2023

An Aggregation-Based Algebraic Multigrid Method with Deflation Techniques and Modified Generic Factored Approximate Sparse Inverses

Anastasia Natsiou
Technological University Dublin, Ireland, anastasia.natsiou@tudublin.ie

George A. Gravvanis
Democritus University of Thrace, Kimmeria, 67100 Xanthi, Greece

Christos K. Filelis-Papadopoulos
University College Cork, T12 XF62 Cork, Ireland

See next page for additional authors

Funder: This research received no external funding

mathematics

MDPI

Article

An Aggregation-Based Algebraic Multigrid Method with Deflation Techniques and Modified Generic Factored Approximate Sparse Inverses

Anastasia A. Natsiou [1], George A. Gravvanis [2], Christos K. Filelis-Papadopoulos [3] and Konstantinos M. Giannoutakis [4,*]

[1] School of Computing, Technological University of Dublin, D07 H6K8 Dublin, Ireland
[2] Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, Kimmeria, 67100 Xanthi, Greece
[3] Department of Computer Science, University College Cork, T12 XF62 Cork, Ireland
[4] Department of Applied Informatics, School of Information Sciences, University of Macedonia, 54636 Thessaloniki, Greece
* Correspondence: kgiannou@uom.edu.gr

Abstract: In this paper, we examine deflation-based algebraic multigrid methods for solving large systems of linear equations. Aggregation of the unknown terms is applied for coarsening, while deflation techniques are proposed for improving the rate of convergence. More specifically, the V-cycle strategy is adopted, in which, at each iteration, the solution is computed by initially decomposing it utilizing two complementary subspaces. The approximate solution is formed by combining the solution obtained using multigrids and deflation. In order to improve performance and convergence behavior, the proposed scheme was coupled with the Modified Generic Factored Approximate Sparse Inverse preconditioner. Furthermore, a parallel version of the multigrid scheme is proposed for multicore parallel systems, improving the performance of the techniques. Finally, characteristic model problems are solved to demonstrate the applicability of the proposed schemes, while numerical results are given.

Keywords: multigrid; deflation; approximate inverses; aggregation-based algebraic multigrid; iterative methods; linear systems

MSC: 65F08; 65F10; 65F50; 65N55

Citation: Natsiou, A.A.; Gravvanis, G.A.; Filelis-Papadopoulos, C.K.; Giannoutakis, K.M. An Aggregation-Based Algebraic Multigrid Method with Deflation Techniques and Modified Generic Factored Approximate Sparse Inverses. *Mathematics* **2023**, *11*, 640. https://doi.org/10.3390/math11030640

Academic Editors: Vladislav Kovalnogov and Nadezhda Yarushkina

Received: 29 November 2022
Revised: 16 January 2023
Accepted: 24 January 2023
Published: 27 January 2023

1. Introduction

Lately, research efforts for efficiently solving large sparse linear systems have focused on multigrid methods [1,2]. Their applicability and efficiency are based on the use of a stationary method as a smoother for higher frequency components of the error, while the lower frequency components are transferred to a coarser level with higher frequency, which can reduce the error [3,4]. Multigrid algorithms consider a recursive application with two-grid coarse grid-correction schemes [5]. In the coarse level, a smoother is used for defecting correction, which is interpolated to the finer level and then added to the solution [3,4].

An algebraic multigrid (AMG) requires the information of the coefficient matrix of the linear system [4]. The only difference between the geometrical and algebraic approaches is the coarsening algorithm; however, they both consist of a smoother and a coarse grid correction. The smoother, is usually the application of some iterations of a fixed-point iterative method (such as Jacobi and Gauss–Seidel) and coarse grid correction is the transformation of the approximate residual to the finest grid. It should be noted that the notion of aggregation has multiple applications in several scientific fields, including gene regulatory networks [6,7].

Moreover, approximate inverses have been extensively used for approximating the inverse of the coefficient matrix of a linear system, resulting in efficient preconditioners for Krylov subspace iterative methods. Their efficiency relies on the fact that they have inherent parallelism and can be computed quickly in parallel [8–10]. Additionally, approximate inverse techniques have been efficiently used in conjunction with multigrid methods for solving partial differential equations discretized with finite difference methods [11], and for a variety of other applications [12,13].

In this paper, a class of Modified Generic Factored Approximate Sparse Inverses (MGenFAspI) based on approximate inverse sparsity patterns is used to solve two-dimensional and three-dimensional partial differential equations, discretized with high-order finite difference schemes. For the computation of these approximate inverses, a fill-in parameter is required which affects the density, in terms of nonzero elements, of the preconditioner [10]. A deflation technique is also utilized in this work, splitting the problem at each multigrid level into two complementary subspaces, aiming to further improve the convergence rate.

The paper is organized as follows: Section 2 introduces the deflation technique and the corresponding algorithm used for the solution of linear systems. Section 3 presents the aggregation-based algebraic multigrid method in conjunction with the deflation techniques and the Modified Generic Factored Approximate Sparse Inverses. The algorithmic procedures for the pairwise aggregation and the preconditioned multigrid are also given. The parallel aspects and techniques for accelerating the computations required by the Modified Generic Factored Approximate Sparse Inverses in hybrid parallel environments are discussed in Section 4. Finally, numerical results regarding the efficiency and convergence behavior of the proposed techniques are presented and discussed in Section 5, by solving characteristic model problems. Concluding remarks and directions for future research are provided in Section 6.

2. Deflation Techniques

Deflation techniques were introduced by Nicolaides in 1987 for improving the rate of convergence of the conjugate gradient method [14]. The main idea is based on the projection of the linear system into two complementary subspaces, in order to separately solve the two produced linear systems. The efficiency of this technique is based on the fact that the projected linear systems could be solved faster than the initial system, and then their solution can be combined for formulating the solution of the initial problem [15].

