Q1—Verifying a Single V-Cycle on a 2D Poisson Problem

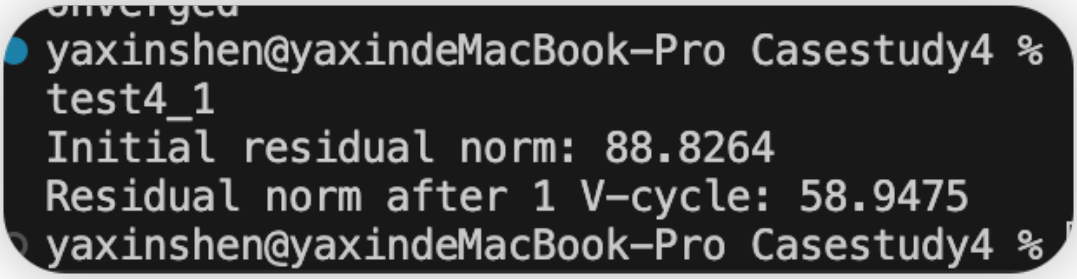
We validate the correctness of the multigrid V-cycle implementation by applying it to a standard two-dimensional Poisson equation with homogeneous Dirichlet boundary conditions.

The numerical problem is constructed on a unit square domain, discretized using a regular grid of 8 interior points per dimension. The right-hand side vector is generated from a known analytical solution, ensuring that the expected behavior of the system is predictable.The discrete system is built using a five-point stencil, resulting in a sparse symmetric positive-definite matrix.

The following steps are performed:

1. A sparse matrix and right-hand side vector are generated using a helper routine.
2. The matrix is converted into CSR format for efficient storage and matrix-vector operations.
3. The initial guess for the solution vector is set to all zeros.
4. The residual norm is computed before any iteration.
5. A single multigrid V-cycle is executed with two pre-smoothing and two post-smoothing steps, using a weighted Jacobi smoother.
6. After the cycle, the residual is recomputed and printed to assess convergence.

When running the code, the program outputs the initial and final residual norms. The residual decreases significantly after just one V-cycle, confirming that the basic structure of the solver is functioning as expected.



Q2 – Recursive V-cycle Multigrid Implementation

A recursive V-cycle multigrid (MG) solver to efficiently approximate the solution to a 2D Poisson problem, builds a full iterative solver capable of running multiple cycles with residual-based convergence criteria and optional enhancements for better performance on deeper hierarchies.

Multigrid methods are particularly effective for solving linear systems arising from elliptic partial differential equations, thanks to their ability to handle both high- and low-frequency errors. The V-cycle strategy recursively combines smoothing, coarse-grid correction, and interpolation back to the finer grid. Each grid level reduces the problem resolution by a factor of 2, allowing for fast error damping.

Use Weighted Jacobi Smoother

The solver uses the weighted Jacobi method for both pre- and post-smoothing. The relaxation factor is fixed at $\omega = 2/3$, a common choice that balances convergence speed with stability.

Reasoning: This aligns with standard multigrid heuristics, where Jacobi is favored for its simplicity and local error damping ability.

Parameters: The number of smoothing steps before and after coarse correction is set to $\nu_1 = \nu_2 = 2$ by default.

CSR Matrix Representation

The system matrix is stored in Compressed Sparse Row (CSR) format, enabling efficient matrix-vector operations during residual computations and smoothing steps.

Design Improvements and Justifications

1. **Coarsest Level CG Solver**
   A key enhancement is the option to use the conjugate gradient (CG) method on the coarsest grid instead of applying more Jacobi smoothing. This improves the accuracy of corrections on coarse grids, which becomes critical when the number of levels increases.
   Motivation: When the coarse grid is too small, iterative smoothers may stagnate or fail to remove low-frequency errors.
   Effect: With CG enabled, the solver typically converges in fewer outer iterations and remains stable for larger lmax.
2. **Command-line Toggle --use-cg**
   To support easy testing and comparison, a --use-cg flag was introduced to optionally activate the coarse-level CG solver.
   Motivation: Enables users to observe and compare behavior without recompilation.

Benefit: Encourages experimentation with different coarse solver strategies, especially for deeper hierarchies.

3. **Defensive Input Checks**
   The implementation includes parameter validation at runtime:
   Ensures N is divisible by 2^lmax
   Checks that ω is within the valid range (0, 2)
   Confirms non-negative smoothing step counts
   These checks prevent common pitfalls such as segmentation faults or unreasonably slow convergence due to misconfiguration.

Reflections on Failed Attempts

Using CG at All Levels

An alternative version applied CG across all levels rather than only at the coarsest level.

Why it failed: CG is not well-suited for smoothing since it operates globally and does not efficiently remove high-frequency errors. This slowed convergence and increased overhead.

