CLASS NOTES CS/MA 166
Numerical Analysis
DAY 9

Dennis C. Smolarski, S.J.
Santa Clara University
BASIC PIVOTING [4-28]

**Strategy:** Interchange the row containing 0 in the crucial place, with the **first** row (below it) containing a non-zero element in the same column.

**Problems:** Inexact machine arithmetic suggests that we should interchange a zero with the **largest** of all possible elements rather than any non-zero element.
PARTIAL PIVOTING

**Strategy:** Interchange the row containing 0 with the row (below it) containing the LARGEST non-zero element in the same column.

⇒ *continuing the example above.*

\[
\begin{pmatrix}
1 & 2 & 1 & | & 2 \\
0 & 0 & -1 & | & 1 \\
0 & -1 & -2 & | & -1 \\
\end{pmatrix} \Rightarrow \text{former } R_3
\]

\[
\begin{pmatrix}
1 & 2 & 1 & | & 2 \\
0 & -1 & -2 & | & -1 \\
0 & 0 & -1 & | & 1 \\
\end{pmatrix} \Rightarrow \text{former } R_2
\]

THUS

\[
\begin{align*}
z &= -1 \\
y &= \frac{1}{-1}(-1 + 2(-1)) = 3 \\
x &= \frac{1}{1}[2 - 2(3) - 1(-1)] = 2 - 6 + 1 = -3
\end{align*}
\]

THEREFORE, \((x, y, z)^T = (-3, 3, -1)^T\).
PIVOTING TO PRESERVE ACCURACY [4-28B]  

BF-9 §6.2, Ex 1, p372

Look at the following system:

\[
\begin{align*}
0.003x_1 + 59.14x_2 &= 59.17 \\
5.291x_1 - 6.130x_2 &= 46.78
\end{align*}
\]

The exact answer is \((x_1, x_2)^T = (10, 1)^T\).

Suppose we try Gaussian elimination using 4-digit arithmetic and rounding. We transform the system to

\[
\begin{align*}
0.003x_1 + 59.14x_2 &= 59.17 \\
-104300x_2 &= -104400
\end{align*}
\]

Backward substitution yields \((x_1, x_2)^T = (-10.000, 1.001)^T\).

Notice that \(x_1\) is \(-10\) instead of the correct value of \(+10\).
If we use pivoting and re-order the system so that the largest element in column 1 is in row 1, we get the following system:

\[
\begin{align*}
5.291x_1 & - 6.130x_2 = 46.78 \\
0.003x_1 & + 59.14x_2 = 59.17
\end{align*}
\]

Using this (pivoted) system, we get the “correct” answer of \((x_1, x_2)^T = (10, 1)^T\).

Note that \(\text{cond}(A) = 11.298\), which is not that large, but the system is nevertheless sensitive when using Gaussian elimination (and limited precision arithmetic).
SCALING OF LINEAR SYSTEMS

The previous system can be “scaled,” i.e., multiplied by some constant (large or small), either row-wise individually or collectively without mathematically changing the solution. For example, multiplying the first equation (row) in the first example by 10,000 yields the system .

\[
30.0x_1 + 591,400x_2 = 591,700 \\
5.291x_1 - 6.130x_2 = 46.78
\]

Column pivoting does not re-arrange rows since \( a_{11} > a_{21} \), so we end up with the same incorrect answer as before (assuming 4-digit rounding arithmetic).
To avoid this problem, we can use an intermediate step to help the reordering of the equations.

— We “scale” (divide) each row by the largest element (in magnitude) in that row.

— Then we reorder the rows based on the largest element in a specific column.

After the reordering, we can make use of the original values to obtain maximal accuracy, if it is appropriate.
COMPLETE PIVOTING [4-29]

Rearranging columns of the system matrix is equivalent to changing the order of the variables. But this does not really change the linear system. Thus, in searching for an element to replace an unwanted 0, we can search both columns and rows.

**Strategy:** Interchange the unwanted 0 element with the largest element in the remaining submatrix, rearranging both rows and columns.

