CME 108/MATH 114
Introduction to Scientific Computing
Summer 2018

Problem Set 3 – Solution

Due: Friday, July 20 at 11:59pm

Warm-up Questions

The "warm-up" questions do not need to be submitted (and won't be graded). However, I *highly* encourage you to work out their solutions! I'll post answers along with the solutions to the assigned problems.

Let A be an $n \times n$ invertible matrix. Note that the product AA is defined, and so is AAA. In general, for any positive integer k, set $A^k = A \cdots A$, the product on the right containig k factors (e.g. $A^4 = AAAA$).

1. Show that if $A^2 = I$ then $A^{-1} = A$.

   Solution

   The uniqueness of inverse implies that if $AA = I$ then $A^{-1} = A$.

2. If $6A^3 - 2A^2 + 4A - 2I = 0$, what is A^{-1}?

   Solution

   To find A^{-1}, add $2I$ to both sides of the equation, and then multiply both sides by $1/2A^{-1}$ to obtain
   $$A^{-1} = 3A^2 - A + 2I.$$

3. Show that $(A^{-1})^{-1} = A$.

   Solution

   For clarity, let $B = A^{-1}$. Note that, by definition, $BA = I$. Uniqueness of inverse implies $B^{-1} = A$. Hence $(A^{-1})^{-1} = A$.

4. *(LU verification, Bradie 3.5.5.)* Let

$$B = \begin{bmatrix} 2 & 7 & 5 \\ 6 & 20 & 10 \\ 4 & 3 & 0 \end{bmatrix}.$$

Verify that each of the following pairs forms an LU decomposition of B, and then use the decomposition to solve the system $Bx = \begin{bmatrix} 0 & 4 & 1 \end{bmatrix}^T$.

(a)

$$L_1 = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 11 & 1 \end{bmatrix}, \quad U_1 = \begin{bmatrix} 2 & 7 & 5 \\ 0 & -1 & -5 \\ 0 & 0 & 45 \end{bmatrix}.$$

(b)

$$L_2 = \begin{bmatrix} 1 & 0 & 0 \\ 3 & -1 & 0 \\ 2 & -11 & 45 \end{bmatrix}, \quad U_2 = \begin{bmatrix} 2 & 7 & 5 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix}.$$

(c)

$$L_3 = \begin{bmatrix} 2 & 0 & 0 \\ 6 & -1 & 0 \\ 4 & -11 & 45 \end{bmatrix}, \quad U_3 = \begin{bmatrix} 1 & 7/2 & 5/2 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{bmatrix}.$$

In general, the LU factorization is not unique! The algorithm we learned in lecture produces the so-called "Doolittle factorization," which forces L to have a unit diagonal ($l_{ii} = 1$ for $1 \leq i \leq n$).

Solution

For the verification, explicit computation shows that $L_1 U_1 = L_2 U_2 = L_3 U_3 = B$.

For each case, we first solve $L_j y = \begin{bmatrix} 0 & 4 & 1 \end{bmatrix}^T$ and then $U_j x = y$, using forward and backward substitution. We obtain:

(a)

$$y = \begin{bmatrix} 0 \\ 4 \\ -43 \end{bmatrix},$$

(b)

$$y = \begin{bmatrix} 0 \\ -4 \\ -43/45 \end{bmatrix},$$

(c)

$$y = \begin{bmatrix} 0 \\ -4 \\ -43/45 \end{bmatrix}.$$

In each case we have $x = \begin{bmatrix} -1/3 & 7/9 & -43/45 \end{bmatrix}^T$.

5. *(LU factorization, Bradie* 3.5.10.*)* Let

$$C = \begin{bmatrix} 1 & 1 & 2 \\ -1 & 0 & 2 \\ 3 & 2 & -1 \end{bmatrix}.$$

Find a lower triangular L with ones along its diagonal and an upper triangular matrix U such that $A = LU$.

