MULTIGRID METHODS

ADITYA KARTHIK SARAVANAKUMAR

Abstract. Multigrid methods are techniques used to accelerate the convergence of standard iterative methods such as the Jacobi and Gauss-Seidel methods. Developed over the last 25 years, multigrid methods are well-developed for elliptic problems and offer the possibility of solving a problem with computation and memory proportional to the number of unknowns $\mathcal{O}(n)$. These methods are also found to be useful in generating effective preconditioners.

Key words. Iterative methods, Multigrid methods, Hybridizable Discontinuous Galerkin methods (HDG), Elliptic PDE.

1. Introduction. Linear systems of equations are ubiquitous in computational science and often arise from finite discretizations of partials differential equations (PDEs). Consequentially, the ability to efficiently solve these systems is of great importance, and research in this direction has become a fundamental part of linear algebra in modern times. Techniques used to solve linear systems can be classified as direct and iterative methods. Most direct methods can be interpreted as variations of the famous Gauss Elimination algorithm, which generates exact solutions at the cost of $\mathcal{O}(n^3)$ for a dense $n \times n$ matrix. However, the matrices that arise from the discretization of physical systems often possess special structures that can be utilized to devise more efficient algorithms. For example, the Thomas algorithm is a direct method that can be used to solve tridiagonal linear systems at $\mathcal{O}(n)$ cost. [6] reviews the various direct method algorithms used for sparse linear systems.

In practice, iterative methods are often preferred over direct methods for large systems of equations. The classic iterative methods, Jacobi and Gauss-Seidel, use iterations of the form $\underline{x}^{n+1} = \underline{x}^n + \underline{\underline{Q}}(\underline{b} - \underline{\underline{A}}\,\underline{x}^n)$ to produce approximate solutions to large linear systems $\underline{\underline{A}}\,\underline{x} = \underline{b}$. The successive over-relaxation (SOR) and weighted Jacobi methods were later introduced to improve the standard GS and Jacobi methods, respectively. More sophisticated iterative methods include the Krylov subspace methods [14] such as the generalized minimum residual (GMRES), Arnoldi, Lanczos, Conjugate Gradient (CG), and BiCGSTAB methods, which often provide improved convergence and stability properties.

Over the last few decades, multigrid methods have been developed to accelerate the convergence of iterative methods (such as Gauss-Seidel and under-relaxed Jacobi methods). This method was first proposed in [7], and its convergence properties were presented in [8]. Typically, iterative methods (including GS and under-relaxed Jacobi) function as smoothers in that they can get rid of the high-frequency components of the error in fewer iterations than the low-frequency components. Furthermore, when the errors are transferred to a coarse grid, the previously low-frequency components become high-frequency components in the coarse grid and can therefore be eliminated cheaply. Multigrid methods exploit these two properties by using a hierarchy of discretizations and grid-transfer operations to efficiently solve systems of equations with great accuracy at modest costs. By extension, these methods also rely on the ideas that smooth functions can be well represented on coarse grids and that coarse grid solutions are cheaper to compute. A good introduction to multigrid methods can be found in [2].

Another approach to improving the efficiency of iterative solutions is using preconditioners. It is well known that the efficiency of iterative methods is often closely

linked to the condition number of the linear system in consideration. Preconditioners improve the condition number of the problem and, therefore, improve iterative methods' performance. Practically, this amounts to multiplying both sides of the linear system $\underline{\underline{A}}\,\underline{x} = \underline{b}$ by a splitting $\underline{\underline{Q}}^{-1}$ and solving the resulting better-conditioned system $\underline{\underline{Q}}^{-1}\,\underline{\underline{A}}\,\underline{x} = \underline{\underline{Q}}^{-1}\,\underline{b}$. The multigrid method offers another use here as it turns out they generate efficient preconditioners for external iterative solvers [4].

Multigrid methods are a relatively new class of iterative methods and are still subject to active research. These methods are currently well-developed for elliptic problems and offer the possibility of solving a problem with computation and memory proportional to the number of unknowns $\mathcal{O}(n)$.

This work reviews multigrid methods and discusses results from an elementary 1D implementation of a multigrid method in an HDG finite element solver framework. Section 2 introduces the multigrid algorithm and the tools required to implement the algorithm. Following this, an exposition of the implementation of the 1D multigrid method, along with a brief overview of the finite element method used, is provided in Section 3. An analysis of the multigrid results is presented in Section 4 with comparisons against other iterative methods along with comparisons between different multigrid variants. Section 5 summarizes the findings of this work and points to active directions of research on this topic.

