Multigrid Method

Fast Iterative Solvers, Project 2

Johannes Leonard Grafen, 380149

July 22, 2023

Contents

List of Figures

1 Remarks on used architecture and compiler options

The code was written in C++, compiled using the Clang compiler and executed on an Apple Silicon M1 Pro Chip featuring the ARM64 architecture. To measure timings, the high resolution clock of the std chrono library was employed. The following flags where passed to the compiler to enhance code performance of the aforementioned architecture: -march=native, -O2. To suppress the output to measure runtime accurately the flag "DISABLEIO" was introduced and passed to the compiler via the "-D" option.

2 Multigrid Method - MG

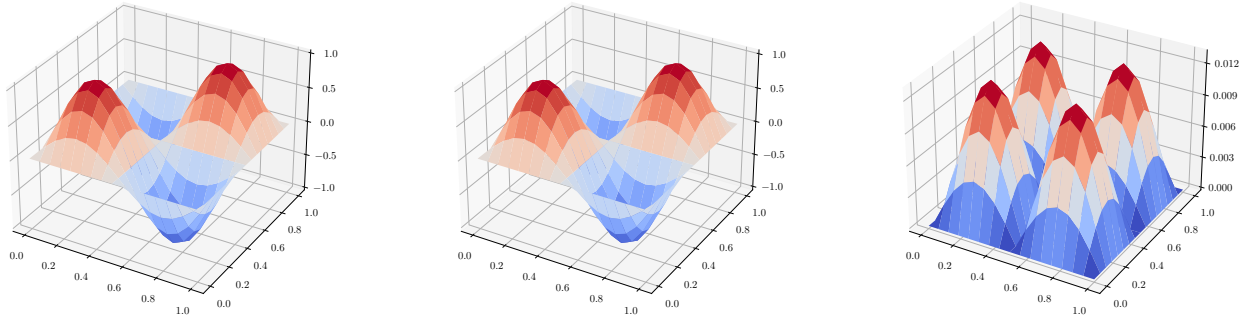
For the multigrid iterations, W-cycles ($\gamma = 2$) were used. Convergence was determined by the criterion

$$\frac{||r^{(m)}||_\infty}{||r^{(0)}||_\infty} < 10^{-10}, \tag{1}$$

were $r^{(m)}$ denotes the residual of the m-th iteration and $r^{(0)}$ the initial residual using $\mathbf{u} = \mathbf{0}$ as an initial guess for the solution vector.

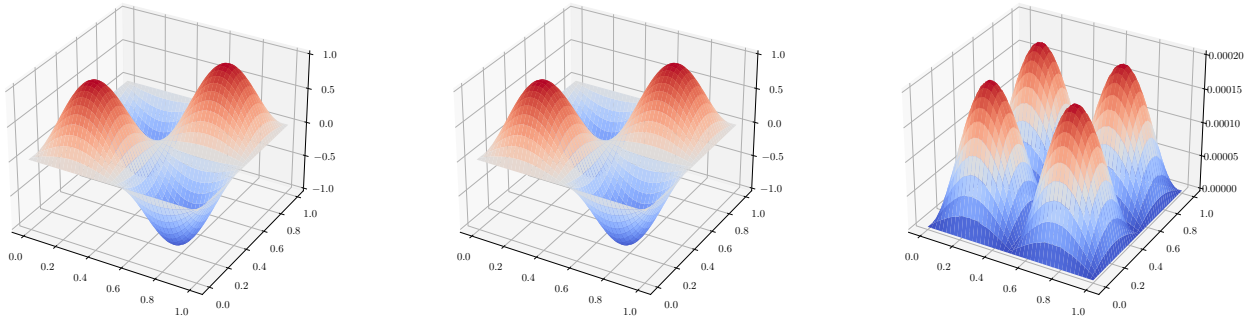
2.1 Validation of MG Method

The solution obtained by the multigrid method is compared to the exact solution $\mathbf{U}_{ex}(x, y) = sin(2\pi x)sin(2\pi y)$ in Fig. 1 for $n = 4$ and in Fig. 2 for $n = 7$. The estimated solution is in good agreement with the exact solution, with a maximum error of $\max_{i,j} e_{i,j} = 1.295 \cdot 10^{-2}$ for $n = 4$ and $\max_{i,j} e_{i,j} = 2.01 \cdot 10^{-4}$ for $n = 7$.



