MTH 9821: Homework 1

Due on Sep 11, 2014

Weiyi Chen, Xia Hua, Sam Pfeiffer, Xiaoyu Zhang

Weiyi Chen: #4, 5, 6, 7, 8
Xia Hua: #1, 2, 6, 7, 8
Sam Pfeiffer: #3, 4, 5, 7, 8
Xiaoyu Zhang: #1, 2, 3, 7, 8

Problem 1

Since L_2 and U_1 are nonsingular, they have inverses. This allows us to say,

$$L_1 U_1 = L_2 U_2 \tag{1}$$
$$L_2^{-1} L_1 U_1 U_1^{-1} = L_2^{-1} L_2 U_2 U_1^{-1} \tag{2}$$
$$L_2^{-1} L_1 = U_2 U_1^{-1} \tag{3}$$

From the properties of triangular matrices, $U_1 U_1^{-1}$ is upper triangular and $L_2^{-1} L_1$ is lower triagular. For them to be equal it must be that $U_2 U_1^{-1}$ and $L_2^{-1} L_1$ are diagonal matrices. Define the diagonal matrix

$$D = U_2 U_1^{-1} = L_2^{-1} L_1 \tag{4}$$

then we have

$$D = U_2 U_1^{-1} \Rightarrow U_2 = D U_1 \tag{5}$$
$$D = L_2^{-1} L_1 \Rightarrow L_2 = L_1 D^{-1} \tag{6}$$

Problem 2

Let's say the given matrix is A. Apply the Pseudocde for LU decomposition without pivoting (Table 2.5 on textbook) to A, i.e.,

```
[L,U] = lu_no_pivoting(A)
```

We generate the output as, the lower triangular matrix with entries 1 on main diagonal,

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix} \tag{7}$$

and the upper triangular matrix

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 16 \end{pmatrix} \tag{8}$$

such that $A = LU$.

Problem 3

(i)

The 2×2 leading principle minor of A is

$$det \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} = 2 \times 1 - (-2) \times (-1) = 0 \tag{9}$$

Problem 3 continued on next page. . .

(ii)

Let L and U be the LU factors of A. The entries of the first row of U are given $U(1,k) = A(1,k)$ for $k = 1:3$:

$$U(1,1) = 2, U(1,2) = -1, U(1,3) = 1 \tag{10}$$

The entries of the first column of L are given by $L(k,1) = A(k,1)/U(1,1)$ for $k = 1:3$:

$$L(1,1) = 1, L(2,1) = \frac{-2}{U(1,1)} = -1, L(3,1) = \frac{4}{U(1,1)} = 2 \tag{11}$$

The current forms of L and U are

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & L(3,2) & 1 \end{pmatrix}, U = \begin{pmatrix} 2 & -1 & 1 \\ 0 & U(2,2) & U(2,3) \\ 0 & 0 & U(3,3) \end{pmatrix} \tag{12}$$

The updated form of the 2×2 matrix $A(3:4, 3:4)$ is

$$A(2:3, 2:3) = A(2:3, 2:3) - L(2:3, 1)U(1, 2:3) \tag{13}$$

$$= \begin{pmatrix} 1 & 3 \\ 0 & -1 \end{pmatrix} - \begin{pmatrix} -1 \\ 2 \end{pmatrix} (-1, 1) \tag{14}$$

$$= \begin{pmatrix} 0 & 4 \\ 2 & -3 \end{pmatrix} \tag{15}$$

The unknown entries from the second row of U and from the second column of L can be computed from the 2×2 matrix above in the same way, the entries of the first row of $U(2:3)$ are given $U(2,k) = A(2,k)$ for $k = 2:3$:

$$U(2,2) = 0, U(2,3) = 4 \tag{16}$$

The entries of the first column of L are given by $L(k,2) = A(k,2)/U(2,2)$ for $k = 2:3$:

$$L(3,2) = \frac{A(3,2)}{U(2,2)} \tag{17}$$

However $U(2,2) = 0$, the division by $U(2,2)$ cannot be performed when trying to compute this second row of L.

