MATH/CS 5466 · NUMERICAL ANALYSIS

Problem Set 2

Posted Friday 12 February 2016. Due Friday 19 February 2016 (5pm).
Students should complete all 4 problems (total of 100 points).

Several of the problems use the *infinity norm* of a function $g \in C[a,b]$, defined by

$$\|g\|_\infty = \max_{a \le x \le b} |g(x)|.$$

This norm defines the usual norm axioms:

(i) $\|g\|_\infty \ge 0$ for all $g \in C[a,b]$ and $\|g\|_\infty = 0$ if and only if $g(x) = 0$ for all $x \in [a,b]$;

(ii) $\|\alpha g\|_\infty = |\alpha| \|g\|_\infty$ for all $g \in C[a,b]$ and all $\alpha \in \mathbb{R}$;

(iii) $\|g+h\|_\infty \le \|g\|_\infty + \|h\|_\infty$ for all $g, h \in C[a,b]$.

1. [25 points]

   In the lectures we discussed the construction of finite difference approximations of differential equations, developing a second-order accurate approximation of the boundary value problem $-u''(x) = g(x)$ for $x \in [0,1]$ with $u(0) = u(1) = 0$.

   For this problem, you will develop a more accurate approximation to this same differential equation. You are welcome to use symbolic computation (Mathematica, WolframAlpha, Maple, etc.) to expedite these calculations, though they can all be worked out by hand if you prefer.

   Throughout, we use the uniformly spaced grid

   $$0 = x_0 < x_1 < \cdots < x_{n-1} < x_n = 1$$

   with $x_j = jh$ for $h = 1/n$.

   (a) Compute the *quartic* (degree 4) polynomial interpolant p_4 to a function $f(x)$ through the five points x_{j-2}, x_{j-1}, x_j, x_{j+1}, and x_{j+2}. (The formula will be intricate. Simplify it as much as possible, so that it only involves x_j, h, and the five function values $f_{-2} \equiv f(x_{j-2})$, $f_{-1} \equiv f(x_{j-1})$, $f_0 \equiv f(x_j)$, $f_1 \equiv f(x_{j+1})$, $f_2 \equiv f(x_{j+2})$.)

   (b) Compute $p_4''(x_j)$ and simplify as much as possible. Your formula should have the form

   $$p_4''(x_j) = \frac{Af_{-2} + Bf_{-1} + Cf_0 + Bf_1 + Af_2}{h^2}$$

   for appropriate constants A, B, and C that you should specify.

   (c) Compute $p_4''(x_{j-1})$ and $p_4''(x_{j+1})$, again simplifying as much as possible. These formulas should have the form

   $$p_4''(x_{j-1}) = \frac{Df_{-2} + Ef_{-1} + Ff_0 + Gf_1 + Hf_2}{h^2}$$

   and

   $$p_4''(x_{j+1}) = \frac{Hf_{-2} + Gf_{-1} + Ff_0 + Ef_1 + Df_2}{h^2}$$

   for appropriate constants D, E, F, G and H (same values in each equation) that you should specify.

   (d) We are now prepared to approximate the solution to the differential equation

   $$-u''(x) = g(x), \qquad u(0) = u(1) = 0.$$

Use the formulas you derived in parts (b) and (c) to construct a linear system of equations

$$\begin{bmatrix} & & \\ & \textit{fill in} & \\ & & \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_{n-2}) \\ g(x_{n-1}) \end{bmatrix}$$

akin to equation (1.24) in the course notes, i.e., each row should encode $p_4''(x_j) = -g(x_j)$ for some $j = 1, \ldots, n-1$. Clearly specify the general form of the entries in the matrix. Most rows will use your answer from part (b), but a few will instead need your answers to part (c). Explain why you choose the particular formulas you use for each value of j.

(e) Produce a MATLAB implementation of your approximation. For $n = 6$:
– write down the 5×5 matrix that your discretization yields;
– edit the code `fd_bvp` on the class website to incorporate your new approximation. The modified code should produce two plots: (i) $u(x)$, the approximation obtained from the quadratic interpolant, and the new approximation obtained from the quartic interpolant.