Let us consider an $n \times n$ linear system of the form:

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

where A is a symmetric and positive definite matrix, b is the right-hand side vector and x is the solution vector. The deflation subspace $V \in \mathbb{R}$ can be represented by a full-rank matrix $V \in \mathbb{R}^{n \times k}$ with $k \ll n$ such that:

$$E = V^T AV. \tag{2}$$

Then, the projection matrix Q can be defined as follows:

$$Q = I - VE^{-1}V^T A, \tag{3}$$

where I is the identity matrix. The matrix Q is the orthogonal projection on the subspace $V^{\perp}$, while $I - Q$ is the orthogonal projection on the subspace V. For the selection of the subspace V, we can use the deflation technique for the generation of the basis of V, proposed in [15], where the domain Ω is decomposed into m nonoverlapping subdomains Ω_j, $j = 1, \ldots, m$, where:

$$v_{i,j} = \begin{cases} 1, & \text{if } x_i \in \Omega_j \\ 0, & \text{if } x_i \notin \Omega_j \end{cases}. \tag{4}$$

The solution of the linear system can be computed as follows

$$x = (I - Q)x_c + Qx_s, \tag{5}$$

where x_c is the solution vector of the projection to subspace V, while x_s is the solution vector from the projection to the subspace $V^{\perp}$. These vectors can be computed as follows:

$$x_c = (I - Q)x \text{ and } x_s = Qx_{it}, \tag{6}$$

where x is an initial approximation for the solution. By applying matrix operations, the solution vector x_c can be defined as:

$$x_c = (I - Q)x = VE^{-1}V^TAx = V(V^TAV)^{-1}V^TAx = V(V^TAV)^{-1}V^Tb. \tag{7}$$

Then, the vector x_{it} can be computed by solving iteratively the following linear system:

$$A(I - V(V^TAV)^{-1}V^TA)x_{it} = b - Ax_c. \tag{8}$$

The procedure for the solution of the linear system with the deflation technique can be summarized by Algorithm 1.

Algorithm 1 Deflation technique for solving linear systems

1: Select the basis of the subspace V.
2: Project vector V based on the subspace V.
3: Compute $E^{-1} = (V^TAV)^{-1}$.
4: Compute the solution from the subspace V, using the transformation matrix $I - Q$, such that $x_c = V(V^TAV)^{-1}V^Tb$.
5: Solve iteratively the linear system $A(V(V^TAV)^{-1}V^T)x_{it} = b - Ax_c$.
6: Multiply the solution x_{it} with Q and define the solution x_s.
7: Compute the approximate solution by adding the two solution vectors from the two complementary subspaces.

The use of the deflation technique in conjunction with an efficient preconditioning scheme can accelerate the convergence behavior of the solution of the system [14]. Preconditioning techniques yield the solution of a linear system of the form:

$$MAx = Mb, \tag{9}$$

where M is a symmetric positive definite preconditioner. An efficient preconditioner should be efficiently computed in parallel, MA should have a "clustered" spectrum, and the $M \times$ vector should be fast to compute in parallel [9,10,16]. Then, the preconditioner can be used in conjunction with a Krylov-subspace iterative method, such as the conjugate gradient or the generalized minimum residual methods for the solution of the linear system [9,10].

3. Aggregation Multigrid Method

Multigrid methods have been widely used by the scientific community for solving linear systems, due to their convergence behavior and efficiency [8,17]. The design of these methods is based on the fact that high-frequency error components can be damped by a stationary iterative method, while low-frequency error components are handled by coarser grids with a higher discretization step, where the high-frequency modes of the error are more oscillatory and can be damped efficiently by a stationary iterative method [4,18]. Multigrid methods are composed of four components: (a) the stationary iterative method (iterative methods such as the Richardson, Jacobi and Gauss–Seidel methods), (b) the restriction operator (transfer operators from finer to coarser grids), (c) the prolongation operator (transfer operators from coarser to finer grids) and (d) the cycle strategy (the

sequence in which the grids are visited until a solution with the predefined tolerance is achieved) [16].

The method utilized for grid coarsening in this paper is the aggregation multigrid, first introduced in [19] and developed by Notay [5]. The nodes of the grid are combined into pairs, creating aggregates, based on a weighting rule.

The coarsening algorithm constructs a prolongation matrix P from the coefficient matrix A, which is the transformation matrix from the coarse to the fine grid. The dimensions of the prolongation matrix are $n \times n_c$, where $n_c < n$ is the number of the nodes of the coarser grid. The transformation from the fine to the coarser grid can be managed by the restriction matrix R, which is computed as follows:

$$R = P^T, \tag{10}$$

and the coarse coefficient matrix A_c is estimated by Galerkin's formula:

$$A_c = RAP. \tag{11}$$

For the formulation of the prolongation matrix, the aggregates G_i, which are disjoint subsets of the variable set, have to be defined. The number of these variables, n_c, is, then, the number of the subsets and is used to define the prolongation matrix P:

$$P_{i,j} = \begin{cases} 1, \text{ if } i \in G_j \\ 0, \text{ otherwise} \end{cases} \quad (1 \le i \le n, 1 \le j \le n_c). \tag{12}$$

Therefore, P is a Boolean matrix with one nonzero entry in each row. In order to define the aggregates, the algorithm searches for the set of nodes S_i to which i is strongly negatively coupled, using a strong/weak coupling threshold β:

$$S_i = \left\{ j \neq i | a_{i,j} < -\beta \max_{a_{i,k}<0} |a_{i,k}| \right\}. \tag{13}$$

Then, one unmarked node at a time is selected, giving priority to the node with the minimal m_i, where m_i is the number of unmarked nodes which are strongly negatively coupled to I [5,20]. Next, one can pick an unmarked node j, which consists of the column of the matrix A with the minimal element $a_{i,j}$ in line i. Then, an aggregate is created between the node i and the node j. The algorithm of the pairwise aggregation is given in Algorithm 2 [5]. It should be noted that the *CheckDD* parameter is optional and confirms if matrix A is diagonal-dominant [5].