(iii)

Since

$$det(A) = -16 \tag{18}$$

then A is nonsingular, we just apply the Pseudocde for LU decomposition with row pivoting (Table 2.10 on textbook) to A, i.e.,

```
[P,L,U] = lu_row_pivoting(A)
```

We generate the output as, the permutation matrix,

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{19}$$

the lower triangular matrix with entries 1 on main diagonal,

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ 0.5 & -1 & 1 \end{pmatrix} \tag{20}$$

and the upper triangular matrix

$$U = \begin{pmatrix} 4 & 0 & -1 \\ 0 & 1 & 2.5 \\ 0 & 0 & 4 \end{pmatrix} \tag{21}$$

such that $PA = LU$.

Problem 4

Pseudocode for the forward substitution corresponding to a lower triangular banded matrix of band m

```
Function Call:
x = forward_subst_band(L,b,m)

Input:
L = nonsingular banded lower triangular matrix of size n and band m
b = column vector of size n
m = band of banded matrix

Output:
x = solution to Lx=b

x[1] = b[1] / L[1,1]
for j = 2:n
    sum = 0
    for k = max{1,(j-m)}:(j-1)
        sum = sum + L[j,k]x[k]
    end
    x[j] = (b[j]-sum)/L[j,j]
end
```

Computing $x(1)$ requires 1 operation. At step $j = 2 : (m + 1)$,

$$\max\{1, (j - m)\} = 1 \tag{22}$$

the "for" loop to compute the term sum requires $2(j - 1)$ operations. Thus, the "for" loop to compute $x(j)$ for $j = 2 : (m + 1)$ requires $2(j - 1) + 2 = 2j$ operations. At step $j = (m + 2) : n$,

$$\max\{1, (j - m)\} = j - m \tag{23}$$

the "for" loop to compute the term sum requires $2m$ operations. Thus, the "for" loop to compute $x(j)$ for $j = (m + 2 : n)$ requires $2m + 2$ operations.
Then the total number of operations required by the Forward Substitution is

$$1 + \sum_{j=2}^{m+1} 2j + \sum_{j=m+2}^{n} 2m + 2 = 1 + m(m + 3) + (n - m - 1)(2m + 2) \tag{24}$$

$$= (2m + 2)n - m^2 - m - 1 \tag{25}$$

$$= 2mn - m^2 + O(n) \tag{26}$$

Problem 5

(Symmetrical to)Pseudocode for the backward substitution corresponding to an upper triangular banded matrix of band m

```
Function Call:
x = backward_subst_band(U,b)

Input:
U = nonsingular banded upper triangular matrix of size n and band m
b = column vector of size n

Output:
x = solution to Ux=b

x[n] = b[n] / U[n,n]
for j = (n-1):1
    sum = 0
    for k = (j+1):min{n,(j+m)}
        sum = sum + U[j,k]x[k]
    end
    x[j] = (b[j]-sum)/U[j,j]
end
```

Computing $x(n)$ requires 1 operation. At step $j = (n-1) : (n-m-1)$, the "for" loop to compute the term sum requires $2(n-j)$ operations. Thus, the "for" loop to compute $x(j)$ for $j = (n-1) : (n-m-1)$ requires $2(n-j)+2 = 2n-2j+2$ operations. At step $j = (n-m-2) : 1$, the "for" loop to compute the term sum requires $2m$ operations. Thus, the "for" loop to compute $x(j)$ for $j = (n-m-2 : 1)$ requires $2m+2$ operations.
Then the total number of operations required by the Forward Substitution is

$$1 + \sum_{j=n-m-1}^{n-1} 2n - 2j + 2 + \sum_{1}^{n-m-2} 2m + 2 = 1 + m(m+3) + (n-m-1)(2m+2) \quad (27)$$

