AMSC 660 Assignment 5

Jiaqi Leng

October 3, 2019

1 Problem 1

(a) Suppose we have an upper triangular matrix $U = (u_{ij})$ for $1 \leq i, j \leq n$, and $b = (b_1, ..., b_n)^T$. We want to solve $Ux = b$ by backward substitution. The algorithm is: for $k = n, n-1, ..., 1$,

$$x_k = \frac{b_k - \sum_{i=k+1}^{n} u_{ki} x_i}{u_{kk}}, \ (u_{n,n+1} = x_{n+1} = 0).$$

At the $k-$th step, there are $(n-k)$ multiplications, $(n-k)$ subtractions, and 1 division, i.e., $2(n-k)+1$ flops, so the total number of flops is

$$\sum_{k=1}^{n} 2(n-k) + 1 = 2n^2 + n - 2(1 + 2 + ... + n) = n^2.$$

The asymptotic number of flops is $O(n^2)$.

(b) There are two ways to perform the matrix product $x^T y z^T$:

(i) $(x^T y) z^T$: $x^T y = \sum_{k=1}^{n} x_i y_i =: c$. There are n multiplications, and $(n-1)$ additions, in total $(2n-1)$ flops. Then $cz^T = (cz_i)_{1 \leq i \leq n}$, i.e., there are n more flops. In total, the number of flops is $2n - 1 + n = 3n - 1$.

(ii) $x^T (y z^T)$: $yz^T = (y_i z_j)_{1 \leq i,j \leq n}$. There are n^2 multiplications. Then, $x^T(yz^T) = (\sum_{i=1}^{n} x_i (y_i z_j))_{1 \leq j \leq n}$, there are n^2 multiplications, $n(n-1)$ additions. In total, the number of flops is $n^2 + n^2 + n(n-1) = 3n^2 - n$.

It is clear that the method (i) is more efficient to perform the matrix multiplication $x^T y z^T$.

(c) In class, the formula for L is $(1 \leq i, j \leq n)$

$$L = \begin{cases} 1 & (i = j) \\ m_{ij} & (i > j) \\ 0 & (\text{otherwise}). \end{cases}$$

Recall that L is defined to be the the inverse of $M_{n-1}M_{k-2}...M_1$, so it suffices to check $LM_{n-1}M_{k-2}...M_1 = I$.

Since M_k is

$$M_k = \begin{cases} 1 & (i = j) \\ -m_{ik} & (i > k) \\ 0 & (\text{otherwise}), \end{cases}$$

we can evaluate the product

$$LM_{n-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ m_{21} & 1 & ... & 0 & 0 \\ ... & ... & ... & ... & ... \\ m_{n1} & m_{n2} & ... & m_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & ... & 0 & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & -m_{n,n-1} & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ m_{21} & 1 & ... & 0 & 0 \\ ... & ... & ... & ... & ... \\ m_{n-1,1} & ... & ... & 1 & 0 \\ m_{n1} & m_{n2} & ... & 0 & 1 \end{pmatrix}.$$

The $(n-1)-$th column is transformed to $(0, ..., 1, 0)$, only having a unit entry on its diagonal. And inductively, $LM_{n-1}...M_k$ returns a matrix with the last $n-k+1$ columns being the last $n-k+1$ standard basis in $\mathbb{R}^n$. It follows that $LM_{n-1}M_{k-2}...M_1 = I$.

2 Problem 2

(a) To reduce $AB = I$ to be $UB = L^{-1}$, it is equivalent to perform a LU factorization. The algorithm is:

For $i = 1 : n-1$:

For $j = i+1 : n$:

$l_{ji} = -a_{ji}/a_{ii}$, for $k = i : n$, $u_{jk} = a_{jk} + l_{ji}a_{ik}$. This algorithm returns

$L^{-1} = (l_{ij})$, $U = (u_{ij})$, and the total number of flops is the same as the LU decomposition:

$$W(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (1 + \sum_{k=i}^{n} 2) \approx \frac{2}{3} n^3.$$

(b) We want to use the backward substitution to solve $UB = L^{-1} = (c_1, ..., c_n)$, with c_k the $k-$th column of L^{-1}. It is equivalent to find the $k-$th column b_k of B such that $Ub_k = c_k$. In problem 1(a), we found the flops of backward substitution is n^2, and here we actually do n times substitution, so the total cost is n^3.

(c) To compute A^{-1}, we need to perform the LU decomposition in part (a) and the backward substitution in part (b), the total cost is $\frac{2}{3}n^3 + n^3 = \frac{5}{3}n^3$.

On the other hand, if we solve $Au = b$ by LU factorization, the algorithm is: (i) $A = LU$; (ii) solve $Ly = b$ by forward substitution; (iii) solve $Uu = b$ by backward substitution. The cost for forward substitution can be in a similar way as what we did in problem 1(a), so the costs of (ii) and (iii) are both n^2. The total cost of this algorithm is $\frac{2}{3}n^3 + 2n^2 \approx \frac{2}{3}n^3$ when n is large.

Now, it is clear that the cost of computing A^{-1} is $\frac{5}{3}n^3$, more than twice as expensive as $\frac{2}{3}n^3$, the cost of solving $Au = b$.