For the solution of a linear system with an algebraic aggregation multigrid, the V-cycle strategy can be used. The linear system is smoothed and the residual is transferred to the coarser level until the resulting linear system is small enough to be solved directly. Then, the solution vector is subject to the procedure of coarse grid correction, by means of the appropriate prolongation [5,20].

During each iteration of the V-cycle, the deflation technique is used to split the problem into two complementary subspaces, and the approximate solution is computed, which is transferred to the coarser level of the grid. The combination of the multigrid and deflation techniques is depicted in Figure 1.

Algorithm 2 Pairwise aggregation

Inputs: matrix $A = (a_{i,j})$ with n rows,
Threshold $\beta = 0.25$,
Logical parameter *CheckDD*
Outputs: number of coarse variables n_c,
Aggregates G_i, $i = 1, \ldots n_c$ (such as $G_i \cap G_j = \emptyset$)

if *CheckDD* **then**
  $U = [1, n] \backslash \{i | a_{i,i} > 5 \sum_{j \neq i} |a_{i,j}|\}$
else $U = [1, n]$
end if
for $i = 1, \ldots, n$ **do**
  $S_i = \left\{ j \in U \backslash \{i\} | a_{i,j} < -\beta \max_{a_{i,k}<0} |a_{i,k}| \right\}$
end for
for $i = 1, \ldots, n$ **do**
  $m = |\{j | i \in S_j\}|$
end for
$n_c = 0$
while $U \neq \emptyset$ **do**
  Select $i \in U$ with minimum m_i, $n_c = n_c + 1$
  Select $j \in U$ such that $a_{i,j} = \min_{k \in U} a_{i,k}$
  if $j \in U$ **then**
    $G_{n_c} = \{i, j\}$
  else $G_{n_c} = \{i\}$
  end if
  $U = U \backslash G_{n_c}$
  For all $k \in G_{n_c}$ update $m_l = m_l - 1$ for $l \in S_k$
end while

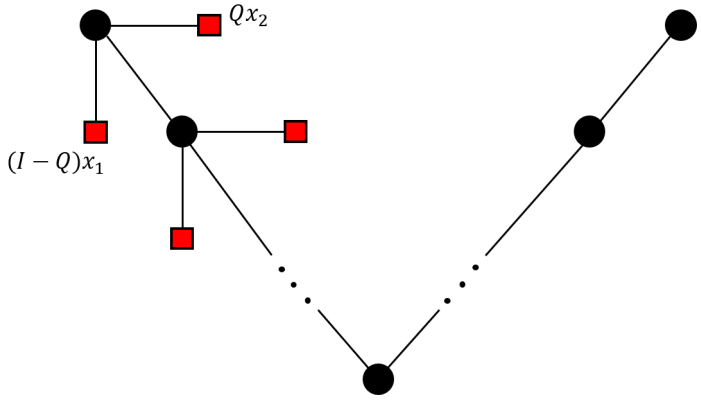

Figure 1. A V-cycle of aggregation multigrid in conjunction with deflation.

Algorithm 3 depicts the procedure of using a multigrid as preconditioner for a V-cycle.

Algorithm 3 Preconditioned V cycle

Inputs: Residual vector r_k
Grid matrix at level k: A_k
Deflation matrix M_k
Prolongation matrix P_k
Matrix of next grid A_{k-1}
Outputs: z_k

function $MGprec(r_k,k)$
  Compute $z_k^{(1)} = M_k^{-1} r_k$
  Calculate the new residual vector $r_k = r_k - A_k z_k^{(1)}$
  Compute $r_{k-1} = P_k^T r_k$
  if k=1 **then**
    $x_{k-1} = A_{k-1}^{-1} r_{k-1}$
  else
    $x_{k-1} = MGprec(r_{k-1}, k-1)$
  end if
  $z_k^{(2)} = P_k x_{k-1}$
  $r_k = r_k - A_k z_k^{(2)}$
  $z_k^{(3)} = M_k^{-1} r_k$
  $z_k = z_k^{(1)} + z_k^{(2)} + z_k^{(3)}$
end function

The function $MGprec$ is executed once at the higher level for $k = l$ using the matrix $A_l = A$, and iteratively calls itself until the parameter k reduces to 1. For smoothing, the symmetric Gauss–Seidel method is used, where

$$M_k = low(A_k) diag(A_k)^{-1} upp(A_k), \tag{14}$$

where low, $diag$ and upp are the lower, diagonal and upper submatrices of A_k [5].

In this paper, the coarse coefficient matrix A_c is inverted by the Modified Generic Factored Approximate Sparse Inverse (MGenFAspI) technique [10]. For a coefficient matrix A, we consider its incomplete LU factorization, i.e.,

$$A = LU + E, \tag{15}$$

where L and U are, respectively, the upper and lower triangular factors and E is the error matrix. The GenFAspI matrix can be computed using the following procedure:

$$A = LU \Leftrightarrow A^{-1} = U^{-1} L^{-1} \Leftrightarrow A^{-1} = M = GH, \tag{16}$$

considering $G = U^{-1}$ and $H = L^{-1}$. In order to compute the G and H, an "a-priori" sparsity pattern is required, [21]. These patterns occur when utilizing a predetermined drop tolerance (droptol) and then raising to a predefined power (level of fill or lfill) the upper and lower triangular factors. After that, the GenFAspI matrix is constructed by solving the following system:

$$M = G_{\text{droptol}}^{\text{lfill}} H_{\text{droptol}}^{\text{lfill}} \Leftrightarrow \begin{cases} U g_{:,j} = e_{:,j} \\ L h_{:,j} = e_{:,j} \end{cases}, 0 \le j < m, \tag{17}$$

