Week3 Computer Project

樊泽羲 2200010816

Experiment1: Piviot v.s. No-Piviot

1. Problem Statement

We investigate the solution of the linear system $A\mathbf{x} = \mathbf{b}$ for a matrix A and vector $\mathbf{b}$ defined as:

$$A = \begin{bmatrix} 6 & 1 & & & \\ 8 & 6 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 8 & 6 & 1 \\ & & & 8 & 6 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 7 \\ 15 \\ \vdots \\ 15 \\ 14 \end{bmatrix} \tag{1}$$

The exact solution is known to be $\mathbf{x}^* = (1, 1, \ldots, 1)^T$. The objective is to compare the accuracy of two algorithms for solving this system as its dimension n grows:

1. **Gaussian Elimination without Pivoting**
2. **Gaussian Elimination with Partial (Column) Pivoting**

The accuracy is measured by calculating the l^2 and l^∞ error norms of the difference between the numerical solution $\mathbf{x}$ and the exact solution $\mathbf{x}^*$.

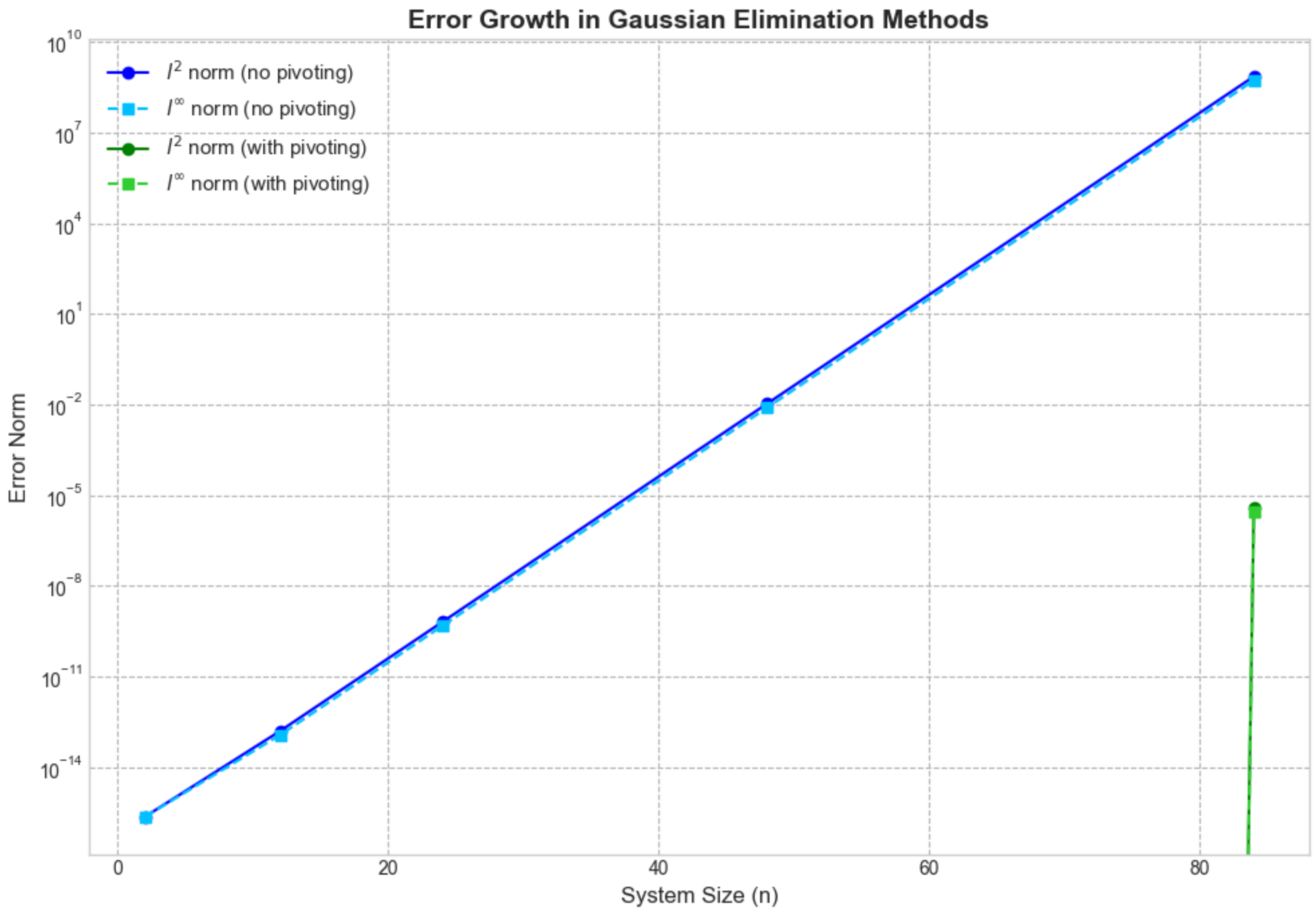
2. Numerical Results

The system was solved for $n = \{2, 12, 24, 48, 84\}$. The resulting error norms are presented in Table 1.

