CME 302: NUMERICAL LINEAR ALGEBRA
FALL 2005/06
LECTURE 7

GENE H. GOLUB

1. Computing the Inverse

Using the LU decomposition, one can compute the inverse of a matrix. A natural method to compute the inverse of an $n \times n$ matrix A is to solve the matrix equation

$$AX = I$$

by solving the systems of equations

$$A\mathbf{x}_j = \mathbf{e}_j, \quad j = 1, \ldots, n.$$

Since only the right-hand side is different in each of these systems, we need only compute the LU decomposition of A once, which requires $2n^3/3$ operations. For simplicity, we assume that pivoting is not required, and note that the case where pivoting is required can be handled in a similar fashion.

Given the LU decomposition, we compute A^{-1} by solving the systems $L\mathbf{y}_j = \mathbf{e}_j$ and $U\mathbf{x}_j = \mathbf{y}_j$ for $j = 1, \ldots, n$. Computing each column $\mathbf{x}_j$ of A^{-1} requires n^2 operations, resulting in a total of $2n^3/3 + n(n^2) = 5n^3/3$ operations.

We can compute A^{-1} more efficiently by noting that $A^{-1} = U^{-1}L^{-1}$ and computing U^{-1}, L^{-1}, and the product $U^{-1}L^{-1}$ directly. Since the inverse of an upper triangular matrix is also an upper triangular matrix, computing column j of U^{-1} requires only approximately $j^2/2$ operations, since, in solving the system $U\mathbf{x}_j = \mathbf{e}_j$, we can ignore the last $n - j$ components of $\mathbf{x}_j$ since we know that they are equal to zero. As a result, computing U^{-1} requires only $n^3/6$ operations. A similar result holds for computing L^{-1}, which is a lower triangular matrix.

To compute the product $A^{-1} = U^{-1}L^{-1}$, we note that if we number the northeast-to-southwest diagonals of A^{-1} starting with 1 for the upper left diagonal (the (1,1) element) and n for the lower right diagonal (the (n, n) element), then elements along diagonal j require only $n - j + 1$ multiplications to compute. It follows that the total operation count to compute the product of U^{-1} and L^{-1} is

$$(2n - 1) + 2(2n - 3) + 3(2n - 5) + \cdots + (n - 1)2 + n \approx \frac{1}{3}n^3.$$

Therefore, the overall operation count to compute A^{-1} using this method is $4n^3/3$.

2. The Simplex Method

In order to implement the Simplex Algorithm, it is necessary to solve three systems of linear equations at each iteration; namely

$$Bx = b \tag{2.1}$$

$$B^\top w = \tilde{c} \tag{2.2}$$

$$Bt^{(r)} = -a^{(r)} \tag{2.3}$$

Date: November 6, 2005, version 1.1.

Notes originally due to James Lambers. Edited by Lek-Heng Lim.

If the LU decomposition of B is known, then it is easy to solve the three systems of equations. We have already shown how to solve systems (2.1) and (2.3), using the LU decomposition. Since $B^\top = U^\top L^\top$, solving (2.2) merely requires the solving of $U^\top y = \tilde{c}$ and then $L^\top w = y$.

In the Simplex Algorithm we change only one column of B at a time. If the LU decomposition of B is known, we can determine the LU decomposition of the new matrix B by simply updating the previous decomposition. This process can be done efficiently and in a manner that insures numerical stability.

Suppose Gaussian elimination with partial pivoting has been used on B so that

$$P^{(m-1)}\Pi^{(m-1)} \cdots P^{(1)}\Pi^{(1)}B = U.$$

Let

$$B = [b^{(1)}, b^{(2)}, \ldots, b^{(m)}] \quad \text{and} \quad U = [u^{(1)}, u^{(2)}, \ldots, u^{(m)}].$$

Because the last $(m-k)$ components of $u^{(k)}$ are zero,

$$P^{(k+1)}\Pi^{(k+1)}u^{(k)} = u^{(k)}$$

since $P^{(k+1)}\Pi^{(k+1)}$ linearly combines the bottom $(m-k)$ elements of $u^{(k)}$. Thus,

$$u^{(k)} = P^{(k)}\Pi^{(k)}P^{(k-1)}\Pi^{(k-1)} \cdots P^{(1)}\Pi^{(1)}b^{(k)}.$$

If we let

$$\bar{B} = [b^{(1)}, b^{(2)}, \ldots, b^{(s-1)}, g, b^{(s+1)}, \ldots, b^{(m)}],$$

and

$$T^{(k)} = P^{(k)}\Pi^{(k)} \cdots P^{(1)}\Pi^{(1)}.$$

Then

$$\begin{aligned} T^{(s-1)} &= [T^{(1)}b^{(1)}, \ldots, T^{(s-1)}b^{(s-1)}, T^{(s-1)}g, T^{(s-1)}b^{(s+1)}, \ldots, T^{(s-1)}b^{(m)}] \\ &= [u^{(1)}, \ldots, u^{(s-1)}, T^{(s-1)}g, T^{(s-1)}b^{(s+1)}, \ldots, T^{(s-1)}b^{(m)}]. \end{aligned}$$

Therefore, to find the new LU decomposition of $\bar{B}$ we need only compute $\bar{\Pi}^{(s)}$, $\bar{P}^{(s)}$, ..., $\bar{\Pi}^{(m-1)}$, $\bar{P}^{(m-1)}$ so that

$$\bar{P}^{(m-1)}\Pi^{(m-1)} \cdots \bar{P}^{(s)}\bar{\Pi}^{(s)}T^{(s-1)}[g, b^{(s+1)}, \ldots, b^{(m)}] = [\bar{u}^{(s)}, \bar{u}^{(s+1)}, \ldots, \bar{u}^{(m)}],$$

where $\bar{u}^{(k)}$ is a new vector whose last $(m-k)$ components are zero. If g replaces $b^{(m)}$, then about $m^2/2$ multiplications are required to compute the new $\bar{U}$. However, if g replaces $b^{(1)}$, the decomposition must be completely recomputed.

