MATH/CS 5466 · NUMERICAL ANALYSIS

Problem Set 4

Posted Wednesday 23 March 2016. Due Wednesday 30 March 2016 (5pm).
Please complete all 4 problems (total of 100 points).

1. [25 points]

   Clenshaw–Curtis quadrature approximates the integral of a function by the integral of the degree-N polynomial that interpolates it at $N+1$ Chebyshev points $x_j = \cos(j\pi/N)$, $j = 0, \ldots, N$.

   A MATLAB routine for computing the nodes and weights of this rule for the interval $[a, b] = [-1, 1]$, called `clencurt.m`, is linked from the course website. (This implementation is from Trefethen's *Spectral Methods in MATLAB* book.)

   (a) Use Clenshaw–Curtis quadrature (with nodes and weights from `clencurt.m`) to approximate $\int_{-1}^{1} f(x)\,\mathrm{d}x$ for each of:

   $$f(x) = \mathrm{e}^{-x^2}, \qquad f(x) = (1 + 25x^2)^{-1}, \qquad f(x) = |x|.$$

   In particular, for each f, produce a `semilogy` plot showing the degree of interpolating polynomial N versus the error between the Clenshaw–Curtis approximation and the true integral (whose values can be computed in MATLAB via `sqrt(pi)*erf(1)`, `2*atan(5)/5`, and `1`), for $N = 1, \ldots, 50$.

   (b) Why do you think the three different functions in part (a) produce such different results?

   (c) For the first function, $f(x) = \mathrm{e}^{-x^2}$, use MATLAB's `tic` and `toc` commands to time how long it takes compute the Clenshaw-Curtis approximation for $N = 20$ (include the time for computing the nodes and weights). Compare this value to the time required to integrate this same f using MATLAB's all-purpose adaptive quadrature routine, `quad`, with precision `1e-15`.

2. [30 points]

   You may use the following facts without proof: For positive integers N and n,

   $$\sum_{k=1}^{N} \sin\left(\frac{2\pi nk}{N}\right) = 0$$

   and

   $$\sum_{k=1}^{N} \cos\left(\frac{2\pi nk}{N}\right) = \begin{cases} N & \text{if } n/N \text{ is an integer;} \\ 0 & \text{otherwise.} \end{cases}$$

   (a) Write down the composite trapezoid rule for approximating

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

   with function evaluations at $x_k = a + kh$ for $h = (b-a)/N$ and $k = 0, \ldots, N$.

(b) Suppose we wish to approximate

$$\int_0^{2\pi} f(x)\,\mathrm{d}x,$$

where f is a 2π-periodic function (that is, $f(x) = f(x + 2\pi)$ for all $x \in \mathbb{R}$). Show that in this case the composite trapezoid rule reduces to

$$\frac{2\pi}{N}\sum_{k=1}^{N} f\Big(\frac{2\pi k}{N}\Big).$$

For the rest of the problem, suppose f is a 2π-periodic function with Fourier series

$$f(x) = \frac{c_0}{\sqrt{2\pi}} + \sum_{n=1}^{\infty} c_{-n}\frac{1}{\sqrt{\pi}}\cos(nx) + \sum_{n=1}^{\infty} c_n\frac{1}{\sqrt{\pi}}\sin(nx)$$

for constants $c_0, c_{-1}, c_1, \ldots$.

(c) Integrate each term of this Fourier series to obtain a simple formula for $\int_0^{2\pi} f(x)\,\mathrm{d}x$.

(d) Write down a descriptive bound for the difference between the true integral found in part (c) and the approximation obtained from the composite trapezoid rule applied to the function f with the above Fourier series.

(e) If f and its first $p > 1$ derivatives are continuous and 2π-periodic, then there exists a constant γ such that $|c_{|n|}| \le \gamma/|n|^p$ for all integers n. Compare the performance of the composite trapezoid rule applied to such an f with the usual composite trapezoid error bound that holds for functions that are in C^2, but not necessarily 2π-periodic.

(f) Produce a `loglog` plot comparing the error in the trapezoid rule approximations to the 2π-periodic problem

$$\int_0^{2\pi} \exp(\sin(x))\,\mathrm{d}x = 7.954926521012845274513 22\ldots$$

and the smooth but not 2π-periodic problem

$$\int_0^{2\pi} \exp(\sin(x/\pi))\,\mathrm{d}x = 13.3094551602297896414536\ldots.$$

3. [25 points]

Let $\langle f, g\rangle$ denote a weighted inner product on $L^2(a, b)$, e.g., $\langle f, g\rangle = \int_a^b f(x)g(x)w(x)\,\mathrm{d}x$ for some appropriate weight function w.

