Fast Solver Report

January 7, 2018

1 Recent advancements

- Implemented and tested Multigrid (the V-cycle).
- Changed the restriction grid operator such that it is the transpose of the prolongation operator. (as this is required for the preconditioned CG - O.Tatebe article)
- Used the multigrid as a preconditioner for the Conjugate Gradients method.

2 Equation example

I will exemplify the results I got so far with the simple differential equation with Dirichlet border conditioning:

$$f : \Omega = [0,1] \times [0,1] \to \mathbb{R}, \text{ with } f(x,y) = \sin(x)\sin(y) \text{ on } \partial\Omega \tag{1}$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} = -2\sin(x)\sin(y) \tag{2}$$

3 Experimental results

3.1 Multigrid as a solver on its own

First of all, we will examine the Multigrid method as a solver on its own. I am solving the differential equation by discretizing it and applying v-cycles on the resulting linear equation. For each grid level, I am using Jacobi / Gauss Seidel, with $\nu_1 = \nu_2 = 3$ iterations.

Below, $N + 1$ is the number of uniformly distributed samples on one axis of the function f. Therefore, the matrix in the linear equation has the size $(N + 1)^2 \times (N + 1)^2$. Let us plot the logarithm of the residual error in terms of the number of iterations. This is illustrated in the figure below for $N \in \{4, 8, 16, 32, 64\}$. We're iterating as long as the residual error euclidean norm is above a specified tolerance value $tol = 10^{-6}$.

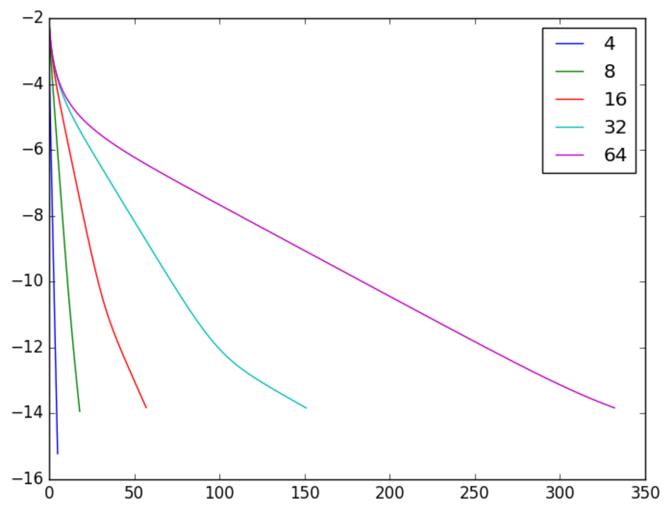

Figure 1: Multigrid with Jacobi smoothing

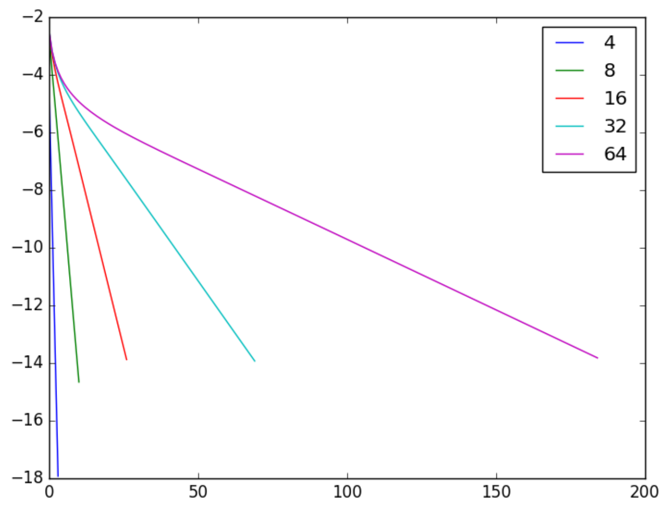

Figure 2: Multigrid with Gauss Seidel smoothing

Using the Multigrid solver, I have also been plotting the solution of f for different levels of discretization below.

