Gaussian Elimination with Partial Pivoting: Examples

Partial? Is there a complete version? Yes, but we don't go there (but give a reader's digest version at the end).

Example. We first look at an example in which round-off errors occur when the leading entry of a system is too small.

$$\begin{aligned} 0.003000x_1 + 59.14x_2 &= 59.17; \\ 5.291x_1 - 6.130x_2 &= 46.78. \end{aligned}$$

Though the numbers look quite contrived, we can spot the exact solution as

$$\begin{aligned} x_1 &= 10.00; \\ x_2 &= 1.000. \end{aligned}$$

Now, let's proceed with 4-digit rounding arithmetic. The multiplier for the second row is

$$m_{21} = \frac{a_{21}}{a_{11}} = \frac{5.291}{0.003000} = 1763.6\overline{6} = 1764$$

after rounding. Performing $(E_2 - m_{21}E_1) \to (E_2)$ and the appropriate rounding yields

$$\begin{aligned} 0.003000x_1 + 59.14x_2 &\approx 59.17; \\ -104300x_2 &\approx -104400, \end{aligned}$$

instead of the exact system

$$\begin{aligned} 0.003000x_1 + 59.14x_2 &= 59.17; \\ -104309.37\overline{6}x_2 &= 104309.37\overline{6}. \end{aligned}$$

The round-off error has already infected $m_{21}a_{13}$ and a_{23}. It is now propagating through the back substitution step

$$x_2 \approx 1.001$$

which is an OK approximation to the true solution $x_2 = 1$. The real trouble arises when back substituting for x_1, i.e.,

$$x_1 \approx \frac{59.17 - (59.14)\,(1.001)}{0.003000} = -10.00.$$

Woah, this is opposite of what we knew for x_1.

So, what's a good remedy? Swapping the rows, we get

$$\begin{aligned} 5.291x_1 - 6.130x_2 &= 46.78; \\ 0.003000x_1 + 59.14x_2 &= 59.17. \end{aligned}$$

Now,

$$m_{21} = \frac{a_{21}}{a_{11}} = \frac{0.003000}{5.291} = 0.0005670$$

after rounding to 4-digit. Then, $(E_2 - m_{21}E_1) \to (E_2)$ gives

$$\begin{aligned} 5.291x_1 - 6.130x_2 &= 46.78; \\ 59.14x_2 &= 59.14. \end{aligned}$$

We retrieve $x_2 = 1.000$ and $x_1 = 10.00$ exactly, avoiding the catastrophe seen earlier.

So, that was a 2×2 system. How about larger ones? What is the general procedure to avoid dividing by small numbers and producing multipliers that lead to large round-off errors?

Gaussian Elimination with Partial Pivoting: Algorithm

Suppose we are now using the k^{th} row to perform Gaussian elimination for the rows below. If a_{kk} is small relative to the entries a_{ij} for $k \leq i \leq n$ and $k \leq j \leq n$, pivoting is performed by selecting an element a_{pq} with a larger magnitude as the pivot and interchanging the k^{th} and p^{th} rows (and swapping the k^{th} and q^{th} columns is also viable, if necessary).

More precisely, this strategy, called **partial pivoting**, is to select an element in the same column (the k^{th} column) that is below the diagonal and has the largest absolute value; we determine the smallest $p \geq k$ such that

$$|a_{pk}| = \max_{k \leq i \leq n} |a_{ik}|$$

(remember, k is fixed here since we are using the k^{th} row to row reduce; the maximum we are searching is over the row index, which means, throw all entries in the k^{th} column below the diagonal).

(1) For $i = 1, \ldots, n$, set $NROW(i) = i$. (Initialize row pointer – keeping the book for swapped rows).

(2) For $i = 1, \ldots, n-1$ (elimination)

(a) Let p be the smallest integer with $i \leq p \leq n$ and

$$|a(NROW(p), i)| = \max_{i \leq j \leq n} |a(NROW(j), i)|$$

where

$$a(NROW(j), i) = a_{NROW(j),i}.$$

This step is checking through the i^{th} column and looking for which entry we should use as the pivot. This step automatically completes the "zero checking" step in the algorithm for Naive Gaussian Elimination.

(b) If $a(NROW(p), i) = 0$, then OUTPUT('no unique solution exists'); STOP.
This step simply find all zeros underneath the diagonal, indicating more than one variable is free and we expect no unique solutions, e.g., for $i = 2$,

$$\left[\begin{array}{ccc|c} 1 & 3 & 1 & 1 \\ 0 & 0 & 5 & 2 \\ 0 & 0 & 9 & 3 \end{array}\right]$$

where we now are focusing on the second row and finding $a(NROW(p), 2) = 0$.

(c) If $NROW(i) \neq NROW(p)$ (if the current leading entry is not the largest), then

$$NCOPY = NROW(i);$$
$$NROW(i) = NROW(p);$$
$$NROW(p) = NCOPY.$$

This step updates the list of swapped rows. You may think about why $NCOPY$ is introduced here.

(d) For $j = i+1, \ldots, n$,

(i) Set

$$m(NROW(j), i) = \frac{a(NROW(j), i)}{a(NROW(i), i)}.$$

(ii) Perform

$$\left(E_{NROW(j)} - m(NROW(j), i) \cdot E_{NROW(i)}\right) \to \left(E_{m(NROW(j))}\right)$$

(3) If $a(NROW(n), n) = 0$, then OUTPUT('no unique solution exists'); STOP.

(4) Set

$$x_n = \frac{a(NROW(n), n+1)}{a(NROW(n), n)}$$

(back substitution).

(5) For $i = n-1, \ldots, 1$, set

$$x_i = \frac{a(NROW(i), n+1) - \sum_{j=i+1}^{n} a(NROW(i), j)\, x_j}{a(NROW(i), i)}.$$

(6) OUTPUT $(x_1, \ldots, x_n)$ .STOP.

Remark 1. You may check the row reduced matrix after completing step 3. If you print out the matrix, you may notice that it is not exactly upper triangular. However, if you do print out the matrix with row numbers following the swapped indices, it must be upper triangular.

Remark 2. It is costly to **actually** swap rows. Instead, it is much less expensive to keep a list of swapped indices, and call upon them when needed. As long as this list is updated, we can proceed without affecting the solutions.

Remark. (Recap)

The main point of partial pivoting is to reduce round-off errors, caused by multiplying possibly large multipliers (obtained from dividing small pivots). We avoid these errors by choosing the pivot carefully for each row.

Complete pivoting involves checking for the maximal element in magnitude in both the k^{th} row and k^{th} column. Therefore, the procedures will be a mix of column and row swaps. Row swap may not affect the order of solution $(x_1, \ldots, x_n)$ but column does, namely, for a swapped p^{th} and q^{th} column, we must swap the indices in the solution

$$(x_1, \ldots, x_p, \ldots, x_q, \ldots x_n) \to (x_1, \ldots, x_q, \ldots, x_p, \ldots x_n)\,.$$

Performing complete pivoting thus involves keep another list of swapped columns that will eventually be used to reorder the output at the end of back substitution.

Noting all the additional procedures needed for complete pivoting, we recommend this method only for systems that demand so high of an accuracy that this increase in execution time can be justified.