On the Construction of AMG Prolongation through Energy Minimization

Giovanni Isotton
g.isotton@m3eweb.it
M3E S.r.l.
Padova, Italy

Andrea Franceschini
andrea.franceschini@unipd.it
University of Padova
Padova, Italy

Carlo Janna
carlo.janna@unipd.it
University of Padova
Padova, Italy

ABSTRACT

Algebraic Multigrid (AMG) is a very popular iterative method used in several applications. This wide diffusion is due to its effectiveness in solving linear systems arising from PDEs discretization. The key feature of AMG is its optimality, i.e., the ability to guarantee a convergence rate independent of the mesh size for different problems. This is obtained through a good interplay between the smoother and the interpolation. Unfortunately, for difficult problems, such as those arising from structural mechanics or diffusion problems with large jumps in the coefficients, standard smoothers and interpolation techniques are not enough to ensure fast convergence. In these cases, an improved prolongation operator is required to enhance the AMG effectiveness. In this work, we present an updated prolongation according to an energy minimization criterion and show how this minimization can be seen as a constrained minimization problem. In detail, we have that the constraint is twofold: the prolongation must be sparse, and its range must represent the operator near-kernel. Even though energy minimization is well-known in the AMG community, it has little application due to both its cost and difficult implementation. Here, we would like to make energy minimization feasible through suitable preconditioners and effective implementation. In particular, to solve this problem, we propose two strategies: a restricted Krylov subspace iterative procedure and the null-space method. Both approaches can be preconditioned to speed up the setup time. Finally, thanks to some numerical experiments, we demonstrate how the convergence rate can be significantly increased at a reasonable setup cost.

CCS CONCEPTS

• **Mathematics of computing** → **Solvers**; **Numerical analysis**;
• **Applied computing** → *Engineering*.

KEYWORDS

algebraic multigrid, energy minimization, restricted Krylov subspaces, null space method

PASC '22, June 27–29, 2022, Basel, Switzerland

ACM ISBN 978-1-4503-9410-9/22/06…$15.00
https://doi.org/10.1145/3539781.3539796

ACM Reference Format:
Giovanni Isotton, Andrea Franceschini, and Carlo Janna. 2022. On the Construction of AMG Prolongation through Energy Minimization. In *Platform for Advanced Scientific Computing Conference (PASC '22), June 27–29, 2022, Basel, Switzerland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3539781.3539796

1 INTRODUCTION

With the increasing availability of powerful computational resources, scientific and engineering applications are becoming more and more demanding in terms of both memory and CPU time. The current simulation models may easily grow up to several millions or even billions of unknowns and the efficient solution of the related sparse linear systems of equations

$$Ax = b \tag{1}$$

where A is a symmetric positive definite (SPD) matrix, may represent one of the most, and often the most [11, 16], expensive tasks in any numerical application.

AMG is a very popular and effective iterative method for the solution of (1). The main feature of AMG is its optimality, i.e., it is characterized by a complexity that grows only linearly with the system size. AMG is a very complex machinery made up of several algorithmic components such as smoother, coarsening and prolongation. In this contribution, we will focus on an improved algorithm to build an accurate prolongation operator. This method was first introduced in the early 1980s for the solution of Poisson problems [2, 25]. In those pioneering works, the system unknowns were partitioned into *Fine* and *Coarse* variables, and the latter were used as primary unknowns of the coarser problem. For historical reasons, any AMG method performing coarsening through a coarse/fine (C/F) partition is called *Classical* AMG. The fact that the constant vector is a good approximation of the near-kernel of the Poisson operator was exploited to build optimal solution strategies for these matrices. Unfortunately, for other problems, such as structural mechanics, a larger near-kernel is required to achieve good results. In the 1990s, the Smoothed Aggregation AMG (SA-AMG) method was introduced to overcome this limitation. In contrast to Classical AMG, in this case, the coarsening consists of aggregating several fine nodes in one coarse level unknown and the interpolation operator is constructed by interpolating exactly a few approximations of the near-kernel [26, 27]. Since then, many other multigrid variations have appeared in the literature, e.g., the element based AMG family, with energy-minimization AMGe [6], element-free AMGe [13] and spectral AMGe [9], but also the adaptive and Bootstrap AMG (BAMG) [3, 4, 7, 10, 18], where the near-kernel of the operator is approximated adaptively during the AMG setup stage. Despite many differences, all the above methods perform coarsening based

on a C/F partitioning or aggregation of the unknowns, hence the common usage of distinguishing between classical or aggregation-based AMG. For further reading on the AMG variants, we suggest the review work of Xu and Zikatanov [28].

In this work, we focus on the prolongation, i.e., on the operator interpolating the correction computed on a coarser level into a finer one. Among other desirable features, it is of paramount importance for the prolongation to interpolate low-frequency (i.e., smooth) error components. *Optimal prolongation* has been derived following this idea [5]. On the other side, minimizing an approximation measure of the AMG operator leads to the definition of the *ideal interpolator*, see [5, 29]. Beyond these theoretical properties, the prolongation has to be sparse, to keep under control both the setup cost of the single level and the overall complexities of the multigrid framework. Here, we propose an efficient way to build a sparse prolongation that approximately satisfies both the ideal and the optimal requirements.

