Math 465A, Computer Projects

Regarding computer projects:

- You may use your own machine or your MSCC account, but all computations should be done in double precision (about 16 digits of accuracy). All projects can be done by writing Matlab function files. However you may not use a Matlab routine such as fft in place of your own routine unless explicitly given permission.
- Interpret your results:
  - Observe what happened.
  - Explain what happened in light of the theory we have studied.
- Turn in your write-up of your interpretations, your program listings, and your program output (in that order), attached together, at the beginning of class on the due date.
- Please start early. No late sets will be accepted. Leave time to interpret your results. Notice how crowded the assignments are toward the end of the quarter.

---

1 Inverse Power Method – due January 25

Write a program that uses the shifted inverse power method to find an approximation to the eigenvalue of a matrix which is closest to a given number α and the corresponding eigenvector. The program should take as inputs a limit, M on the number of iterations and a tolerance ϵ (<u>not</u> necessarily the machine "epsilon"). It should also take as input an initial approximation, v, to the eigenvector. The program should terminate when successive estimates of the eigenvalue differ by less than ϵ or when the number of iterations reaches m. The program should use Rayleigh quotients to estimate the eigenvalue. Run the program on the following matrices and find all of their eigenvalues. The first digits of the eigenvalues are given. In each case use $v = [1, \sqrt{2}, \sqrt{3}, \ldots, \sqrt{n}]^T$ as initial guess.

$$\begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} 100 & 20 & 0 & 0 \\ 20 & 110 & 16 & 0 \\ 0 & 16 & 120 & -8 \\ 0 & 0 & -8 & 130 \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} 4 & 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 0 & 0 & 0 & 1 & 4 \end{bmatrix} \tag{3}$$

First digits of the eigenvalues:

(a) 3, 3, 2, 1, .7, .1

(b) 1e2, 1e2, 1e2, 8e1

(c) 5, 5, 4, 3, 2

2 Bairstow's Method – due February 17

Write a program to do Bairstow's method. The program should take as input and integer n and a vector a of dimension $n+1$ representing a polynomial of degree n. It should also take as input a tolerance tol, a maximum number of iterations it and an initial guess u, v for a quadratic factor $x^2 - ux - v$. The program should return c, d where $x^2 - cx - d$ is your final estimate for a quadratic factor. The program should terminate when either $\|(u_i - u_{i-1}, v_i - v_{i-1})\| < tol$ or $i \geq it$. Find real quadratic factors of $x^4 + x^2 + 1$ and $x^3 + x^2 - 2$.

3 Orthogonal Polynomials – due March 1

Write a program which takes as inputs an n, w, x, y, m, d where n is the number of data points, w is an n dimensional vector of weights, the data points are (x_i, y_i), $i = 1, \ldots, n$, $m + 1$ is the number of evaluation points and d is the maximum degree of the set of orthogonal polynomials. The program should generate polynomials $q_j(x), j = 0, \ldots, d$ which are orthogonal with respect to the inner product $< f, g >= \sum_1^n w_i f(x_i) g(x_i)$. These polynomials should be calculated according to the three term recurrence relation described on pages 268-270. The program should compute the "Fourier" coefficients $c_k =< y, q_k > / < q_k, q_k >$ of the "function" y and the values of the approximation $p(x) = \sum_0^d c_k q_k(x)$ at $m+1$ equally spaced points in the interval $[x_1, x_n]$.

Run your program with the following data and plot the approximation p.

$n = 11,\ m = 100,$
$w = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),$
$x = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10),$
$y = (3, 2, 3, 5, 3, 4, 3, 2, 2, 3, 2),$
$d = 2, 5, 10,$ (one run for each value of d)

4 Fast Fourier Transform – due March 12

Write a program to compute the discrete Fourier transform of 2^m points using the fast Fourier transform algorithm. Let $\mathcal{F}(x)$ denote the discrete Fourier transform of the vector x. Use the program to compute $\mathcal{F}(x)$ for the following vectors:

(a) $x = (1, 2, 3, 4, 5, 6, 7, 8)$

(b) $\mathcal{F}(x)$, where $x = (1, 2, 3, 4, 5, 6, 7, 8)$

(c) $x = (1, -1, 2, -2, 3, -3, 4, -4)$

I will give a handout describing an FFT algorithm.