(a) $\mathbf{U}_{ex}$ (b) $\mathbf{U}_{est}$ (c) Error $\mathbf{e} = |\mathbf{U}_{ex} - \mathbf{U}_{est}|$

Figure 1: Plots of exact solution, approximated solution and corresponding error for $n = 4$

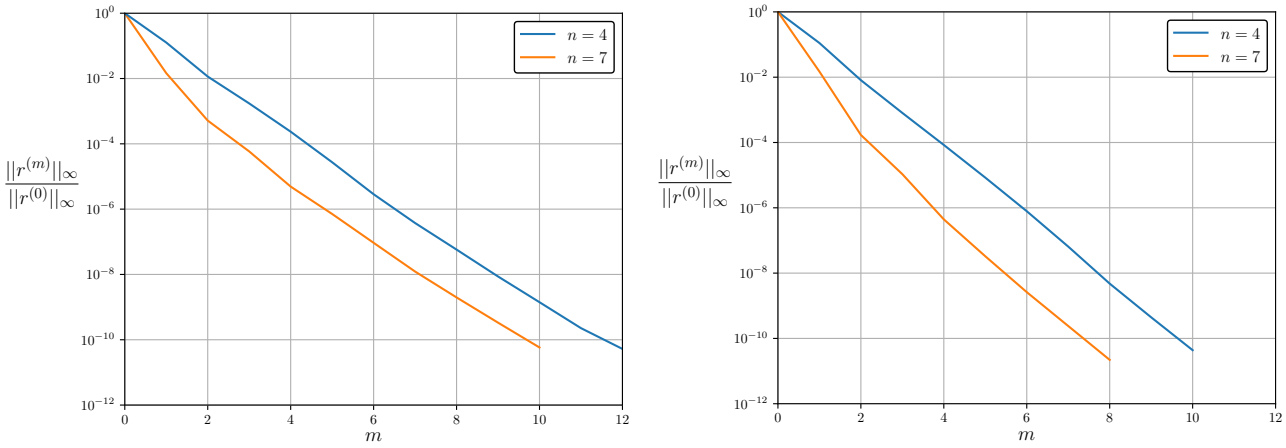


(a) $\mathbf{U}_{ex}$ (b) $\mathbf{U}_{est}$ (c) Error $\mathbf{e} = |\mathbf{U}_{ex} - \mathbf{U}_{est}|$

Figure 2: Plots of exact solution, approximated solution and corresponding error for $n = 7$

2.2 Convergence Plots

In Fig. 3 the convergence of the MG method for $\nu_1 = \nu_2 = 1$ and $\nu_1 = 2, \nu_2 = 1$ for two different meshes ($n = 4, n = 7$) is shown. The simulation is considered converged when the criterion as in Eq. (1) is satisfied. Notably, the finer mesh ($n = 7$) exhibits faster convergence compared to the coarser mesh ($n = 4$). Increasing the number of pre-smoothing iterations ν_1 leads to accelerated convergence. Specifically, the finer mesh with $n = 7$ achieves convergence after 8 iterations for $\nu_1 = 2$, while a single pre-smoothing iteration ($\nu_1 = 1$) requires 10 iterations to meet the criterion in Eq. (1). A similar trend is observed for the coarser mesh as well. The influence of the number of pre-smoothing iterations ν_1 is further investigated in Sec. 2.3.



(a) $\nu_1 = \nu_2 = 1$ (b) $\nu_1 = 2, \nu_2 = 1$

Figure 3: Convergence against multigrid iterations m for meshes $n = 4$ and $n = 7$