Solution

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -1 & 1 \end{bmatrix} \text{ and } U = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 4 \\ 0 & 0 & -3 \end{bmatrix}.$$

6. With B as defined in 4 above, use partial pivoting to find matrices L, U, and P so that $PB = LU$.

Solution

$$L = \begin{bmatrix} 1 & & \\ 2/3 & 1 & \\ 1/3 & -1/31 & 1 \end{bmatrix}, \; U = \begin{bmatrix} 6 & 20 & 10 \\ & -31/3 & -20/3 \\ & & 45/31 \end{bmatrix}, \text{ and } P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

7. *(Block matrices.)* Let A and B be $n \times n$ matrices. For $1 \leq k \leq n$, set

$$\begin{aligned} S_{11} &= A(1:k, 1:k), \\ S_{12} &= A(1:k, k+1:n), \\ S_{21} &= A(k+1:n, 1:k), \\ S_{22} &= A(k+1:n, k+1:n). \end{aligned}$$

Similarly, let

$$\begin{aligned} T_{11} &= B(1:k, 1:k), \\ T_{12} &= B(1:k, k+1:n), \\ T_{21} &= B(k+1:n, 1:k), \\ T_{22} &= B(k+1:n, k+1:n). \end{aligned}$$

Note that A can be partitioned as

$$A = \left[\begin{array}{c|c} S_{11} & S_{12} \\ \hline S_{21} & S_{22} \end{array}\right],$$

and that a similar partition applies for B.

Show that

$$AB = \left[\begin{array}{c|c} S_{11}T_{11} + S_{12}T_{21} & S_{11}T_{12} + S_{12}T_{22} \\ \hline S_{21}T_{11} + S_{22}T_{21} & S_{21}T_{12} + S_{22}T_{22} \end{array}\right].$$

Note the sizes and order of the factors!

Solution

This result follows from the definition of matrix multiplication, and the indexing can be slightly confusing. For instance, consider any $1 \leq i, j \leq k$. Then

$$\begin{aligned} (S_{11}T_{11} + S_{12}T_{21})_{ij} &= (S_{11}T_{11})_{ij} + (S_{12}T_{21})_{ij} \\ &= \sum_{p=1}^{k} (S_{11})_{ip}(T_{11})_{pj} + \sum_{p=1}^{n-k} (S_{12})_{ip}(T_{21})_{pj} \\ &= \sum_{p=1}^{k} a_{ip}b_{pj} + \sum_{p=1}^{n-k} a_{i,p+k}b_{p+k,j} \\ &= \sum_{p=1}^{n} a_{ip}b_{pj} \\ &= (AB)_{ij}. \end{aligned}$$

The first and second equalities follow by definition of matrix addition and multiplication. The third equality follows by definition of S_{rs} and T_{rs}, and the last equality follows again by definition of matrix multiplication.

The result for other indices follows by analogous reasoning, and we omit the argument here.

Assignment problems

Solutions to the following problems should be submitted.

1. (30 points) *(Permuted LU).* Consider the linear system $Ax = b$. In MATLAB, implement the LU factorization algorithm with partial pivoting.

To reduce the storage needs of our algorithm, we require that the algorithm overwrites A with its LU factors. That is, you may not at any point create new arrays to store the L and U factors: their entries should be stored in place of the entries of A.

Next, implement triangular solvers (backward and forward substitution) which can be used to solve the systems $Ly = b$ and $Ux = y$.

Define

$$A = \begin{bmatrix} 2 & 5 & -9 & 3 \\ 5 & 6 & -4 & 2 \\ 3 & -4 & 2 & 7 \\ 11 & 7 & 4 & -8 \end{bmatrix} \text{ and } b = \begin{bmatrix} 151 \\ 103 \\ 16 \\ -32 \end{bmatrix},$$

and use your triangular solvers to find x. Print out L, U, and the permutation matrix P. Verify that $x =$ `A\b`.

Hint: Explicitly swapping rows of A as needed and creating a permutation matrix will make your life easier. (You should opt for implicit swaps and a permutation vector if you want to write high-perfomance code but explicit swaps and a full permutation matrix suffice here.)