The remainder of this work is organized as follows. In Section 2, a brief overview of the multigrid methods is presented, with a focus on the key components, i.e., smoother and near-null space. Then, in Section 3, we propose a strategy to improve the prolongation operator based on energy minimization. Two algorithms to implement this energy minimization, i.e., the restricted Krylov subspace method and the null space method, are presented and analyzed in Section 4, where we also introduce the use of a preconditioner to speed up the setup stage. After this, in Section 5, thanks to several numerical results, we assess the performances of the enhanced prolongation. Finally, some conclusions are drawn in Section 6.

2 CLASSICAL ALGEBRAIC MULTIGRID

The main components of any AMG method are: i) a multilevel hierarchy, ii) an interpolation operator, iii) a smoother. The complementary between coarse-grid correction and relaxation is the reason behind the optimality of this class of solvers. In this work, we adopt a classical AMG approach, where coarse variables are a subset of the fine level variables.

The smoother is usually a stationary iterative method. Its task is to reduce the high-frequency errors, i.e., the error components along the eigenvectors with the largest eigenvalues of A. While in many AMG methods the smoother is a point-wise relaxation method, such as (block) Jacobi or Gauss-Seidel, in our multigrid we adopt a factorized approximate inverse (FSAI), as described in Janna et al. [15]. The FSAI preconditioner is given by

$$M^{-1} = FF^T \approx A^{-1} \tag{2}$$

where F is computed to provide an optimal preconditioner, i.e., for a chosen pattern, F minimizes the Frobenius norm of

$$\|I - FL\|_F \tag{3}$$

where L is the exact Cholesky lower factor of $A = LL^T$.

Another key component of AMG is the near-null space, i.e., the subspace of the smooth vectors, associated with the components of the error that are not reduced by the smoother. Indeed, FSAI, as long as any single level preconditioners, can accurately represent only the higher part of the eigenspectrum of A, with the lower part just roughly approximated. To build an effective AMG solver, we need an accurate approximation of the subspace spanned by the eigenvectors associated with the smallest eigenvalues of

$$I - M^{-1}A = I - FF^T A \tag{4}$$

Generally, we start from an initial approximation, which is given by rigid body modes in elasticity or the constant vector for Poisson problems, and improve this initial guess through a Simultaneous Rayleigh Quotient Minimization by Conjugate Gradients (SRQCG), as introduced by [17] and used by [11].

The last component in AMG is the prolongation operator P. Since the optimality of the multigrid method relies on the complementarity between smoothing and coarse-grid correction, it is of paramount importance that the prolongation operator accurately represents the low-frequency components of the error. In the literature, there are several approaches to build P. Among them, we recall the *ideal prolongation* [29] and the *optimal prolongation* [5] which are strongly connected with the proposed approach. In the next section we will focus on a prolongation setup strategy, based on energy minimization, that tries to combine the nice features of the two aforementioned methods.

3 PROLONGATION SETUP THROUGH ENERGY MINIMIZATION

Before focusing on energy minimization, we briefly recall the guidelines for optimal prolongation construction. Given the generalized eigenproblem:

$$Av = \lambda Mv \tag{5}$$

the optimal prolongation is constructed by collecting in the rectangular matrix V the eigenvectors associated with the n_c smallest eigenvalues, where n_c is the size of the coarse set:

$$P_{\text{opt}} = V \tag{6}$$

Clearly, in a practical setting, the generalized eigenproblem is solved only approximately for a few eigenpairs, n_v, and it is required that the range of prolongation includes this reduced size V.

From this simplification, it is then possible to construct the energy minimizing prolongation following the two guidelines below:

(1) on one side, in line with the *optimal prolongation*, the range of prolongation must include V, the approximate near kernel of A:
$$V \subseteq \text{range}(P_{\text{opt}}) \tag{7}$$
(2) on the other side, in line with the *ideal prolongation* construction, the energy of each column of P is minimized over the set of all possible prolongations having a given non-zero pattern $\mathcal{P}$:
$$P_{\text{ideal}} = \underset{P \in \mathcal{P}}{\text{argmin}} \left(\text{tr}(P^T AP) \right) \tag{8}$$

Being P_{ideal} a dense matrix, it is important to define a non-zero pattern $\mathcal{P}$ for the prolongation operator. There are several strategies to define this non-zero pattern [5], however, this topic is beside the scope of this work and only one method to build $\mathcal{P}$ will be presented below.

The idea of energy minimization AMG has been exploited for both symmetric and non-symmetric operators in several works, see for instance [19, 20, 23, 24], and, though requiring some additional effort in the setup, gives rise to improved preconditioners.

