CS450: Numerical Analysis
Fall 2015

Homework 4

University of Illinois, Urbana-Champaign, Prof. Paul Fischer

Due: October 16, 2015 (Friday), 5.00 pm. Submit PDF + code files on Moodle.

Problem 1.

Write a program implementing Lanczos iteration as given in Section 4.5.7. Test your program using a random real symmetric matrix A of order n having eigenvalues 1, 2, ..., n. To generate such a matrix, first generate an $n \times n$ matrix B with random entries uniformly distributed on the interval [0,1). Then compute the QR factorization $B = QR$. Next compute $A = QDQ^T$, where $D = \text{diag}(1, ..., n)$.

The Lanczos algorithm generates only the tridiagonal matrix T_k at iteration k, so you will need to compute its eigenvalues (i.e. the Ritz values γ_i) at each iteration using a library routine. Run the Lanczos algorithm for the full n iterations for this assignment.

To see graphically how the Ritz values behave as iterations proceed, construct a plot with the iteration number on the vertical axis and the Ritz values at each iteration on the horizontal axis. Plot each pair (γ_i, k), $i = 1, ..., k$, as a discrete point at each iteration k similar to figure 4.4. As iteration proceed and the number of Ritz values grows correspondingly, you should see vertical "trails" of Ritz values converging on the true eigenvalues.

Use $n = 20$ for this assignment. You may use library routines to help compute the random real symmetric matrix A. You may also use library routines to compute the eigenvalues of the tridiagonal matrix at each iteration.

Coding notes: You may find the following functions useful for this problem: For Python `scipy.linalg.qr` and `scipy.linalg.eig`. For Matlab `qr()` and `eig()`.

Problem 2.

Let $A =$

$$\begin{bmatrix} -261 & 209 & -49 \\ -530 & 422 & -98 \\ -800 & 631 & -144 \end{bmatrix} \tag{1}$$

(a) Use 10 QR iterations to find the (approximate) eigenvalues of A and plot the absolute value of the lower triangular entries using a logarithmic scaling of the y-axis. Your plot should contain 3 lines with 1 line per nonzero in the strictly lower triangular portion of A depicting the changes in the magnitude of that nonzero between iterations.

(b) Repeat the previous process using shifted QR iterations using the median eigenvalue approximated from part (a).

(c) Compare the approximate eigenvalues from part (a) and (b) after 2 iterations with the exact eigenvalues found using library function.

Coding notes: You may find the following functions useful for this problem: `np.linalg.eigvals` and `plt.semilogy` for Python, `eig` and `semilogy` for Matlab.

Problem 3.

For the equation $f(x) = x^2 - 3x + 2 = 0$, each of the following functions yields an equivalent fixed-point problem:

$$\begin{aligned} g_1(x) &= (x^2 + 2)/3 \\ g_2(x) &= \sqrt{3x - 2} \\ g_3(x) &= 3 - 2/x \\ g_4(x) &= (x^2 - 2)/(2x - 3) \end{aligned} \tag{2}$$

(a) Analyze the convergence properties of each of the corresponding fixed-point iteration schemes for the root $x = 2$ by considering $|g_i'(2)|$.

(b) Confirm your analysis by implementing each of the schemes and verifying its convergence (or lack thereof) and approximate convergence rate.

Problem 4.

Implement the bisection, Newton, and secant methods for solving nonlinear equations in one dimension, and test your implementations by finding at least one root for each of the following equations. What termination criterion should you use? What convergence rate is achieved in each case? Verify that your implementation has the correct converging behavior as you expect. Compare your solutions with those for a library routine for solving nonlinear equations. (You don't have to find one-to-one comparisons for all three methods.)

(a) $x^3 - 2x - 5 = 0$
(b) $e^{-x} = x$
(c) $x \sin x = 1$
(d) $x^3 - 3x^2 + 3x - 1 = 0$

Coding notes: You may find the following library functions useful: `scipy.optimize.bisect` or `scipy.optimize.newton` for Python, `fzero()` or `fsolve()` for Matlab.