where $g_{:,j}$ denote the elements of G, $h_{:,j}$, the elements of H; $e_{:,j}$ are the elements of the identity matrix and m the order of the coefficient matrix of the linear system [22]. A modified version of GenFAspI (MGenFAspI) is used in order to enhance the performance during the computation of the approximate inverse [10]. The MGenFAspI matrix is used to compute each column of the factor of the approximate inverse, separately, by a restricted solution

process of only the factors. The elements that do not belong to G and H factors are set to zero; thus, Equation (17) is rewritten as:

$$\begin{cases} Ug_{:,j} = e_{:,j} \\ Lh_{:,j} = e_{:,j} \end{cases} 0 \leq j < m \Leftrightarrow \begin{cases} g_{i,j} = 0, (i,j) \notin G^{\text{lfill}}_{\text{droptol}} \\ h_{i,j} = 0, (i,j) \notin H^{\text{lfill}}_{\text{droptol}} \end{cases}. \tag{18}$$

The advantage of the modified version of GenFAspI is that the construction of the approximate inverse can be easily parallelized, since the elements of the matrices G and H or L and U are not required to be identified. The algorithmic procedure of MGenFAspI is given in [10].

4. Parallel Techniques

In this section, the parallel algorithm for the Modified Generic Factored Approximate Sparse Inverse and the parallel V-cycle are discussed. The matrices G and H can be computed by solving the following equations:

$$\begin{cases} UG = I \\ LH = I \end{cases}, \tag{19}$$

where U and L are the upper and lower triangular matrices that are derived from the incomplete LU factorization of the coefficient matrix A. Thus, for calculating the columns of the matrices G and H, the linear system of Equation (18) should be solved.

The MGenFAspI algorithm allows the individual computation of the columns of matrices G and H, and, due to this inherent parallelism, it can be efficiently implemented on hybrid parallel systems that consist of distributed and multicore CPUs. Each node of the distributed system can be assigned into a block of columns of the matrices, while, within each node, the multiple cores can further distribute the workload.

The computation of the elements of each column of matrices G and H does not require all rows of matrices U or L, respectively, but only the rows that correspond to nonzero elements of each column. For this reason, there is no need to broadcast the whole matrices U and L, yielding a reduced communication cost, especially for large-scale matrices. A visual representation of the distribution of matrix G on the available nodes is given in Figure 2, while the workload of two cores of a distributed node is depicted in Figure 3.

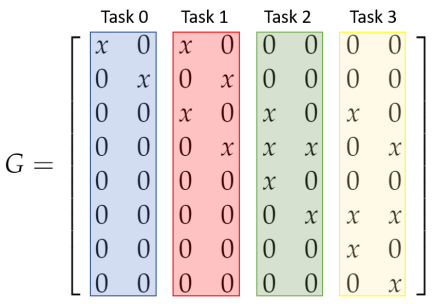

Figure 2. Distribution of columns of matrix G on four tasks.

The same process was implemented for the computation of the elements of matrix H, where the rows of matrix L were used. Since this technique does not require any communication between the distributed processes, the parallel speedup and parallel efficiency are expected to be high. The derived matrices (G and H) are distributed on the nodes; however, they can be used on any distributed system for executing a column-wise matrix$\times$vector multiplication. The algorithm for the parallelization of the MGenFAspI has been presented in [10]. In the context of this work, the parallelization on multicore systems was considered.

$$
\begin{array}{c} \\ \text{R2} \\ \text{R3} \\ \text{R4} \end{array}
\begin{bmatrix} & & & & & & & \\ 0 & 0 & x & 0 & x & 0 & x & 0 \\ 0 & 0 & 0 & x & x & x & 0 & x \\ 0 & 0 & 0 & 0 & x & 0 & 0 & 0 \\ & & & & & & & \end{bmatrix}
\overset{g_{:,4}}{\begin{bmatrix} 0 \\ 0 \\ x \\ x \\ x \\ 0 \\ 0 \\ 0 \end{bmatrix}} =
\overset{e_{:,4}}{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}}
$$

$$
\begin{array}{c} \\ \text{R3} \\ \\ \text{R5} \end{array}
\begin{bmatrix} & & & & & & & \\ 0 & 0 & 0 & 0 & x & x & x & x \\ & & & & & & & \\ 0 & 0 & 0 & 0 & x & x & x & x \\ & & & & & & & \end{bmatrix}
\overset{g_{:,5}}{\begin{bmatrix} 0 \\ 0 \\ 0 \\ x \\ 0 \\ x \\ 0 \\ 0 \end{bmatrix}} =
\overset{e_{:,5}}{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}}
$$

Figure 3. Computations performed from the two cores of Task 2 for the formulation of fourth and fifth columns of matrix G of the MGenFAspI.

5. Numerical Results

The applicability, performance and convergence behavior of the proposed schemes are given in this section. Various model problems are considered and numerical results are obtained, experimentally demonstrating the behavior of the proposed methods. It should be noted that the experimental results were obtained on an Intel Xeon 2420v2 with 6 cores and 12 threads, 48GB of RAM memory, running Linux CentOS 7. All algorithmic techniques were implemented in C++ 11.

For the evaluation of the aggregation-based multigrid method with V-cycle, the 2D Poisson problem was initially selected with the number of elements per direction $nx = 50, 100, 200, 500$. Four iterations of the Jacobi method were performed as a pre-smoother and four iterations of the Jacobi method were used for post-smoothing in each level of grid hierarchy. In the last level of the multigrid, the linear system is solved by a direct method, such as LU factorization. The termination criterion is $||r_i||_2 < tol||r_0||_2$, where $||r_i||_2$ is the norm of the residual vector of the i-th iteration, $||r_0||_2$ is the norm of the residual of the initial problem and $tol = 10^{-8}$. In the parallel experiments, the resolution at the lowest level was held by the PARDISO method (parallel direct solver) contained in the parallel libraries of Intel [23].