We can update the LU decomposition in a more efficient manner which unfortunately requires more storage. Let us write

$$B_0 = L_0U_0.$$

Let the column s_0 of B_0 be replaced by the column vector g_0. As long as we revise the ordering of the unknowns accordingly we may insert g_0 into the last column position, shifting columns s_0+1 through m of B_0 one position to the left to make room. We will call the result B_1, and we can easily check that it has the decomposition

$$B_1 = L_0H_1,$$

where H_1 is a matrix that is *upper Hessenberg* in its last $m - s_0 + 1$ columns, and upper-triangular in its first $s_0 - 1$ columns.

The first $s_0 - 1$ columns of H_1 are identical with those of U_0. The next $m - s_0$ are identical with the last $m - s_0$ columns of U_0, and the last column of H_1 is the vector $L_0^{-1}g_0$.

H_1 can be reduced to upper-triangular form by Gaussian elimination with row interchanges. Here, however, we need only concern ourselves with the interchanges of pairs of adjacent rows. Thus, U_1 is gotten from H_1 by applying a sequence of simple transformations:

$$U_1 = P_1^{(m-1)}\Pi_1^{(m-1)} \cdots P_1^{(s_0)}\Pi_1^{(s_0)}H_1 \tag{2.4}$$

where each $P_1^{(k)}$ is the identity matrix with a single nonzero subdiagonal element $g_k^{(1)}$ in the $(k+1, k)$ position, and each $\Pi^{(k)}$ is either the identity matrix or the identity matrix with the kth and $(k+1)st$ rows exchanged, the choice being made so that $|g_k^{(1)}| \leq 1$.

The essential information in all of the transformations can be stored in $m - s_0$ locations plus an additional $m - s_0$ bits (to indicate the interchanges). If we let

$$L_1^{-1} = P_1^{(m-1)} \Pi_1^{(m-1)} \cdots P_1^{(s_0)} \Pi_1^{(s_0)} L_0^{-1},$$

then we have achieved the decomposition

$$B_1 = L_1 U_1.$$

The transition from B_1 to B_{i+1}, where i represents the ith time through steps (2)-(7) of the Simplex Algorithm, is to be made exactly as the transition from B_0 to B_1. Any system of linear equations involving the matrix B_i for any i is to be solved by applying the sequences of transformations defined by (2.4) and then solving the upper triangular system of equations.

As we have already pointed out, it requires

$$m^3/3 + O(m^2)$$

multiplication-type operations to produce an initial LU decomposition,

$$B_0 x = v.$$

The solution for any system $B_i x = v$ must be found according to the LU decomposition method by computing

$$y = L_i^{-1} v, \tag{2.5}$$

followed by solving

$$U_i x = y. \tag{2.6}$$

The application of L_0^{-1} to v in (2.5) will require $m(m-1)/2$ operations. The application of the remaining transformations in L_i^{-1} will require at most $i(m-1)$ operations. Solving (2.6) costs $m(m+1)/2$ operations. Hence, the cost of (2.5) and (2.6) together is not greater than

$$m^2 + i(m-1)$$

operations, and a reasonable expected figure would be $m^2 + \frac{i}{2}(m-1)$.

3. Gauss-Jordan Elimination

A variant of Gaussian elimination is called *Gauss-Jordan elimination.* It entails zeroing elements above the diagonal as well as below, using elementary *row* operations (cf. LDU factorization). The result is a decomposition $A = LM^\top D$, where L is a unit lower triangular matrix, D is a diagonal matrix, and M is also a unit lower triangular matrix. We can then solve the system $A\mathbf{x} = \mathbf{b}$ by solving the systems

$$L\mathbf{y} = \mathbf{b}, \quad M^\top \mathbf{z} = \mathbf{y}, \quad D\mathbf{x} = \mathbf{z}.$$

The benefit of Gauss-Jordan elimination is that it maintains full vector lengths throughout the algorithm, making it particularly suitable for vector computers.

4. *LDU* Factorization

A variant of LU factorization is called LDU factorization. It entails zeroing elements above the diagonal as well as below, using elementary *column* operations (cf. Gauss-Jordan elimination) that are similar to the elementary row operations used in Gaussian elimination. The result is a decomposition $A = LDU$, where L is a unit lower triangular matrix, D is a diagonal matrix, and U is also a unit upper triangular matrix. We can then solve the system $A\mathbf{x} = \mathbf{b}$ by solving the systems

$$L\mathbf{y} = \mathbf{b}, \quad D\mathbf{z} = \mathbf{y}, \quad U\mathbf{x} = \mathbf{z}.$$

5. Uniqueness of the LU Decomposition

It is natural ask whether the LU decomposition is unique. To determine this, we assume that A has two LU decompositions, $A = L_1U_1$ and $A = L_2U_2$. From $L_1U_1 = L_2U_2$ we obtain $L_2^{-1}L_1 = U_2U_1^{-1}$. The inverse of a unit lower triangular matrix is a unit lower triangular matrix, and the product of two unit lower triangular matrices is a unit lower triangular matrix, so $L_2^{-1}L_1$ must be a unit lower triangular matrix. Similarly, $U_2U_1^{-1}$ is an upper triangular matrix. The only matrix that is both upper triangular and unit lower triangular is the identity matrix I, so we must have $L_1 = L_2$ and $U_1 = U_2$.

Department of Computer Science, Gates Building 2B, Room 280, Stanford, CA 94305-9025
E-mail address: `golub@stanford.edu`