2. Multigrid Algorithm. Here we start by describing a simple two-grid algorithm that presents the fundamental multigrid ideas. Each cycle of the two-grid method can be decomposed into three stages : 1) pre-smoothing iterations 2) coarse grid correction and 3) post-smoothing iterations. The two smoothing iterations are carried out by a iterative method that eliminates the high-frequency error components in just a few iterations. Eigenvalue analysis can sometimes be used to reveal whether or not an iterative method functions as smoother [2]. The second stage involves generating a correction to our approximate using a coarse-grid solution for the same problem. The error equation for a linear system of the form $\underline{\underline{A}}\,\underline{x} = \underline{b}$ can be derived in terms of the error $\underline{e}$ and the residual $\underline{r}$ as

$$\underline{\underline{A}}\,\underline{x} - \underline{\underline{A}}\,\underline{x}^{est} = \underline{b} - \underline{\underline{A}}\,\underline{x}^{est} \quad \Longrightarrow \quad \underline{\underline{A}}\,(\underline{x} - \underline{x}^{est}) = \underline{\underline{A}}\,\underline{e} = \underline{r} \tag{2.1}$$

Starting from an initial guess $\underline{x}^n$, pre-smoothing iterations are carried out to obtain $\underline{x}^{n+1/3}$. The residual obtained on the fine grid (of mesh size h) can be transferred to a coarse grid (say, of size $2h$) using a procedure called **restriction** as

$$\underline{r}_{2h} = \underline{\underline{I}}^{h}_{2h}\,\underline{r}_h \quad \textit{and} \quad \underline{\underline{A}}_{2h}\,\underline{e}_{2h} = \underline{r}_{2h} \tag{2.2}$$

Note that we consider the case where the coarse grid has twice the mesh size as the fine grid here, and this common practice as there is generally no advantage in using different grid spacing ratios.

We expect it to be quite smooth since our initial approximation has already been subject to pre-smoothing iterations. Based on this, we can expect the residual to be reasonably well-represented after restriction to a coarse grid. The simplest restriction procedure would be *injection*, wherein we just pick the alternate grid point values from the fine grid vector to form the coarse grid vector as, $r_{2h,i} = r_{h,2i}$ for $i = 1, \ldots, (n-1)/2$. The error equation derived above is then used to compute the residual on the coarse grid, at a relatively cheap computational cost, as $\underline{e}_{2h} = \underline{\underline{A}}^{-1}_{2h}\,\underline{r}_{2h}$. The computed error can be transferred back to the fine grid using a procedure called **prolongation** and used to improve our approximation as

$$\underline{e}_h = \underline{\underline{I}}_h^{2h}\,\underline{e}_{2h} \quad \textit{and} \quad \underline{x}_h^{n+2/3} = \underline{x}_h^{n+1/3} + \underline{e}_h \tag{2.3}$$

Finally, the approximation is further updated by post-smoothing iterations to obtain $\underline{x}^{n+1}$. The two-grid algorithm is summarized below

Algorithm 2.1 One cycle of a two-grid algorithm

1: Choose an initial guess $\underline{x}_h^0$
2: Relax ν_1 iterations of $\underline{\underline{A}}_h\,\underline{x}_h^0 = \underline{b}_h \rightarrow \underline{x}_h^{1/3}$ ▷ Pre-smoothing iterations
3: Compute the residual as $\underline{r}_h = \underline{b}_h - \underline{\underline{A}}_h\,\underline{x}_h^{1/3}$ ▷ Coarse grid correction
4: Restrict residual to coarse grid as $\underline{r}_{2h} = \underline{\underline{I}}_{2h}^{h}\,\underline{r}_h$
5: Compute coarse grid error as $\underline{\underline{A}}_{2h}\,\underline{e}_{2h} = \underline{r}_{2h} \rightarrow \underline{e}_{2h}$
6: Prolongate error to fine grid as $\underline{e}_h = \underline{\underline{I}}_h^{2h}\,\underline{e}_{2h}$
7: Correct $\underline{x}_h^{2/3} = \underline{x}_h^{1/3} + \underline{e}_h$
8: Relax ν_2 iterations of $\underline{\underline{A}}_h\,\underline{x}_h^{2/3} = \underline{b}_h \rightarrow \underline{x}_h^1$ ▷ Post-smoothing iterations

3. Implementation. In order to validate the multigrid method, we apply it to the linear system that arises from discretizing the 1D Poisson equation shown below,

$$-\underline{\nabla}\cdot(\kappa\,\underline{\nabla} u) = \underline{f} \tag{3.1}$$

For simplicity, we will assume homogeneous boundary conditions and that $\kappa = 1$ for each test case. Various numerical methods can be used to discretize the PDE, including finite difference methods and finite volume methods, but in this work, we employ a hybridizable discontinuous Galerkin (HDG) finite element method.