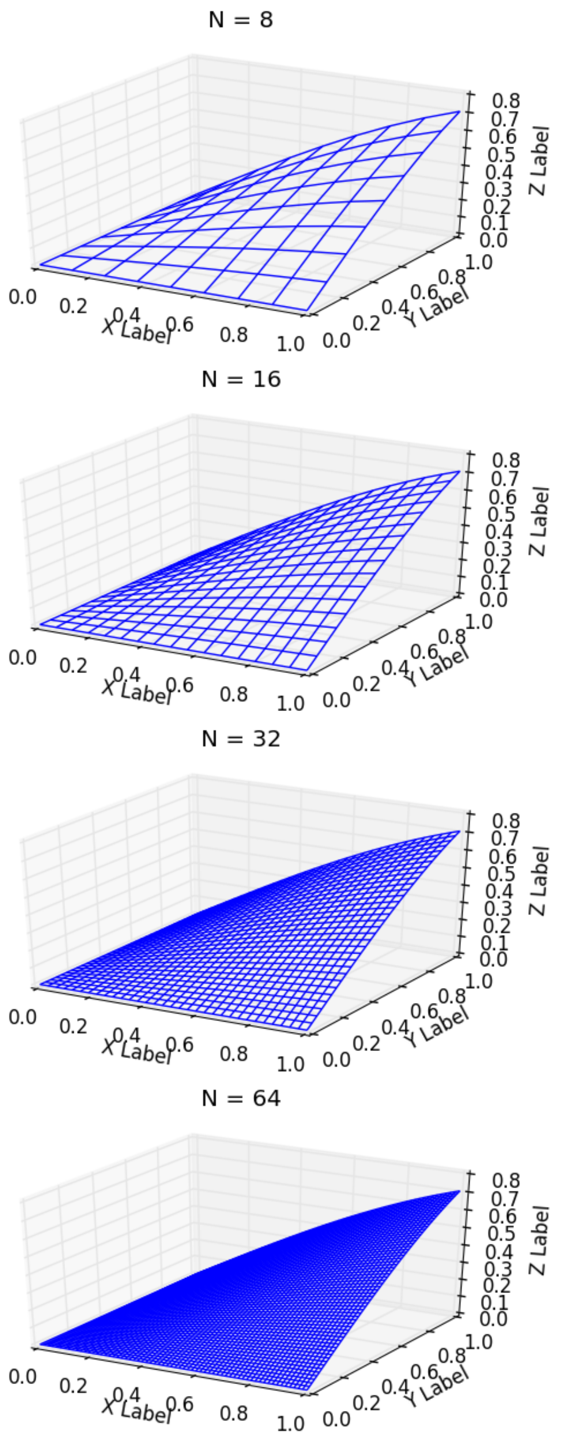

Figure 3: Plots of f at different levels of discretization obtained with Multigrid as a solver

3.2 Preconditioning Conjugate Gradients with Multigrid

Let us now expose some results for using the Multigrid as a preconditioner for Conjugate Gradients. The advantage of the implementation is that we do not have to specifically compute the preconditioning matrix, but we can just simulate the action of it by modifying the original Conjugate Gradients algorithms, using one iteration of the v-cycle to solve a linear equation withing the CG loop.

As proven in O.Tatebe's article, the preconditioned matrix remains symmetric and positive definite as long as $\nu_1 = \nu_2$ (the number of GS/Jacobi iterations in the pre-smoothing step = number of iterations in the post-smoothing step) and $R = \alpha P^T$, where R is the restriction matrix, P is the prolongation/interpolation matrix, and α is a positive real number.

Let us plot the logarithm of the residual norm in terms of the number of iterations. We do not expect any special form plots in this chart, but we need this representation because the error diminishes very fast. For the example below, we are using one Multigrid v-cycle per step, with $\nu_1 = \nu_2 = 3$ and with Jacobi smoothing.

