Math 465, Computer Projects

Latest revision as of December 22, 1999

Introduction

Regarding computer projects:

- The projexts are meant to give you experience with
  - programming numerical routines,
  - learning to relate computational and theoretical results
  - developing a report writing style which communicates in an organized manner:
    * the problem considered
    * the methods used
    * the results obtained
    * the conclusions or interpretations reached.
- For the programming portion of the project you may use your own machine or your MSCC or other UW account. All computations should be done in double precision with a machine epsilon or unit roundoff giving about 16 digits of accuracy. All projects can be done by writing Matlab or Splus script and function files, taking advantage of the high level matrix and vector operations available in these languages. Other programming language (C, C++, Fortran) main and function or subroutine programs can also be used to for the calculational part of the projects. However you may not use a Matlab routine such as eig or fft or a library routine such as linpack dgefa in place of your own routine unless I explicitly give you permission.

- Any plotting required may be done directly in the main Matlab or Splus script/function, if you have used one of those languages for the calculations. You may also use any other plotting package, such as Excel (or gnuplot available on Saul), but you should document how you tranfer data from your computational program to the plotting program used, so any effects of the data transfer on accuracy can be understood.
- In your report you must include all the points described above.
- Your results and interpretations must include:
  - Observation/description of what happened when the program was run.
  - Explaintion of what happened in light of the theory we have studied.
- Your report should also include (probably as appendixes referred to at various points in your report)
  - Your program listings.
  - Your program output.
- The report is due at the beginning of class on the due date. Remember any written work must be coherent; the point of any report is to explain not just give answers.
- Please start early. Leave time to interpret your results.

1 Inverse Power Method – due Monday, January 24

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. In addition to the matrix A and the number α, the program should take as inputs a limit, m, on the number of iterations and a tolerance ϵ (not necessarily the machine "epsilon") whose role is explained below. 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. The program should return the final estimate of the eigenvalue, the corresponding eigenvector estimate and information as to whether the method converged.

An optional addition to the return values of the program would be a list (vector) with all the (Rayleigh quotient) estimates of the eigenvalue, and a matrix with the corresponding eigenvector extimates. Two optional additional programs and or additions to the program would be

a) An implementation of Richardson extrapolation to produce an improved eigenvalue estimate together with an upper bound on the absolute error in the improved estimate. (This should not be restricted to symmetric matrices, even though the examples below are symmetric.)

(ii) An adaptation of the discussion on p.94 of Johnson and Reiss, and Problem 7, in section 3.2 of JR, to use ratios of successive differences in eigenvalue estimates for symmetric matrices to generate estimates of the eigenvalue which is second closest to α. (The adaptation of Problem 7 to Rayleigh quotient eigenvalue estimates for **symmetric** A will involve $(\lambda_2/\lambda_1)^{2i}$. The further adaptation to the shifted inverse power method is then a question of algebra. Any analysis/discussion associated with this addition should explain/derive the method/formulae used.)

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: (Be sure to explain how you have used this information in relation to the theory of the convergence of the shifted inverse power method and the general theory of eigenvalues to assure that you have obtained all the eigenvalues for the given matrices.)

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

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

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

2 Bairstow's Method – due Monday, February 14

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 Monday, February 28

Write a program to compute

1. The orthogonal polynomials with respect to a discrete weighted inner product on a data set of x_i.
2. The expansion coefficients of a least squares polynomial approximation to a given function based on these polynomials and the data values $y_i = f(x_i)$.
3. The values at a given number of equally spaced points of any least squares polynomial based on these orthogonal polynomials.

The inputs to the overall program should be 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 subprogram for (1) 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 subprogram (2) should compute the "Fourier" coefficients $c_k =< y, q_k > / < q_k, q_k >$ of the "function" y. And the subprogram (3) should calculate 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.

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

Also run and plot the approximations when $y = 1/(1+x^2)$. (The $d = 10$ case is a polynomial interpolant (Why?) and should remind you of one of the polynomial interpolants plotted last quarter.)

4 Fast Fourier Transform – due Friday, March 10

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.