Solution

Refer to the MATLAB code below for implementation details.

```
%Load matrix and vector data
A = [2 5 -9 3; 5 6 -4 2; 3 -4 2 7; 11 7 4 -8];
b = [151; 103; 16; -32];
```

```
N = length(A(1,:));

%Initialize permutation P as identity
P = eye(N);

%Permuted LU factorization
for k = 1:N-1
    %Find pivot
    [a_max,imax] = max(abs(A(k:N,k)));
    if imax+k-1 ~= k
        temp = A(k,:);
        A(k,:) = A(k+imax-1,:);
        A(k+imax-1,:) = temp;
        temp = P(k,:);
        %Compute permutation matrix
        P(k,:) = P(k+imax-1,:);
        P(k+imax-1,:) = temp;
    end
    %Perform Gaussian elimination update step
    for i = k+1:N
        A(i,k) = A(i,k)/A(k,k);
        for j=k+1:N
        A(i,j) = A(i,j)-A(i,k)*A(k,j);
        end
    end
end

%Permute b vector using P
b_star = P*b;

%Forward substitution triangular solver
y(1) = b_star(1);
for i=2:N
    y(i) = b_star(i)-A(i,1:i-1)*y(1:i-1)';
end

%Backward substitution triangular solver
x(N) = y(N)/A(N,N);
```

```
for i=N-1:-1:1
        x(i) = (y(i)-A(i,i+1:N)*x(i+1:N)')/A(i,i);
end

%Report solution
x

%Output LU factors and permutation matrix
L = tril(A, -1) + eye(N)
U = triu(A)
P

%Verify factorization
L*U - P*[2 5 -9 3; 5 6 -4 2; 3 -4 2 7; 11 7 4 -8]
```

To the nearest ten-thousanth, the program reports

$$L = \begin{bmatrix} 1.0000 & & & \\ 0.2727 & 1.0000 & & \\ 0.1818 & -0.6308 & 1.0000 & \\ 0.4545 & -0.4769 & 0.5882 & 1.0000 \end{bmatrix},$$

$$U = \begin{bmatrix} 11.0000 & 7.0000 & 4.0000 & -8.0000 \\ & -5.9091 & 0.9091 & 9.1818 \\ & & -9.1538 & 10.2462 \\ & & & 3.9882 \end{bmatrix},$$

$$P = \begin{bmatrix} & & & 1 \\ & & 1 & \\ 1 & & & \\ & 1 & & \end{bmatrix}, \text{ and}$$

$$x = \begin{bmatrix} 3 \\ 5 \\ -11 \\ 7 \end{bmatrix}.$$

2. (30 points) *(Tridiagonal LU.)*

(a) Compute an LU decomposition of the tridiagonal matrix A by hand, with

$$A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & -1 & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{bmatrix}.$$

Let $b = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$. Use the computed LU factors to solve the system $Ax = b$ (by hand). You will encounter this matrix later in the course, in the context of PDEs.

(b) Specialize the solver you wrote in Problem 1 to the case where A is tridiagonal.

Note that tridiagonal matrices are *sparse*, which means that most of their entries are 0. As you discovered in Part (a), the LU factors also turn out to be sparse in that case. In the context of solving the system $Ax = b$, sparsity of A implies that much less work is required to obtain a solution. **Your specialized solver must reflect this.** You may assume no pivoting is required. Comment on how the number of computations performed by your specialized solver depends on N, the number of non-zero entries in A.

(c) Use the specialized solver you implemented in Part (b) to verify the calculations you performed in Part (a).

Solution

(a) Using Gaussian elimination, we readily find that

$$L = \begin{bmatrix} 1 & & & \\ -\frac{1}{2} & 1 & & \\ & -\frac{2}{3} & 1 & \\ & & -\frac{3}{4} & 1 \end{bmatrix}, \text{ and } U = \begin{bmatrix} 2 & -1 & & \\ & \frac{3}{2} & -1 & \\ & & \frac{4}{3} & -1 \\ & & & \frac{5}{4} \end{bmatrix}.$$

With the LU factors in hand, we first use forward substitution to find y sucht that $Ly = b$. With a few simple calculations we find that

$$y = \begin{bmatrix} 1 & \frac{3}{2} & 2 & \frac{5}{2} \end{bmatrix}^T.$$

Finally, we solve $Ax = b$ by applying backward substitution to the relation $Ux = y$. A few calculations show that

$$x = \begin{bmatrix} 2 & 3 & 3 & 2 \end{bmatrix}^T.$$

(b) Refer to the MATLAB code below for implementation details. Pay particular attention to the comments, which highlight how the specialized solver reduces the cost of general LU factorization.

