Worksheet: A bad way to compute determinants

It turns out that in serious tasks on a computer one does *not* use determinants. Certainly we will *not* compute them the way you were taught! The reasons why this is a bad idea, explained on this worksheet, is a good example of numerical analysis thinking.

Definition. The *determinant* of a square matrix is a number computed from the entries:

- For a 1×1 matrix: $\det([a_{11}]) = a_{11}$.
- For a 2×2 matrix:
$$\det\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right) = a_{11}a_{22} - a_{12}a_{21}.$$
- For a 3×3 matrix it can be computed by multiplying certain submatrices below the first row, the *minors*, by entries in the first row, and then combining the results using alternating signs:
$$\det\left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\right) = +a_{11}\det\left(\begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}\right) - a_{12}\det\left(\begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{bmatrix}\right) + a_{13}\det\left(\begin{bmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}\right).$$
- For an $n \times n$ matrix A it can be computed recursively by multiplying the determinants of the minors by entries of the first row and alternating signs:
$$\det(A) = \sum_{j=1}^{n} (-1)^{j-1} a_{1j} \det(A_{1j}),$$
where A_{1j} is defined here as the $(n-1) \times (n-1)$ matrix which remains after removing the first row and jth column.

The following fact is proven in any linear algebra class; you may use it at any time.

Lemma. Given a square matrix A, suppose we construct matrix B by exchanging the ith and jth rows of A, where $i \neq j$. Then $\det(B) = -\det(A)$. The same is true if we swap columns.

(a) Use the above facts to compute the determinants by hand:

$$\det\left(\begin{bmatrix} 0 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right) =$$

$$\det\left(\begin{bmatrix} 4 & 2 & 5 & 6 \\ 0 & 2 & 0 & -1 \\ 7 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}\right) =$$

If possible, confirm your answers using MATLAB.

(b) Count the number of multiplications needed to compute the determinant of a generic 2×2 matrix. Do the same for a 3×3 matrix and a 4×4 matrix. (*You don't need to count the $\pm$ decisions as multiplications.*)

(c) Write a pseudocode for a determinant function *that calls itself*, that is, that is recursive. In fact, fill in blanks below to get a functional MATLAB code.

```
function z = mydet(A)
% MYDET [put some documentation here if you want]

n = size(A,1);
if size(A,2) ~= n,  error('only works on square matrices'),  end

if n == 1
    z = A(1,1);
elseif n == 2
    z =
else





end
```

(d) Explain why the number of multiplications needed for `mydet()` to compute an $n \times n$ determinant exceeds $n!$. Assuming that this is the case, argue that a computer capable of a billion floating-point operations per second—way too optimistic for your laptop—would take more time than the age of the universe to compute `mydet(A)` of a 30×30 matrix.

(e) If you have access to MATLAB, compute the determinant of a random 50×50 matrix:

```
>> A = randn(50,50);
>> det(A)
```

How long did this take? (*More generally, how do you time computations in* MATLAB?)

(f) Apparently the implementation of `det()` is not the same as `mydet()`. Fortunately, the following fact is also true:

Lemma. $\det(AB) = \det(A)\det(B)$.

Use this and the LU decomposition idea (section 7.2.2) to suggest how `det()` might be doing it. Confirm this online by finding the appropriate MATLAB technical documentation page.