§ 7.2.3. LU w/ Pivoting

GE may fail:

Take $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$, $b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \Rightarrow x_1 = 1 = x_2$$

But in GE: mult $= A_{21} / \underbrace{A_{11}}_{"0"!}$ is not defined.

(This may happen at any stage of GE.)

Solution to this: to interchange the rows

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

Strategy: 1) test if the denominator in "mult" is zero;

2) if so, search for a nonzero entry in the **column**, interchange the rows, and do the elimination.

Pivoting: the process of interchanging rows;

Pivot (element): the nonzero entry

This will work in exact arithmetic, but not in finite precision one used on computers:

$$\begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$x_1 \approx 1$ & $x_2 \approx 1$ as $x_1 = 1 + \frac{1}{10^{20}-1}$ & $x_2 = 1 - \frac{1}{10^{20}-1}$

In our GE/LU code:

$$\left(\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 1 & 1 & 2 \end{array}\right) \xrightarrow{r_2 - 10^{20} r_1} \left(\begin{array}{cc|c} 10^{-20} & 1 & 1 \\ 0 & \underbrace{1-10^{20}}_{-10^{20}} & \underbrace{2-10^{20}}_{-10^{20}} \end{array}\right)$$

due to rounding!

$\Rightarrow$ $x_2 \approx 1$, but $10^{-20} x_1 + 1 = 1$ $\Rightarrow$ $x_1 = 0$

Again, we could have interchanged the rows:

$$\left(\begin{array}{cc|c} 1 & 1 & 2 \\ 10^{-20} & 1 & 1 \end{array}\right) \xrightarrow{r_2 - r_1 10^{-20}} \left(\begin{array}{cc|c} 1 & 1 & 2 \\ 0 & \underbrace{1}_{1-10^{-2} \approx 1} & \underbrace{1}_{1-2\cdot 10^{-20} \approx 1} \end{array}\right)$$

$\Rightarrow$ $x_1 = 1 = x_2$!

To avoid both difficulties: with 0 or tiny entries, <u>partial pivoting</u> is used, i.e. choose the pivot to be abs(largest entry).

Partial pivoting:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}$$

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 0 & 2 \end{array}\right) \xrightarrow{\text{pivot}} \left(\begin{array}{ccc|c} 7 & 8 & 0 & 2 \\ 4 & 5 & 6 & 0 \\ 1 & 2 & 3 & 1 \end{array}\right) \xrightarrow[r_3 - r_1(1/7)]{r_2 - r_1(4/7)}$$

$$\left(\begin{array}{ccc|c} 7 & 8 & 0 & 2 \\ 0 & 3/7 & 6 & -8/7 \\ 0 & 6/7 & 3 & 5/7 \end{array}\right) \xrightarrow{\text{pivot}} \left(\begin{array}{ccc|c} 7 & 8 & 0 & 2 \\ 0 & 6/7 & 3 & 5/7 \\ 0 & 3/7 & 6 & -8/7 \end{array}\right)$$

$$\xrightarrow{r_3 - r_2(1/2)} \left(\begin{array}{ccc|c} 7 & 8 & 0 & 2 \\ 0 & 6/7 & 3 & 5/7 \\ 0 & 0 & 9/2 & -3/2 \end{array}\right) \Rightarrow \begin{aligned} x_3 &= -1/3 \\ x_2 &= 2 \\ x_1 &= -2 \end{aligned}$$

See handout on partial pivoting GE in matrix form (LU w/ pivoting).

Note: one can perform "complete" pivoting permuting both rows & columns

```
% Gaussian elimination with partial pivoting.
% This is (roughly) what Matlab can do when it computes PLU factorizations.

for j=1:n-1                    % Loop over columns.

 [pivot,k] = max(abs(A(j:n,j)));  % Find the pivot element in column j.
                               % pivot is the largest absolute
                               % value of an entry; k+j-1 is its
                               % index.
 if pivot==0,                  % If all entries in the column are 0,
   disp(' Matrix is singular.')   % return with an error message.
   break;
 end;
 temp = A(j,:);                % Otherwise,
 A(j,:) = A(k+j-1,:);            % Interchange rows j and k+j-1.
 A(k+j-1,:) = temp;
 tempb = b(j);
 b(j) = b(k+j-1);
 b(k+j-1) = tempb;

 for i=j+1:n                   % Loop over rows below j.
   mult = A(i,j)/A(j,j);       % Subtract this multiple of row j from
                               % row i to make A(i,j)=0.
   A(i,j:n) = A(i,j:n) - mult*A(j,j:n);
   b(i) = b(i) - mult*b(j);
 end
end
```

Operations count:

No additional cost is required for pivoting in terms of FLOPs, but there is $O(n^2)$ cost to find the largest entry & additional cost of data movement. Still, $O(n^2) < \underbrace{O(\frac{2}{3}n^3)}_{\text{\# FLOPs for modified GE code}}$