AMSC/CMSC 666

NUMERICAL ANALYSIS I

HOMEWORK # 2 (Pbs 1-5 due Th 10/3)

1 (10 pts). *Bidiagonalization.* This problem shows how to reduce a square matrix $\mathbf{A}$ of order n to an upper bidiagonal matrix $\mathbf{B}$; this is useful to compute the SVD. Consider the following basic step

$$(\mathbf{P}_1 \cdot \mathbf{A}) \cdot \mathbf{Q}_1 = \begin{bmatrix} x & x & 0 & \cdots & 0 \\ 0 & x & x & \cdots & x \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & x & x & \cdots & x \end{bmatrix}$$

where $\mathbf{P}_1$ and $\mathbf{Q}_1$ are both Householder matrices.

(a) Explain how to construct $\mathbf{P}_1$ and $\mathbf{Q}_1$ and perform an operation count. To do so, exploit the structure of $\mathbf{P}_1$ and $\mathbf{Q}_1$ to avoid matrix-matrix multiplication.

(b) Explain how to construct a sequence of orthogonal matrices $\{\mathbf{P}_i\}_{i=1}^{n-1}$ and $\{\mathbf{Q}_i\}_{i=1}^{n-2}$ such that

$$\mathbf{P}_{n-1} \cdots \mathbf{P}_1 \cdot \mathbf{A} \cdot \mathbf{Q}_1 \cdots \mathbf{Q}_{n-2} = \mathbf{B},$$

and perform an operation count for the entire process.

2 (10 pts). *Roots of Orthogonal Polynomials.* Let $\{p_i\}$ denote the set of monic polynomials (leading coefficient equal to 1) which are orthogonal with respect to a weight ω. They satisfy the three term recursion

$$p_j(x) = (x - \delta_j)p_{j-1}(x) - \gamma_j^2 p_{j-2}(x), \qquad \delta_j = \frac{\langle xp_{j-1}, p_{j-1}\rangle}{\langle p_{j-1}, p_{j-1}\rangle}, \quad \gamma_j^2 = \frac{\langle p_{j-1}, p_{j-1}\rangle}{\langle p_{j-2}, p_{j-2}\rangle},$$

for $1 \leq j \leq n$ where $p_0(x) = 1, p_{-1}(x) = 0$ and $\gamma_1^2 = \langle p_0, p_0\rangle$. Show that the normalized polynomials

$$q_0 = \frac{p_0}{\gamma_1}, \quad q_1 = \frac{p_1}{\gamma_1\gamma_2}, \quad q_2 = \frac{p_2}{\gamma_1\gamma_2\gamma_3}, \cdots, q_n = \frac{p_n}{\gamma_1\gamma_2\gamma_3\cdots\gamma_{n+1}}$$

satisfy the recursion

$$\gamma_j q_{j-2}(x) + \delta_j q_{j-1}(x) + \gamma_{j+1} q_j(x) = x q_{j-1}(x) \quad 1 \leq j \leq n.$$

Write this expression in matrix form using the following tridiagonal matrix T_n and vector $\mathbf{q}_n(x)$

$$T_n = \begin{bmatrix} \delta_1 & \gamma_2 & & & \\ \gamma_2 & \delta_2 & \gamma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-1} & \delta_{n-1} & \gamma_n \\ & & & \gamma_n & \delta_n \end{bmatrix}, \qquad \mathbf{q}_n(x) = \begin{bmatrix} q_0(x) \\ q_1(x) \\ \vdots \\ q_{n-2}(x) \\ q_{n-1}(x) \end{bmatrix}.$$

Let now $x = x_j$ be any root of q_n for $1 \leq j \leq n$. Show that x_j is an eigenvalue of T_n with eigenvector $\mathbf{q}_n(x_j)$. Computing the eigenvalues of T_n is a stable process to find the nodes of Gaussian quadrature rules. It turns out that the corresponding weights ω_j can be computed as well. If $\mathbf{v}_j \in \mathbb{R}^n$ is an eigenvector corresponding to x_j, i.e. $\mathbf{v}_j$ is proportional to $\mathbf{q}_n(x_j)$, then $\omega_j = \frac{(v_j^1)^2}{\|\mathbf{v}_j\|_2^2} \int_a^b \omega(x)dx$ where v_j^1 is the first component of $\mathbf{v}_j$ and ω is the weight of the desirable integral.