(f) Edit the code `fd_bvp_conv` on the class website to incorporate your new approximation. You should now produce a `loglog` plot showing

$$\max_{0 \le j \le n} |u(x_j) - u_j|$$

for the approximations with $n = 16, 32, 64, \ldots, 512$ for both the quadratic approximation and the new quartic approximation.
– The quadratic approximation yielded $O(h^2)$ accuracy. What accuracy does the quartic approximation produce? Add a dashed line to your plot to reflect the appropriate convergence rate of the quartic approximation.

(g) (optional: if the first part of the problem was simple for you, please try this extension)
Suppose we change the left boundary condition to $u'(0) = 0$. Discuss how you would implement this boundary condition while maintaining the accuracy of the approximation. Test your method out on the equation $-u''(x) = \cos(\pi x/2)$ with exact solution $u(x) = (4/\pi^2)\cos(\pi x/2)$.

2. [25 points]

Solve these three problems from Gautschi's text.

(a) Consider the piecewise cubic function

$$S(x) = \begin{cases} p(x) & x \in [0, 1]; \\ (2-x)^3 & x \in [1, 2]. \end{cases}$$

Find the cubic p such that $S(0) = 0$ and S is a cubic spline for the knots $x_0 = 0$, $x_1 = 1$, and $x_2 = 2$. Do you get a natural spline?

(b) Consider the set of knots $x_0 < x_1 < \cdots x_n$. One could extend the idea of splines presented in class to give functions S that are polynomials of degree d on each interval $[x_j, x_{j+1}]$, for $j = 0, \ldots, n$ with $S \in C^p[x_0, x_n]$, i.e., S and its first p derivatives are all continuous on $[x_0, x_n]$. What is the dimension of the space of such functions? (That is, how many free variables are left in S, after the continuity conditions are imposed? Do not count any interpolation conditions.)

(c) Let Π denote the linear operator that maps $f \in C[a,b]$ to its *piecewise linear interpolant* at $a = x_0 < x_1 < \cdots < x_n = b$, (i.e., Πf denotes the piecewise linear interpolant to f).

— Show that $\|\Pi g\|_\infty \le \|g\|_\infty$ for any $g \in C[a,b]$.

— Let p_* denote any *piecewise linear polynomial for these interpolation points* $x_0, \ldots, x_n$ (e.g., p_* could minimize $\|f - p\|_\infty$ over all piecewise linear p for these interpolation points). Show that

$$\|f - \Pi f\|_\infty \le 2\|f - p_*\|_\infty.$$

(In other words, the piecewise linear interpolant approximates f to within a factor of 2 of the optimal approximation.)

3. [25 points]

This problem continues the theme of the last problem, but now with standard degree-n polynomial interpolation replacing piecewise linear interpolation.

Let Π_n denote the linear operator that maps $f \in C[a,b]$ to the polynomial p_n that interpolates f at the distinct points $x_0, \ldots, x_n$, $\{x_j\}_{j=0}^n \subset [a,b]$. In other words, $\Pi_n f = p_n$, where p_n is the unique polynomial of degree n (or less) for which $f(x_j) = p_n(x_j)$ for $j = 0, \ldots, n$.

(a) Explain why Π_n is a *projector*: That is, for any $f \in C[a,b]$, show that $\Pi_n(\Pi_n f) = \Pi_n f$. (Hint: What does $\Pi_n p_n$ equal if p_n is a polynomial of degree n?)

This infinity norm induces the *operator norm*

$$\|\Pi_n\|_\infty = \max_{f \in C[a,b], f \neq 0} \frac{\|\Pi_n f\|_\infty}{\|f\|_\infty} = \max_{\|f\|_\infty = 1} \|\Pi_n f\|_\infty.$$

(b) Show that if $x_0 = a$ and $x_1 = b$, then $\|\Pi_0\|_\infty = \|\Pi_1\|_\infty = 1$.

(c) Recall that we can write the polynomial $p_n = \Pi_n f$ in the Lagrange form

$$\Pi_n f = \sum_{j=0}^n f(x_j)\ell_j(x),$$

where ℓ_k denotes the kth Lagrange basis polynomial.

Prove that $\|\Pi_n\|_\infty = \max_{x \in [a,b]} \sum_{j=0}^n |\ell_j(x)|$.

