Math 105AL – Lab 5

Haoyu Zhang, 25327328

1 Objective

In this lab, I implemented and compared two methods to compute the determinant of a square matrix: 1. The recursive cofactor expansion method, the "detA_recursive.m" based on the mathematical definition of the determinant; 2. The Gaussian Elimination method, "detA_GE.m" which performs row reduction to upper-triangular form.

I examined their performance, computational efficiency, and numerical stability by measuring the time for random matrices of increasing sizes and testing the round-off sensitivity on a large diagonal matrix.

2 Procedure

For Q1, I implemented the function "detA_recursive.m", which recursively expands along a row or column and computes minor until reaching base cases of one by one or 2 by 2 matrices.

For Q2, I created the function "detA_GE.m" to compute the determinant using Gaussian Elimination Algorithm with partial pivoting. Each row swap flips the determinant's sign. The final result is the product of the diagonal entries of the upper-triangular matrix.

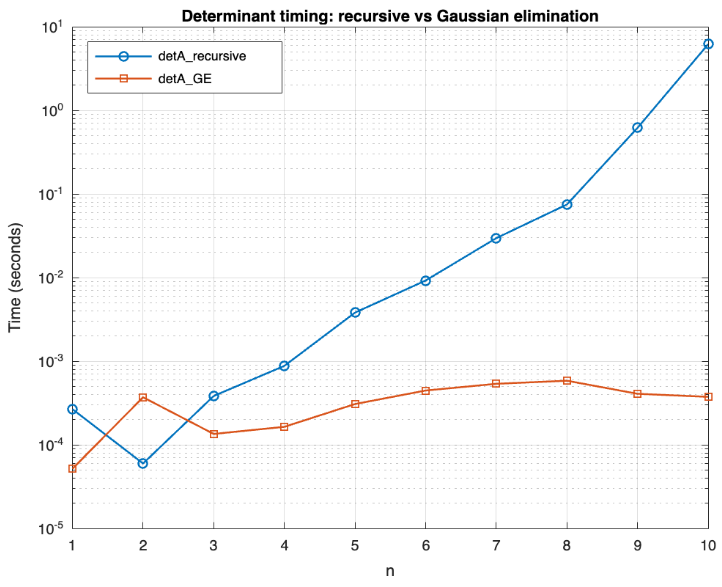
For Q3, I had two experiments: 1. Timing: for n = 1, 2, …, 10, I generated random matrices A = randn(n, n). I used "tic" and "toc" to record the computation time for both functions and plotted time and n graph; 2. Round-off sensitivity: for n = 1000, I defined a diagonal matrix D = labda I with labda = 0.01. I computed "detA_GE(D)" and analyzed the effects of numerical underflow.

3 Results

The timing results showed that the recursive method's grows extremely fast, reaching over 6s at n=10. However, the GE method remains around 0.0005s across all tested sizes.

```
--- Timing experiment (randn matrices) ---
n= 1 | time(rec)=0.0002s  time(GE)=0.0001s  | det diff=0.000e+00
n= 2 | time(rec)=0.0000s  time(GE)=0.0001s  | det diff=0.000e+00
n= 3 | time(rec)=0.0000s  time(GE)=0.0001s  | det diff=9.714e-17
n= 4 | time(rec)=0.0001s  time(GE)=0.0001s  | det diff=3.886e-16
n= 5 | time(rec)=0.0002s  time(GE)=0.0001s  | det diff=0.000e+00
n= 6 | time(rec)=0.0013s  time(GE)=0.0001s  | det diff=7.105e-15
n= 7 | time(rec)=0.0093s  time(GE)=0.0003s  | det diff=1.776e-15
n= 8 | time(rec)=0.0809s  time(GE)=0.0005s  | det diff=6.821e-13
n= 9 | time(rec)=0.6275s  time(GE)=0.0005s  | det diff=1.377e-13
n=10 | time(rec)=6.1991s  time(GE)=0.0005s  | det diff=1.847e-13
```

The Plot demonstrates a nearly exponential curve for recursion and a likely flat curve for GE one. The determinant values from both methods matched closely. I think because of the floating-point rounding, their differences are around 10^-13.



For the round-off sensitivity test, the determinant of D was det(D) = (0.01)^1000 = 10^-2000, which underflows to zero in double precision. The log one was approximately to -4605.17.

```
--- Round-off sensitivity (diagonal D) ---
n = 1000, lambda = 0.01
detA_GE(D)          = 0   (double precision; underflow expected)
sign(det(D))        = 1
log(|det(D)|)       = -4605.17019  ->  |det(D)| = exp(logAbsDet) ~ 10^{-2000.0}
Is D invertible?  YES (all diagonal entries nonzero)

Note: The exact determinant is lambda^n = (1/100)^{1000} = 10^{-2000},
      which is far below double-precision min (~5e-324), so it underflows to 0.
```

4 Conclusion

In conclusion, the recursive determinant algorithm implements the definition. Because of the factorial complexity, it becomes computationally infeasible. However, the Gaussian Elimination one with O(n^3) complexity is much faster and keeps the accuracy.

The round-off test showed that determinants with very small magnitude can underflow to zero. Although the matrix is invertible, it can also be realized. I think the logarithmic determinant computation, log(det), is more stable for complex situations.

5 Challenges and Bugs

1. There's a unrecognized function or variable 'ternary' mentioned from MATLAB in Question 3 when I was coding it. I thought about it and delete them. Instead, I wrote a "if else" logical structure to solve the problem. (On the website, it shows because helper functions at the bottom of scripts require function-file structure.)
2. There are some indexing errors in the recursive version when I was coding.
3. Some timing results at small n were inconsistent due to floating-point timing.