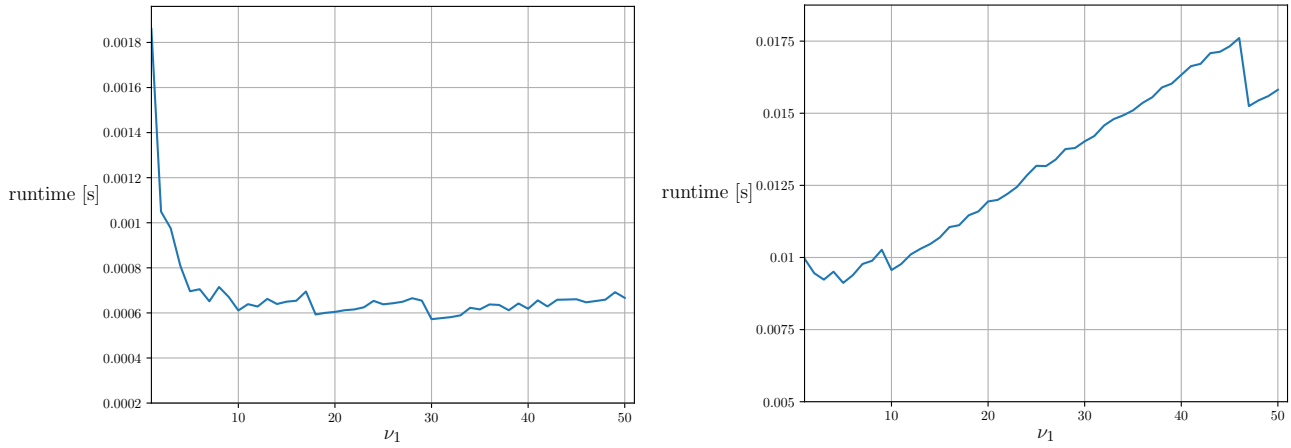
3 Further Analysis

This section focuses on analyzing the impact of the number of pre-smoothing iterations ν_1 used in the Gauss-Seidel Smoother and the number of coarse grid cycles γ required to achieve sufficient convergence as defined in Eq. (1).

3.1 Convergence for different ν_1

In Fig. 4, simulations were conducted on two meshes ($n = 4, 7$) using varying numbers of pre-smoothing iterations $\nu_1 = 1...50$. The number of post-smoothing iterations ν_2 was fixed at 1 for all simulations, and two coarse grid cycles ($\gamma = 2$) were applied.

Observing Fig. 4, it can be inferred that there is no further decrease in runtime beyond $\nu_1 > 10$ for both meshes. For the coarser mesh (Fig. 4a), the runtime remains relatively constant for $\nu_1 > 10$, while simulations on the finer mesh (Fig. 4b) exhibit a linear increase in runtime beyond the same threshold.



(a) $n = 4$ (b) $n = 7$

Figure 4: Runtime against number of smoothing iterations $\nu_1 = 1...50$ for meshes $n = 4$ and $n = 7$

3.2 Convergence for different γ

In this section, we explore the effect of different numbers of coarse grid cycles $\gamma = 1...10$, while keeping the number of pre- and post-smoothing iterations constant at $\nu_1 = \nu_2 = 1$. Similar to the previous section, we analyze different γ values for both a coarse and fine mesh.

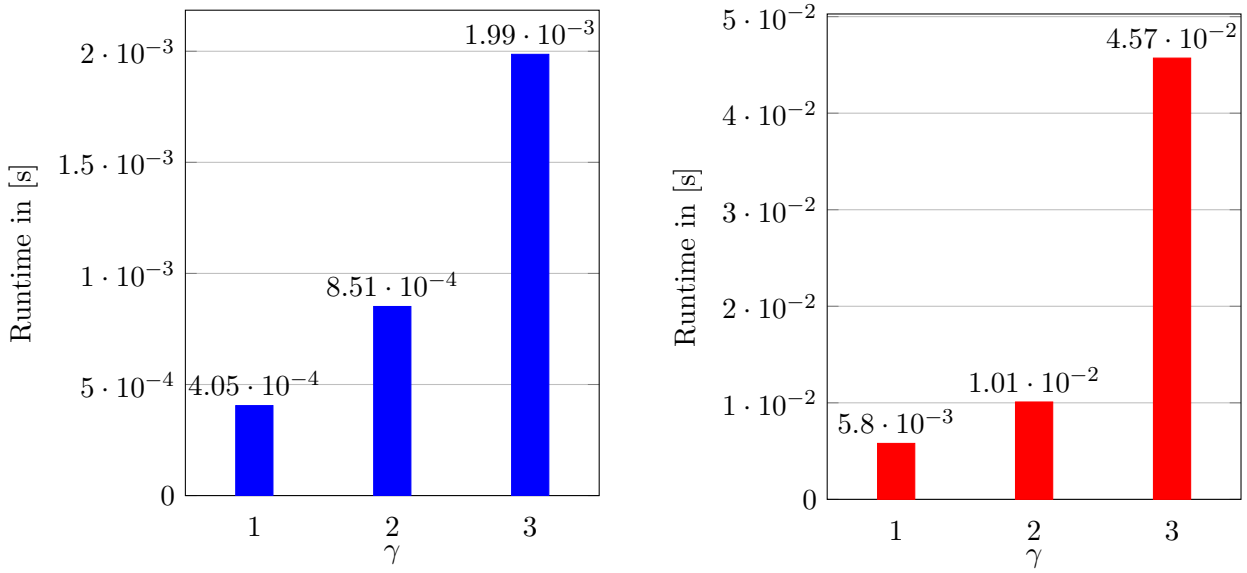
Figure 5 illustrates the runtime for different γ values within the range 1...3, following the theoretical trend for the workload W_k discussed in the lecture:

