International Scholarly Research Network
ISRN Applied Mathematics
Volume 2012, Article ID 127647, 49 pages
doi:10.5402/2012/127647

Review Article

Preconditioning for Sparse Linear Systems at the Dawn of the 21st Century: History, Current Developments, and Future Perspectives

Massimiliano Ferronato

Dipartimento ICEA, Università di Padova, Via Trieste 63, 35121 Padova, Italy

Correspondence should be addressed to Massimiliano Ferronato, ferronat@dmsa.unipd.it

Received 1 October 2012; Accepted 22 October 2012

Academic Editors: J. R. Fernandez, Q. Song, S. Sture, and Q. Zhang

Iterative methods are currently the solvers of choice for large sparse linear systems of equations. However, it is well known that the key factor for accelerating, or even allowing for, convergence is the preconditioner. The research on preconditioning techniques has characterized the last two decades. Nowadays, there are a number of different options to be considered when choosing the most appropriate preconditioner for the specific problem at hand. The present work provides an overview of the most popular algorithms available today, emphasizing the respective merits and limitations. The overview is restricted to algebraic preconditioners, that is, general-purpose algorithms requiring the knowledge of the system matrix only, independently of the specific problem it arises from. Along with the traditional distinction between incomplete factorizations and approximate inverses, the most recent developments are considered, including the scalable multigrid and parallel approaches which represent the current frontier of research. A separate section devoted to saddle-point problems, which arise in many different applications, closes the paper.

1. Historical Background

The term "preconditioning" generally refers to *"the art of transforming a problem that appears intractable into another whose solution can be approximated rapidly"* [1], while the "preconditioner" is the mathematical operator responsible for such a transformation. In the context of linear systems of equations,

$$A\mathbf{x} = \mathbf{b}, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$, the preconditioner is a matrix which transforms (1.1) into an equivalent problem for which the convergence of an iterative method is much faster.

Preconditioners are generally used when the matrix A is large and sparse, as it typically, but not exclusively, arises from the numerical discretization of Partial Differential Equations (PDEs). It is not rare that the preconditioner itself is the operator which makes it possible to solve numerically the system (1.1), that otherwise would be intractable from a computational viewpoint.

The development of preconditioning techniques for large sparse linear systems is strictly connected to the history of iterative methods. As mentioned by Benzi [2], the term "preconditioning" used to identify the acceleration of the iterative solution of a linear system can be found for the first time in a 1968 paper by Evans [3], but the concept is pretty older and was already introduced by Cesari back in 1937 [4]. However, it is well known that the earliest tricks to solve numerically a linear system were already developed in the 19th century. As reported in the survey by Saad and Van der Vorst [5] and according to the work by Varga [6], Gauss set the pace for iterative methods in an 1823 letter, where the famous German mathematician describes a number of clever tricks to solve a singular 4×4 linear system arising from a geodetic problem. Most of the strategies suggested by Gauss compose what we currently know as the stationary Gauss-Seidel method, which was later formalized by Seidel in 1874. In his letter, Gauss jokes saying that the method he suggests is actually a pleasant entertainment, as one can think about many other things while carrying out the repetitive operations required to solve the systems. Actually, it is rather a surprise that such a method attracted a great interest in the era of automatic computing! In 1846 another famous algorithm was introduced by Jacobi to solve a sequence of 7×7 linear systems enforcing diagonal dominance by plane rotations. Other similar methods were later developed, such as the Richardson [7] and Cimmino [8] iteration, but no significant improvements in the field of numerical solution of linear systems can be reported until the 40s. This period is referred to as the *prehistory* of this subject by Benzi [2].

As it often occurred in the past, a dramatic and tragic historical event like World War II increased the research fundings for military reasons and helped accelerate the development and progress also in the field of numerical linear algebra. World War II brought a great development of the first automatic computing machines, which later led to the modern digital electronic era. With the availability of such new computing tools, the interest in the numerical solution of relatively large linear systems of equations suddenly grew. The more significant advances were obtained in the field of direct methods. Between the 40s and 70s the algorithms of choice for solving automatically a linear system are based on the Gaussian elimination, with the development of several important variants which helped greatly improve the native method. First, it was recognized that pivoting techniques are of paramount importance to stabilize the numerical computations of the triangular factors of the system matrix and reduce the effects of rounding errors, for example, [9]. Second, several reordering techniques were developed in order to limit the fill-in of the triangular factors and the number of operations required for their computation. The most famous bandwidth reducing algorithms, such as Cuthill-McKee and its reverse variant [10], nested dissection [11–13], and minimum degree [14–16], are still very popular today. Third, appropriate scaling strategies were introduced so as to balance the system matrix entries and help stabilize the triangular factor computation, for example, [17, 18].

