BARUCH, MFE

MTH 9821 Homework One *

Chu, Hongshan
Wang, Jiaxi
Wei, Zhaoyue
Yin, Gongshun
Zhou, ShengQuan

September 4, 2016

*Team work:
Chu, Hongshan #1, #2, #3
Wang, Jiaxi #4, #5
Wei, Zhaoyue #5, #6
Yin, Gongshun #7, #8
Zhou, ShengQuan #1, #2, #3, #7 ,#8

1 UNIQUENESS OF LU-DECOMPOSITION

Let L_1 and L_2 be nonsingular lower triangular matrices and let U_1 and U_2 be nonsingular upper triangular matrices. If $L_1U_1 = L_2U_2$, show that there exists a nonsingular diagonal matrix D such that

$$L_1 = L_2D, \quad \text{and} \quad U_1 = D^{-1}U_2.$$

Proof: Since L_1, L_2, U_1, and U_2 are nonsingular, apply L_2^{-1} from the left and U_1^{-1} from the right:

$$\begin{aligned}
L_1U_1 = L_2U_2 \quad &\Rightarrow \quad L_2^{-1}L_1U_1U_1^{-1} = L_2^{-1}L_2U_2U_1^{-1} \\
&\Rightarrow \quad \underbrace{L_2^{-1}L_1}_{\text{lower-triangular}} = \underbrace{U_2U_1^{-1}}_{\text{upper-trianglar}} \\
&\Rightarrow \quad L_2^{-1}L_1 = U_2U_1^{-1} = D,
\end{aligned}$$

where D is a nonsingular diagonal matrix. In other words,

$$L_1 = L_2D, \quad \text{and} \quad U_1 = D^{-1}U_2.$$

2 *LU*-Decomposition of a Special Matrix

Find the LU-decomposition without pivoting of the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix}$$

Solution: This LU-decomposition can be done by hand exactly:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 1 & 8 \\ 0 & 0 & 0 & 0 & 16 \end{pmatrix}.$$

The entries of U on the rightmost column are unbound relative to the those of the original matrix as the matrix dimension grows. This is a classic example of a matrix whole LU-decomposition is unstable.

3 LU-DECOMPOSITION WITH NO-PIVOTING AND ROW-PIVOTING

Let

$$A = \begin{pmatrix} 2 & -1 & 1 \\ -2 & 1 & 3 \\ 4 & 0 & -1 \end{pmatrix}$$

(i) Show that the 2×2 leading principal minor of A is 0.
Proof: Direct computation,

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

(ii) Attempt to do the LU-decomposition without pivoting of the matrix A, and show that the division by U_{22} cannot be performed when trying to compute the second row of L.
Solution: We perform LU-decomposition with no pivoting at stages:

Stage 1:

$$A = \begin{pmatrix} 2 & -1 & 1 \\ -2 & 1 & 3 \\ 4 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & \times & 0 \\ 2 & \times & \times \end{pmatrix} \times \begin{pmatrix} 2 & -1 & 1 \\ 0 & \times & \times \\ 0 & 0 & \times \end{pmatrix}$$

Stage 2: we have

$$\begin{pmatrix} -1 \\ 2 \end{pmatrix} \times \begin{pmatrix} -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -2 & 2 \end{pmatrix} \Rightarrow A = \begin{pmatrix} 2 & -1 & 1 \\ -2 & 1 & 3 \\ 4 & 0 & -1 \end{pmatrix} \to \begin{pmatrix} 2 & -1 & 1 \\ -2 & \boxed{0} & 4 \\ 4 & 2 & -3 \end{pmatrix}$$

At this stage, we get $U_{22} = A_{22} = 0$. Thus, the division by U_{22} cannot be performed when trying to compute the second row of L.

(iii) Show that the matrix A is nonsingular, and compute the LU-decomposition with row pivoting of A.
Solution: Let $P = (1, 2, 3)$. We perform LU-decomposition with row pivoting at stages:

Stage 1: Interchange row 1 and row 3, $P = (3, 2, 1)$,

$$A \to \begin{pmatrix} 4 & 0 & -1 \\ -2 & 1 & 3 \\ 2 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1/2 & \times & 0 \\ 1/2 & \times & \times \end{pmatrix} \times \begin{pmatrix} 4 & 0 & -1 \\ 0 & \times & \times \\ 0 & 0 & \times \end{pmatrix}$$