3.1. Finite element methods. Finite element methods are robust numerical methods that are widely used in computational science. These methods allow for *hp*-adaptivity over unstructured meshes that can handle complex geometries, making them useful for a variety of problems. In these methods, the domain is discretized into a finite set of elements, and the solution on each element is approximated using a locally defined basis function. The classic FEM schemes are the Galerkin schemes, of which the continuous Galerkin (CG) and discontinuous Galerkin (DG) methods are two types. A detailed introduction to these methods can be found in [15]. The HDG methods for convection-diffusion problems were introduced in [22], and they offer computational savings over the DG method by reducing the size of the global linear that needs to be solved. The discretization and flux choices in HDG methods are set up such that the solution across the domain can be computed by solving a global system for the unknowns along the skeleton of the grid ($\underline{\Lambda}$) (referred to as the numerical traces),

$$\underline{\underline{K}}_g\,\underline{\Lambda} = \underline{F}_g \tag{3.2}$$

Following this, these numerical traces are used to reconstruct the local solution on each element.

3.2. Test cases. We will be using a few test cases to validate the multigrid method and make comparisons with other competing algorithms. The method of

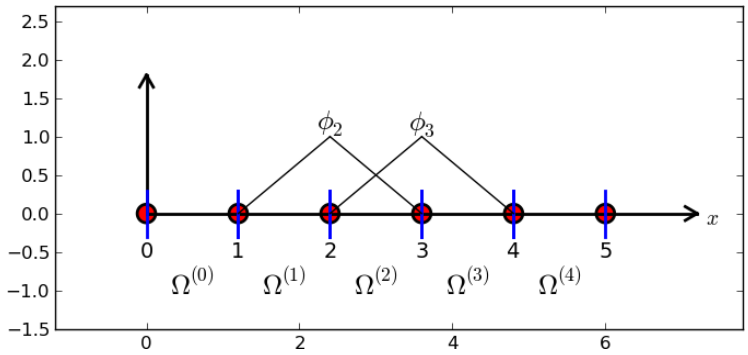


Fig. 1: The numerical trace is represented in terms of piece-wise linear basis functions (image taken from [16])

manufactured solutions has been used to generate these problems [23]. The first problem (referred to as P1 henceforth) is defined as,

$$-u_{xx} = 20x^3 - 30x^2 + \frac{108}{8}x - \frac{25}{16}, \quad 0 < x < 1, \quad u(0) = u(1) = 0 \tag{3.3}$$

$$u(x) = x(x - 1/4)(x - 1/2)(x - 3/4)(x - 1) \tag{3.4}$$

Note that since the solution is a fifth-order polynomial, we expect the HDG method to generate an exact solution when $p \geq 5$.

For the second problem, we construct a solution that lies outside the polynomial space such that the numerical method cannot generate an exact solution. The second problem (referred to as P2 henceforth) is defined as,

$$-u_{xx} = 25\pi^2(\sin(5\pi x) + 9\sin(15\pi x)) \quad 0 < x < 1, \quad u(0) = u(1) = 0 \tag{3.5}$$

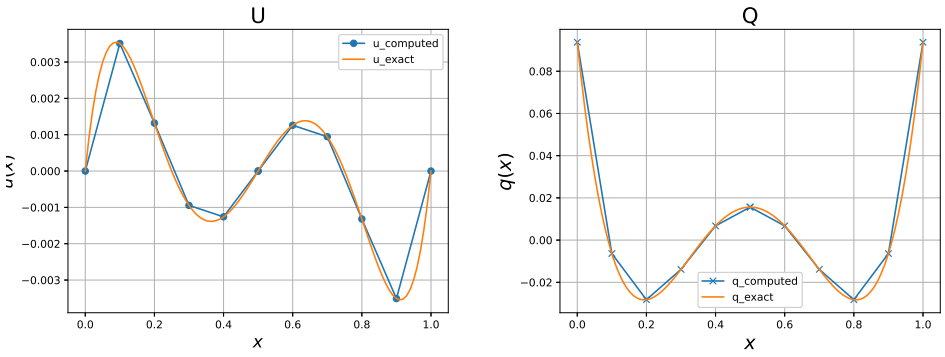
$$u(x) = \sin(5\pi x) + \sin(15\pi x) \tag{3.6}$$