In the meantime, iterative methods lived in a kind of niche, despite the publication in 1952 of the famous papers by Hestenes and Stiefel [19] and Lanczos [20] who discovered independently and almost simultaneously the Conjugate Gradient (CG) method for solving symmetric positive definite (SPD) linear systems. These papers did not go unnoticed, but at that time CG was interpreted as a direct method converging in m steps, if m is the number

of distinct eigenvalues of the system matrix A. As it was soon realized [21], this property is lost in practice when working in finite arithmetics, so that convergence is actually achieved in a number of iterations possibly much larger than m. This is why CG was considered an attractive alternative to Gaussian elimination in a few lucky cases only and was not rated much more than just an elegant theory. The iterative method of choice of the period was the Successive Overrelaxation (SOR) algorithm, which was introduced independently by Young [22] in his Ph.D. thesis and Frankel [23] as an acceleration of the old Gauss-Seidel stationary iteration. The possibility of estimating theoretically the optimal overrelaxation parameter for a large class of matrices having the so-called property A [22] gained a remarkable success for SOR especially in problems arising from the discretization of PDEs of elliptic type, for example, in groundwater hydrology or nuclear reactor diffusion [6].

In the 60s the Finite Element (FE) method was introduced in structural mechanics. The new method had a great success and gave rise to a novel set of large matrices which were very sparse but neither diagonally dominant nor characterized by property A. Unfortunately, SOR techniques were not reliable in many of these problems, either converging very slowly or even diverging. Therefore, it is not a surprise that direct methods soon became the reference techniques in this field. The irregular sparsity pattern of the FE-discretized structural problems gave a renewed impulse to the development of more appropriate ordering strategies based on the graph theory and more efficient direct algorithms, leading in the 70s and early 80s to the formulation of the modern multifrontal solvers [13, 24, 25]. In the 80s the first pioneering commercial codes for FE structural computations became available and the solvers of choice were obviously direct methods because of their reliability and robustness. At this time the solution of sparse linear systems by direct techniques appears to be quite a mature field of research, well summarized in famous reference textbooks such as the ones by Duff et al. [26] and Davis [27].

In contrast, during the 60s and 70s iterative solvers were living their infancy. CG was rehabilitated as an iterative method by Reid in 1971 [28] who showed that for reasonably well-conditioned sparse SPD matrices the convergence could have been reached after far fewer than n steps, being n the size of A. This work set the pace for the extension of CG to nonsymmetric and indefinite problems, with the introduction of the earliest Krylov subspace methods such as the Minimal Residual (MINRES) [29] and the Biconjugate Gradient (Bi-CG) [30] algorithms. However, the crucial event for the future success of Krylov subspace methods was the publication in 1977 by Meijerink and van der Vorst of the Incomplete Cholesky CG (ICCG) algorithm [31]. Incomplete factorizations had been already introduced in the Soviet Union [32] and independently by Varga [33] in the late 50s, and the seminal ideas for improving the conditioning of the system matrix in the CG iteration can be found also in the original work by Hestenes and Stiefel [19] and in Engeli et al. [21], but Meijerink and van der Vorst were the first who put the things together and showed the great potential of the preconditioned CG. The idea was later extended by Kershaw [34] to an SPD matrix not necessarily of M-type and gained soon a great popularity. The key point was that CG as well as any other Krylov subspace method with a proper preconditioning could become competitive with the latest direct methods, though requiring much less memory and so being attractive for large three-dimensional (3D) simulations.

