Case Study 4: MG

Aidan Coleman (colemaa3@tcd.ie)

May 2025

1 4.1 Basics: The Poisson problem, again.

Consider the Poisson problem on a unit square

$$x = (x_1, x_2) \in \Omega = (0,1)^2$$

with source function $f : \Omega \to \mathbb{R}$. We seek $u : \Omega \to \mathbb{R}$ such that

$$\begin{cases} -\Delta u(x) = f(x), & x \in \Omega, \\ u(x) = 0, & x \in \partial\Omega, \end{cases}$$

where

$$\Delta u(x) \;\equiv\; \frac{\partial^2 u}{\partial x_1^2}(x) \;+\; \frac{\partial^2 u}{\partial x_2^2}(x).$$

We can symmetric finite-difference approximations to the second derivatives in the Laplacian to rewrite this boundary-value problem as the linear system

$$A\,y \;=\; b,$$

where y is the vector of approximations

$$y_k = u(i\,h,\; j\,h), \qquad 0 \le i, j \le N,$$

on a regular 2D grid of spacing $h = 1/N$. We do this by considering the approximation as follows: Introduce a uniform grid of mesh-size

$$h = \frac{1}{N}, \qquad x_{i,j} = (i\,h,\; j\,h), \quad i, j = 0, 1, \ldots, N,$$

and let

$$u_{i,j} \approx u(x_{i,j}),$$

with the Boundary Conditions:

$$u_{i,j} = 0 \quad \text{whenever } i = 0, N \text{ or } j = 0, N.$$

Symmetric Finite Difference: Five-point stencil for Δu

On an interior node (i, j), $1 \le i, j \le N-1$, approximate each second derivative by the centered difference

$$\frac{\partial^2 u}{\partial x_1^2}\Big|_{i,j} \approx \frac{u_{i+1,j} - 2\,u_{i,j} + u_{i-1,j}}{h^2}, \qquad \frac{\partial^2 u}{\partial x_2^2}\Big|_{i,j} \approx \frac{u_{i,j+1} - 2\,u_{i,j} + u_{i,j-1}}{h^2}.$$

Hence the discrete Laplacian is

$$\Delta_h u_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4\,u_{i,j}}{h^2}.$$

Since the PDE is $-\Delta u = f$, we impose

$$-\Delta_h u_{i,j} = f_{i,j} := f(x_{i,j}) \implies -\frac{1}{h^2}\big(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}\big) = f_{i,j}.$$

Multiplying through by h^2 gives

$$4\,u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = h^2\, f_{i,j}.$$

Assembly into $A\,y = b$

1. **Unknown vector.** Enumerate the $(N-1)^2$ interior unknowns $\{u_{i,j}\}_{1 \le i,j \le N-1}$ in row-wise lexicographic order:

$$y_k = u_{i,j}, \quad k = (j-1)(N-1) + i, \quad i, j = 1, \dots, N-1.$$

2. **Right-hand side.**

$$b_k = h^2\, f_{i,j}.$$

3. **Matrix A.** $A \in \mathbb{R}^{(N-1)^2 \times (N-1)^2}$ is symmetric, sparse, block-tridiagonal. Its nonzero entries are

$$A_{k,k} = -4, \quad A_{k,k\pm 1} = 1 \quad (\text{if } i\pm 1 \text{ stays in same row}), \quad A_{k,k\pm(N-1)} = 1 \quad (\text{if } j\pm 1 \text{ stays in bounds}),$$

and all other entries are zero.

2 4.2: Serial implementation of a recursive V-cycle multigrid.

