Numerical Analysis Hilary Term 2018
Lecture 5: LU Factorization

The basic operation of Gaussian Elimination, row $i \leftarrow$ row $i + \lambda *$ row j, can be achieved by pre-multiplication by a special lower-triangular matrix

$$M(i,j,\lambda) = I + \begin{array}{cc} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{array}{c} \\ \leftarrow i \\ \\ \end{array} \\ \begin{array}{ccc} & \uparrow & \\ & j & \end{array} & \end{array}$$

where I is the identity matrix.

Example: $n = 4$,

$$M(3,2,\lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad M(3,2,\lambda) \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a \\ b \\ \lambda b + c \\ d \end{bmatrix},$$

i.e., $M(3,2,\lambda)A$ performs: row 3 of $A \leftarrow$ row 3 of $A + \lambda *$ row 2 of A and similarly $M(i,j,\lambda)A$ performs: row i of $A \leftarrow$ row i of $A + \lambda *$ row j of A.

So GE for e.g., $n = 3$ is

$$\begin{array}{ccccccc} M(3,2,-l_{32}) & \cdot & M(3,1,-l_{31}) & \cdot & M(2,1,-l_{21}) & \cdot & A = U = (\urcorner). \\ l_{32} = \dfrac{a_{32}}{a_{22}} & & l_{31} = \dfrac{a_{31}}{a_{11}} & & l_{21} = \dfrac{a_{21}}{a_{11}} & & \text{(upper triangular)} \end{array}$$

The l_{ij} are called the **multipliers**.

Be careful: each multiplier l_{ij} uses the data a_{ij} and a_{ii} that *results from the transformations already applied*, not data from the original matrix. So l_{32} uses a_{32} and a_{22} that result from the previous transformations $M(2,1,-l_{21})$ and $M(3,1,-l_{31})$.

Lemma. If $i \neq j$, $(M(i,j,\lambda))^{-1} = M(i,j,-\lambda)$.

Proof. Exercise.

Outcome: for $n = 3$, $A = M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) \cdot U$, where

$$M(2,1,l_{21}) \cdot M(3,1,l_{31}) \cdot M(3,2,l_{32}) = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} = L = (\llcorner). \quad \text{(lower triangular)}$$

This is true for general n:

Theorem. For any dimension n, GE can be expressed as $A = LU$, where $U = (\urcorner)$ is upper triangular resulting from GE, and $L = (\llcorner)$ is unit lower triangular (lower triangular with ones on the diagonal) with l_{ij} = multiplier used to create the zero in the (i, j)th position.

Most implementations of GE therefore, rather than doing GE as above,

$$\begin{array}{rll} \text{factorize} & A = LU & (\approx \frac{1}{3}n^3 \text{ adds} + \approx \frac{1}{3}n^3 \text{ mults}) \\ \text{and then solve} & Ax = b & \\ \text{by solving} & Ly = b & \text{(forward substitution)} \\ \text{and then} & Ux = y & \text{(back substitution)} \end{array}$$

Note: this is much more efficient if we have many different right-hand sides b but the same A.

Pivoting: GE or LU can fail if the pivot $a_{ii} = 0$. For example, if

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

GE fails at the first step. However, we are free to reorder the equations (i.e., the rows) into any order we like. For example, the equations

$$\begin{array}{c} 0 \cdot x_1 + 1 \cdot x_2 = 1 \\ 1 \cdot x_1 + 0 \cdot x_2 = 2 \end{array} \quad \text{and} \quad \begin{array}{c} 1 \cdot x_1 + 0 \cdot x_2 = 2 \\ 0 \cdot x_1 + 1 \cdot x_2 = 1 \end{array}$$

are the same, but their matrices

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

have had their rows reordered: GE fails for the first but succeeds for the second $\Longrightarrow$ better to interchange the rows and then apply GE.

Partial pivoting: when creating the zeros in the jth column, find

$$|a_{kj}| = \max(|a_{jj}|, |a_{j+1j}|, \ldots, |a_{nj}|),$$

then swap (interchange) rows j and k.