3.3. Restriction and prolongation. Defining the restriction and prolongation operators used to transfer vectors between grids is an integral part of multigrid methods. In the 1D HDG formulation used here, hat functions are used as basis functions for the numerical traces as $\lambda(x) = \sum_{i=1}^{N} \lambda_i \phi_i(x)$. An illustration of these hat functions is shown in Fig 1

Therefore, here we use a simple linear interpolation as our 1D prolongation operator wherein the values at the intermediate nodes are approximated as averages of the neighbouring node values. This operation can be represented in matrix form as,

$$\underline{\underline{I}}^h_{2h}\underline{u}_{2h} = \begin{bmatrix} 1 & & \\ 1/2 & 1/2 & \\ & 1 & \\ & 1/2 & 1/2 \\ & & 1 \end{bmatrix} \begin{bmatrix} u_{2h,1} \\ u_{2h,2} \\ u_{2h,3} \end{bmatrix} = \begin{bmatrix} u_{h,1} \\ u_{h,2} \\ u_{h,3} \\ u_{h,4} \\ u_{h,5} \end{bmatrix} = \underline{u}_h \tag{3.7}$$

When the restriction and interpolation can be represented as



(a) $u(x) - P1$ (b) $q(x) - P1$

Fig. 2: Validation of the multigrid method for P1

$$\underline{\underline{I}}^{h}_{2h} = c\,(\underline{\underline{I}}^{2h}_{h})^{T}, \qquad c \in \mathbb{R} \tag{3.8}$$

they are said to satisfy a variational property that is useful for theoretical convergence proofs. We follow suit with this to define our restriction operation to be a weighted linear operator defined as

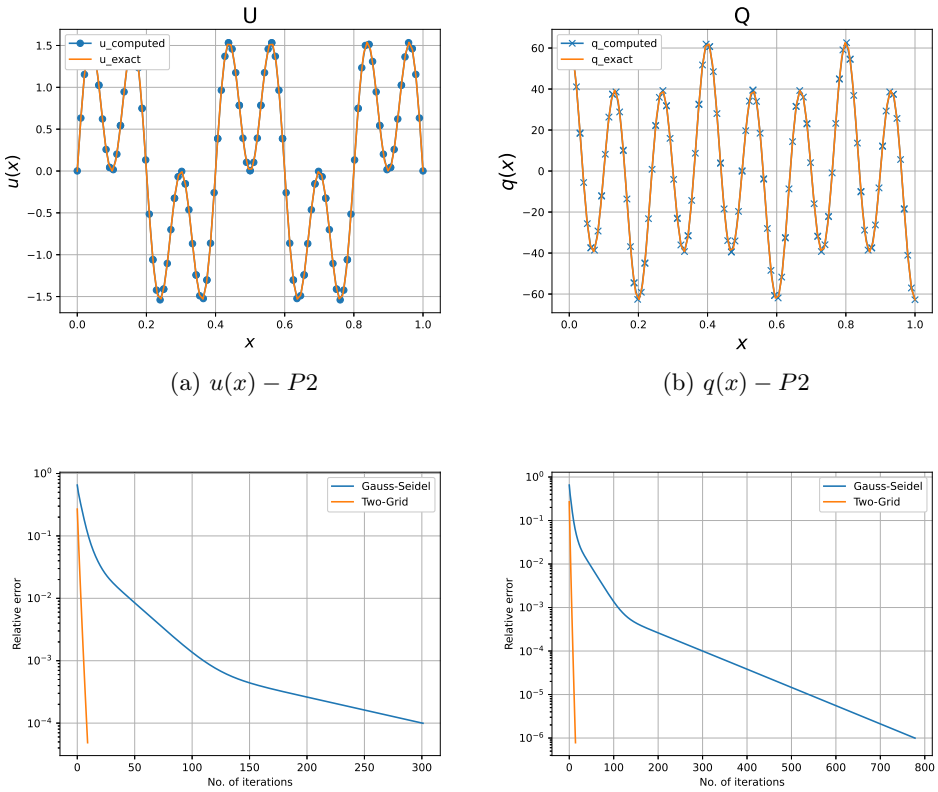
$$\underline{\underline{I}}^{2h}_{h}\,\underline{u}_h = \frac{1}{4}\begin{bmatrix} 2 & 1 & & & \\ & 1 & 2 & 1 & \\ & & & 2 & 1 \end{bmatrix}\begin{bmatrix} u_{h,1} \\ u_{h,2} \\ u_{h,3} \\ u_{h,4} \\ u_{h,5} \end{bmatrix} = \begin{bmatrix} u_{2h,1} \\ u_{2h,2} \\ u_{2h,3} \end{bmatrix} = \underline{u}_{2h} \tag{3.9}$$