The 80s and 90s are the years of the great development of Krylov subspace methods. In 1986 Saad and Schultz introduced the Generalized Minimal Residual (GMRES) method [35] which soon became the algorithm of choice among the iterative methods for nonsymmetric linear systems. Some of the drawbacks of Bi-CG were addressed in 1989 by Sonnenveld who developed the Conjugate Gradient Squared (CGS) method [36], on the basis of which in 1992

van der Vorst presented the Biconjugate Gradient Stabilized (Bi-CGSTAB) algorithm [37]. Along with GMRES, Bi-CGSTAB is the most successful technique for nonsymmetric linear systems, with the advantage of relying on a short-term recurrence. As demonstrated in the famous Faber-Manteuffel theorem [38], Bi-CGSTAB is not optimal in the sense that it does not theoretically lead to an approximate solution with some minimum property in the current Krylov subspace; however, its convergence can be faster than GMRES. Another family of Krylov subspace methods includes the Quasi-Minimal Residual (QMR) algorithm [39], with its transpose-free (TFQMR) and symmetric (SQMR) variants [40, 41]. Several other variants of the methods above, including truncated, restarted, and flexible versions, along with nested optimal techniques, for example, [42, 43], have been introduced by several researchers until the late 90s. A recent review of Krylov subspace methods for linear systems is available in [44], while several textbooks have been written on this subject at the beginning of the 21st century, among which the most famous is perhaps the one by Saad [45].

Initially, Krylov subspace methods were seen with some suspicion by the practitioners, especially those coming from structural mechanics, because of their apparent lack of reliability. Numerical experiments in different fields, such as FE elastostatics, geomechanics, consolidation of porous media, fluid flow, and transport, for example, [46–51], showed a growing interest for these methods, mainly because they potentially could allow for the solution of much bigger and more detailed problems. Whereas direct methods typically scale poorly with the matrix size, especially in 3D models, Krylov subspace methods appeared to be virtually mandatory with large problems. On the other hand, it became soon clear that the key factor to improve the robustness and the computational efficiency of any iterative solver is preconditioning. Nowadays, it is quite well accepted that it is preconditioning, rather than the selected Krylov algorithm, the most important issue to address.

This is why research on the construction of effective preconditioners has significantly grown over the last two decades, while advances on Krylov subspace methods have progressively faded. Currently, preconditioning appears to be a much more active and promising research field than either direct and iterative solution methods, and particularly so within the context of the fast evolution of the hardware technology. On one hand, this is due to the understanding that there are virtually no limits to the available options for obtaining a good preconditioner. On the other hand, it is also generally recognized that an optimal general-purpose preconditioner is unlikely to exist, so that there is the possibility to improve the solver efficiency in different ways for any specific problem at hand within any specific computing environment. Generally, the knowledge of the governing physical processes, the structure of the resulting system matrix, and the available computer technology are factors that cannot be ignored in the design of an appropriate preconditioner. It is also recognized that theoretical results are few, and frequently "empirical" algorithms may work surprisingly well despite the lack of a rigorous foundation. This is why finding a good preconditioner for solving a sparse linear system can be viewed rather as *"a combination of art and science"* [45] than a rigorous mathematical exercise.

2. Basic Concepts

Roughly speaking, preconditioning means transforming system (1.1) into an equivalent mathematical problem which is expected to converge faster using an iterative solver. For instance, given a nonsingular matrix M, premultiplying (1.1) by M^{-1} yields

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}. \tag{2.1}$$

If the matrix $M^{-1}A$ is better conditioned than A for a Krylov subspace method, then M^{-1} is the *preconditioner* and (2.1) denotes the *left preconditioned* system. Similarly, M^{-1} can be applied on the right:

$$AM^{-1}\mathbf{y} = \mathbf{b}, \qquad \mathbf{x} = M^{-1}\mathbf{y}, \tag{2.2}$$

thus producing a *right preconditioned* system. The use of left or right preconditioning depends on a number of factors and can produce quite different behaviors; see, for example, [45]. For instance, right preconditioning has the advantage that the residual of the preconditioned system is the same as the native system, while with left preconditioning it is not, and this can be important in the convergence check or using a residual minimization algorithm such as GMRES. By the way, with the right preconditioning a back transformation of the auxiliary variable $\mathbf{y}$ must be carried out to resume the original unknown $\mathbf{x}$. If the preconditioner can be written in a factorized form:

$$M^{-1} = M_2^{-1}M_1^{-1}, \tag{2.3}$$

a *split preconditioning* can be also used:

$$M_1^{-1}AM_2^{-1}\mathbf{y} = M_1^{-1}\mathbf{b}, \qquad \mathbf{x} = M_2^{-1}\mathbf{y}, \tag{2.4}$$

thus potentially exploiting the advantages of both left and right formulations.