Suppose that a C/F partition of the unknowns of A has already been found with n_f fine nodes and n_c coarse nodes (collected in the $\mathcal{F}$ and C sets, respectively), and that A has been ordered accordingly:

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{fc}^T & A_{cc} \end{bmatrix} \tag{9}$$

where $A_{cf} = A_{fc}^T$ since A is SPD. In the context of classical AMG, the prolongation takes the form:

$$P = \begin{bmatrix} W \\ I \end{bmatrix} \tag{10}$$

so that only the entries in W have to be computed. We assume that a Strength of Connection (SoC) [22] matrix is available, and we denote by S the binary matrix representing the non-zero pattern of SoC after filtering out the weak connections. Suppose now that we have already computed a tentative prolongation $\overline{P}_0$, the non-zero pattern of the final prolongation must be chosen. One way to determine $\mathcal{P}$ is by computing this product between binary matrices:

$$\overline{P} = S^k \overline{P}_0 \tag{11}$$

for a small power k (typically $k = 1$), and enforcing the pattern of the identity for the block of coarse unknowns.

Condition (7) is equivalent to require the existence of a rectangular matrix, X with the same number of columns of V, such that $PX = V$. Using the C/F splitting defined in (9), the near kernel of V is partitioned as:

$$V = \begin{bmatrix} V_f \\ V_c \end{bmatrix} \tag{12}$$

Using the definition of P in (10), it is possible to write:

$$\begin{cases} WX = V_f \\ X = V_c \end{cases} \tag{13}$$

and the ability to interpolate low-frequency errors expressed in ((7)) reads:

$$W\, V_c = V_f \tag{14}$$

which can be viewed as a set of n_f conditions on the rows of W. By denoting as w_i^T and v_i^T the i-th rows of W and V, respectively, condition (14) can be written as:

$$V_c^T w_i = v_i \qquad \forall i \in \mathcal{F} \tag{15}$$

By exploiting the fact that $\mathcal{P}$ must be sparse, and thus a fine node has to be interpolated using just a few coarse nodes, we can rewrite (15) considering only the nonzeros of w_i:

$$V_c(\mathcal{J}_i, :)^T \widetilde{w}_i = v_i \qquad \forall i \in \mathcal{F} \tag{16}$$

where $\widetilde{w}_i$ are small vectors collecting nonzeros of w_i, and $\mathcal{J}_i$ is the set of column indices of the prescribed nonzeros of the i-th row of W. The matrix $V_c(\mathcal{J}_i, :)$, which is a dense submatrix of V_c has n_{tv} columns, with n_{tv} the number of test vectors. It is important to note that the n_f constraints (16) are completely independent each other, that is each unknown row w_i^T of W must satisfy only one constraint.

On the other side, minimizing (8) is equivalent to minimizing the energy of the individual columns p_i of P:

$$p_i = \underset{p \in \mathcal{P}}{\operatorname{argmin}}\, p^T A p \qquad \forall i \in C \tag{17}$$

Denoting by I_i the set of nonzero row indices of the i-th column of W, and by $\widetilde{h}_i$ the vector collecting the nonzero entries of the i-th column of W, the minimization above is equivalent to solving:

$$A(\mathcal{I}_i, \mathcal{I}_i)\widetilde{h}_i = -A(\mathcal{I}_i, i) \qquad \forall i \in C \tag{18}$$

As before, we note that each column of P can be found independently from the others, by solving a small dense linear system of equations.

The pair of conditions 1 and 2 gives rise to a constrained minimization problem, whose solution is the desired energy minimal prolongation with a range matching the near kernel and respecting the prescribed non-zero pattern $\mathcal{P}$. We stress that this approach, as many other AMG approaches, requires a sparse prolongation to be feasible. If too many non-zeroes are included in P, the minimization would become practically impossible and the AMG application too expensive.

A widely-used class of methods to solve this kind of problems are the Lagrange multipliers, which reduce the constrained minimization problem to the solution of a saddle point system:

$$\begin{bmatrix} K & B \\ B^T & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{19}$$

In the expression (19), p is the vector form of the prolongation entries that we are computing, λ is the vector of Lagrange multipliers (the auxiliary variable used to impose the constraint and that we do not really need to know), K is a block diagonal matrix collecting the small and dense matrices $A(\mathcal{I}_i, \mathcal{I}_i)$ for every column of P, B is a matrix collecting $V_c(\mathcal{J}_i, :)$ for every row or P, f is a vector collecting $-A(\mathcal{I}_i, i)$ for every column of P and, finally, g is a vector collecting $\widetilde{v}_i$ for every row or P.

Note that, both K and B can be sorted so that they become block diagonal matrices. If we number the nonzero entries of P by columns, then K is block diagonal, conversely, if we number by rows, than B is block diagonal, but unfortunately, there is no sorting able to produce two block diagonal matrices at the same time.

4 SOLUTION ALGORITHMS

The saddle-point linear system (19) that arises from the energy minimization is a classical problem in numerical linear algebra. To solve it, there are several techniques available in the literature. In this context, the key factor is represented by the efficiency. In fact, solution to (19) is only a part of the AMG setup, so that, if we want a competitive method, we must solve this problem very quickly.

We propose two possible solution algorithms:

- a restricted conjugate gradient;
- the null space method.

The former has been originally proposed in [23] while the latter is a classical method to solve saddle-point systems [1]. We show that, with a specific choice of the null space and using the conjugate gradient on the reduced systems, the two methods are equivalent and require the same number of iterations to reach the desired accuracy. Finally, we show how preconditioning can be used to accelerate convergence.