4. Results and Comparisons. In this section, we first validate the two-grid method described earlier and then proceede to make comparisons against other iterative methods. Finally, we briefly introduce other variants of the commonly used multigrid methods.

4.1. Validation. We validate the two-grid correction method (algorithm 2.1) by solving P1 with $\nu_1 = 1$ and $\nu_2 = 1$. Figure 2 shows that the computed solution and derivative ($q(x) = u'(x)$) match closely with the analytical solutions. The domain was discretized by two 5th order $p = 5$ elements and after 2 multigrid iterations, the L_2 error of the computed solution was down to machine precision ϵ_{mach}.

4.2. Comparison with classic iterative methods. The Gauss-Seidel (GS) method, introduced in the late 1800s, is a classic iterative method used to solve linear systems $\underline{\underline{A}}\,\underline{x} = \underline{b}$. The procedure involves decomposing the matrix as $\underline{\underline{A}} = \underline{\underline{D}} - \underline{\underline{L}} - \underline{\underline{U}}$, where $\underline{\underline{D}}$ is diagonal, $\underline{\underline{L}}$ is strictly lower trianglular and $\underline{\underline{U}}$ is strictly upper triangular. Using this decomposition, we can define the GS iteration as

$$\underline{x}^{n+1} = (\underline{\underline{D}} - \underline{\underline{L}})^{-1}\underline{\underline{U}}\,\underline{x}^{n} + (\underline{\underline{D}} - \underline{\underline{L}})^{-1}\underline{b} \tag{4.1}$$



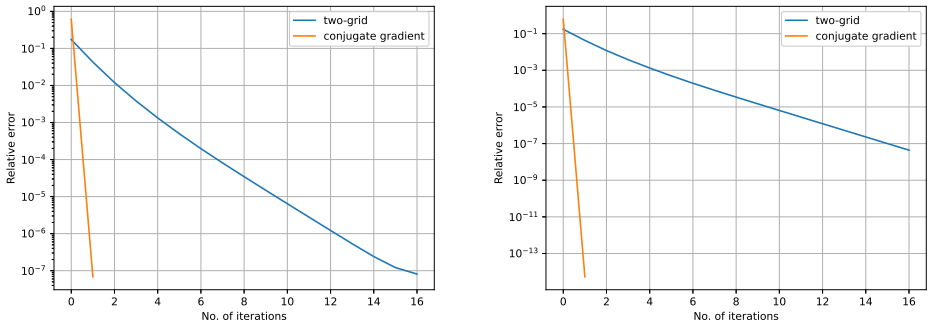
(a) $u(x) - P2$ (b) $q(x) - P2$

(c) $p = 2$ (d) $p = 3$

Fig. 3: Validation of the multigrid method for P2 using 32 elements

These GS iterations will converge to the solution when $\underline{\underline{A}}$ is either diagonally dominant or symmetric positive definite (SPD). The matrices that arise from the test cases used in this study are always diagonally dominant, making this method applicable. Although the GS method should not be considered an algorithm that competes with multigrid methods, comparisons against this method serves to illustrate the effectiveness of multigrid methods.

We solve P2 using the GS method and the two-grid method (with $\nu_1 = \nu_2 = 1$) using two different orders of discretizations ($p = 2, 3$). A comparison of the number of iterations required to converge to a specified tolerance is shown in Fig 3, and it can be clearly seen that the multigrid method requires far fewer iterations than GS. We note that each multigrid cycle involves two GS smoothing iterations, two matrix-vector multiplications for grid-transfers and a direct solve for the coarse grid problem. However, when the number of unknowns n is very large, the solution to the coarse grid problem is negligible compared to the fine grid problem. Further, even when the multigrid iteration count is scaled by $(\nu_1 + \nu_2)$, the iteration count is much smaller than the GS iteration count.