**Problem:** Since rearranging columns reorders the variables, the column rearrangement must somehow be remembered!

**Comment:** tests done on complete pivoting vs. partial pivoting are inconclusive as to which method is better overall.
SCALING and SCALED PIVOTING

Dividing both sides of an equation by a constant does NOT change the equality at all.

Thus, to minimize round off error induced by large and small elements being together in the same matrix, “scaling” is sometimes employed (as noted earlier). “Scaling” is the conversion of the largest (in magnitude) element in each row to 1 (or to some number between 1 and 10) and “scaling” the other elements in the row correspondingly.

PROBLEM: Sometimes the added work does not help at all.
IMPLEMENTATIONS

Many “canned” routines exist for Gaussian elimination exist. BEWARE that you don’t “reinvent the wheel” and produce a lesser quality product!

Some implementation concerns are described in published books. E.g., does the code actually “rearrange” rows and columns (very time-consuming for large systems) or use an index array to “remember” the re-arrangement instead? How do you “remember” the column re-arrangement?
SINGULAR SYSTEM MATRIX

What are the implications of a singular system matrix, i.e., when \( \det(A) = 0 \)? This occurs when a row/column is a linear combination of other rows/columns.

Two possible results:

**CASE 1**: underdetermined system: The system is not linearly independent — one equation (i.e., both sides of an equal sign) is a linear combination of other equations. RESULT: infinite number of solutions.

**CASE 2**: inconsistent system: The variable side of an equation is a linear combination of other equations but the constant side is NOT the same combination. RESULT: NO solution.
EXTENSIONS

(1) One can solve the matrix-matrix equation \( AX = B \) by modifying methods for solving \( A\vec{x} = \vec{b} \). One simply augments the system matrix to \( (A|B) \) and use the same strategy.

EX. Given 2 systems with the same \( A \) as follows:

\[
\begin{align*}
3x + 2y &= 1 \\
5x - 6y &= 2 \\
3x + 2y &= 5 \\
5x - 6y &= 1
\end{align*}
\]

we have

\[
A = \begin{pmatrix} 3 & 2 \\ 5 & -6 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 5 \\ 2 & 1 \end{pmatrix}
\]

The augmented matrix will be \( (A|B) = \begin{pmatrix} 3 & 2 & 1 & 5 \\ 5 & -6 & 2 & 1 \end{pmatrix} \).
The solution matrix \( X = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \) is such that the first column, 
\[
\begin{pmatrix} x_{11} \\ x_{21} \end{pmatrix},
\]
is the solution for the first system and the second column, 
\[
\begin{pmatrix} x_{12} \\ x_{22} \end{pmatrix},
\]
is the solution for the second system.
(2) **Gauss-Jordan** is a variant of Gaussian elimination. (BF-9 §6.1, prob #12, p 370.) Instead of reducing the system matrix $A$ to an upper triangular matrix, one reduces it to the identity matrix. The augmented column then immediately becomes the solution. The process is basically:

(a) perform the standard **FORWARD ELIMINATION**.

(b) scale each row s.t. $a_{ii} = 1$.

(c) perform backward **ELIMINATION** (and **not** **SUBSTITUTION**).

(3) **Finding** $A^{-1}$ can be easily accomplished by solving the **MATRIX-MATRIX** equation, $AX = I_n$. (A simple way would be to use Gauss-Jordan.) The augmented matrix $(A|I_n)$ is transformed into $(I_n|A^{-1})$. 
The residual is the difference between the actual constant vector and the approximate “constant” vector obtained from the computed solution.

\[ \vec{r}_n = \vec{b} - A\vec{x}_n \]  
(or in some books \( A\vec{x}_n - \vec{b} \))

or

\[ R_n = B - AX_n \]

where \( \vec{x}_n, X_n \) are the computed solution.

Large residual error indicates errors in \( \vec{x}_n \). However, small residual error may NOT necessarily guarantee accuracy of \( \vec{x}_n \).

The Residual is also related to the gradient, i.e., giving the direction of steepest descent. cf. BF-9 §7.6, p 481.