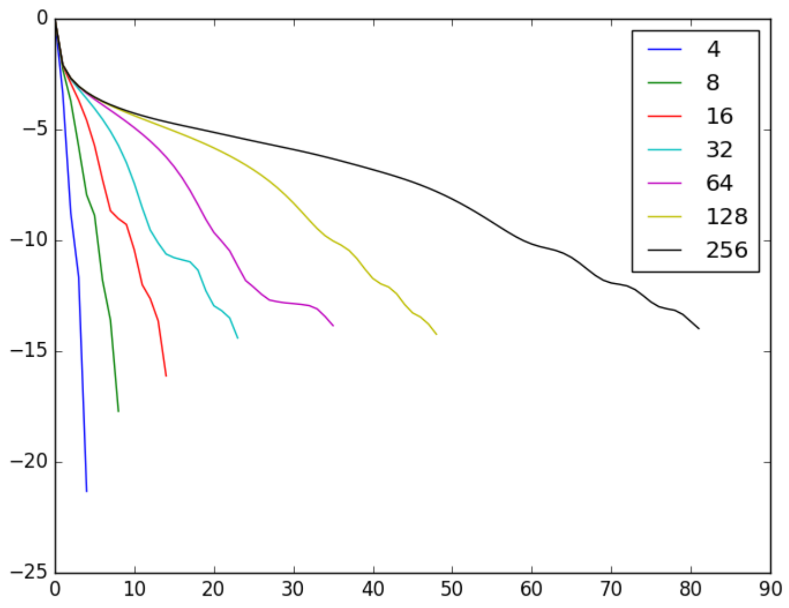

Figure 4: Conjugate Gradients with Multigrid as a preconditioner

This is a good solver, as for each N label above, the size of the linear system is $(N+1)^2 \times (N+1)^2$.

3.3 Other experiments

Let us also take a look at how the restriction operator in the Multigrid impacts the convergence for it as a solver on its own. Also, let us take a look at how the preconditioned CG behaves to the simple CG method.

3.3.1 The restriction operator in Multigrid

As noted before, we need the prolongation operator to be proportional to the tranpose of the restriction operator if we want the Multrigrid to work as a preconditioner for CG. Let us analyze how the iteration count of v-cycles is impacted by using two different restriction operators.

The first restriction operator we are looking at is simply dropping unnecessary values, it can be defined by the stencil: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.

For the second restriction operator, we are using the transpose of the linear interpolation operator.

We can easily see below how the second, more accurate, restriction operator yields a faster convergence. We are plotting the logarithms of the residual error norm in terms of the iteration count.

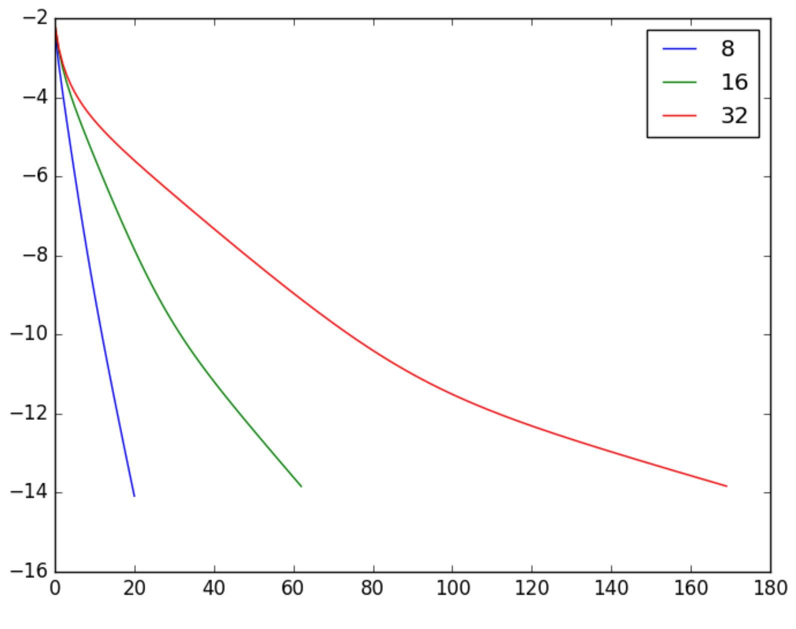

Figure 5: Using the simple restriction operator

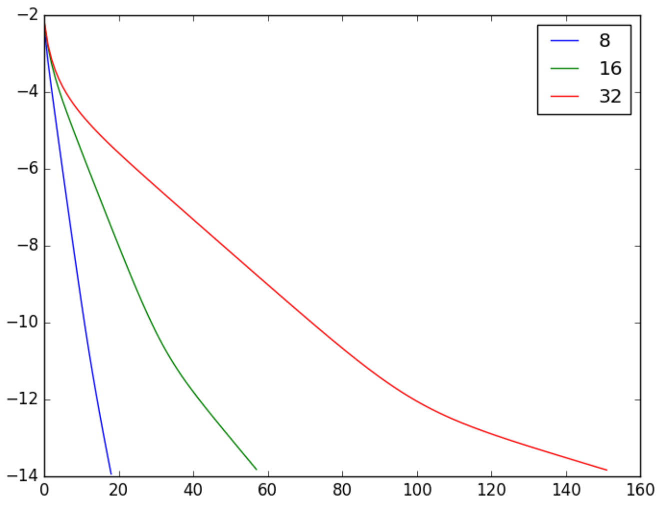

Figure 6: Using the second restriction operator

In order to check that our restrictions make sens, let us also plot how the trivial restriction works, which is just appending the necessary number of zeros. This makes some sense for our example, as the function has small values everywhere (moreover, in the general case, it will just be equivalent to applying the second half of the v-cycle).