Table 1: Comparison of numerical errors for Gaussian elimination with and without partial pivoting.

n	Without Pivoting (l^2 error)	With Pivoting (l^2 error)	Without Pivoting (l^∞ error)	With Pivoting (l^∞ error)
2	2.2204e-16	0.0000e+00	2.2204e-16	0.0000e+00
12	1.5374e-13	0.0000e+00	1.1369e-13	0.0000e+00
24	6.2965e-10	0.0000e+00	4.6563e-10	0.0000e+00
48	1.0564e-02	0.0000e+00	7.8120e-03	0.0000e+00
84	7.2594e+08	3.7828e-06	5.3684e+08	2.7974e-06

To better visualize the trend, the data can be plotted on a log-linear scale, where the exponential growth of error in the non-pivoted method becomes apparent as a straight line.



3. Discussion of Results

The data reveals a stark contrast in the performance and stability of the two methods.

3.1 Gaussian Elimination without Pivoting: A Case of Instability

The method without pivoting exhibits classic signs of numerical instability.

- For small n (2, 12, 24), the error is small, though it grows steadily.
- A dramatic shift occurs between $n = 24$ and $n = 48$, where the error jumps from $\approx 10^{-10}$ to $\approx 10^{-2}$. This indicates that the accumulation of floating-point round-off errors is beginning to corrupt the solution significantly.
- By $n = 84$, the error has exploded to $\approx 10^8$. The resulting numerical solution is completely meaningless and has no relation to the true solution.

This instability is caused by the structure of matrix A. At each step of the forward elimination, the algorithm calculates a multiplier $m_{ij} = a_{ij}/a_{ii}$ to eliminate elements below the diagonal. In our matrix, the first step involves a multiplier of $m_{21} = a_{21}/a_{11} = 8/6 \approx 1.33$. Because this multiplier has a magnitude greater than 1, any round-off error present in the first row is amplified when it is used to modify the second row. This process repeats and compounds at each step, leading to exponential error growth.

3.2 Gaussian Elimination with Partial Pivoting: A Stable Approach

The partial pivoting strategy fundamentally changes the algorithm's behavior. Before each elimination step, the algorithm inspects the current column and swaps rows to ensure that the largest absolute value is in the pivot position (the diagonal).

- In our matrix, at the first step, the algorithm would compare $|a_{11}| = 6$ and $|a_{21}| = 8$. It would swap the first two rows, making 8 the pivot element.
- The new multiplier would be $m_{21} = 6/8 = 0.75$.

- By ensuring the pivot is always the largest element in its column, the partial pivoting strategy guarantees that all multipliers will have a magnitude $|m_{ij}| \leq 1$. This prevents the amplification of round-off errors.

The results in Table 1 confirm this stability. The error remains exceptionally low across all tested system sizes. The "0.0000e+00" values likely represent errors at the level of machine precision (around 10^{-16}), which are too small to be displayed by the output format. Even at $n = 84$, the error is on the order of 10^{-6}, which is astronomically smaller than the 10^8 error from the non-pivoted method, demonstrating its superior stability and accuracy.

4. Conclusion and Perspective on Gaussian Elimination

This analysis leads to a clear and definitive conclusion: Standard Gaussian elimination is not a generally reliable algorithm for solving systems of linear equations in a computational setting. While mathematically correct, it is numerically unstable if the pivot elements are small relative to the other elements in their column. For certain matrices, like the one studied here, this leads to a catastrophic loss of accuracy.

The inclusion of a partial pivoting strategy is an essential and transformative modification. By preventing the use of small pivots and ensuring multipliers do not exceed 1 in magnitude, it suppresses the growth of round-off errors and renders the algorithm numerically stable. The small computational overhead of searching for a pivot at each step is an insignificant price to pay for the accuracy and reliability it guarantees.

Therefore, for any practical scientific or engineering problem, a pivoted version of Gaussian elimination (or other stable methods like LU decomposition with pivoting) should always be used. Relying on a non-pivoted implementation is an unnecessary risk that can lead to completely invalid results.

Experiment2: Possion Solvers Comparison

1. Problem Statement

The numerical task is to solve the 2D Poisson equation with Dirichlet boundary conditions on a unit square $\Omega = (0,1) \times (0,1)$:

$$\begin{cases} -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f(x,y), & (x,y) \in \Omega, \\ u = 0, & (x,y) \in \partial\Omega, \end{cases} \tag{2}$$

where the source function is $f = 2\pi^2 \sin(\pi x) \sin(\pi y)$ and the exact solution is $u = \sin(\pi x) \sin(\pi y)$.

Using a five-point central difference scheme on a uniform grid with spacing $h = 1/N$, the problem is transformed into a system of linear equations $A_h U_h = h^2 F_h$. The resulting matrix A_h is a large, sparse, symmetric positive-definite, and block-tridiagonal matrix of size $(N-1)^2 \times (N-1)^2$. This specific structure, where non-zero elements are clustered around the main diagonal, is key to its efficient solution.