It should be noted that the choice of pre-smoothing and post-smoothing iterations affects the convergence behavior, e.g., increased values lead to better overall performance. However, since the stationary iterative methods in the context of multigrids are used to damp the high-frequency components of the error, an arbitrary increase does not guarantee substantial improvements. In the context of an algebraic multigrid, the choices for pre-smoothing and post-smoothing iterations vary between 1 and 4 for smoothers, such as Jacobi or Gauss–Seidel, in the literature. Thus, determination of the optimal parameters in the context of a multigrid depends on the problem to be solved and can be determined by experimentation [5,18].

The convergence behavior and the performance of the AGAMG method for different numbers of the unknowns per dimension for the 2D Poisson problem are given in Table 1.

Table 1. Convergence behavior and performance of V cycle of AGAMG for 2D Poisson model problem.

Level		***nx***				
		50	**100**	**200**	**300**	**500**
2	Execution time	0.0967	0.8053	8.7529	40.4147	275.5410
	Iterations	25	27	27	28	28
3	Execution time	0.1118	0.5446	4.2991	15.6946	90.7970
	Iterations	41	45	52	55	57
4	Execution time	0.125	0.6148	3.7552	11.7529	59.7511
	Iterations	49	59	63	64	65
5	Execution time	0.1483	0.7674	3.9582	10.4572	39.3368
	Iterations	61	85	98	105	115

The number of iterations required for convergence slightly increases as the size of matrix A increases, for the different number of levels. Moreover, increasing the number of levels while increasing the number of points per dimension (nx) yields reduced execution time. This is due to the size of the matrix at the last level, which is solved by a direct method. Thus, for large-scale systems, an increased number of levels is recommended.

In order to examine the applicability and efficiency of the deflation techniques, the same model problem was executed by considering deflation with V-cycle. For the integration of the two methods, the four Jacobi pre-smoothing iterations were replaced with one deflation iteration. The deflation algorithm computes, using a direct method, the solution in subspace $I - Q$:

$$x_c = (I - Q)x = PE^{-1}RAx = P(RAP)^{-1}RAx, \tag{20}$$

by inverting the matrix E with the Jacobi method (i.e., inverting only the main diagonal). The solution in subspace Q is computed with six iterations of the BiCGStab method for the following linear system:

$$A(PE^{-1}R)x_{it} = b - Ax_c. \tag{21}$$

The efficiency and convergence behavior of the aggregation-based algebraic multigrid method with deflation are given in Table 2, for various numbers of levels and nx. It should be noted that the Jacobi method with $n_2 = 4$ was used as post-smoother.

Table 2. Convergence behavior and performance of V cycle of AGAMG in combination with deflation technique for 2D Poisson model problem.

Level		***nx***				
		50	**100**	**200**	**300**	**500**
2	Execution time	1.6971	1.2749	11.1617	42.9594	296.577
	Iterations	20	26	29	27	29
3	Execution time	0.3221	1.5183	7.8454	22.2745	117.9680
	Iterations	42	45	47	46	56
4	Execution time	0.3351	2.0727	9.1327	24.3004	100.505
	Iterations	40	60	59	61	66
5	Execution time	0.3964	2.9885	16.976	41.8845	170.785
	Iterations	46	88	124	130	181

To further improve the convergence behavior, the Modified Generic Factored Approximate Sparse Inverse (MGenFAspI) was used for approximating matrix E^{-1}. The $lfill$ parameter was set to 1 and the $droptol$ to zero. The corresponding results are given in Table 3, for various values of nx and numbers of levels for the 2D Poisson problem. It can be seen that in most of the cases, the number of iterations is reduced.

Table 3. Convergence behavior and performance of V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 1$ for 2D Poisson model problem.

Level		***nx***				
		50	**100**	**200**	**300**	**500**
2	Execution time	0.6232	4.2415	24.0091	72.6027	374.277
	Iterations	17	25	27	27	28
3	Execution time	1.3349	16.3954	39.3299	93.8537	378.846
	Iterations	32	96	54	54	67
4	Execution time	2.9044	33.2589	52.0096	379.832	561.296
	Iterations	65	185	70	218	105

In Table 4, the convergence behavior and the performance of the proposed scheme with the use of MGenFAspI preconditioner for various values of nx and number of levels for solving the 3D Poisson problem are given. Moreover, in Tables 5 and 6, numerical results are given for the 2D and 3D Poisson problem with $lfill = 2$ and $droptol = 0.0$ for various numbers of levels.

Table 4. Convergence behavior and performance of V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 1$ for 3D Poisson model problem.

Level		***nx***				
		50	**100**	**200**	**300**	**500**
2	Execution time	9.1539	47.4512	573.946	2146.98	5649.41
	Iterations	8	10	16	20	18
3	Execution time	9.9429	48.6524	571.837	1261.92	3664.08
	Iterations	8	10	26	21	23
4	Execution time	10.4677	54.8827	327.732	535.449	2954.17
	Iterations	8	11	28	23	57

Table 5. Convergence behavior and performance of V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 2$ for 2D Poisson model problem.

Level		***nx***				
		50	**100**	**200**	**300**	**500**
2	Execution time	1.3201	6.9443	36.0793	97.401	441.648
	Iterations	21	26	28	27	28
3	Execution time	1.9226	11.4484	50.453	135.453	454.7931
	Iterations	27	40	42	48	52
4	Execution time	2.093	13.0483	73.6803	191.0220	555.9120
	Iterations	28	43	59	65	64
5	Execution time	1.9354	17.4081	100.6990	282.1470	860.0930
	Iterations	25	56	79	96	103

Table 6. Convergence behavior and performance of V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 2$ for 3D Poisson model problem.