The key here is to observe the structure in the LU factors computed in (a). Note that since A is tridiagonal, the factors are *bidiagonal*. This is a direct consequence of the limited *reach* of A and is a property of general tridiagonal matrices, not just of the given A. Note that because A is tridiagonal, the kth Gaussian elimination update step only *reaches* the $(k-1)$st, kth, and $(k+1)$st entries in the kth row. Therefore, we need only update a few entries at each step.

```
%Load matrix and vector data
%To save memory, you should store the matrix as a sparse matrix.
%Given the structure of this problem, the intereseted reader
%should check out MATLAB's 'tridiag' command.
A = [2 -1 0 0; -1 2 -1 0; 0 -1 2 -1; 0 0 -1 2];
N = length(A(1,:));
b = ones(N,1);

%Permuted LU factorization
for k = 2:N
    %Perform Gaussian elimination update step
    %Since L and U have bandwidth 1, we need only compute a single entry
    %in each pass
    A(k,k-1) = A(k,k-1)/A(k-1, k-1);
    A(k,k) = A(k,k) - A(k,k-1)*A(k-1,k);
end

%Forward substitution triangular solver
y(1) = b(1);
for i=2:N
    %Note only ONE multiplication is required
    y(i) = b(i)-A(i,i-1)*y(i-1)';
end

%Backward substitution triangular solver
```

```
x(N) = y(N)/A(N,N);
for i=N-1:-1:1
    %Note only TWO multiplications are required
        x(i) = (y(i)-A(i,i+1)*x(i+1))/A(i,i);
end

%Report solution
x

%Output LU factors
L = tril(A, -1) + eye(N)
U = triu(A)


%Verify factorization
L*U - [2 -1 0 0; -1 2 -1 0; 0 -1 2 -1; 0 0 -1 2];
```

In particular, note that the number of operations performed by the program above is directly proportional to N. That is, the cost of the specialized solution algorithm is $O(N)$. In fact, you can verify that the total number of multiplications performed is about $5N$. Note that the work does **not** depend on the size of the matrix but on its number of non-zero entries.

For completeness, we note that the computational cost of general LU factorization is $O(n^3)$ when A is $n \times n$.

(c) Running the program given above yields $x = \begin{bmatrix} 2 & 3 & 3 & 2 \end{bmatrix}^T$, which agrees with the calculations performed in (a).

3. *(Facts about LTs.)* In this problem we'll prove some facts about lower triangular matrices, which we assumed during lecture.

(a) First show that if L is any $n \times n$ lower triangular *invertible* matrix, then $l_{jj} \neq 0$ for every $1 \leq j \leq n$. You can do this by induction, starting with the case $n = 1$ and using the definition of invertibility.

Note that in part (b) you will use this result to prove that the inverse of a lower triangular matrix is lower triangular, so you may **not** assume that L^{-1} is lower triangular. (You don't need to anyways!)

Hint: For the inductive step, partition

$$L = \left[\begin{array}{c|c} L_n & 0_{n\times 1} \\ \hline l_{n+1} & l_{n+1,n+1} \end{array}\right],$$

with $L_n = L(1:n, 1:n)$ and $l_{n+1} = L(n+1, 1:n)$, and consider the equation $LL^{-1} = I$. (Refer to warm-up problem on block matrices.)

(b) Using part (a) and the definition of invertibility, show that if L is an $n \times n$ invertible lower triangular matrix, then L^{-1} is also lower triangular.

Hint: Consider l_{ij} for $j > i$ row by row, starting with $i = 1$.

(c) Recall the so-called "Gauss transforms" defined in Problem 3 of HW2. If $L_k = I + g_k e_k^T$, show that

$$L_1 L_2 \cdots L_{n-1} = I + \sum_{k=1}^{n-1} g_k e_k^T.$$

Solution

i. We prove the claim by induction. Since L is invertible, there exists a matrix $M = L^{-1}$ such that, in particular, $LM = I$.
The base case $n = 1$ is simple: the equality $LM = [l_{11} m_{11}] = [1]$ directly implies $l_{11} \neq 0$.
Now suppose that the diagonal entries of *any* $n \times n$ invertible lower triangular matrix are non-zero and consider an $(n+1) \times (n+1)$ invertible lower triangular L. Following the hint, we partition the product $LM = I$ as follows:

$$LM = \left[\begin{array}{c|c} L_n & 0_{n\times 1} \\ \hline l_{n+1} & l_{n+1,n+1} \end{array}\right] \left[\begin{array}{c|c} M_n & m_{n+1} \\ \hline \tilde{m}_{n+1} & m_{n+1,n+1} \end{array}\right] = \left[\begin{array}{c|c} I_n & 0_{n\times 1} \\ \hline 0_{1\times n} & 1 \end{array}\right]. \tag{1}$$