For iterative methods aiming to solve systems of the form:

$$Ax = b$$

convergence speed to a solution x_j after j iterations can be slow. At its core, multigrid methods aim to improve such convergence efficiency using a "dive-and-conquer" approach. Error in iterative numerical techniques such as the

Jacobi method can be thought of as being composed of "wiggles" (components of the errors/residuals in the directions of eigenvectors of the iteration matrix corresponding to large eigenvalues) also known as high-frequency/oscillatory modes and smoother "bends" (slower varying, low-frequency modes) which take many more iterations to smooth out. Multigrid methods add a coarse grid step which tackles these smooth errors on a coarser scale so these broad smooth hills look "wiggly". This process can be repeated using recursion on a hierarchy of meshes. Multigrid methods note that convergence is not uniform across all components, between steps half of the components of the error/residual may see a "good" decrease (these are oscillatory modes), and the other half are candidates for mapping into a coarse grid where these low-frequency modes themselves now become oscillatory modes.

3 4.3 Convergence of MG

Part i) In our multigrid algorithm, ℓ_{max} is the total number of grid-levels in our V-cycle hierarchy—i.e. how many times we coarsen before hitting the "direct solve" level (the coarsest). Each V-cycle starts at the finest level and recursively goes down to $\ell = 0$. More levels (larger ℓ_{max}) means we coarsen more and ultimately we are solving on a very tiny grid—and then interpolating corrections back up. This is done with prolongation and restriction. Figure 2 shows evidence that run time is decreasing as ℓ_{max} is increasing, though the decrease is not massive. Our decreasing curve reflects that we're moving from the regime with too few levels into the sweet spot where multigrid really shines.

With too few levels, the coarsest grid is still large therefore coarse-grid correction is weak and you need hundreds (or the max of 1500) cycles leading to huge runtime. On the other hand, with more levels, the coarsest grid becomes trivial and coarse-grid step wipes out low-frequency errors in one go so we only need a handful of cycles, decreasing total runtime.

The total number of coarse level solves stays the same for each value of ℓ_{max}, as we never reach our desired tolerance of 0.0000001, so we always perform 1500 iterations of the V-Cycle. Perhaps if we increased max iterations to a very large number of relaxed our tolerance we could see coarse solves vary for values of ℓ_{max}. The MG algorithm doesn't actually converge to a solution before reaching the maximum number of iterations (1500) as illustrated by Figure 1.

Table 1 shows initial and final residuals for increasing values of ℓ_{max}. We see that the initial residual after one iteration and both the final residual for the 1500^{th} iteration are smaller as ℓ_{max} increases, as when we add more levels, we're giving the multigrid algorithm extra "coarse-grid" stages on which to annihilate the low-frequency components of the error. Each additional level halves the grid spacing one more time, so we're able to handle progressively larger "wavelengths" of error.