Level		nx				
		50	**100**	**200**	**300**	**500**
2	Execution time	8.9012	48.5778	709.2940	1974.3400	5529.2900
	Iterations	8	10	15	16	16
3	Execution time	9.986	49.2396	519.0140	1704.4200	4300.770
	Iterations	8	10	15	20	22
4	Execution time	10.2313	54.0935	427.9530	1100.960	2115.110
	Iterations	8	11	17	23	25
5	Execution time	13.5096	51.6422	605.7950	1135.4700	1956.0400
	Iterations	10	10	24	25	27

The convergence behavior and the performance of the proposed schemes with the use of MGenFAspI for various numbers of the $lfill$ parameter and number of levels for the 2D Poisson problem with $nx = 300$ is given in Table 7.

Table 7. Convergence behavior and performance of V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and different values of $lfill$ for 2D Poisson model problem with $nx = 300$.

Level		$lfill$			
		1	**2**	**3**	**4**
2	Execution time	72.6027	98.6010	132.5990	174.9720
	Iterations	27	27	27	27
3	Execution time	93.8537	138.0190	208.5200	310.3720
	Iterations	54	48	48	50
4	Execution time	379.8320	191.1260	318.5270	398.9170
	Iterations	218	65	71	61
5	Execution time	-	283.9880	407.6990	730.2840
	Iterations	-	96	89	111

The performance and applicability were also tested against the Sherman problem (https://sparse.tamu.edu/HB/sherman1 (accessed on 26 January 2023)). In Table 8, the convergence behavior and performance of the proposed schemes with the use of the MGenFAspI preconditioner, for various values of the parameter $lfill$ and number of levels for the Sherman1 problem, are given in Table 8. In Table 9, the parallel aggregation-based algebraic multigrid method in conjunction with deflation and Modified Generic Factored Approximate Sparse Inverses for the 3D Poisson problem, with $nx = 70$ and $lfill = 2$, is given.

The addition of deflation is expected to increase the computational time, since it involves matrix inversion operations (approximate) and matrix-by-matrix products. However, for difficult problems, e.g., Sherman1, where iterations increase substantially when following the original AGMG approach, deflation leads to substantial improvements and even ensures convergence within the prescribed maximum iterations. Moreover, the additional computational work is composed of inherently parallel operations which can be accelerated on modern hardware, leading to an efficient solution approach.

Table 8. Convergence behavior and performance of V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and different values of $lfill$ for Sherman1 problem.

Level		$lfill$			
		1	**2**	**3**	**4**
2	Execution time	0.8757	0.9876	1.0288	1.0522
	Iterations	369	367	369	357
3	Execution time	1.1042	1.2073	1.3506	1.1743
	Iterations	365	368	366	341
4	Execution time	1.3235	1.5063	1.6007	1.6300
	Iterations	369	371	369	354
5	Execution time	1.5504	1.6823	1.8967	1.8641
	Iterations	375	360	367	339

Table 9. Convergence behavior and performance of a parallel V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 2$ for 3D Poisson model problem with $nx = 70$.

Level		$lfill$			
		1	**2**	**3**	**4**
2	Execution time	2867.37	1804.98	1125.15	820.11
	Iterations	54	37	43	40
3	Execution time	2129.46	1410.57	1087.21	866.76
	Iterations	34	41	35	33
4	Execution time	3542.25	2145.36	1621.55	1299.05
	Iterations	53	37	41	36
5	Execution time	3083.26	2145.08	1506.31	1201.97
	Iterations	45	46	30	34

Direct parallelization of algebraic multigrid algorithms has been studied extensively in the literature and has been shown to be limited [24]. The advantages of a truncated approach, which is similar to the one adopted in this manuscript, for the parallelization of the V cycle have been discussed extensively in [25].

Finally, the speedups and efficiency of the parallel aggregation-based algebraic multigrid method in conjunction with deflation and Modified Generic Factored Approximate Sparse Inverses for the 3D Poisson problem, with $nx = 70$ and $lfill = 2$, are given in Table 10 and Table 11, respectively. The corresponding figures for the execution times, speedups and efficiency are given in Figure 4, Figure 5 and Figure 6, respectively.

Table 10. Speedup of parallel V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 2$ for 3D Poisson model problem with $nx = 70$.

Level	**Threads**		
	2	**4**	**6**
2	1.59	2.55	3.77
3	1.51	1.96	2.46
4	1.65	2.18	2.73
5	1.44	2.05	2.57

Table 11. Efficiency of parallel V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 2$ for 3D Poisson model problem with $nx = 70$.

Level	Threads		
	2	**4**	**6**
2	0.79	0.64	0.58
3	0.75	0.49	0.41
4	0.83	0.558	0.45
5	0.72	0.51	0.43

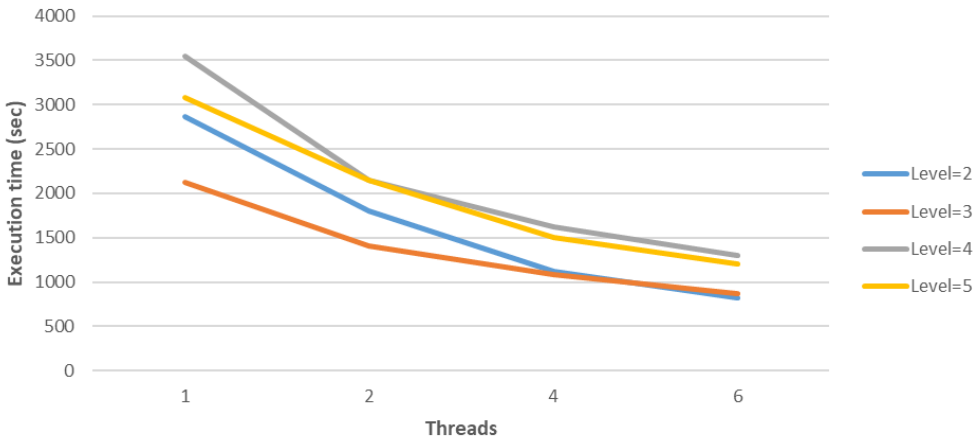

Figure 4. Execution times (in seconds) of parallel V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 2$ for 3D Poisson model problem with $nx = 70$.