$$W_k \leq C \frac{1}{1 - \frac{\gamma}{4}} N_k. \tag{2}$$

As expected, both the runtime and the workload increase with an increasing γ. However, an interesting observation is that for very fine meshes with $n \geq 10$, the runtime for $\gamma = 2$ was found to be lower than for $\gamma = 1$. For example, for a mesh with $N = 1024$ nodes ($n = 10$), the runtime for

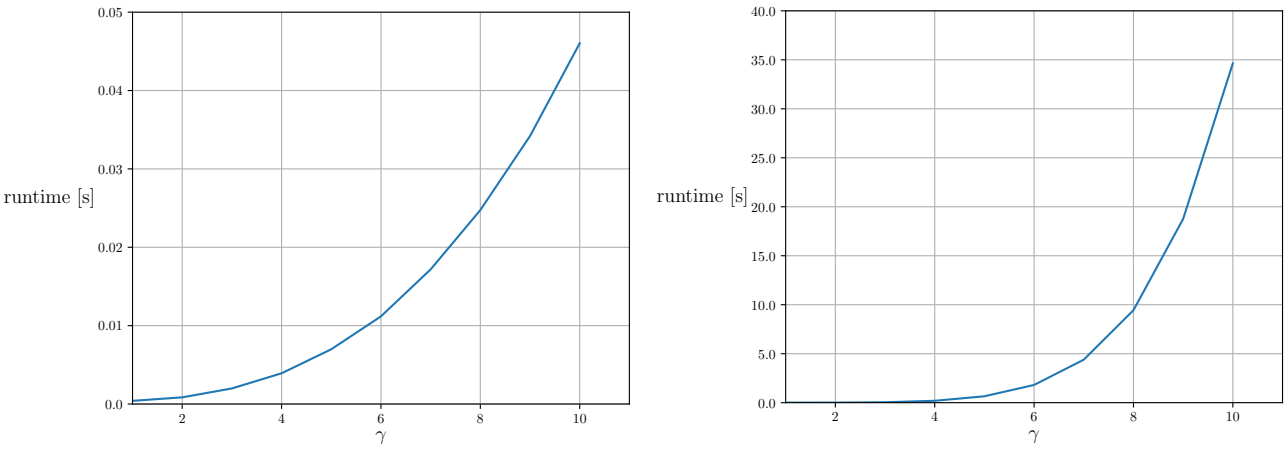
a single coarse grid cycle is $0.453s$, while for two coarse grid cycles, the runtime decreases to $0.368s$. This is in contrast to the trend observed for meshes with a parameter $n < 10$.

Furthermore, in the lecture was proven that we require $\gamma < 4$, otherwise the workload becomes unbounded. Nevertheless, I conducted further investigations with $\gamma = 1...10$, which resulted in an exponential increase in runtime, in agreement with our theoretical prediction as shown in Fig. 6.



(a) coarse mesh $n = 4$ (b) fine mesh $n = 7$

Figure 5: Runtime for $\gamma = \{1, 2, 3\}$ for both meshes



(a) $n = 4$ (b) $n = 7$

Figure 6: Runtime against number of coarse grid cycles $\gamma = 1...10$ for meshes $n = 4$ and $n = 7$