$$= (2m+2)n - m^2 - m - 1 \quad (28)$$

$$= 2mn - m^2 + O(n) \quad (29)$$

Problem 6

Given the fact that described in the problem, the pseudocode for the LU composition without pivoting for banded matrices of band $m \geq 1$ is

```
Function Call:
[L,U] = lu_no_pivoting_band(A)

Input:
A = nonsingular banded matrix of size n and band m with LU decomposition

Output:
L = lower banded triangular matrix with entries 1 on main diagonal and band m
U = uppder banded triangular matrix with band m
```

```
such that A = LU

Initialize L = 0, U = 0
for i = 1:(n-1)
    for k = i:min{(i+m),n}
        U(i,k) = A(i,k)
        L(k,i) = A(k,i)/U(i,i)
    end
    for j = (i+1):min{(i+m),n}
        for k = (i+1):min{(i+m),n}
            A(j,k) = A(j,k) - L(j,i)U(i,k)
        end
    end
end
L(n,n) = 1, U(n,n) = A(n,n)
```

At step $i = 1 : (n-m-1)$, the "for" loop to compute the i-th row of U and the i-th column of L requires $m+1$ operations. The double "for" loop to update $A(i+1 : i+m, i+1 : i+m)$ requires

$$\sum_{i+1}^{i+m} \sum_{i+1}^{i+m} 2 = 2m^2 \tag{30}$$

operations. Then accounting for the outside "for" loop for $i = 1 : (n-m-1)$, we obtain that the operation count for the LU decomposition without pivoting is

$$\sum_{i=1}^{n-m-1} (2m^2 + m + 1) = (n-m-1)(2m^2 + m + 1) \tag{31}$$

At step $i = (n-m) : (n-1)$, the "for" loop to compute the i-th row of U and the i-th column of L requires $n-i+1$ operations. The double "for" loop to update $A(i+1 : n, i+1 : n)$ requires

$$\sum_{j=i+1}^{n} \sum_{k=i+1}^{n} 2 = 2(n-i)^2 \tag{32}$$

operations. Then accounting for the outside "for" loop for $i = (n-m) : (n-1)$, we obtain that the operation count for the LU decomposition without pivoting is

$$\sum_{i=n-m}^{n-1} [2(n-i)^2 + (n-i+1)] = \frac{2}{3}m^3 + \frac{3}{2}m^2 + \frac{11}{6}m \tag{33}$$

Therefore the total operation count is

$$\sum_{i=1}^{n-m-1} (2m^2 + m + 1) + \sum_{i=n-m}^{n-1} [2(n-i)^2 + (n-i+1)] \tag{34}$$

$$= (2m^2 + m + 1)n - \frac{1}{6}m(m+1)(8m+1) \tag{35}$$

$$= 2m^2 n - \frac{4}{3}m^3 + O(mn) \tag{36}$$

Problem 7

C++ codes for problem 7 and 8:

Problem 7 continued on next page. . .

```
#ifndef LU_DECOMPOSITION_HPP_
#define LU_DECOMPOSITION_HPP_

#include <Eigen/Dense>
#include <boost/tuple/tuple.hpp>

using namespace Eigen;

VectorXd forward_subst(MatrixXd L, VectorXd b){
    /*
    @summary: Forward substitution
    @param L: nonsingular lower triangular matrix of size n
    @param b: column vector of size n
    @return x: solution to Lx=b
    */
    int n = sqrt(L.size());
    VectorXd x(n);
    x(0) = b(0) / L(0,0);
    for (int j = 1; j < n; ++j){
        double sum = 0;
        for (int k = 0; k < j; ++k)
            sum += L(j,k) * x(k);
        x(j) = (b(j) - sum) / L(j,j);
    }
    return x;
}

VectorXd backward_subst(MatrixXd U, VectorXd b){
    /*
    @summary: Backward substitution
    @param U: nonsingular upper triangular matrix of size n
    @param b: column vector of size n
    @return x: solution to Ux=b
    */
    int n = sqrt(U.size());
    VectorXd x(n);
    x(n-1) = b(n-1) / U(n-1,n-1);
    for (int j = n-2; j >= 0; --j){
        double sum = 0.0;
        for (int k = j+1; k < n; ++k)
            sum += U(j,k) * x(k);
        x(j) = (b(j) - sum) / U(j,j);
    }
    return x;
}

boost::tuple<MatrixXd, MatrixXd> lu_no_pivoting(MatrixXd A) {
    /*
    @summary: LU decomposition without pivoting
    @param A: nonsingular matrix of size n with LU decomposition
    @return L: lower triangular matrix with entries 1 on main diagonal
    @return U: upper triangular matrix such that A=LU
    */
```

