Determinants

Numerical evaluation of a determinant (using row operations)

This section ties together results from previous sections, and gives us another way to compute determinants. The key results (that we’ve already proven) are

- The determinant of an upper (or lower) triangular matrix is the product of its diagonal entries
- The observation that elementary row ops have certain effects on the determinant; and that in particular \( cR_j + R_i \rightarrow R_i \) leaves the determinant unchanged

The conclusion is that, since cofactor expansion can be a long and messy process for anything larger than a 3 \( \times \) 3, it’s nice to know that it’s safe to perform Gaussian elimination carefully to introduce zeros to a matrix, making it easier to compute the determinant.

So, the only thing to do here is look at an example of using the row ops to introduce zeros before computing the determinant. There is one thing I do want to emphasize with this approach: you have to do it by hand. rref won’t help you here, because rref may involve row swapping, or multiplying by a constant, both of which have an effect of the determinant. If you perform those row operations, you need to keep track of the effects (row swap = sign change, multiply by constant = determinant gets multiplied by that constant). And you do have to be careful with how you recover the original determinant.

**Example:** For the matrix

\[
A = \begin{bmatrix}
2 & 4 & -1 & 7 \\
0 & 1 & -1 & 4 \\
3 & -1 & 5 & 0 \\
0 & 0 & 1 & -3
\end{bmatrix}
\]

- Use elementary row ops to produce a matrix in upper triangular form. Keep track of any row swaps or multiplications.
- Call your final result \( B \).
- Find \(|B|\) by computing the product of the diagonal entries.
- Then, recover \(|A|\) from \(|B|\).
Note of interest:

The process of Gaussian elimination is an $O(n^3)$ process. (“On the order of $n^3$”). This means that if the number of operations needed to reduce an $n \times n$ matrix to upper triangular form is counted, the result can be shown to be a cubic polynomial: number of ops = $an^3 + bn^2 + cn + d$. For large matrices in particular, the $n^3$ part of the formula is the dominant term, which is why algorithms are measured loosely with the $O$ notation - it’s not the exact number of operations that matters so much as the largest power. Consider a 100 by 100 matrix - Gaussian elimination would take (roughly) $100^3$ or one million operations to reduce the matrix. (In the world of numerical methods, by the way, 100 by 100 is a small matrix!). If there were an algorithm that could reduce a matrix that was only $O(n^2)$, it would only take $100^2 = 10000$ operations - a huge saving. There isn’t, by the way - it’s been proven that in general, $O(n^3)$ is the best you can do. So, computing a determinant by using Gaussian elimination is $O(n^3)$.

Now, how does that stack up to cofactor expansion? It can be shown that cofactor expansion is $O(n!)$. For a 100 by 100 matrix, the number of operations is roughly 100! ... which is about, oh, $9 \times 10^{157}$. $10^{157}$ vs. one million. Needless to say, cofactor expansion is not used to compute determinants as a general rule, and you really start noticing the effects as early as the 4 by 4 matrices. It’s still a useful process to know, since there is one context where you work with a lot of 3 by 3 determinants - vectors in three dimensional space.