rate) of this composite Simpson rule.

   (a) Plugging in the Fourier cosine series for $f(x)$, show that

$$\tilde{I}_N = I + \frac{\pi}{3}\left[-a_N + 3a_{2N} - a_{3N} + 3a_{4N} - \cdots\right].$$

Hint: You are given the following formula for $\sum_{n \text{ odd}} \cos(nk\pi/N)$ (for even N), similar to the one in the previous problem:

$$\sum_{\substack{0<n<N \\ n \text{ odd}}} \cos(nk\pi/N) = \frac{N}{2} \begin{cases} (-1)^m & k = mN \\ 0 & \text{otherwise} \end{cases}.$$

[You need not prove this summation formula; the proof is very similar to the one from the previous problem except that you multiply by $\cos(2k\pi/N)$.]

(b) You are *given* the following two series formulas (famously derived by Euler[4]):

$$\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots = \frac{\pi^2}{6}, \qquad \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \cdots = \frac{\pi^2}{8}.$$

For a typical (smooth) $f(x)$ where the cosine series coefficients go asymptotically as $a_k \sim \#/k^2$ for some #, plug this decay rate into your series from the previous part, combined with the two formulas here, to show that the $1/N^2$ term in the error $\tilde{I}_N - I$ *cancels*.

Therefore, when looking at the convergence rate of the a_k coefficients, we must continue our integration-by-parts analysis from class and the cosine-series handout to look at *the next biggest term*. Hence the error in the composite Simpson rule typically goes as $1/N$ **to what power**?

(c) Try the composite Simpson rule on a computer for $f(x) = e^x$. For example, you can evaluate the error with $N = 100$ in Matlab (available on Athena[5]) by typing the following two (one-line) commands

```
N=100

2*pi/(3*N) * ((exp(0)+exp(pi))/2 + sum(exp([1:N-1]*pi/N))
+ sum(exp([1:2:N-1]*pi/N))) - (exp(pi)-exp(0))
```

Divide the error for $N = 100$ by the error for $N = 200$ to verify that the error is decreasing at the rate you expected from the previous part. (If you are more ambitious, you might try plotting the |error| vs. N on a log–log scale.)

[4] See e.g. E. Sandifer, "Euler's Solution of the Basel Problem—The Longer Story" in *Euler at 300: An Appreciation*, Bradley et al., eds. (Math Assoc. Am., 2007), available online at tinyurl.com/2gyctkf

[5] For instructions on running Matlab on Athena, see tinyurl.com/39n8xvl and for a quick reference to Matlab commands see tinyurl.com/3yk5d3z