4.1 Solution through restricted conjugate method

We perform energy minimization starting from a tentative prolongation, P_0, that already satisfies the near kernel constraints. In other words, by denoting by p_0 the initial prolongation in vector form, we assume p_0 is such that:

$$B^T p_0 = g \tag{20}$$

This condition can always be guaranteed, provided that the prolongation pattern is large enough that B^T is full rank.

Defining the final prolongation as the tentative prolongation p_0 plus the correction Δp, the problem can be recast as that of finding:

$$\Delta P = \underset{\Delta P_{\text{trial}} \in \mathcal{P}}{\operatorname{argmin}} \operatorname{tr}\left((P_0 + \Delta P_{\text{trial}})^T A (P_0 + \Delta P_{\text{trial}})\right) \tag{21}$$

subject to the constraint

$$B^T \Delta p = 0 \tag{22}$$

By recalling that ΔP has nonzero components only in W, that is $\Delta P = \begin{bmatrix}\Delta W^T & 0\end{bmatrix}^T$, and using the C/F partition (9), we can write:

$$\begin{aligned} &\operatorname{tr}((P_0 + \Delta P)^T A (P_0 + \Delta P)) = \\ &= \operatorname{tr}(\Delta W^T A_{ff} \Delta W) + 2\operatorname{tr}(W_0^T A_{ff} \Delta W) + 2\operatorname{tr}(A_{fc}^T \Delta W) + \\ &+ \operatorname{tr}(W_0^T A_{ff} W_0) + 2\operatorname{tr}(A_{fc}^T W_0) + \operatorname{tr}(A_{cc}) \end{aligned} \tag{23}$$

where the last three terms are independent of the increment ΔP. Thus the problem becomes that of minimizing:

$$\delta w^T K \delta w + 2 w_0^T K \delta w + 2 f^T \delta w \tag{24}$$

subject to the constraint $B^T \delta w = 0$, where K and f are defined as in (19) and δw is the vector form of ΔW.

The minimization above can be performed using the (preconditioned) conjugate gradient (CG), as suggested in Olson et al. [23], ensuring that the initial solution and the search direction satisfy the constraint, namely equations (20) and (22) are satisfied.

By defining the projection orthogonal to B:

$$\Pi_B = I - B(B^T B)^{-1} B^T \tag{25}$$

any vector that is projected by Π_B satisfies the homogeneous constraint (22):

$$B^T \Pi_B x = 0, \quad \forall x \tag{26}$$

and allow us to define a restricted linear system which is solvable though CG:

$$\Pi_B K \Pi_B \Delta w = -\Pi_B (f + K w_0) \tag{27}$$

starting from $\Delta w = 0$. Due to its block diagonal structure, it is easy to find a QR decomposition of B, $B = QR$, and the projection becomes:

$$\Pi_B = I - QQ^T \tag{28}$$

By denoting $K_\Pi = \Pi_B K \Pi_B$ and $\bar{f} = f + K w_0$, the Krylov subspace that is constructed by CG is:

$$\mathcal{K}_m = \operatorname{span}\{\Pi_B \bar{f}, K_\Pi \bar{f}, K_\Pi^2 \bar{f}, \ldots, K_\Pi^m \bar{f}\} \tag{29}$$

The CG pseudocode is provided in algorithm 1. The same projection can be applied to a nonsymmetric Krylov subspace method, such as GMRES, allowing for the extension of the energy minimization procedure to the case of non-symmetric A.

Algorithm 1. CG for energy minimization

```
1:  procedure EMIN_CG(maxit,K,Π_B,w_0,w)
2:      Δw = 0;
3:      r = Π_B(f + K w_0);
4:      for i = 1, ..., maxit do
5:          γ = r^T r;
6:          if i = 1 then
7:              y = r;
8:          else
9:              β = γ/γ_old;
10:             y = r + β y;
11:         end if
12:         γ_old = γ;
13:         ỹ = Π_B K y;
14:         α = γ/(y^T ỹ);
15:         Δw = Δw + α y;
16:         r = r + α ỹ;
17:     end for
18:     w = w_0 + Δw;
19: end procedure
```

4.2 Solution through the null space method

Another solution strategy to (19) is the use of the null space method [1]. To follow this idea, we need a null-space for B^T, that is Z such that $B^T Z = 0$, and a particular solution $\hat{p}$ such that $B^T \hat{p} = g$. After defining the general solution as:

$$p = Zv + \hat{p} \tag{30}$$

and substituting this expression for p into the first block of equations of (19), after pre-multiplying by the full-rank matrix Z^T, we have:

$$Z^T K Z v = Z^T (f - K\hat{p}) \tag{31}$$

which is an SPD system of reduced size $n - m$ in the unknown v. Once v is known, the final value of p can be easily recovered using (30).