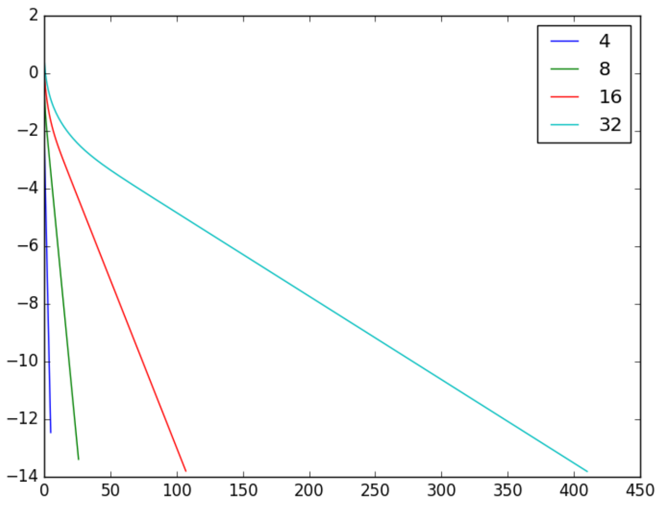

Figure 7: Using the trivial restriction operator, the full zero vector

3.3.2 CG vs Multigrid preconditioned CG

Let us compare the number of iterations required by Conjugate Gradients and by its preconditioned version for $N = 256$ with $tol = 10^{-5}$.

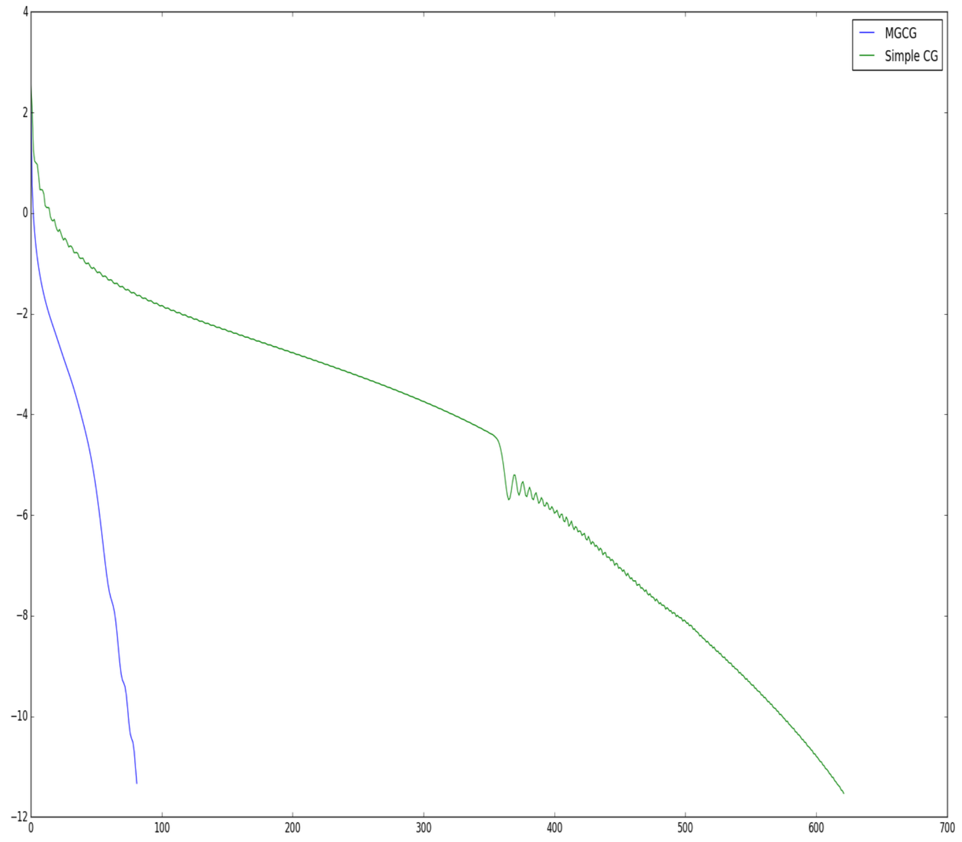

Figure 8: Using the trivial restriction operator, the full zero vector