Performance of the Scheduled Relaxation Jacobi method in a geometric multilevel setting. I. Linear case

Eloisa Bentivegna
IBM Research Europe, Keckwick Lane, Daresbury WA4 4AD, United Kingdom
E-mail: eloisa.bentivegna@ibm.com

Keywords: Relaxation Methods, Geometric Multigrid, Parallel Algorithms

Abstract
I investigate the suitability of the Scheduled-Relaxation-Jacobi method as a smoother within a geometric multilevel (ML) solver. Its performance in the solution of a linear elliptic equation is measured, based on two metrics: absolute performance (measured by the residual reduction in a fixed number of iterations), and parallel scalability. I discuss the theoretical expectations on the effect of this hybrid scheme on the solution iterate and, especially, the solution error, and confirm them numerically.

1. Introduction and motivation

Multigrid methods, based on the combined solution of an elliptic problem on multiple grids, are notoriously able to accelerate the reduction of residuals in a robust and versatile way. Multigrid-inspired recipes exist for nonlinear problems and problems with space-dependent coefficients; the multigrid paradigm has been also successfully applied to problems with do not stem from the discretization of a partial differential equation (PDE).

Due to the complex infrastructure needed to solve elliptic problems of scientific relevance, it is imperative that the basic multigrid building blocks, and in particular the relaxation operator used on each level, work as efficiently and accurately as possible. A standard choice for this operator is provided by the Jacobi scheme; whilst traditional improvements exist to accelerate this method, they are based on the concepts of overrelaxation and in-place update, both of which are known to be incompatible with the multigrid scheme.

Recently, however, a new strategy to accelerate the Jacobi method has been proposed [1] and successively improved [2] and contrasted to other methods [3]. SRJ is based on an alternate underrelaxation and overrelaxation strategy. Its compatibility with a multigrid approach is unknown; this work aims to address this question by measuring the performance of an SRJ-endowed multigrid solver on a common linear PDE. I will briefly describe the method and a set of related open questions in the next section.

2. The Scheduled Relaxation Jacobi method

Given an equation
$$O(\psi) = 0,$$
iterative methods aim at reducing its residual by successive updates to the solution $\psi_i$, according to:
$$\psi_{i+1} = \psi_i + \omega r_i,$$
where the residual $r_i = O(\psi_i)$, and $\omega$ is the relaxation factor, which affects the method’s stability and convergence rate (measured by the number of iterations required to reduce the residual by a predefined amount).

In order to improve on this algorithm, the SRJ approach starts with the key observation that $\omega$ needs not be a constant, but that it can take on a different value at each iteration, $\omega_i$. This relaxation vector provides further flexibility to speed up the scheme’s convergence; the optimal vectors for, e.g., the Laplace equation have been found and shown to accelerate the Jacobi method by several orders of magnitude [1].
The key property of optimal relaxation vectors is that they alternate over- and under-relaxation in a way which reduces all error frequencies efficiently, but at the same time preserves the algorithm’s stability asymptotically. This improved behavior raises the question of whether such an algorithm would be a more effective smoother in a multigrid setting, as it does not suffer from the systematic amplification of the high-frequency error components that afflicts SOR within a multigrid setup. At the same time, it would accelerate the Jacobi smoother (and, asymptotically, the Gauss-Seidel (GS) one), using occasional overrelaxations to accelerate the reduction of the low-frequency error components.

The analysis of [1–4] leaves, however, a number of open questions. In this paper I will address two:

- The optimal coefficients for the 2D Laplace equation with Neumann boundary conditions have been calculated, for schemes with \( P = 6 \) to \( P = 15 \) levels and a number of different grid sizes [2]. Strictly speaking, these coefficients are only optimal for this specific elliptic problem; changing the dimensionality, the equation, or the boundary conditions impacts the optimization problem. This issue has been partially addressed by the measurement of the scheme’s performance for the Poisson equation with Dirichlet boundary conditions (in both 2D Cartesian and 3D polar coordinates), and for the GradShafranov equation [2]. In all cases, the coefficients, albeit not optimal, provided a reasonable speedup, supporting hope that the optimum for one equation can be safely reutilized (with minimum performance loss) for the solution of closely-related ones. As the calculation of the coefficients might become quite laborious for schemes with many levels, this is an important practical result. This property, however, needs to be checked on a case-by-case basis; in this paper I will address some of the cases not treated previously.

- The previous observations are particularly crucial when considering the application of SRJ within a multilevel setting. In this case, the error equation, rather than the original equation, has to be solved on certain levels. Whether the relaxation vector has to be modified accordingly is a question that has not been addressed yet.

I will address these questions with numerical experiments in the following section.

3. Using SRJ as a smoother in a ML solver

In the following, I will describe a number of performance measurements of the SRJ algorithm used as a smoother in a ML solver (hereafter abbreviated as ML+SRJ). In particular, I will explore the effectiveness of this method, compared to a classic (GS-smoothed) ML solver and single-level SRJ relaxation, for a linear problem and a number of grid sizes.

3.1. Infrastructure

For all tests, I use the CT_MultiLevel solver [5], distributed with the Einstein Toolkit [6]. I solve all equations on the unit cube in three dimensions, \([0, 1]^3\), with \( N \) points per side. I run each solver for \( 10^4 \) iterations; in the SRJ-enhanced ML case, I make sure that the relaxation vector is completely traversed before switching levels.