(a) Suppose $\{\phi_j\}_{j=0}^n$ is a system of *monic* orthogonal polynomials, and let

$$\phi_{n+1}(x) = x\,\phi_n(x) - \sum_{j=0}^{n} c_{n,j}\phi_j(x)$$

denote the next polynomial in this system. For $n \ge 1$, show that we can write

$$\phi_{n+1}(x) = x\,\phi_n(x) - \alpha_n\phi_n(x) - \beta_n\phi_{n-1}(x),$$

where α_n and β_n are constants that you should specify. (This was worked out in the lectures, but the result is sufficiently important that you should work through the details for yourself.)

(b) Consider the *Jacobi* matrix and vector

$$\mathbf{J} = \begin{pmatrix} \alpha_0 & \sqrt{\beta_1} & & & \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \sqrt{\beta_2} & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \sqrt{\beta_n} \\ & & & \sqrt{\beta_n} & \alpha_n \end{pmatrix}, \qquad \mathbf{v}(x) = \begin{pmatrix} \phi_0(x) \\ \phi_1(x)/\sqrt{\beta_1} \\ \phi_2(x)/\sqrt{\beta_1\beta_2} \\ \vdots \\ \phi_n(x)/\sqrt{\beta_1 \cdots \beta_n} \end{pmatrix}.$$

(For $n \geq 1$, α_n and β_n are as in part (a); α_0 is defined by $\phi_1(x) = x\phi_0(x) - \alpha_0\phi_0(x)$.)

Verify that $(\lambda, \mathbf{v}(\lambda))$ is an eigenpair of $\mathbf{J}$ if and only if λ is a root of ϕ_{n+1}.

N.B. *This observation provides a way to compute Gaussian quadrature nodes. The Gaussian quadrature weights can be extracted from the eigenvectors.*

4. [25 points]

Monte Carlo methods provide an alternative to interpolatory quadrature schemes. Given $f \in C[a, b]$, from a probabilistic perspective one can interpret the integral of f as its *expected value*:

$$\int_a^b f(x)\,\mathrm{d}x =: \mathsf{E}[f].$$

The expected value $\mathsf{E}[f]$ is just another name for the *mean*, which can be estimated by averaging values of f sampled at uniformly distributed random points in $[a, b]$. Hence, the N-point *Monte Carlo* estimate of the integral is given by

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{N} \sum_{k=1}^{N} f(\xi_j),$$

where $\xi_1, \ldots, \xi_N$ are N independent samples of a uniform random variable over $[a, b]$, as can be generated using MATLAB's `rand` command. The Central Limit Theorem suggests that this estimate will converge to the exact integral at a rate of $N^{-1/2}$. (For details, see R. E. Caflisch, "Monte Carlo and quasi-Monte Carlo methods," *Acta Numerica*, 1998. For alternative methods for multidimensional integrals, read up on *Smolyak grids*.)

(a) Consider the function $f(x) = \sin(x)$ over $x \in [0, 2]$. Produce a `loglog` plot comparing N versus error for the N-point Monte Carlo method and the trapezoid rule based on N function evaluations, for various values of N. (Take your largest N to be at least 10^5.) Which method is superior?

The rest of this problem concerns approximation of a 10-dimensional integral,

$$\int_a^b \int_a^b \int_a^b \int_a^b \int_a^b \int_a^b \int_a^b \int_a^b \int_a^b \int_a^b f\big(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\big)\,\mathrm{d}x_1\mathrm{d}x_2\mathrm{d}x_3\mathrm{d}x_4\mathrm{d}x_5\mathrm{d}x_6\mathrm{d}x_7\mathrm{d}x_8\mathrm{d}x_9\mathrm{d}x_{10},$$

a type of problem that arises in mathematical physics and financial modeling.

(b) Write a MATLAB code based on nested trapezoid rules to evaluate this ten-dimensional integral. If each trapezoid rule uses n function evaluations, then the total procedure should require n^{10} function evaluations.

(c) Use your code from part (b) to approximate the integral for $a = 0$, $b = 2$, and

$$f\big(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\big) = \sin\big(x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}\big)$$

for several small values of n (e.g., $n = 3, 4, 5$). Produce a table showing your estimate for the integral, as well as the total error. (The exact integral is approximately 174.467318369179.)

(d) Compare your result from (c) to the Monte Carlo approximation

$$\frac{(b-a)^{10}}{N} \sum_{k=1}^{N} f\big(\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6, \xi_7, \xi_8, \xi_9, \xi_{10}\big)$$

for $f(\xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6, \xi_7, \xi_8, \xi_9, \xi_{10}) = \sin(\xi_1 \xi_2 \xi_3 \xi_4 \xi_5 \xi_6 \xi_7 \xi_8 \xi_9 \xi_{10})$ on $[a, b] = [0, 2]$. Use the same number of total function evaluations that you used for the trapezoid rule in part (c): i.e., take $N = n^{10}$ for the same values of n used in part (c).

Here $\xi_1, \ldots, \xi_{10}$ denote independent, identically distributed random variables uniformly distributed over $[a, b] = [0, 2]$.

(e) Which approach is superior for the 10-dimensional integral? (Include a rough explanation.)