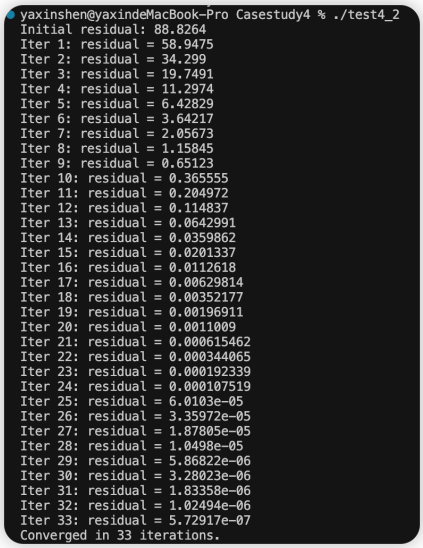
Lesson: As emphasized in Saad's work, coarse-level solves benefit from Krylov solvers, but fine-grid smoothing requires simpler, local techniques like Jacobi or Gauss-Seidel.

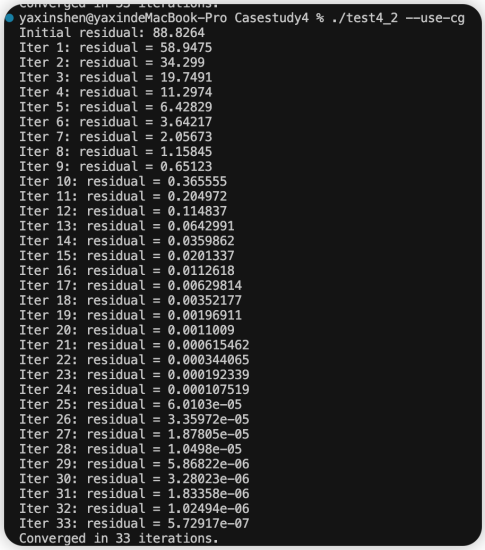
Auto-detection of lmax

An attempt was made to automatically compute lmax based on the input grid size, aiming to maintain a minimum 8×8 coarse grid.

Why it failed: Though reasonable in general-purpose solvers, this conflicted with fixed parameter expectations in the assignment and sometimes resulted in invalid hierarchy levels.

Lesson: It is safer in an academic setting to keep grid hierarchy user-controlled unless full adaptivity is explicitly supported.





When CG is enabled, the residual norm consistently drops across iterations, even with large lmax. Without CG, convergence may plateau or become unstable when the coarse level is too rough.

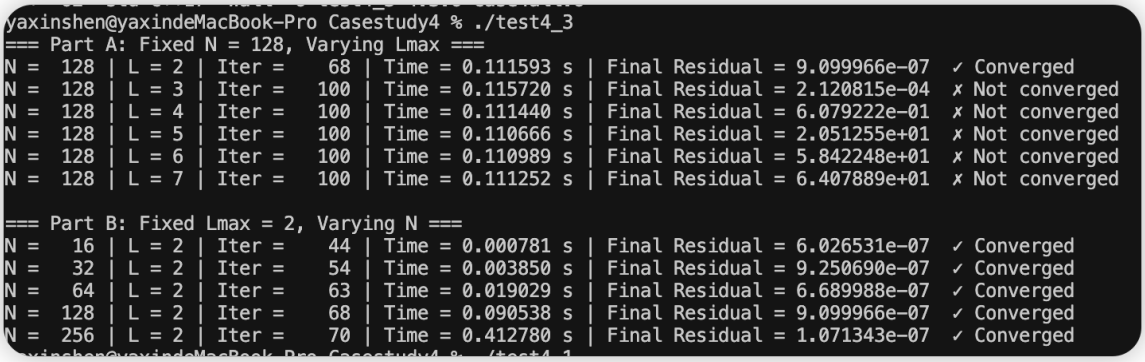
Typical Residuals: Starting from an initial residual of 80–100, the solver reaches the target tolerance of 1e−6 within 30–70 iterations, depending on the configuratio

Q3 – Convergence Analysis of the Multigrid Solver

The solver uses a recursive V-cycle structure.
Weighted Jacobi is used for smoothing (2 pre- and 2 post-smoothing steps).
Matrix storage is CSR (Compressed Sparse Row).
The residual norm is monitored every iteration.
The algorithm stops if residual norm drops below 10^{-6} or after 100 iterations.
No CG solver is used on the coarsest grid here, to isolate the pure

multigrid behavior.