(a) $P2 - p = 2$ (b) $P2 - p = 3$

Fig. 4: Comparison between conjugate gradient method and two-grid method

4.3. Comparison with the conjugate gradient method. It can be shown that the linear system that arises from an HDG discretization of the 1D Poisson equation is symmetric positive-definite []. Therefore, a suitable alternative to the multigrid algorithm for this problem would be the conjugate gradient (CG) method (a Krylov subspace method). In this subsection, we compare the CG algorithm against the two-grid method. The CG algorithm can be summarised as

Algorithm 4.1 Conjugate gradient algorithm

$\underline{r}_0 = \underline{b} - \underline{\underline{A}}\,\underline{x}_0$
$\underline{\rho}_0 = \underline{r}_0$
$k = 0$
while $\underline{r}_{k+1} \geq$ tolerance **do**
  $\alpha_k = \underline{r}_k^T \underline{r}_k / \underline{\rho}_k^T \underline{\underline{A}}\,\underline{\rho}_k$ ▷ step size
  $\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{\rho}_k$
  $\underline{r}_{k+1} = \underline{r}_k - \alpha_k \underline{\underline{A}}\,\underline{\rho}_k$ ▷ new residual
  $\beta_k = \underline{r}_{k+1}^T \underline{r}_{k+1} / \underline{r}_k^T \underline{r}_k$
  $\underline{\rho}_{k+1} = \underline{r}_{k+1} + \beta_k \underline{\rho}_k$ ▷ new search direction
  $k = k + 1$

Although the CG algorithm has many computations per cycle, the computation of the matrix-vector product $\underline{\underline{A}}\,\underline{\rho}_k$ is most expensive and therefore dominates over the remaining computations. In contrast, the two-grid algorithm has $(\nu_1 + \nu_2)$ matrix-vector products for the smoothing iterations followed by two more products for the grid transfer operations. We compare the two methods by solving $P2$ using 32 elements and find that CG converges in just two iterations while the two-grid method takes many more iterations. It seems clear that the CG is the better choice for the 1D elliptic problem we have considered here. However, it is useful to note that CG only works for SPD matrices, whereas multigrid methods are more generally applicable.

4.4. V-cycle algorithm. The most expensive step in the two-grid scheme presented in Algorithm 2.1 is the direct solve needed to compute the coarse grid error. Noticing that $\underline{\underline{A}}\,\underline{x}_h = \underline{f}_h$ and $\underline{\underline{A}}\,\underline{e}_{2h} = \underline{r}_{2h}$ have the same structure, we can

improve on the two-grid method by solving the error equations using an even coarser grid. This forms the fundamental idea behind the v-cycle algorithm described below.

Algorithm 4.2 One cycle of a V-cycle algorithm

1: Choose an initial guess $\underline{x}_h^0$ ▷ $VG_h(\underline{x}_h^n, \underline{b}_h) \to \underline{x}_h^{n+1}$
2: Relax ν_1 iterations of $\underline{\underline{A}}_h\, \underline{x}_h^0 = \underline{b}_h \to \underline{x}_h^{1/3}$ ▷ Pre-smoothing iterations
3: Compute the residual as $\underline{r}_h = \underline{b}_h - \underline{\underline{A}}_h\, \underline{x}_h^{1/3}$
4: Restrict residual to coarse grid as $\underline{r}_{2h} = \underline{\underline{I}}_{2h}^h\, \underline{r}_h$
5: **if** h corresponds to coarsest grid **then**
6: $\quad \underline{x}_h^{2/3} = \underline{x}_h^{1/3}$
7: $\quad$ Skip to step 13
8: **else**
9: $\quad$ Compute coarse grid error as $VG_h(0, \underline{r}_{2h}) \to \underline{e}_{2h}$ ▷ Coarse grid correction
10: $\quad$ Prolongate error to fine grid as $\underline{e}_h = \underline{\underline{I}}_h^{2h}\, \underline{e}_{2h}$
11: $\quad$ Correct $\underline{x}_h^{2/3} = \underline{x}_h^{1/3} + \underline{e}_h$
12: **end if**
13: Relax ν_2 iterations of $\underline{\underline{A}}_h\, \underline{x}_h^{2/3} = \underline{b}_h \to \underline{x}_h^1$ ▷ Post-smoothing iterations =0

As we deal with coarser and coarser grids, the cost of operating on the corresponding vectors and matrices sharply decreases. Therefore, we can expect an improved performance of the V-cycle in comparison to the two-grid method. This method was implemented, and it was found to perform marginally better than the two-grid method for our test cases.