Finding a suitable Z in our case is relatively cheap, since we can easily compute a QR decomposition of B, since it consists of a set of n_f blocks of n_{tv} columns which are orthogonal each other. Once the QR decomposition $QR = B$ is computed, we set $Z = [q_{m+1} q_{m+2} \ldots q_n]$, with q_i the i-th column of Q, thus guaranteeing that $B^T Z = 0$. As in the previous case, we can assume that we already have a tentative prolongation p_0 satisfying $B^T p_0 = g$ and, if this is not the case, we can easily recover it. Note that the matrix $Z^T K Z$ needs not to be formed, as its product times a vector is more economically performed through the successive multiplication by Z, K and Z^T, and the use of CG is the most effective option.

4.3 Equivalence between the restricted conjugate gradient and the null space method

Using CG for solving the nullspace system $Z^T K Z v = Z^T (f - K\hat{p})$ is equivalent to applying restricted CG. In fact, recalling that $Q \in \mathbb{R}^{n \times m}$ in (28) and $Z \in \mathbb{R}^{n \times (n-m)}$ in (30) are disjoint parts of the

complete QR factorization of B:

$$B = [Q, Z] \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{32}$$

it is immediately seen that $QQ^T + ZZ^T = I$, that allows us to write the projector Π_B as:

$$\Pi_B = I - QQ^T = ZZ^T \tag{33}$$

The restricted CG iteration matrix thus becomes $\Pi_B K \Pi_B = Z\left(Z^T K Z\right) Z^T$, which clearly has m null eigenvalues, whose eigenvectors span the range of Q. The non-trivial eigenvectors v can be written as linear combination of the columns of Z:

$$v = Zw \tag{34}$$

giving rise to the following eigenproblem:

$$Z\left(Z^T K Z\right) Z^T v = Z\left(Z^T K Z\right) Z^T Z w = \lambda Z w \tag{35}$$

which, after noting that $Z^T Z = I$, clearly shows that the nonzero part of the eigenspectrum of $\Pi_B K \Pi_B$ is the same as the eigenspectrum of $Z^T K Z$.

4.4 Preconditioning

Preconditioning is used to accelerate the iterative solution to (31). We refer to the null space system $Z^T K Z$ as the two methods are equivalent. Two viable options that we explore are the following:

(1) using a Jacobi preconditioner of $Z^T K Z$. Note that forming the entire matrix $Z^T K Z$ is not necessary, but only its diagonal entries can be computed;
(2) using the projection over Z of an effective preconditioner of K, that is preconditioning with $Z^T M^{-1} Z$ with $M^{-1} \simeq K^{-1}$. Note that, due to the block diagonal structure of K, M^{-1} is easily computable also in the form of an exact or incomplete Cholesky factorization of K.

We notice that projecting an effective preconditioner for K onto Z is not guaranteed to be a good choice. We found experimentally that this choice is quite effective, as will be shown in the numerical examples, and its theoretical explanation is the focus of ongoing research.

4.5 Observations

Supposing that P has on average r nonzeros per row, the matrix K in (19) would be of size $n = r\, n_c \frac{n_f}{n_c} = r\, n_f$, while B would have $m = n_f\, n_{tv}$ columns. Hence, the overall dimension of K and B is significantly much larger than that of the original system. From our numerical experiments, we have observed that K easily reaches 20 times the nonzeros of A in elasticity problems. For this reason, the solution of (19) has to be performed with special care, taking advantage of its special structure. In some circumstances it may also be useful to apply K and B in a matrix free fashion, i. e., without explicitly storing them in memory but grabbing the needed entries directly from A and V whenever necessary. The main issue of this *matrix free* approach is that, if we want to use a projected preconditioner for K, then we must compute it from scratch at every application.

Table 1: Size and number of non-zeroes of the matrices used in the tests. The application giving to the problem is also provided.

Matrix	source	n	nnz	nnz/row
Cube_105k	elasticity	105,597	4,079,357	38.63
Cube_739k	elasticity	739,167	29,610,351	40.06
Cube_5317k	elasticity	5,317,443	222,268,213	41.80
Mech_447k	elasticity	447,703	18,243,793	40.75
finger4m	porous flow	4,718,592	23,591,424	5.00
Pflow_742	porous flow	742,793	37,138,461	50.00
cavity	CFD	1,000,000	6,940,000	6.94

5 NUMERICAL EXPERIMENTS

The implementation used in the numerical tests is based on a C++ library implementing all the most compute-intensive kernels. The library exploits a shared-memory parallelization through OpenMP directives [8] and is called by MATLAB [21] thanks to a MEX (MATLAB executable) interface. All the numerical examples have been run on a local server running Ubuntu 20.04, equipped with a dual Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz with ten cores each, and 64 GB of RAM. Even though part of the implementation is still in native MATLAB and sequential, all 20 cores have been used in the parallelized part of the algorithm. We remind that since the implementation is still under development, timings are only provided to show the relative advantage of the proposed approach over a more traditional one, but further improvements are still possible with a fully parallel and optimized implementation.