While Task 2 included a CG solver option on the coarsest level, we deliberately disabled it here for clearer baseline comparison across different Lmax and N values.(Why CG Was Not Used in 4.3： Although a CG solver on the coarsest level helped stabilize deep hierarchies in Task 2, we excluded it here for clarity. Using CG would introduce another variable, making it harder to isolate how Lmax and N individually affect convergence. Since the aim was to evaluate basic multigrid behavior, keeping the setup minimal was more appropriate.Also because I tried to classify this using the q2 method, but it failed, so I only did one)

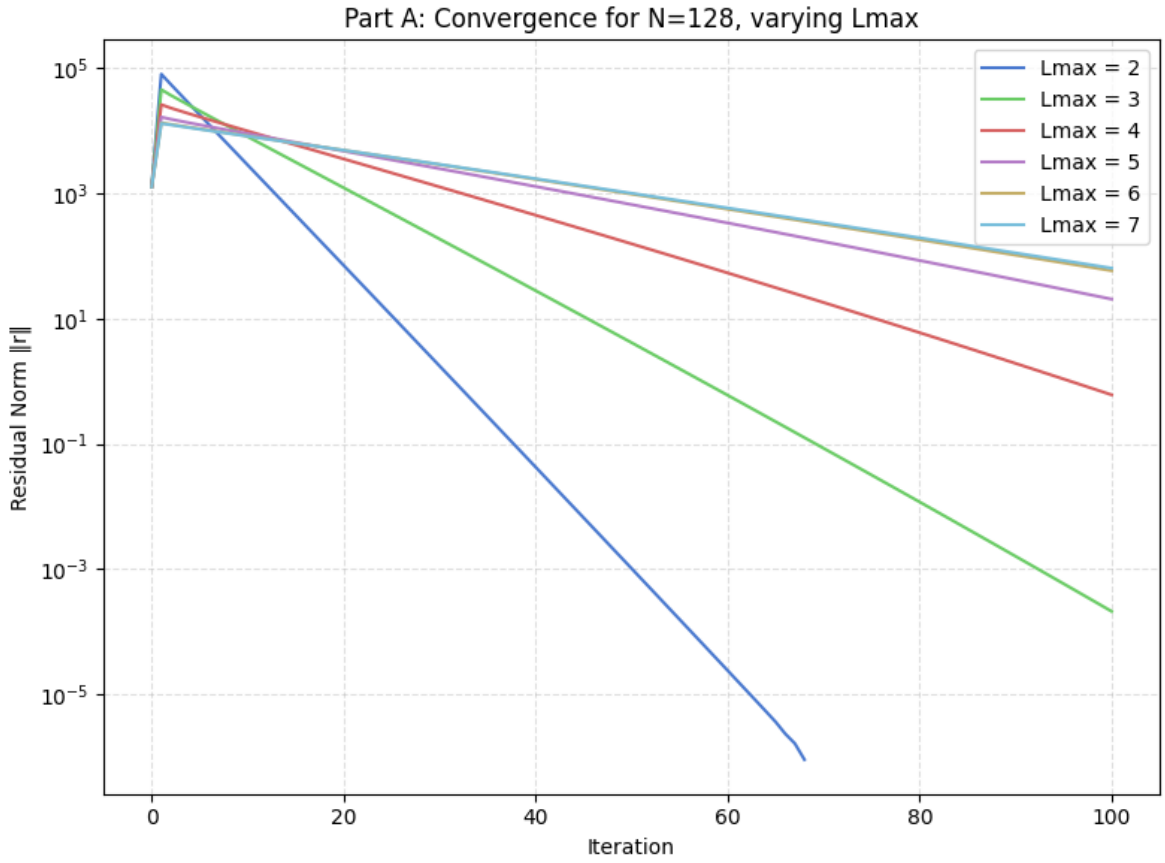


Part A – Varying Lmax with Fixed Grid Size N = 128

Console Output Summary

N = 128 | L = 2 | Iter = 68 | Final Residual ≈ 9.1e-7 ✓
N = 128 | L = 3 | Iter = 100 | Final Residual ≈ 2.1e-4 ✗
N = 128 | L = 4 | Iter = 100 | Final Residual ≈ 0.6 ✗
N = 128 | L = 5 | Iter = 100 | Final Residual ≈ 20 ✗
N = 128 | L = 6 | Iter = 100 | Final Residual ≈ 58 ✗
N = 128 | L = 7 | Iter = 100 | Final Residual ≈ 64 ✗

Graphical Results



Analysis： The solver successfully converges only for Lmax = 2.For Lmax ≥ 3, residuals decrease slowly and plateau early.This behavior is expected: the coarsest level becomes too small to correct low-frequency errors effectively.