5. Conclusion. Multigrid methods are definitely amongst the most exciting algorithms for linear systems that have come up in the last few decades. Currently, these methods are well-developed for elliptic problems and are especially popular for 2D and 3D problems. Efficient application of such methods for other systems of equations, unstructured grids and different discretizations [5, 9, 11, 13, 17] are subjects of active research. The iterative nature of these multigrid methods also suggests that they can be suitable candidates for nonlinear problems, and this avenue has been explored. An interesting question is whether these methods are applicable when no grid is associated with the linear system. Algebraic multigrid (AMG) methods address this very situation, and an introduction to the fundamental ideas can be found in [3]. On the whole, multigrid methods are receiving substantial amounts of well-deserved attention, giving rise to possibilities of exciting developments in the near future.

REFERENCES

[1] *A Multigrid Tutorial, Second Edition*, https://epubs.siam.org/doi/book/10.1137/1.9780898719505 (accessed 2022-05-03).

[2] J. H. Bramble, *Multigrid methods*, Chapman and Hall/CRC, New York, Nov. 2019, https://doi.org/10.1201/9780203746332.

[3] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive Algebraic Multigrid*, SIAM Journal on Scientific Computing, 27 (2006), pp. 1261–1286, https://doi.org/10.1137/040614402, https://epubs.siam.org/doi/abs/10.1137/040614402 (accessed 2022-05-11). Publisher: Society for Industrial and Applied Mathematics.

[4] L. Chen, J. Wang, Y. Wang, and X. Ye, *An auxiliary space multigrid preconditioner for the weak Galerkin method*, Computers & Mathematics with Applications, 70 (2015),

pp. 330–344, https://doi.org/10.1016/j.camwa.2015.04.016, https://www.sciencedirect.com/science/article/pii/S0898122115001972 (accessed 2022-05-03).

[5] B. Cockburn, O. Dubois, J. Gopalakrishnan, and S. Tan, *Multigrid for an HDG methodâ* , IMA Journal of Numerical Analysis, 34 (2014), pp. 1386–1425, https://doi.org/10.1093/imanum/drt024, https://doi.org/10.1093/imanum/drt024 (accessed 2022-03-29).

[6] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar, *A survey of direct methods for sparse linear systems*, Acta Numerica, 25 (2016), pp. 383–566, https://doi.org/10.1017/S0962492916000076, https://www.cambridge.org/core/journals/acta-numerica/article/survey-of-direct-methods-for-sparse-linear-systems/8AE7AC55909389F7EA1F027855AC4044 (accessed 2022-05-04). Publisher: Cambridge University Press.

[7] R. P. Fedorenko, *A relaxation method for solving elliptic difference equations*, Journal of Comput. Math. Math. Phys., (1962), pp. 1092–1096.

[8] R. P. Fedorenko, *The speed of convergence of one iterative process*, Journal of Comput. Math. Math. Phys., (1964), pp. 227–235.

[9] N. Fehn, P. Munch, W. A. Wall, and M. Kronbichler, *Hybrid multigrid methods for high-order discontinuous Galerkin discretizations*, Journal of Computational Physics, 415 (2020), p. 109538, https://doi.org/10.1016/j.jcp.2020.109538, https://www.sciencedirect.com/science/article/pii/S0021999120303120 (accessed 2022-05-03).

[10] N. Fehn, P. Munch, W. A. Wall, and M. Kronbichler, *Hybrid multigrid methods for high-order discontinuous Galerkin discretizations*, Journal of Computational Physics, 415 (2020), p. 109538, https://doi.org/10.1016/j.jcp.2020.109538, https://www.sciencedirect.com/science/article/pii/S0021999120303120 (accessed 2022-03-30).

[11] D. Fortunato, C. H. Rycroft, and R. Saye, *Efficient Operator-Coarsening Multigrid Schemes for Local Discontinuous Galerkin Methods*, SIAM Journal on Scientific Computing, 41 (2019), pp. A3913–A3937, https://doi.org/10.1137/18M1206357, https://epubs.siam.org/doi/abs/10.1137/18M1206357 (accessed 2022-05-03). Publisher: Society for Industrial and Applied Mathematics.