Below, we show some results obtained on a few matrices arising from both elasticity and Poisson problems using different levels of refinement. The size and the number of non-zeroes of the matrices are shown in Table 1. We first define a baseline for both elasticity and flow problems. For the former we use the smoothed-BAMG prolongation (SMBAMG) and for the latter the extended+i interpolation (EXTI) that are the recommended methods in the Chronos package [12, 14]. We adopt a preconditioned conjugate gradient (PCG) as Krylov solver preconditioned with a single V(1,1)-cycle of AMG. The iterative procedure stops when the relative residual norm is reduced by 8 orders of magnitude. To compare different runs, the performances are measured in terms of iterations count (n_{it}), number of AMG levels (n_l) and grid (C_{gd}) and operator (C_{op}) complexities. These complexities consist of the sum overall levels of the number of rows/entries of the operators, respectively. Finally, timings are reported for the prolongation setup and PCG solution stages, named T_P and T_s, respectively.

First, we analyze the so called Cube tests, a linear elasticity benchmark consisting in a tetrahedral discretization of a cube with three levels of refinement (see Figure 1) whose results are shown in Table 2. EMIN(X) denote the use of X iterations of restricted CG to minimize prolongation energy. We see that just a few iterations are enough to be more effective than the standard SMBAMG: the number of iterations n_{it} are almost halved without increasing the operator complexity C_{op} and both prolongation setup and solution times, T_P and T_s respectively, are reduced. The convergence profiles for the preconditioned conjugate gradient using AMG with

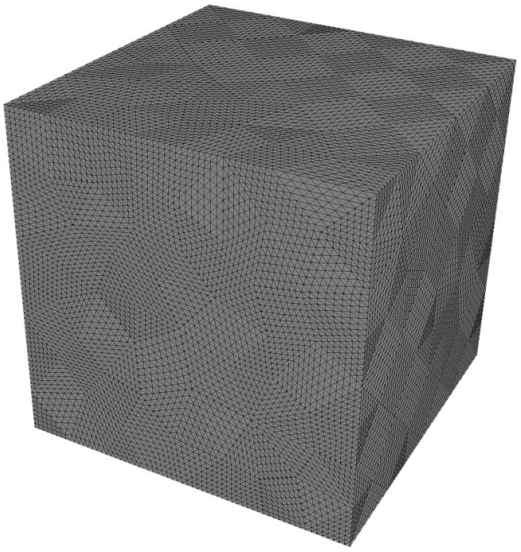

Figure 1: Mesh for the `Cube` tests. The three test cases are obtained through successive refinement levels.

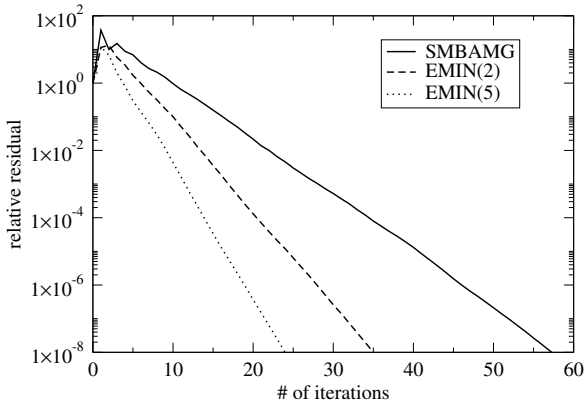

Figure 2: Converge profiles for the `Cube_739k` linear elasticity benchmark.

different prolongations for the case `Cube_739k` are shown in Figure 2. The effectiveness of the energy minimization process can be easily noticed. Moreover, we point out that the effectiveness of the energy minimization increases with the problem size as well as the speed-up of EMIN over SMBAMG. The EMIN method proved effective also on the industrial test case `Mech_447k` (see Figure 3) and the same considerations made above in terms of speed-up hold.

Now, we refer to the application of the EMIN method to Poisson problems, where the results seem more controversial. In two test cases, `finger4m` and `Pflow_742`, the EMIN method proves more effective than the standard EXTI. As seen before for the mechanical problems, the number of iterations needed to reach the solution is almost halved, but the cost for the prolongation setup is slightly larger than the standard EXTI. In any case, the benefit in the solution phase is such that the overall speed-up remains large. On the other hand, in the last test case `cavity` the EMIN method does not represent a valid alternative to EXTI: the setup cost increases but there is no benefit in the solution phase. We observe, however, that in this test case EXTI interpolation was already very effective requiring a very low number of iterations for convergence, so it is very difficult to improve its performance. The EMIN procedure is then recommended for tough test cases only.

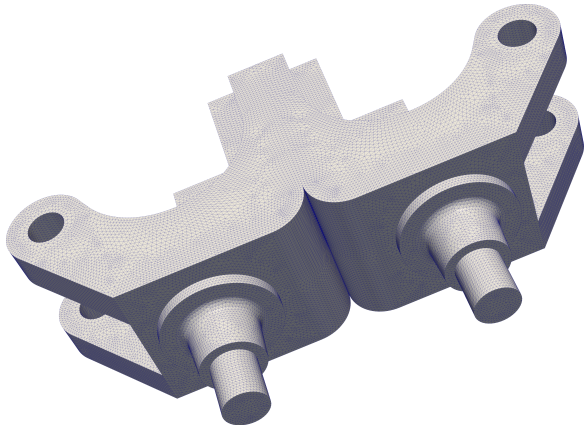

Figure 3: Mesh for the `Mech_447k` industrial study case. It consists in a poorly constrained steel hook discretized with tetrahedral finite elements.