3 (15 pts). *Lobatto quadrature.* This is a gaussian rule for integrating $I(f) = \int_{-1}^1 f(x)dx$ except that it includes ± 1 as two *preassigned* abscissas. It has the form

$$I[f] \approx Q_n[f] = \omega_0 f(-1) + \omega_1 f(x_1) + \cdots + \omega_n f(x_n) + \omega_{n+1} f(1),$$

with nodes x_i and weights w_i chosen so as to maximize the order of the integration method. Counting degrees of freedom we expect the formula to be exact for polynomials of degree $\leq 2n + 1$. This problem explains how to determine the nodes and weights for any n and find these quantities for $n = 2$.

(a) Show that any polynomial $p \in \mathbb{P}_{2n+1}$ can be written as $p(x) = \ell(x) + (1 - x^2)q(x)$ where $q \in \mathbb{P}_{2n-1}$ and $\ell \in \mathbb{P}_1$ is the linear interpolant of p at the nodes ± 1.

(b) Let $p_n \in \mathbb{P}_n$ be an orthogonal polynomial to $\mathbb{P}_{n-1}$ with the scalar product $\langle f, g\rangle = \int_{-1}^{1} f(x)g(x)(1-x^2)dx$. Determine the nodes $\{x_i\}_{i=1}^n$ and weights $\{w_i\}_{i=1}^n$ for a Gauss quadrature $\widehat{Q}_n[f] = \sum_{i=1}^n w_i f(x_i)$ that approximates the integral $\widehat{I}[f] = \int_{-1}^{1} f(x)(1-x^2)dx$ and is exact for $f \in \mathbb{P}_{2n-1}$.

(c) Exploit the relation

$$I[p-\ell] = \widehat{I}[q] = \widehat{Q}[q] = \sum_{i=1}^{n} w_i \frac{p(x_i) - \ell(x_i)}{1 - x_i^2} = Q[p] - Q[\ell]$$

to determine the weights $\{\omega_i\}_{i=1}^n$ and impose the condition $Q[\ell] = I[\ell]$ to find the weights ω_0, ω_{n+1}.

(d) Let $f \in C^{2n+2}[-1,1]$. Show the error estimate

$$\Big| I[f] - Q_n[f] \Big| \le \frac{C}{(2n+2)!} \|f^{2n+2}\|_\infty.$$

(e) Take $n = 2$, compute explicitly p_0, p_1, p_2 and show that

$$x_1 = -\frac{1}{\sqrt{5}}, x_2 = \frac{1}{\sqrt{5}}, \quad \omega_0 = \omega_3 = \frac{1}{6}, \quad \omega_1 = \omega_2 = \frac{5}{6}.$$

4 (15 pts). *Fast Fourier Transform and Denoising.* This problem shows how a noisy signal can be transformed to the frequency domain via the Fast Fourier Transform (FFT), clean via thresholding of the smallest coefficients, and transform back to obtain a signal with less noise.

(a) Given the equally spaced 2^8 sampling points `x = [1:256]*2*pi/256` in the interval $[0, 2\pi]$, consider the function values `y = sin(5*x)` and the *noisy* function values `z = sin(5*x) + 0.1*randn(256,1)'`. The command `randn(256,1)` generates a normal distribution of 256 random numbers with zero mean and variance one. Use `plot(x,y,x,z)` to plot both functions.

(b) Compute `f = fft(z)`. Write a MATLAB function `t = thresh(f,a)` which computes the modulus of each component of `f` using the commands `conj` and `sqrt`, and then zeros all entries with values `< a`.

(c) Compute `s = ifft(t)`, the inverse FFT of `t`, for `a = 2, 3, 4`. Plot the three cases using `plot(x,s)` and draw conclusions.

5 (10 pts). *Discrete convolution.* Let $\Pi_N = \left\{(a_i)_{i=-\infty}^{\infty} : a_i \in \mathbb{C}, a_{i+N} = a_i\right\}$ denote the space of bi-infinite N-periodic complex sequences. If $\mathbf{a}, \mathbf{b} \in \Pi_N$, let the convolution $\mathbf{c} = \mathbf{a} \star \mathbf{b}$ be defined by

$$c_k = \sum_{j=0}^{N-1} a_j b_{k-j} \quad 0 \le k \le N-1.$$

(a) Perform an operation count (multiplication only) to show that this process takes $O(N^2)$ flops.

(b) Prove that the discrete Fourier transform converts convolution into multiplication, namely $(\mathcal{F}_N \mathbf{c})_k = (\mathcal{F}_N \mathbf{a})_k (\mathcal{F}_N \mathbf{b})_k$ for all $0 \le k \le N-1$. Perform an operation count for the procedure that first computes $\mathcal{F}_N \mathbf{c}$ and next finds $\mathbf{c}$ upon inverting $\mathcal{F}_N$. Draw conclusions.