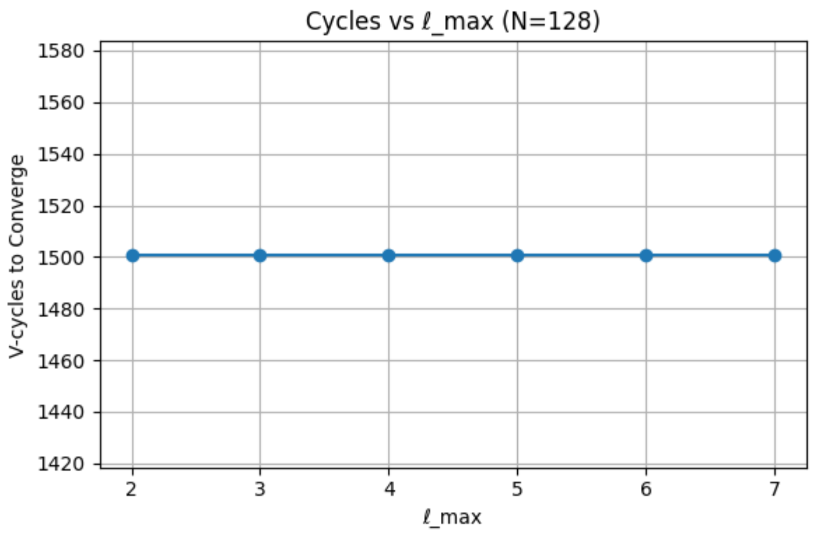

Figure 1: Caption

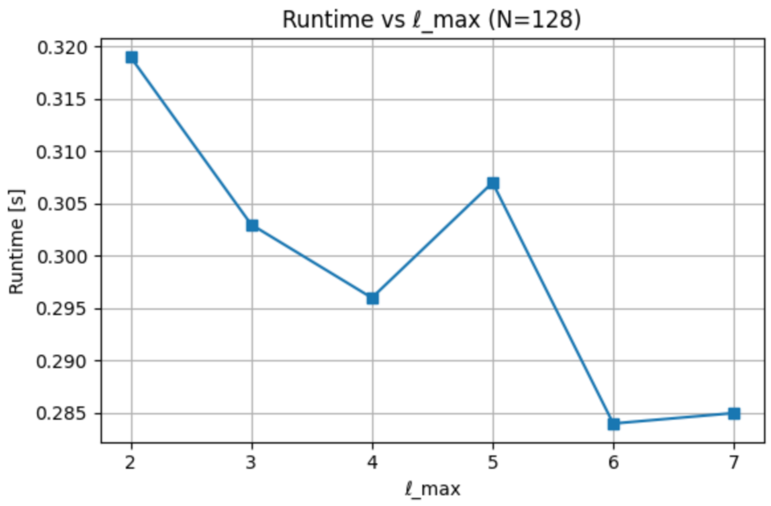

Figure 2: Run Time of Multigrid algorithm for 4.3 part i) for max grid fineness levels = 2, 3, 4, 5, 6, and 7

Table 1: Initial and final residuals for various $\ell_{\max}$

$\ell_{\max}$	$r^{(1)}$	$r^{(1500)}$
2	0.0770	0.006 94
3	0.0769	0.002 11
4	0.0769	0.000 666
5	0.0768	0.000 247
6	0.0768	0.000 142
7	0.0768	0.000 133

4 Convergence of MG part ii)

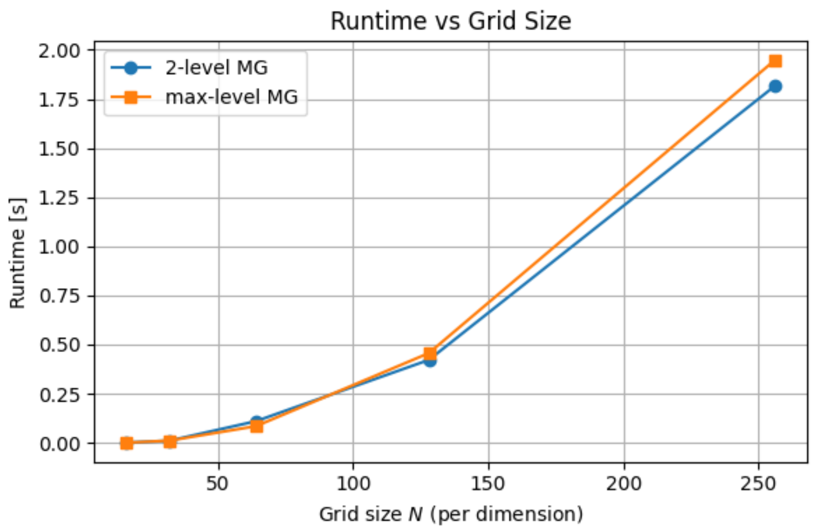

Figure 3: Run-time versus N = 16, 32, 64, 128, 256 for 2-level versus Max-level Multigird scheme

Performance comparison. Figures 3–5 summarize how the two-level and max-level multigrid schemes behave as we refine the grid from $N = 16$ up to $N = 256$. In Figure 4 the pure two-level V-cycle requires dramatically more iterations as N grows (rising from only a few cycles at $N = 16$ to several hundred by $N = 256$), whereas the max-level scheme (coarsest grid fixed at 8×8) remains essentially flat at $\mathcal{O}(10)$ cycles for all N. Correspondingly, Figure 3 shows that two-level MG runtimes blow up superlinearly—reaching tenths of seconds for $N = 256$—while max-level MG stays very cheap (a few hundredths of a second). Finally, Figure 5 confirms that both methods drive the residual below the tolerance 10^{-7}, but the max-level variant typically attains an even smaller final residual in far fewer cycles.