Table 2: Comparison between different prolongation algorithms $P-type$ on the test cases reported in Table 1. For each run, the following information are provided: the number of PCG iteration n_{it} and the number of AMG levels n_l, the grid C_{gd} and operator C_{op} complexities, the prolongation setup time T_P and the iteration time T_s, in seconds.

Matrix	$P-type$	n_{it} (n_l)	C_{gd}/C_{op}	T_P	T_s
`Cube_105k`	SMBAMG	35 (3)	1.06/1.53	0.9	1.0
`Cube_105k`	EMIN(2)	23 (3)	1.06/1.50	0.7	0.6
`Cube_105k`	EMIN(5)	18 (3)	1.06/1.50	0.8	0.5
`Cube_739k`	SMBAMG	58 (4)	1.06/1.60	10.6	12.4
`Cube_739k`	EMIN(2)	36 (4)	1.06/1.51	5.1	7.3
`Cube_739k`	EMIN(5)	24 (4)	1.06/1.51	5.8	5.1
`Cube_5317k`	SMBAMG	157 (4)	1.04/1.28	68.7	229.4
`Cube_5317k`	EMIN(2)	82 (4)	1.04/1.28	30.6	111.5
`Cube_5317k`	EMIN(5)	45 (4)	1.04/1.28	38.4	71.0
`Mech_447k`	SMBAMG	42 (3)	1.04/1.27	4.2	15.5
`Mech_447k`	EMIN(5)	33 (3)	1.04/1.27	4.4	13.7
`finger4m`	EXTI	29 (5)	1.41/2.35	8.8	25.7
`finger4m`	EMIN(5)	14 (5)	1.40/3.08	15.2	13.0
`Pflow_742`	EXTI	505 (4)	1.22/4.01	33.9	216.9
`Pflow_742`	EMIN(5)	268 (4)	1.23/2.85	38.2	75.6
`cavity`	EXTI	12 (4)	1.33/3.91	4.2	2.2
`cavity`	EMIN(5)	13 (4)	1.32/6.99	7.6	2.8

6 CONCLUSIONS

In this work an energy minimization-based prolongation has been presented to improve AMG effectiveness. The technique is not completely new, as it has already been presented in [23], but our contribution stands in making it feasible in real applications through a detailed analysis, the use of appropriate preconditioners to accelerate the convergence of the constrained minimization problem and effective implementation. Two approaches have been discussed: a restricted Krylov subspace iterative method and the null space method coupled with a standard iterative algorithm. The two methods turn out to be equivalent, thus the same considerations on preconditioning effects hold for both.

The numerical results on large test cases show an improved convergence for the AMG based on a prolongation enhanced by energy minimization, especially in hard problems.

A distributed memory implementation of the proposed approach is under development and will be included in the Chronos library [12, 14] to be used in large-scale applications on modern supercomputers.

The next steps of our research will concern:

- the investigation of different options to build the initial prolongation pattern $\mathcal{P}$;
- the minimization of approximated energy expressions able to reduce setup time and memory footprint with acceptable differences in the overall convergence rate;
- the use of relaxed constraints to facilitate the energy minimization procedure without threatening the overall AMG effectiveness.

REFERENCES

[1] M. Benzi, G. H. Golub, and J. Liesen. 2005. Numerical solution of saddle point problems. *Acta Numer.* 14 (2005), 1–137. https://doi.org/10.1017/S0962492904000212

[2] A. Brandt. 1986. Algebraic Multigrid Theory: The Symmetric Case. *Appl. Math. Comput.* 19, 1-4 (jul 1986), 23–56. http://dx.doi.org/10.1016/0096-3003(86)90095-0

[3] A. Brandt, J. Brannick, K. Kahl, and I. Livshits. 2011. Bootstrap AMG. *SIAM J. Sci. Comput.* 33, 2 (2011), 612–632. https://doi.org/10.1137/090752973

[4] A. Brandt, J. Brannick, K. Kahl, and I. Livshits. 2015. Bootstrap Algebraic Multigrid: Status Report, Open Problems, and Outlook. *Numerical Mathematics: Theory, Methods and Applications* 8, 1 (2015), 112–135. https://doi.org/10.4208/nmtma.2015.w06si

[5] J. Brannick, F. Cao, K. Kahl, R. D. Falgout, and X. Hu. 2018. Optimal interpolation and compatible relaxation in classical algebraic multigrid. *SIAM Journal on Scientific Computing* 40, 3 (2018), A1473–A1493. https://doi.org/10.1137/17M1123456

[6] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge. 2001. Algebraic Multigrid Based on Element Interpolation (AMGe). *SIAM J. Sci. Comput.* 22, 5 (2001), 1570–1592. https://doi.org/10.1137/S1064827598344303

[7] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. 2005. Adaptive Smoothed Aggregation (αSA) Multigrid. *SIAM Rev.* 47, 2 (2005), 317–346. https://doi.org/10.1137/050626272

[8] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald. 2001. *Parallel programming in OpenMP.* Morgan kaufmann.