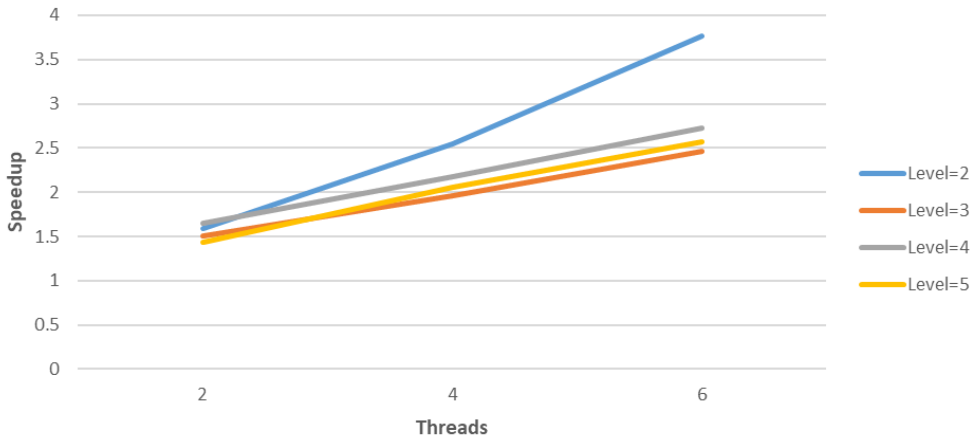

Figure 5. Speedups of parallel V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 2$ for 3D Poisson model problem with $nx = 70$.

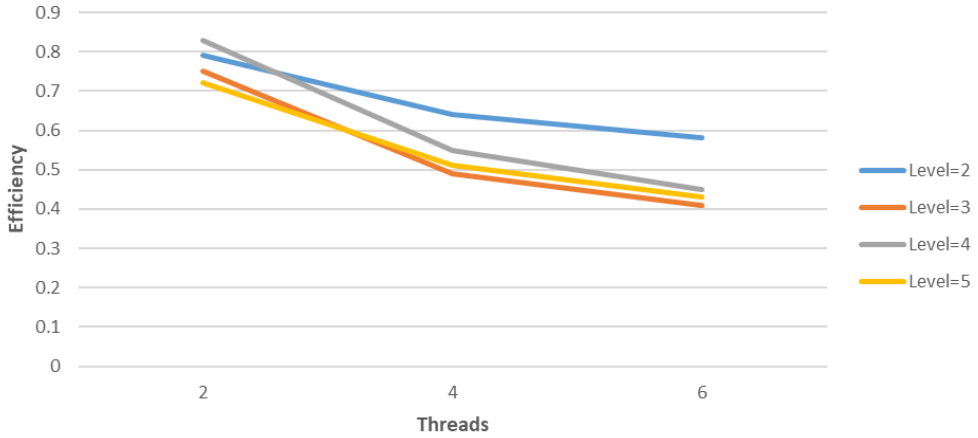

Figure 6. Efficiency of parallel V cycle of AGAMG in combination with deflation technique and MGenFAspI with $droptol = 0$ and $lfill = 2$ for 3D Poisson model problem with $nx = 70$.

6. Discussion and Conclusions

The Modified Generic Factored Approximate Sparse Inverse technique based on incomplete factorization has been recently proposed and used as a preconditioner for Krylov subspace methods. In this work, algebraic multigrid methods were examined that are based on aggregation, while a V cycle was used in conjunction with deflation techniques to improve the convergence behavior. The proposed schemes were discussed and numerical results from solving two model problems were presented.

The evaluation of the proposed methods demonstrates the improvement in the rate of convergence of the multigrid techniques using deflation, for various values of the parameters nx and $lfill$. The extensive experimentation that was carried out validated that the V cycle of AGAMG in combination with deflation and MGenFAspI was the most efficient approach in terms of convergence behavior; however, with greater execution time. For this, the paper took advantage of the inherent parallelism of the approximate inverse scheme and the aggregation-based multigrid method, and resulted in reduced execution times on a shared memory parallel system. The parallel numerical results obtained demonstrated good parallel results with good potential for large-scale parallel systems.

Future work includes the use of the aggregation-based multigrid method in conjunction with deflation techniques and approximate inverses as preconditioners for Krylov-subspace iterative methods. Moreover, the use of domain decomposition methods will be examined, in order to further improve the parallel efficiency of the methods.