(d) Let p_* denote any polynomial of degree n (e.g., p_* minimizes $\|f - p\|_\infty$ over all $p \in \mathcal{P}_n$). Prove that $\|f - p_n\|_\infty \le (1 + \|\Pi_n\|_\infty)\|f - p_*\|_\infty$.

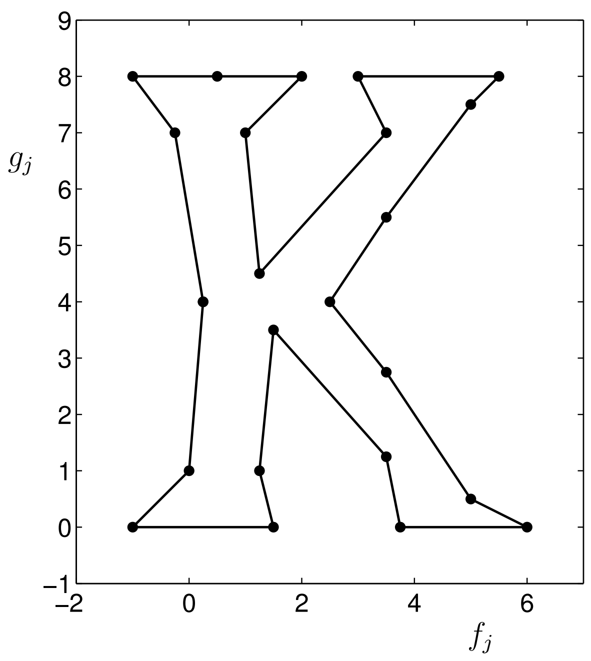
(e) Computationally estimate $\|\Pi_n\|_\infty$ for $n = 1, \ldots, 20$ with (i) uniformly spaced points $x_j = -1 + 2j/n$ and (ii) Chebyshev points $x_j = \cos(j\pi/n)$ over $[-1,1]$.

4. [25 points]

Splines in font design. The font in which this text is set was designed by Donald Knuth using his remarkable METAFONT software. To make appealing letters, the font designer establishes fixed points that guide *Beziér curves*, defined via *Bernstein polynomials.* These curves do not interpolate the guide points, but a similar system based on spline functions, which do interpolate, has also been proposed. Here you will try your hand at spline font design: design a stylized 'K' character using cubic splines with natural boundary conditions. The craft would be the same if you were designing an airplane fuselage or a new sports car.

Consider the following table of data. The (f_j, g_j) values specify the skeleton for our 'K', as shown on the right. Our goal is to replace the straight lines by smooth curves generated from splines.

x_j	f_j	g_j
0	.25	4
1	0	1
2	-1	0
3	1.5	0
4	1.25	1
5	1.5	3.5
6	3.5	1.25
7	3.75	0
8	6	0
9	5	0.5
10	3.5	2.75
11	2.5	4
12	3.5	5.5
13	5	7.5
14	5.5	8
15	3	8
16	3.5	7
17	1.25	4.5
18	1	7
19	2	8
20	0.5	8
21	-1	8
22	$-.25$	7
23	.25	4



(a) Write a MATLAB routine

```
function S = cBspline(x, x0, h)
```

that computes the value of a cubic B-spline at a point `x` $\in \mathbb{R}$, given the initial knot `x0` $\in \mathbb{R}$ and uniform grid spacing `h`, i.e., $x_j = x_0 + j$`h`.

(b) Using your code from part (a), or otherwise, construct two *natural* cubic splines, one, called $S_1(x)$, interpolating the (x_j, f_j) values, the other, called $S_2(x)$, interpolating (x_j, g_j). (Each spline should be the linear combination of $n + 3 = 26$ B-splines. Further details are provided in the lecture notes; the variables f_j and g_j are defined in the MATLAB file `Kdata.m` on the class website.)

(c) Produce a plot showing $S_1(x)$ and $S_2(x)$ for $x \in [0, 23]$, along with the points (x_j, f_j) and (x_j, g_j), to verify that your splines interpolate the data as desired.

(d) In a separate figure, plot $(S_1(x), S_2(x))$ for $x \in [0, 23]$. You should obtain a picture like the skeleton above, but with the straight lines replaced by more interesting curves. Superimpose the (f_j, g_j) points to verify that your splines interpolate the data points.