The accuracy of a numerical solution U_h is measured by the discrete error norm:

$$e_N = h \left(\sum_{i=1}^{N+1} \sum_{j=1}^{N+1} |u_{i,j} - U_{i,j}|^2 \right)^{1/2} \tag{3}$$

2. Numerical Results

The linear system was solved for grid sizes $N = 16, 32, 64, 128, 256$ using three methods. The computation time and resulting error are compiled in Table 1.

Table 1: Comparison of Solver Performance and Accuracy

N	System Size	Method	Time (s)	Error (e_N)
16	225	Gaussian Elim.	0.067461	1.609482e-03
16	225	Cholesky (LDL^T)	0.024481	1.609482e-03
16	225	Banded Cholesky	0.000611	1.609482e-03
32	961	Gaussian Elim.	0.089273	4.017888e-04
32	961	Cholesky (LDL^T)	0.017111	4.017888e-04
32	961	Banded Cholesky	0.079579	4.017888e-04
64	3969	Gaussian Elim.	0.756738	1.004109e-04
64	3969	Cholesky (LDL^T)	0.884303	1.004109e-04
64	3969	Banded Cholesky	0.461064	1.004109e-04
128	16129	Gaussian Elim.	19.129262	2.510046e-05
128	16129	Cholesky (LDL^T)	20.525627	2.510046e-05
128	16129	Banded Cholesky	0.390676	2.510046e-05
256	65025	Gaussian Elim.	inf	nan
256	65025	Cholesky (LDL^T)	inf	nan
256	65025	Banded Cholesky	1.402262	6.274973e-06

3. Discussion of Results

3.1 Accuracy and Error Convergence

The results in Table 1 show that the numerical error e_N is identical for all three methods at each grid size N. This is expected, as they are all direct solvers that should produce the same solution up to floating-point precision.

More importantly, the error demonstrates a clear convergence trend. As we refine the grid by doubling N (and halving the step size h), the error decreases by a factor of approximately four. For instance, from $N = 64$ to $N = 128$, the error drops from 1.004×10^{-4} to 2.510×10^{-5}, a ratio of 3.99. This confirms that the five-point difference scheme is **second-order accurate**, with the error e_N being proportional to h^2.

3.2 Computational Performance and Scalability

The primary distinction between the methods lies in their computational cost, which includes both time and memory usage.

Gaussian Elimination and Dense Cholesky:
These two methods treat the coefficient matrix A_h as a dense structure, ignoring its sparsity.

- **Time Complexity:** For an $M \times M$ system (where $M = (N-1)^2$), their computational complexity is $O(M^3)$. This leads to a rapid increase in solution time. For example, between $N = 64$ ($M = 3969$) and $N = 128$ ($M = 16129$), the system size quadruples, and the runtime for Gaussian Elimination explodes from ~0.76s to ~19.1s, a factor of ~25.
- **Memory Complexity:** The memory required to store the full matrix is $O(M^2)$. For $N = 256$, the system size is $M = 65025$, requiring storage for over 4.2 billion floating-point numbers. This exceeds the memory capacity of most standard computers, which is why the solvers failed (`inf` time, `nan` error) for this case.

Banded Cholesky:
This specialized solver is designed to exploit the banded structure of the matrix, storing only the non-zero diagonals.

- **Time Complexity:** The complexity for a banded solver is approximately $O(M \cdot k^2)$, where k is the matrix half-bandwidth (here, $k = N - 1$). This is significantly better than $O(M^3)$. The practical benefit is dramatic: at $N = 128$, the banded solver takes only **0.39 seconds** compared to ~20 seconds for the dense solvers—a **50x speedup**.
- **Memory Complexity:** The memory usage is only $O(M \cdot k)$, which is vastly more efficient. This efficiency is why the banded solver could easily handle the $N = 256$ case, solving a system with over 65,000 unknowns in just **1.4 seconds**, while the other methods failed entirely.

4. Conclusion

This analysis highlights a fundamental principle in numerical linear algebra: algorithms should be chosen to match the structure of the problem.

1. Accuracy is Method-Independent: All direct solvers tested produced equally accurate results, with the error being governed by the underlying finite difference discretization, which was shown to be second-order accurate.
2. Dense Solvers Are Impractical: General-purpose solvers like Gaussian Elimination are robust but inefficient for large, sparse systems. Their prohibitive time and memory complexity make them unsuitable for solving PDEs or other problems that generate structured matrices.
3. Exploiting Structure is Key: The Banded Cholesky method, by leveraging the specific banded and symmetric positive-definite nature of the matrix, provides an enormous performance advantage. It is not only orders of magnitude faster but is also the only feasible option for achieving high-resolution (large N) solutions on standard hardware.

For scientific and engineering problems that lead to structured linear systems, the use of specialized solvers is not just an optimization—it is an absolute necessity.