MATH 2160, Chapter 4 Summary & Exercises

Richard M. Slevinsky

A Conversation with Slevinsky

Problems	Solutions
Given a sufficiently smooth function, how do I use function samples to estimate derivatives?	Finite difference formulæ provide numerical estimates of derivatives. A method whose error scales as $\mathcal{O}(h^p)$ for some $p > 0$ is p^{th} order. Taylor series with Lagrange's remainder allow us to obtain simple order estimates. Just expand each nearby function sample about x and add up all similar terms.
In finite-precision, rounding error limits their applicability.	For infinitely smooth functions, complex differences provide a way around the instability.
How can I numerically differentiate *programmatically*?	Dual numbers extend a field $\mathbb{F}$ ($= \mathbb{R}$ or $\mathbb{C}$) to the quotient ring $\mathbb{D} = \mathbb{F}[\varepsilon]/\varepsilon^2$. They are like complex numbers but the unit dual number satisfies $\varepsilon^2 = 0$ rather than $\mathrm{i}^2 = -1$. Since: $$(a + b\varepsilon)(c + d\varepsilon) = ac + (ad + bc)\varepsilon,$$ and since: $$f(a + b\varepsilon) = f(a) + f'(a)b\varepsilon,$$ for differentiable f, they naturally implement both the product rule and the chain rule for differentiation.
How can I integrate a function using samples at equispaced points?	Newton–Cotes quadrature formulæ are derived by integrating a polynomial interpolant at equispaced points.
But what about the Runge phenomenon?	The Runge phenomenon in interpolation translates to the development of negative quadrature weights for $n \geq 9$, making high order Newton–Cotes formulæ useless. If equispaced points must be used, then one solution is to use composite formulæ, by splitting the original integration interval into numerous subintervals.
Interesting. And what is the error?	Generically, we have Theorem 4.2.2, but in many cases we can get tighter bounds, using the integral mean value theorem and a little sweat.

What if I can evaluate the function anywhere in my domain?

Gaussian quadrature is the result of treating *both* weights *and* nodes as variables, seeking to exactly integrate as high a degree of polynomial as possible. Using $2n$ degrees of freedom, we should be able to integrate polynomials of degree-$(2n-1)$ exactly.

Great! But the same setup I used to find the Newton–Cotes weights gives a *nonlinear* system of equations for Gaussian quadrature. What do I do now?

Theorem 4.4.1 highlights the connection between nodes of Gaussian quadrature, and the zeros of orthogonal polynomials. From the monic recurrence relation:

$$\hat{\pi}_{n+1}(x) = (x - \alpha_n)\hat{\pi}_n(x) - \beta_n \hat{\pi}_n(x),$$

we setup the (symmetric) Jacobi matrix:

$$J := \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & & \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \sqrt{\beta_{n-2}} & \alpha_{n-2} & \sqrt{\beta_{n-1}} \\ & & & \sqrt{\beta_{n-1}} & \alpha_{n-1} \end{bmatrix},$$

and solve for the nodes as the eigenvalues. The weights are the squares of the first components of the eigenvectors times the zeroth moment $\mu_0 = \int_D \mathrm{d}\mu(x)$.

In higher dimensions, the "curse of dimensionality" sets in, and my once glorious Gaussian quadrature rule is no longer as effective. Where should I place my next bet?

Monte Carlo integration uses a large number of function samples from randomly distributed points in the domain to obtain an estimate of an integral as the expectation (mean value). For N points, the standard deviation is $\mathcal{O}(N^{-1/2})$, *independent of the dimension* of the problem! Monte Carlo integration can be used to numerically estimate integrals in $D \subset \mathbb{R}^d$ with d well into the millions.

Oh, by the way, what is Richardson extrapolation?

Richardson extrapolation exploits the structure of the error in an approximation (a power series in a small tuneable parameter h) in order to try to extrapolate from current best estimates.

The system of equations satisfied by the Richardson extrapolant is a *Vandermonde* system, and we develop a sneaky way to solve it in $\mathcal{O}(n^2)$ operations by only requesting the extrapolant, and not the extra degrees of freedom that have been introduced. The extrapolants are the ratios of two sequences of Newton's divided differences that can even be done by hand in some cases!

Exercises

1. The forward difference:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

is a first order method for approximating $f'(x)$. Using two different values of h, say, h_0 and h_1, derive the Richardson extrapolant. If $h = h_0 = 2h_1$, carefully explain why you arrive at:

$$f'(x) \approx \frac{-f(x+h) + 4f(x+\frac{h}{2}) - 3f(x)}{h}.$$

What is the order of this method? Use it to numerically differentiate e^x at $x = 1$. How does it compare to the centred difference formula?

2. Complex differences can be extended to higher order methods. If we used three points in the complex plane on the circle of radius h centred at the point x, it seems reasonable that the three points that would lead to a formula with the smallest rounding error would be points that are equidistant on the circle (the further away two function samples, the larger the differences, leading to less amplification of subtractive cancellation). Show that the formula:

$$f'(x) \approx \Re\left\{\frac{f(x+h) + e^{-\mathrm{i}2\pi/3} f(x + e^{\mathrm{i}2\pi/3}h) + e^{-i4\pi/3} f(x + e^{\mathrm{i}4\pi/3}h)}{3h}\right\},$$

is third order. Use it to numerically differentiate e^x at $x = 1$. At which point does the subtractive cancellation ruin the results?[1]

3. In the assignment, you created the nodes and weights of the Gauss–Hermite quadrature rule. Tabulate the result when you integrate:

$$\begin{aligned}
&\int_{\mathbb{R}} \cos(x) e^{-x^2}\,\mathrm{d}x = \sqrt{\pi} e^{-\frac{1}{4}}, \\
&\int_{\mathbb{R}} \frac{e^{-x^2}}{x^2 + 2^2}\,\mathrm{d}x = [1 - \Phi(2)]\frac{\pi e^4}{2} \approx 0.401\,174\,590\,227\,840\,8\ldots, \quad \text{and,} \\
&\int_{\mathbb{R}} |x|^3\, e^{-x^2}\,\mathrm{d}x = 1.
\end{aligned}$$

for $n = 10, 20, 30, \ldots, 150$. Here Φ is the error function (available in JULIA via `erf` in the `SpecialFunctions` package, `Pkg.add("SpecialFunctions")`).

Gaussian quadrature is designed to integrate functions that look like polynomials. Does this help you explain why Gauss–Hermite works well for the first two integrals, but not the third?

4. Use Monte Carlo integration to estimate the multivariate normal distribution:

$$\left(\frac{2}{\sqrt{\pi}}\right)^d \int_{[0,1]^d} e^{-(x_1^2 + x_2^2 + \cdots + x_d^2)}\,\mathrm{d}x_1\,\mathrm{d}x_2 \cdots \mathrm{d}x_d = [\Phi(1)]^d \approx 3.692\,667\,955\,058\,481 \times 10^{-08},$$

for $d = 100$. Clearly, d-dimensional numerical integration is not necessary for the normal distribution, because the variables separate. But not every probability density function allows separation of variables!

[1] This exercise is in some way a preview into the next chapter on Fourier analysis. In this chapter, we will create methods of arbitrarily high order, which will result in very low errors due to rounding and subtractive cancellation.