Stage 2:

$$\begin{pmatrix} -1/2 \\ 1/2 \end{pmatrix} \times \begin{pmatrix} 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & 1/2 \\ 0 & -1/2 \end{pmatrix} \Rightarrow A \to \begin{pmatrix} 4 & 0 & -1 \\ -2 & 1 & 5/2 \\ 2 & -1 & 3/2 \end{pmatrix}$$

No exchange of rows is needed,

$$A \to \begin{pmatrix} 4 & 0 & -1 \\ -2 & 1 & 5/2 \\ 2 & -1 & 3/2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/2 & -1 & \times \end{pmatrix} \times \begin{pmatrix} 4 & 0 & -1 \\ 0 & 1 & 5/2 \\ 0 & 0 & \times \end{pmatrix}.$$

Stage 3: $-1 \times 5/2 = -5/2$. Thus, no exchange of rows,

$$A \to \begin{pmatrix} 4 & 0 & -1 \\ -2 & 1 & 5/2 \\ 2 & -1 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/2 & -1 & 1 \end{pmatrix} \times \begin{pmatrix} 4 & 0 & -1 \\ 0 & 1 & 5/2 \\ 0 & 0 & 4 \end{pmatrix}.$$

In summary,

$$\underbrace{\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}}_{P} \times \underbrace{\begin{pmatrix} 2 & -1 & 1 \\ -2 & 1 & 3 \\ 4 & 0 & -1 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/2 & -1 & 1 \end{pmatrix}}_{L} \times \underbrace{\begin{pmatrix} 4 & 0 & -1 \\ 0 & 1 & 5/2 \\ 0 & 0 & 4 \end{pmatrix}}_{U}.$$

The existence of an LU-decomposition with row pivoting serves as a proof that the matrix A is nonsingular.

4 Pseudocode for the Forward Substitution for a Lower Triangular Banded matrix

Write the pseudocode for the forward substitution corresponding to a lower triangular banded matrix of band m, i.e. for solving $Lx = b$ where b is an n vector and L is an $n \times n$ lower triangular matrix such that

$$L_{jk} = 0, \quad \forall 1 \leq j, k \leq n, \quad j - k > m.$$

What is the corresponding operation count?

```
Function Call:
x = forward_subst_banded(L,b)


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

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
```

The operation count of the above forward substitution for a lower triangular banded matrix is given by

$$1 + \sum_{j=2}^{m+1} \left[2(j-1) + 2\right] + \sum_{j=m+2}^{n} (2m+2) = 2mn - m^2 + O(m).$$

5 Pseudocode for the Backward Substitution for an Upper Triangular Banded matrix

Write the pseudocode for the backward substitution corresponding to an upper triangular banded matrix of band m, i.e. for solving $Ux = b$ where b is an n vector and U is an $n \times n$ upper triangular matrix such that

$$U_{jk} = 0, \quad \forall 1 \leq j, k \leq n, \quad k - j > m.$$

What is the corresponding operation count?

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


Input:
U = nonsingular upper triangular banded matrix of band m of size n
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
```

The operation count of the above forward substitution for a lower triangular banded matrix is given by

$$1 + \sum_{j=n-m}^{n-1} \left[2(n-j)+2\right] + \sum_{j=1}^{n-m-1} (2m+2) = 2mn - m^2 + O(m).$$

6 Pseudocode for the LU-Decomposition without Pivoting for Banded Matrices

6.1 LU-Decomposition without Pivoting

Write the pseudocode for the LU-decomposition without pivoting for banded matrices of band m. What is the operation count?

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

Input:
A = nonsingular banded matrix of band m

Output:
L = lower triangular matrix with entries 1 on main diagonal
U = upper triangular matrix
such that A = LU

for i = 1:(n-1)
    for k = i:n
        U(i,k) = A(i,k);
        L(k,i) = A(k,i)/U(i,i);
    end
    for j = (i+1):n
        for k = (i+1):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);
```

The derivation of the operation count is given in the lecture,

$$\text{Operation Count} = \frac{2}{3}n^3 + O\left(n^2\right).$$

6.2 LU-Decomposition without Pivoting: Band Simplification

Use (without proving) the fact that the L and U factors from the LU-decompostion without pivoting of a banded matrix of band m are a banded lower triangular matrix of band m and a banded upper triangular matrix of band m, respectively. What is the corresponding operation count?