[9] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski. 2003. Spectral AMGe (ρAMGe). *SIAM J. Sci. Comput.* 25, 1 (2003), 1–26. https://doi.org/10.1137/S106482750139892X

[10] P. D'Ambra, S. Filippone, and P. S. Vassilevski. 2018. BootCMatch: A Software Package for Bootstrap AMG Based on Graph Weighted Matching. *ACM Trans. Math. Softw.* 44, 4 (jun 2018), 39:1–39:25. http://doi.acm.org/10.1145/3190647

[11] A. Franceschini, V. A. P. Magri, G. Mazzucco, N. Spiezia, and C. Janna. 2019. A robust adaptive algebraic multigrid linear solver for structural mechanics. *Computer Methods in Applied Mechanics and Engineering* 352 (2019), 389–416. https://doi.org/10.1016/j.cma.2019.04.034

[12] Matteo Frigo, Giovanni Isotton, and Carlo Janna. 2021. Chronos web page. https://www.m3eweb.it/chronos. https://www.m3eweb.it/chronos

[13] V. E. Henson and P. S. Vassilevski. 2001. Element-Free AMGe: General Algorithms for Computing Interpolation Weights in AMG. *SIAM J. Sci. Comput.* 23, 2 (2001), 629–650. https://doi.org/10.1137/S1064827500372997

[14] G. Isotton, M. Frigo, N. Spiezia, and C. Janna. 2021. Chronos: a general purpose classical AMG solver for high performance computing. *SIAM J. Sci. Comput.* 43, 5 (2021), C335–C357. https://doi.org/10.1137/21M1398586

[15] C. Janna, M. Ferronato, F. Sartoretto, and G. Gambolati. 2015. FSAIPACK: a software package for high-performance factored sparse approximate inverse preconditioning. *ACM Trans. Math. Software* 41, 2 (2015), Art. 10, 26. https://doi.org/10.1145/2629475

[16] S. Koric and A. Gupta. 2016. Sparse matrix factorization in the implicit finite element method on petascale architecture. *Computer Methods in Applied Mechanics and Engineering* 302 (2016), 281–292. https://doi.org/10.1016/j.cma.2016.01.011

[17] D. E. Longsine and S. F. McCormick. 1980. Simultaneous Rayleigh-quotient minimization methods for Ax= λBx. *Linear Algebra Appl.* 34 (1980), 195–234. https://doi.org/10.1016/0024-3795(80)90166-4

[18] V. A. P. Magri, A. Franceschini, and C. Janna. 2019. A novel algebraic multigrid approach based on adaptive smoothing and prolongation for ill-conditioned systems. *SIAM Journal on Scientific Computing* 41, 1 (2019), A190–A219. https://doi.org/10.1137/17M1161178

[19] J. Mandel, M. Brezina, and P. Vaněk. 1999. Energy optimization of algebraic multigrid bases. *Computing* 62, 3 (1999), 205–228. https://doi.org/10.1007/s006070050022

[20] T. A. Manteuffel, L. N. Olson, J. B. Schroder, and B. S. Southworth. 2017. A root-node-based algebraic multigrid method. *SIAM J. Sci. Comput.* 39, 5 (2017), S723–S756. https://doi.org/10.1137/16M1082706

[21] MATLAB. 2020. *Version 9.9.0 (R2020b).* The MathWorks Inc., Natick, Massachusetts.

[22] L. N. Olson, J. Schroder, and R. .S Tuminaro. 2010. A new perspective on strength measures in algebraic multigrid. *Numerical Linear Algebra with Applications* 17, 4 (2010), 713–733. https://doi.org/10.1002/nla.669

[23] L. N. Olson, J. B. Schroder, and R. S. Tuminaro. 2011. A general interpolation strategy for algebraic multigrid using energy minimization. *SIAM J. Sci. Comput.* 33, 2 (2011), 966–991. https://doi.org/10.1137/100803031

[24] M. Sala and R. S. Tuminaro. 2008. A new Petrov-Galerkin smoothed aggregation preconditioner for nonsymmetric linear systems. *SIAM J. Sci. Comput.* 31, 1 (2008), 143–166. https://doi.org/10.1137/060659545

[25] K. Stüben. 1983. Algebraic Multigrid (AMG): Experiences and Comparisons. *Appl. Math. Comput.* 13, 3-4 (jan 1983), 419–451. http://dx.doi.org/10.1016/0096-3003(83)90023-1

[26] P. Vaněk. 1992. Acceleration of Convergence of a Two-Level Algorithm by Smoothing Transfer Operator. *Appl. Math. (Prague)* 37 (1992), 265–274. https://eudml.org/doc/15715

[27] P. Vaněk, J. Mandel, and M. Brezina. 1996. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing* 56, 3 (Sep 1996), 179–196. https://doi.org/10.1007/BF02238511

[28] J. Xu and L. Zikatanov. 2017. Algebraic multigrid methods. *Acta Numerica* 26 (2017), 591–721. http://dx.doi.org/10.1017/S0962492917000083

[29] X. Xu and C.-S. Zhang. 2018. On the ideal interpolation operator in algebraic multigrid methods. *SIAM J. Numer. Anal.* 56, 3 (2018), 1693–1710. https://doi.org/10.1137/17M1162779