Author Contributions: Conceptualization, G.A.G. and C.K.F.-P.; Investigation, A.A.N., G.A.G., C.K.F.-P. and K.M.G.; Methodology, A.A.N., G.A.G., C.K.F.-P. and K.M.G.; Software, A.A.N. and C.K.F.-P.; Supervision, G.A.G.; Validation, A.A.N., C.K.F.-P. and K.M.G.; Visualization, C.K.F.-P. and K.M.G.; Writing—original draft, A.A.N., G.A.G., C.K.F.-P. and K.M.G.; Writing—review and editing, A.A.N., G.A.G., C.K.F.-P. and K.M.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AMG	Algebraic multigrid
AGAMG	Aggregation-based algebraic multigrid
CPU	Central processing unit

GenFAspI	Generic Factored Approximate Sparse Inverse
MGenFAspI	Modified Generic Factored Approximate Sparse Inverse

References

1. Paludetto Magri, V.A.; Franceschini, A.; Janna, C. A Novel Algebraic Multigrid Approach Based on Adaptive Smoothing and Prolongation for Ill-Conditioned Systems. *SIAM J. Sci. Comput.* **2019**, *41*, A190–A219. [CrossRef]
2. Demidov, D. AMGCL: An efficient, flexible, and extensible algebraic multigrid implementation. *Lobachevskii J. Math.* **2019**, *40*, 535–546. [CrossRef]
3. Briggs, W.L.; Henson, V.E.; McCormick, S.F. *A Multigrid Tutorial*; SIAM: Philadelphia, PA, USA, 2000.
4. Wesseling, P. Theoretical and Practical Aspects of a Multigrid Method. *SIAM J. Sci. Stat. Comput.* **1982**, *3*, 387–407.
5. Notay, Y. An aggregation-based algebraic multigrid method. *Electron. Trans. Numer. Anal.* **2008**, *37*, 123–146.
6. Li, H.; Liu, Y.; Wang, S.; Niu, B. State Feedback Stabilization of Large-Scale Logical Control Networks via Network Aggregation. *IEEE Trans. Autom. Control.* **2021**, *66*, 6033–6040. [CrossRef]
7. Wang, S.; Li, H. Aggregation method to reachability and optimal control of large-size Boolean control networks. *Sci. China Inf. Sci.* **2022**, *66*, 1869–1919. [CrossRef]
8. Bröker, O.; Grote, M.J.; Mayer, C.; Reusken, A. Robust Parallel Smoothing for Multigrid Via Sparse Approximate Inverses. *SIAM J. Sci. Comput.* **2001**, *23*, 1396–1417.
9. Giannoutakis, K.M.; Gravvanis, G.A. High performance finite element approximate inverse preconditioning. *Appl. Math. Comput.* **2008**, *201*, 293–304. [CrossRef]
10. Filelis-Papadopoulos, C.; Gravvanis, G. A class of generic factored and multi-level recursive approximate inverse techniques for solving general sparse systems. *Eng. Comput.* **2016**, *33*, 74–99. [CrossRef]
11. Filelis-Papadopoulos, C.K.; Gravvanis, G.A. On the Multigrid Method Based on Finite Difference Approximate Inverses. *Comput. Model. Eng. Sci.* **2013**, *90*, 233–253.
12. Gravvanis, G.; Filelis-Papadopoulos, C.; Matskanidis, P. Algebraic Multigrid Methods Based on Generic Approximate Banded Inverse Matrix Techniques. *Comput. Model. Eng. Sci.* **2014**, *100*, 323–345.
13. Kyziropoulos, P.; Filelis-Papadopoulos, C.; Gravvanis, G. Parallel N-body simulation based on the PM and P3M methods using multigrid schemes in conjunction with generic approximate sparse inverses. *Math. Probl. Eng.* **2015**, *2015*, 450980. [CrossRef]
14. Nicolaides, R.A. Deflation of Conjugate Gradients with Applications to Boundary Value Problems. *SIAM J. Numer. Anal.* **1987**, *24*, 355–365.
15. Frank, J.; Vuik, C. On the Construction of Deflation-Based Preconditioners. *SIAM J. Sci. Comput.* **2001**, *23*, 442–462.
16. Gravvanis, G.; Filelis-Papadopoulos, C. On the multigrid cycle strategy with approximate inverse smoothing. *Eng. Comput. Int. J. Comput.-Aided Eng.* **2014**, *31*, 110–122. [CrossRef]
17. Haelterman, R.; Vierendeels, J.; Van Heule, D. Non-stationary two-stage relaxation based on the principle of aggregation multi-grid. In *Proceedings of the Computational Fluid Dynamics 2006*; Deconinck, H., Dick, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 243–248.
18. Hackbusch, W. *Multi-Grid Methods and Applications*; Springer: Berlin/Heidelberg, Germany, 1985; Volume 4. [CrossRef]
19. Bulgakov, V.E. Multi-level iterative technique and aggregation concept with semi-analytical preconditioning for solving boundary-value problems. *Commun. Numer. Methods Eng.* **1993**, *9*, 649–657. [CrossRef]
20. Emans, M. Aggregation schemes for k-cycle AMG. *Proc. Conf. Algoritm.* **2015**, 141–150. Available online: http://www.iam.fmph.uniba.sk/amuc/ojs/index.php/algoritmy/article/view/324 (accessed on 28 November 2022).
21. Chow, E. A Priori Sparsity Patterns for Parallel Sparse Approximate Inverse Preconditioners. *SIAM J. Sci. Comput.* **2000**, *21*, 1804–1822.
22. Grylonakis, E.N.G.; Filelis-Papadopoulos, C.K.; Gravvanis, G.A. On the Numerical Solution of the Generalized Dirichlet-Neumann Map for the 2D Laplace Equation Using Modified Generic Factored Approximate Sparse Inverse Preconditioning. In Proceedings of the 19th Panhellenic Conference on Informatics, Athens, Greece, 1–3 October 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 13–18. [CrossRef]
23. Schenk, O.; Gärtner, K. PARDISO. In *Encyclopedia of Parallel Computing*; Padua, D., Ed.; Springer: Boston, MA, USA, 2011; pp. 1458–1464. [CrossRef]
24. Ulrich Trottenberg, U.; Oosterlee, C.W.; Schuller, A. *Multigrid*; Academic Press: Cambridge, MA, USA; Elsevier: Amsterdam, The Netherlands, 2000.
25. Filelis-Papadopoulos, C.K.; Gravvanis, G.A. Parallel multigrid algorithms based on generic approximate sparse inverses: An SMP approach. *J. Supercomput.* **2014**, *67*, 384–407. [CrossRef]