[12] G. Fu and W. Kuang, *Uniform block-diagonal preconditioners for divergence-conforming HDG Methods for the generalized Stokes equations and the linear elasticity equations*, arXiv:2104.13886 [cs, math], (2022), http://arxiv.org/abs/2104.13886 (accessed 2022-05-03). arXiv: 2104.13886.

[13] J. Gopalakrishnan and S. Tan, *A Convergent Multigrid Cycle for the Hybridized Mixed Method*, 2009.

[14] M. H. Gutknecht, *A Brief Introduction to Krylov Space Methods for Solving Linear Systems*, in Frontiers of Computational Science, Y. Kaneda, H. Kawamura, and M. Sasai, eds., Berlin, Heidelberg, 2007, Springer, pp. 53–62, https://doi.org/10.1007/978-3-540-46375-7_5.

[15] J. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, vol. 54, Jan. 2007.

[16] M. LLC, *Finite element basis functions*, http://hplgit.github.io/INF5620/doc/pub/sphinx-fem/._main_fem003.html (accessed 2022-05-10).

[17] P. Lu, A. Rupp, and G. Kanschat, *Homogeneous multigrid for HDG*, IMA Journal of Numerical Analysis, (2021), p. drab055, https://doi.org/10.1093/imanum/drab055, https://doi.org/10.1093/imanum/drab055 (accessed 2022-03-29).

[18] S. Muralikrishnan, T. Bui-Thanh, and J. N. Shadid, *A multilevel approach for trace system in HDG discretizations*, Journal of Computational Physics, 407 (2020), p. 109240, https://doi.org/10.1016/j.jcp.2020.109240, https://www.sciencedirect.com/science/article/pii/S0021999120300140 (accessed 2022-05-03).

[19] S. Muralikrishnan, M.-B. Tran, and T. Bui-Thanh, *iHDG: An Iterative HDG Framework for Partial Differential Equations*, SIAM Journal on Scientific Computing, 39 (2017), pp. S782–S808, https://doi.org/10.1137/16M1074187, https://epubs.siam.org/doi/abs/10.1137/16M1074187 (accessed 2022-05-03). Publisher: Society for Industrial and Applied Mathematics.

[20] S. Muralikrishnan, M.-B. Tran, and T. Bui-Thanh, *An improved iterative HDG approach for partial differential equations*, Journal of Computational Physics, 367 (2018), pp. 295–321, https://doi.org/10.1016/j.jcp.2018.04.033, https://www.sciencedirect.com/science/article/pii/S0021999118302584 (accessed 2022-05-03).

[21] C. R. Nastase and D. J. Mavriplis, *High-order discontinuous Galerkin methods using an hp-multigrid approach*, Journal of Computational Physics, 213 (2006), pp. 330–357, https://doi.org/10.1016/j.jcp.2005.08.022, https://www.sciencedirect.com/science/article/pii/S0021999105003839 (accessed 2022-05-03).

[22] N. C. Nguyen, J. Peraire, and B. Cockburn, *An implicit high-order hybridizable discontinuous Galerkin method for linear convectionâdiffusion equations*, Journal of Computational Physics, 228 (2009), pp. 3232–3254, https://doi.org/10.1016/j.jcp.2009.01.030, https://www.sciencedirect.com/science/article/pii/S0021999109000308 (accessed 2022-05-11).

[23] P. J. Roache, *The Method of Manufactured Solutions for Code Verification*, in Computer Simulation Validation: Fundamental Concepts, Methodological Frameworks, and Philosophical Perspectives, C. Beisbart and N. J. Saam, eds., Simulation Foundations, Methods and Applications, Springer International Publishing, Cham, 2019, pp. 295–318, https://doi.org/10.1007/978-3-319-70766-2_12, https://doi.org/10.1007/978-3-319-70766-2_12 (accessed 2022-05-11).

[24] T. Wildey, S. Muralikrishnan, and T. Bui-Thanh, *Unified Geometric Multigrid Algorithm for Hybridized High-Order Finite Element Methods*, SIAM Journal on Scientific Computing, 41 (2019), pp. S172–S195, https://doi.org/10.1137/18M1193505, https://epubs.siam.org/doi/abs/10.1137/18M1193505 (accessed 2022-05-03). Publisher: Society for Industrial and Applied Mathematics.