

ACM 106a, Problem Set 4

Due: Thursday, November 29, 2007

Theory

1. Do Stoer & Bulirsch exercise 4.3 (p. 281).
2. Supposing you already know A^{-1} , the previous problem tells you how to compute the inverse $(A + uv^T)^{-1}$ of a slightly modified matrix, without going through the expensive inversion process again. In pseudocode, give an algorithm for computing this inverse. What is the complexity of your algorithm, i.e., how many operations are required for an $n \times n$ matrix? How does this compare to inverting a general matrix?
(This is called a *rank 1 update formula*, because it lets you update the inverse when small changes are made to a matrix, such as replacing a row or column. Note that, although this was discussed in class for updating the matrix *factorization*, updating the *inverse* is a different problem.)
3. Do Stoer & Bulirsch exercise 4.5 (p. 282).
4. The previous problem describes the *singular value decomposition* $U^T AV = D$ for a real, nonsingular, square matrix A . In pseudocode, give an algorithm to compute the orthogonal matrices U and V , and the diagonal matrix D . (Hint: use Householder transformations to bring A to bidiagonal form, i.e., nonzero entries on the diagonal and upper band only. Then perform additional transformations to get a diagonal matrix.)
5. Do Stoer & Bulirsch exercise 4.9 (p. 284).

How to write pseudocode. Problems 2 and 4 ask you to write *pseudocode* for your algorithms. This should specify the general steps in the algorithms fully, mostly in English, and without going into the kind of detail necessary in actual computer code (such as detailed indices, variable names, etc.). Then describe the complexity of each of your higher-level steps. Here's a sample pseudocode for the Gaussian elimination method with partial pivoting by rows:

for each column $1, \dots, n-1$, do the following:

 find the largest column entry on or below the diagonal, and make it the pivot

 perform row operations to zero out all column entries below the diagonal

This algorithm loops $O(n)$ times. For each loop, finding the largest pivot takes $O(n)$ operations. Eliminating the entries below the diagonal takes $O(n)$ row operations, and each row has $O(n)$ entries, for a total of $O(n^2)$ operations. Therefore, the total number of operations is $O(n^3)$. (Note that partial pivoting only takes $O(n^2)$ steps, so its cost is negligible.)