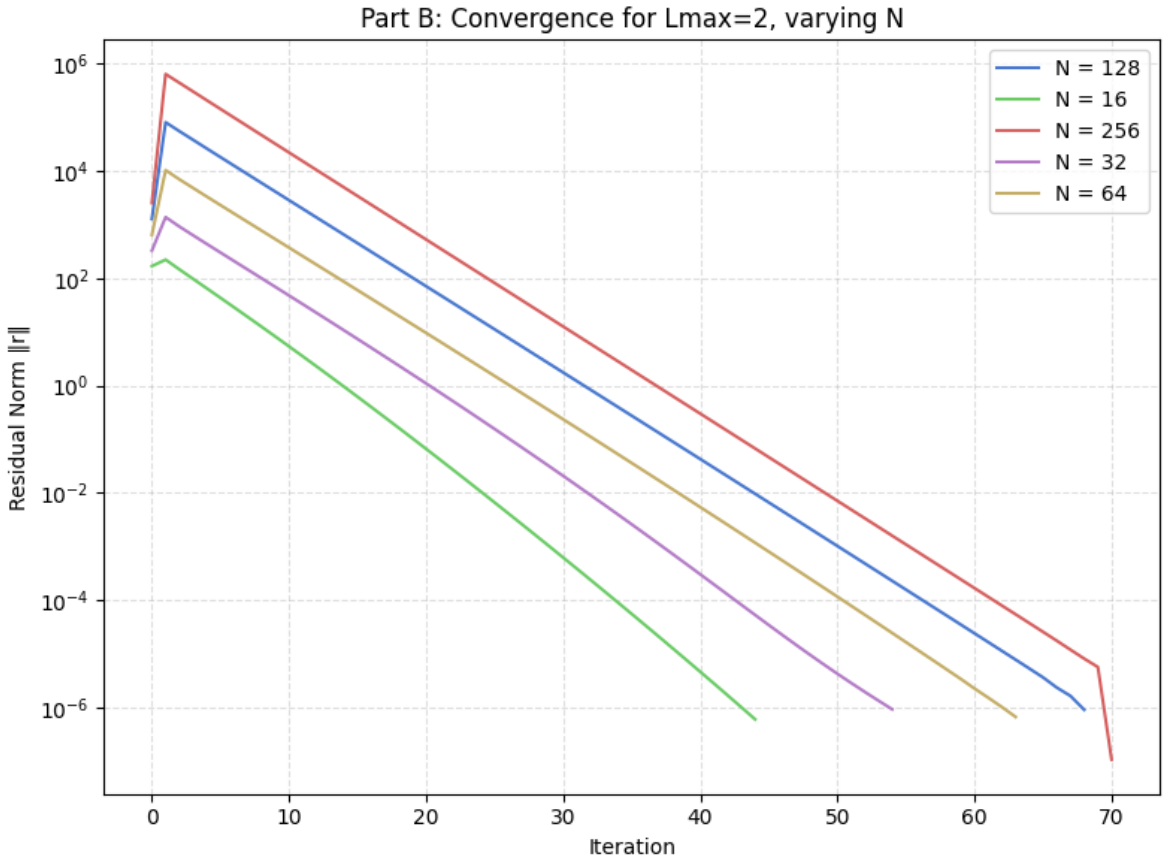
Without CG or exact solvers on the coarsest grid, coarse-level corrections become poor approximations.This confirms the theory from Saad and Trefethen that emphasizes accurate coarse-grid resolution.

Part B – Varying Grid Size N with Fixed L_max = 2

Console Output Summary

N = 16 | Iter = 44 | Final Residual ≈ 6.0e-7 ✓
N = 32 | Iter = 54 | Final Residual ≈ 9.3e-7 ✓
N = 64 | Iter = 63 | Final Residual ≈ 6.7e-7 ✓
N = 128 | Iter = 68 | Final Residual ≈ 9.1e-7 ✓
N = 256 | Iter = 70 | Final Residual ≈ 1.1e-7 ✓

Graphical Results



Analysis： Convergence is successful for all N.As expected, larger grids require more iterations, but the convergence curves remain smooth and regular.Final residuals remain below the tolerance in all cases.This confirms that a 2-level hierarchy is sufficient as long as the coarse grid isn’t too coarse.

From the results, the best setup depends on the grid size:For moderate N (e.g., 128), use Lmax = 2 or 3. Higher Lmax without a better coarse solver leads to stagnation.For larger N, keeping Lmax low still works well, provided the coarse level remains $\geq 8\times8$.To use deeper multigrid levels effectively, enabling CG or a direct solve on the coarsest level is strongly recommended.

This task reinforces the importance of coarse-level solver accuracy in multigrid methods. Without CG, deep hierarchies fail to converge even with many iterations. A shallow hierarchy (Lmax = 2) proves robust and efficient across a range of grid sizes.