Here, L_n and M_n denote the first n rows and columns of L and M, respectively. Similarly, $l_{n+1} = L(n+1, 1:n)$, $m_{n+1} = M(1:n, n+1)$, and $\tilde{m}_{n+1} = M(n+1, 1:n)$.
The top-left equality implies $L_n M_n + 0_{n\times 1}\tilde{m}_{n+1} = I_n$, or equivalently,

$$L_n M_n = I_n.$$

The last equality implies the invertibility of L_n. The inductive hypothesis then implies that the diagonal entries of L_n are non-zero.

Since the diagonal entries of L_n are the first n diagonal entries of L, we have that $l_{ii} \neq 0$ for $i = 1, \ldots, n$.
It remains to show that $l_{n+1,n+1} \neq 0$.
The top-right equality in Equation 1 implies $L_n m_{n+1} + 0_{n\times 1} m_{n+1,n+1} = 0_{n\times 1}$, or equivalently,

$$L_n m_{n+1} = 0.$$

Since L_n is invertible, the last system has the unique solution $m_{n+1} = 0$.
The bottom-right equality implies $l_{n+1} m_{n+1} + l_{n+1,n+1} m_{n+1,n+1} = 1$, or equivalently,

$$l_{n+1,n+1} m_{n+1,n+1} = 1.$$

Therefore $l_{n+1,n+1} \neq 0$.

ii. Since L is invertible, there is a matrix $M = L^{-1}$ such that $LM = I$. We show that $l_{ij} = 0$ whenever $j > i$, by induction on i. For the base case, let $i = 1$ and consider the $(1, j)$-entry of $LM = I$, for any $j > 1$. By definition of matrix product, we have

$$\sum_{k=1}^{n} l_{1k} m_{kj} = (LM)_{1j} = I_{1j} = 0.$$

But $l_{1k} = 0$ for $p = 2, \ldots, n$, since L is lower triangular. Thus the last equality is equivalent to $l_{11} m_{1j} = 0$, so that $m_{1j} = 0$ by part (a). Now suppose $m_{pj} = 0$ whenever $j > p$ for *every* $p \leq i$, for some $i \geq 1$, and consider the case $i + 1$. For any $j > i + 1$, the $(i + 1, j)$-entry of $LM = I$ gives

$$\sum_{k=1}^{n} l_{i+1,k} m_{kj} = (LM)_{i+1,j} = I_{i+1,j} = 0.$$

Since L is lower triangular, the last equality becomes

$$\sum_{k=1}^{i} l_{i+1,k} m_{kj} + l_{i+1,i+1} m_{i+1,j} = 0.$$

The sum evaluates to 0 by the inductive hypothesis, since $j > i+1 > k$, so that the equality becomes $l_{i+1,i+1} m_{i+1,j} = 0$. Then $m_{i+1,j} = 0$, since $l_{i+1,i+1} \neq 0$ by part (a), completing our induction.

iii. Recall that, by definition, the first k entries of g_k are 0.
Again, we proceed by induction on n.
The base case $n = 2$ is trivial, so suppose the equality is true for some $n \geq 2$ and consider the case $n + 1$. Then,

$$\begin{aligned} L_1 L_2 \cdots L_n &= (L_1 L_2 \cdots L_{n-1}) L_n \\ &= \left(I + \sum_{k=1}^{n-1} g_k e_k^T \right) (I + g_n e_n^T) \\ &= I + \sum_{k=1}^{n} g_k e_k^T + \sum_{k=1}^{n-1} g_k (e_k^T g_n) e_n^T \\ &= I + \sum_{k=1}^{n} g_k e_k^T + \sum_{k=1}^{n-1} g_k (0) e_n^T \\ &= I + \sum_{k=1}^{n} g_k e_k^T, \end{aligned}$$

completing the induction. The first equality follows by associativity, the second by the inductive hypothesis, and the third by direct calculation. The fourth equality follows from the fact that, by definition, the first n entries of g_n are 0. (In this case each L_k is $(n+1) \times (n+1)$.)