For example,

$$\begin{bmatrix} a_{11} & \cdot & a_{1j-1} & a_{1j} & \cdot & \cdot & \cdot & a_{1n} \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & a_{j-1j-1} & a_{j-1j} & \cdot & \cdot & \cdot & a_{j-1n} \\ 0 & \cdot & 0 & a_{jj} & \cdot & \cdot & \cdot & a_{jn} \\ 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & a_{kj} & \cdot & \cdot & \cdot & a_{kn} \\ 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & a_{nj} & \cdot & \cdot & \cdot & a_{nn} \end{bmatrix} \to \begin{bmatrix} a_{11} & \cdot & a_{1j-1} & a_{1j} & \cdot & \cdot & \cdot & a_{1n} \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & a_{j-1j-1} & a_{j-1j} & \cdot & \cdot & \cdot & a_{j-1n} \\ 0 & \cdot & 0 & a_{kj} & \cdot & \cdot & \cdot & a_{kn} \\ 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & a_{jj} & \cdot & \cdot & \cdot & a_{jn} \\ 0 & \cdot & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & a_{nj} & \cdot & \cdot & \cdot & a_{nn} \end{bmatrix}$$

Property: GE with partial pivoting cannot fail if A is nonsingular.

Proof. If A is the first matrix above at the jth stage,

$$\det[A] = a_{11} \cdots a_{j-1j-1} \cdot \det \begin{bmatrix} a_{jj} & \cdot & \cdot & \cdot & a_{jn} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{kj} & \cdot & \cdot & \cdot & a_{kn} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ a_{nj} & \cdot & \cdot & \cdot & a_{nn} \end{bmatrix}.$$

Hence $\det[A] = 0$ if $a_{jj} = \cdots = a_{kj} = \cdots = a_{nj} = 0$. Thus if the pivot $a_{k,j}$ is zero, A is singular. So if A is nonsingular, all of the pivots are nonzero. (Note: actually a_{nn} can be zero and an LU factorization still exist.)

The effect of pivoting is just a permutation (reordering) of the rows, and hence can be represented by a permutation matrix P.

Permutation matrix: P has the same rows as the identity matrix, but in the pivoted order. So

$$PA = LU$$

represents the factorization—equivalent to GE with partial pivoting. E.g.,

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} A$$

has the 2nd row of A first, the 3rd row of A second and the 1st row of A last.

Matlab example:

```
>> A = rand(5,5)
A =
      0.69483      0.38156      0.44559       0.6797      0.95974
       0.3171      0.76552      0.64631       0.6551      0.34039
      0.95022       0.7952      0.70936      0.16261      0.58527
     0.034446      0.18687      0.75469        0.119      0.22381
      0.43874      0.48976      0.27603      0.49836      0.75127
>> exactx = ones(5,1);  b = A*exactx;
>> [LL, UU] = lu(A)  % note "psychologically lower triangular" LL
LL =
      0.73123     -0.39971      0.15111            1            0
      0.33371            1            0            0            0
            1            0            0            0            0
     0.036251        0.316            1            0            0
      0.46173      0.24512     -0.25337      0.31574            1
UU =
      0.95022       0.7952      0.70936      0.16261      0.58527
            0      0.50015      0.40959      0.60083      0.14508
            0            0      0.59954    -0.076759      0.15675
            0            0            0      0.81255      0.56608
            0            0            0            0      0.30645
```

```
>> [L, U, P] = lu(A)
L =
             1             0             0             0             0
       0.33371             1             0             0             0
      0.036251         0.316             1             0             0
       0.73123      -0.39971       0.15111             1             0
       0.46173       0.24512      -0.25337       0.31574             1
U =
       0.95022        0.7952       0.70936       0.16261       0.58527
             0       0.50015       0.40959       0.60083       0.14508
             0             0       0.59954     -0.076759       0.15675
             0             0             0       0.81255       0.56608
             0             0             0             0       0.30645
P =
     0     0     1     0     0
     0     1     0     0     0
     0     0     0     1     0
     1     0     0     0     0
     0     0     0     0     1

>> max(max(P'*L - LL)))  % we see LL is P'*L
ans =
     0

>> y = L \ (P*b);  % now to solve Ax = b...
>> x = U \ y
x =
             1
             1
             1
             1
             1

>> norm(x - exactx, 2)  % within roundoff error of exact soln
ans =
   3.5786e-15
```