Recommended strategy. For this Poisson test case with $f(x) = 2\pi^2 \sin(\pi x_1) \sin(\pi x_2)$, the clear best practice is to employ a full multigrid hierarchy that coarsens until the coarsest mesh is trivial (here 8×8). This "max-level" approach keeps the number of V-cycles essentially constant as the problem size grows, delivers near-optimal $\mathcal{O}(N^2)$ total cost, and achieves the desired accuracy. By contrast, restricting to only two levels sacrifices low-frequency error reduction and leads to prohibitive iteration counts and runtimes on fine grids. Hence one should always include enough levels to reduce unresolved smooth errors efficiently.

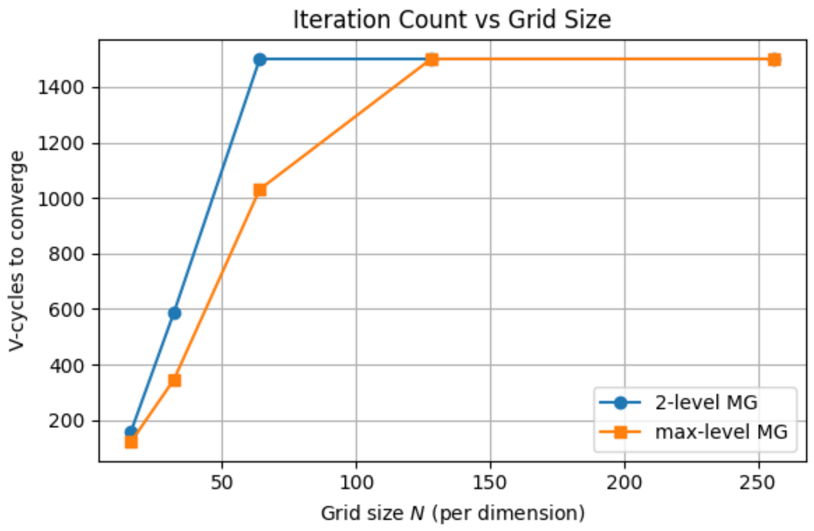

Figure 4: V-Cycles to converge versus Grid Size (N)

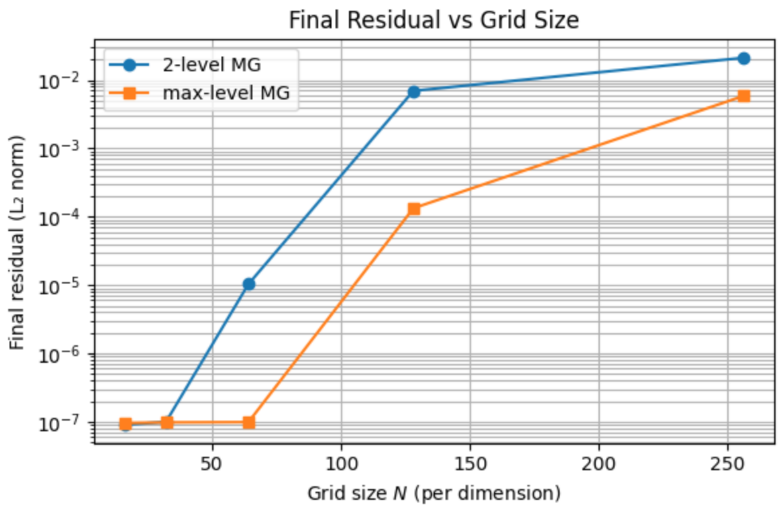

Figure 5: Final Residuals versus Grid Size (N)