For fairness of comparison, when the performance of a multilevel scheme is compared to that of a single-level one, the grid size of the single-level scheme is set equal to the size of the topmost level in the multilevel one. This quantity is used to label the figures in the rest of the paper.

3.2. Results

As a benchmark for linear elliptic equations, I will use the Poisson equation:

\[
\Delta \psi = \rho
\]

where \( \rho \) is set via the Method of Manufactured Solutions (MMS) [7], \( \rho = \Delta \tilde{\psi} \), having chosen a Gaussian function as the exact solution

\[
\tilde{\psi} = e^{-(x-x_0)^2+(y-y_0)^2+(z-z_0)^2)/2\sigma^2}
\]

I solve this equation with \( x_0 = y_0 = z_0 = 1/2, \sigma = 1/4 \), and exact Dirichlet conditions on all boundaries.

I begin by comparing the performance of the ML algorithm, enhanced by SRJ, compared to a ML solver and to SRJ relaxation alone for a reference grid size with \( N = 128 \). The results are illustrated in figure 1. The presence of coarser levels combined with the use of SRJ relaxation proves to be a substantial advantage for the ML+SRJ method at the start, where the residual can be immediately reduced by up to two orders of magnitude compared to the ML case, and up to three compared to SRJ. The two latter schemes catch up by iteration 1000, and single-level SRJ continues to reduce the residual exponentially all the way down to roundoff level, while the other two methods reach a plateau just below \( 10^{-2} \).
This feature is typical in ML schemes, and is caused by the difference in the magnitude of the discretization error between different levels \cite{5}, which cause a jump in the equation residual whenever prolongation is used to transition between the different levels. I provide evidence of the nature of this convergence limit in, e.g., figure 2: here, the plateaus are shown for different grid sizes, and it is apparent that finer grids can reach a lower value, as the scale of the numerical truncation errors is arguably lower (notice that the residual average value in this phase is compatible with the second-order differencing stencils I am using). Naturally, this effect is counterbalanced by the fact that larger grids will generally take a higher number of iterations to converge, so that for a fixed and sufficiently small number of final iterations, the residual grows with $N$. This can be readily inferred from figure 2.

Another important, related aspect to notice is that, whilst the residual of the discrete problem can be reduced to roundoff, a similar reduction in the solution error, defined as the difference between the current solution iterate and the exact solution, is not necessarily guaranteed. The limit is, again, the discretization error. The right panel of figure 1 illustrates this property: here, the final error achieved by the three algorithms is quite close, the only major difference laying in the number of iterations needed to reach this value. The ML+SRJ scheme requires around 70\% fewer iterations to converge. As this is an inherent convergence limitation on a specific grid size, the only way to lower the solution error norm is to adopt a finer mesh. The right panels of figures 2, 3, and 4 show that this prediction is indeed confirmed.

The fact that these expectations are confirmed also proves the feasibility of solving the error equation (as is customary in ML schemes) using the same relaxation vector as for the original equation.

4. Conclusions

I have measured the performance of a ML solver, enhanced with a SRJ smoother, and found that the scheme is generally well-behaved. It is also sufficiently robust for practical applications: the optimal coefficients for the 2D
The Laplace equation works well in three dimensions, after the addition of an inhomogeneous term, and for the associated error equations.

Acknowledgments

The author is grateful to Isabel Cordero-Carrión for suggesting this application, and acknowledges helpful discussions with Roland Haas, Kirk Jordan, Vassilios Mewes, and Vedant Puri, as well as hospitality at the National Center for Supercomputing Applications (NCSA) during the preliminary phases of this project. This work was supported by the STFC Hartree Centres Innovation Return on Research programme, funded by the Department for Business, Energy & Industrial Strategy.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

ORCID iDs

Eloisa Bentivegna © https://orcid.org/0000-0003-1229-1653

References

[1] Yang XI A and Mittal R 2014 Acceleration of the Jacobi iterative method by factors exceeding 100 using scheduled relaxation J. Comput. Phys. 274 708
[2] Adsuara J E, Cordero–Carrión I, Cerdá–Durán P and Aloy M A 2016 Scheduled Relaxation Jacobi method: improvements and applications J. Comput. Phys. 321 369–413
[3] Adsuara J E, Cordero-Carrión I, Cerdá-Durán P, Mewes V and Aloy M A 2017 On the equivalence between the Scheduled Relaxation Jacobi method and Richardson’s non-stationary method J. Comput. Phys. 332 446–60

[4] Adsuara J E, Cordero-Carrión I, Cerdá-Durán P and Aloy M A 2017 Speeding up a few orders of magnitude the Jacobi method: high order Chebyshev-Jacobi over GPUs (arXiv:1705.00103)

[5] Bentivegna E 2014 Solving the Einstein constraints in periodic spaces with a multigrid approach Class. Quant. Grav. 31 035004

[6] Löffler F et al 2012 The Einstein toolkit: a community computational infrastructure for relativistic astrophysics Class. Quant. Grav. 29 115001

[7] Roache P J 1998 Verification and Validation in Computational Science and Engineering (Albuquerque, NM: Hermosa Publishers)