Problem 7 continued on next page. . .

```
    int n = sqrt(A.size());
    MatrixXd L(n,n), U(n,n);
    L.setZero();
    U.setZero();
    for (int i = 0; i < n-1; ++i) {
        for (int k = i; k < n; ++k) {
            U(i,k) = A(i,k);
            L(k,i) = A(k,i) / U(i,i);
        }
        A.block(i+1,i+1,n-i-1,n-i-1) -= L.block(i+1,i,n-i-1,1)*U.block(i,i+1,1,n-i-1);
    }
    L(n-1,n-1) = 1;
    U(n-1,n-1) = A(n-1, n-1);
    return boost::make_tuple(L, U);
}

VectorXd linear_solve_LU_no_pivoting(MatrixXd A, VectorXd b) {
    /*
    @summary: linear solver using LU decomposition without pivoting
    @param A: nonsingular square matrix of size n with LU decomposition
    @param b: column vector of size n
    @return x: solution to Ax=b
    */
    int n = sqrt(A.size());
    MatrixXd L(n,n), U(n,n);
    boost::tie(L, U) = lu_no_pivoting(A);
    VectorXd y(n), x(n);
    y = forward_subst(L, b);
    x = backward_subst(U, y);
    return x;
}

boost::tuple<VectorXi, MatrixXd, MatrixXd> lu_row_pivoting(MatrixXd A) {
    /*
    @summary: LU decomposition with row pivoting
    @param A: nonsingular matrix of size n
    @return P: permutation matrix, stored as vector of its diagonal entries
    @return L: lower triangular matrix with entries 1 on main diagonal
    @return U: upper triangular matrix such that PA=LU
    */
    int n = sqrt(A.size());
    VectorXi P = VectorXi::LinSpaced(n,1,n); // initialize P as an identity matrix
    MatrixXd L(n,n), U(n,n);
    L.setIdentity();                         // initialize L as an identity matrix
    for (int i = 0; i < n; ++i){
        ArrayXd::Index i_row, i_column;
        A.block(i,i,n-i,1).array().abs().maxCoeff(&i_row, &i_column);
        ArrayXd::Index i_max = i_row + i;    // find i_max, index of largest entry in absolute value
            from vector A(i:n,i)
        A.row(i).swap(A.row(i_max));         // switch rows i and i_max of A
        P.row(i_max).swap(P.row(i));                 // update matrix P
        if (i > 0)
            L.block(i,0,1,i).swap(L.block(i_max,0,1,i));   // switch rows i and i_max of L
```

```
        for (int j = i; j < n; j++) {
            L(j,i) = A(j,i) / A(i,i);        // compute column i of L
            U(i,j) = A(i,j);                 // compute row i of U
        }
        A.block(i+1,i+1,n-i-1,n-i-1) -= L.block(i+1,i,n-i-1,1)*U.block(i,i+1,1,n-i-1);
    }
    L(n-1,n-1) = 1;
    U(n-1,n-1) = A(n-1,n-1);
    return boost::make_tuple(P,L,U);
}

#endif /* LU_DECOMPOSITION_HPP_ */
```