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

Input:
A = nonsingular banded matrix of band m

Output:
L = lower triangular matrix with entries 1 on main diagonal
U = upper triangular matrix
such that A = LU

L=0, U=0; // all entries zero
for i = 1:(n-1)
    for k = i : min{n,i+m}
        U(i,k) = A(i,k);
        L(k,i) = A(k,i)/U(i,i);
    end

    for j = (i+1) : min{n,i+m}
        for k = (i+1) : min{n,i+m}
            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);
```

The operation count is divided into two parts at $i = n - m$:

$$\sum_{i=1}^{n-m-1}\left(m+1+\underbrace{\sum_{j=1}^{i+m}\sum_{j=1}^{i+m}2}_{=2m^2}\right)+\sum_{i=n-m}^{n-1}\left(n-i+1+\underbrace{\sum_{j=1}^{n-i}\sum_{j=1}^{n-i}2}_{2(n-i)^2}\right)=\left(2m^2+m+1\right)n-\frac{4}{3}m^3+O\left(m^2\right).$$

7 C++ Codes for Backward and Forward Substitution

```
#include <triangular_solve.h>
#include <Eigen/Dense>
#include <cassert>

Eigen::VectorXd forward_subst(const Eigen::MatrixXd & L,
                              const Eigen::VectorXd & b)
{
    int n = b.size();
    assert(L.rows() == n);
    assert(L.cols() == n);

    Eigen::VectorXd x(n);
    x(0) = b(0)/L(0,0);
    for (int i=1; i<n; i++) {
        double sum = 0;
        for (int j=0; j<i; j++) {
            sum += L(i,j)*x(j);
        }
        x(i) = (b(i)-sum)/L(i,i);
    }

    return x;
}

Eigen::VectorXd backward_subst(const Eigen::MatrixXd & U,
                               const Eigen::VectorXd & b)
{
    int n = b.size();
    assert(U.rows() == n);
    assert(U.cols() == n);

    Eigen::VectorXd x(n);
    x(n-1) = b(n-1)/U(n-1,n-1);
    for (int i=n-2; i>=0; i--) {
        double sum = 0;
        for (int j=i+1; j<n; j++) {
            sum += U(i,j)*x(j);
        }
        x(i) = (b(i)-sum)/U(i,i);
    }

    return x;
}
```

8 C++ Codes for *LU*-Decomposition

```
#include <lu.h>
#include <Eigen/Dense>
#include <cassert>
#include <tuple>

static void lu_helper(int k, int n,
                      Eigen::MatrixXd * A,
                      Eigen::MatrixXd * L,
                      Eigen::MatrixXd * U)
{
    for (int i=k; i<n; i++) {
        (*U)(k,i) = (*A)(k,i);
        (*L)(i,k) = (*A)(i,k)/(*U)(k,k);
    }

    for (int i=k+1; i<n; i++) {
        for (int j=k+1; j<n; j++) {
            (*A)(i,j) -= (*L)(i,k)*(*U)(k,j);
        }
    }
}

std::tuple<Eigen::MatrixXd, Eigen::MatrixXd>
    lu_no_pivoting(const Eigen::MatrixXd & A)
{
    Eigen::MatrixXd Acopy = A;
    int n = Acopy.rows();
    assert(n == Acopy.cols());

    Eigen::MatrixXd L(n,n);
    Eigen::MatrixXd U(n,n);

    L.triangularView<Eigen::StrictlyUpper>().setZero();
    U.triangularView<Eigen::StrictlyLower>().setZero();

    for (int k=0; k<n-1; k++) {
        lu_helper(k, n, &Acopy, &L, &U);
    }

    L(n-1,n-1) = 1;
    U(n-1,n-1) = Acopy(n-1,n-1);

    return std::make_tuple(L,U);
}
```

```cpp
std::tuple<Eigen::VectorXi, Eigen::MatrixXd, Eigen::MatrixXd>
    lu_row_pivoting(const Eigen::MatrixXd & A)
{
    Eigen::MatrixXd Acopy = A;
    int n = Acopy.rows();
    assert(n == Acopy.cols());

    Eigen::VectorXi p = Eigen::VectorXi::LinSpaced(n,1,n);
    Eigen::MatrixXd L = Eigen::MatrixXd::Zero(n,n);;
    Eigen::MatrixXd U = Eigen::MatrixXd::Zero(n,n);;

    for (int k=0; k<n-1; k++) {
        int maxRow, maxCol;
        Acopy.block(k,k,n-k,1).array().abs().maxCoeff(&maxRow, &maxCol);
        Acopy.row(k).swap(Acopy.row(maxRow+k));
        p.row(k).swap(p.row(maxRow+k));
        L.row(k).swap(L.row(maxRow+k));
        lu_helper(k, n, &Acopy, &L, &U);
    }

    L(n-1,n-1) = 1;
    U(n-1,n-1) = Acopy(n-1,n-1);

    return std::make_tuple(p,L,U);
}
```