Writing the preconditioned Krylov subspace algorithms is relatively straightforward. Simply, the basic algorithms can be implemented replacing A with either $M^{-1}A$, or AM^{-1}, or $M_1^{-1}AM_2^{-1}$, and then back substituting the original variable $\mathbf{x}$ where the auxiliary vector $\mathbf{y}$ is used. It can be easily observed that it is not necessary to build the preconditioned matrix, which could be much less sparse than A, but just to compute the product of M^{-1}, or M_1^{-1} and M_2^{-1}, if the factorized form (2.3) is used, by a vector. This operation is called *application* of the preconditioner.

Generally speaking, there are three basic requirements for obtaining a good preconditioner:

(i) the preconditioned matrix should have a clustered eigenspectrum away from 0,

(ii) the preconditioner should be as cheap to compute as possible,

(iii) its application to a vector should be cost-effective.

The origin of condition (i) relies on the convergence properties of CG. It is quite intuitive that if M^{-1} resembles in some sense A^{-1} the preconditioned matrix should be close to the identity, thus making the preconditioned system somewhat "easier" to solve and accelerating the convergence. If A and M^{-1} are SPD, it has been proved that the iteration count n_{iter} of the CG algorithm to go below the tolerance ε depends on the spectral conditioning number ξ of the preconditioned matrix G, for example [52]:

$$n_{\text{iter}} \leq \frac{1}{2}\sqrt{\xi(G)}\ln\frac{2}{\varepsilon} + 1, \tag{2.5}$$

so that clustering the eigenvalues of G away from 0 is of paramount importance to accelerate convergence. If A is not SPD, things can be much more complicated. For example, the performance of GMRES depends also on the conditioning of the matrix of the eigenvectors of G, and the fact that the preconditioned matrix has a clustered eigenspectrum does not guarantee a fast convergence [53]. Anyway, it is quite a common experience that a clustered eigenspectrum away from 0 often yields a rapid convergence also in nonsymmetric problems.

The importance of conditions (ii) and (iii) depends on the specific problem at hand and may be highly influenced by the computer architecture. From a practical point of view, they can also be more important than condition (i). For example, if a sequence of linear systems has to be solved with the same matrix A or with slight changes only, as it often may occur when using a Newton method for a set of nonlinear equations, the same preconditioner can be used several times, with the cost for its computation easily amortized. In this case, it could be convenient spending more time to build an effective preconditioner. On the other hand, the computational cost for the preconditioner application basically depends on its density. As M^{-1} has to be applied once or twice per iteration, according to the selected Krylov subspace method, it is of paramount importance that the increased cost of each iteration is counterbalanced by an adequate reduction of their number. Typically, pursuing condition (i) reduces the iteration count but is in conflict with the need for a cheap computation and application of the preconditioner, so that in the end preconditioning efficiently the linear system (1.1) is always the result of an appropriate tradeoff.

An easy way to build a preconditioner is based on the decomposition of A. Recalling the definition of stationary iteration,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + C\mathbf{r}_k, \tag{2.6}$$

where $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$ is the current residual, the matrix C can be used as a preconditioner. Splitting A as $D - E - F$, where D, $-E$, and $-F$ are the diagonal, the lower, and the upper parts of A, respectively, then

$$M_J^{-1} = D^{-1}, \tag{2.7}$$

$$M_S^{-1} = (D - E)^{-1}, \tag{2.8}$$

$$M_\omega^{-1} = \omega(D - \omega E)^{-1}, \tag{2.9}$$

$$M_\Omega^{-1} = \omega(2 - \omega)\,(D - \omega F)^{-1} D (D - \omega E)^{-1} \tag{2.10}$$

are known as the Jacobi, Gauss-Seidel, SOR, and Symmetric SOR preconditioners, respectively, where ω is the overrelaxation factor. Replacing E with F in (2.8) and (2.9) gives rise to the backward Gauss-Seidel and SOR preconditioners, respectively. The preconditioners obtained from the splitting of A do not require additional memory and have a null computation cost, while their application can be simply done by a forward or backward substitution. With M_J^{-1} a simple diagonal scaling is actually required.

Because of their simplicity, Jacobi, Gauss-Seidel, SOR, and Symmetric SOR preconditioners are still used in some applications. Assume, for instance, that matrix A has two clusters of eigenvalues due to the heterogeneity of the underlying physical problem. This is

a quite common experience in geotechnical soil/structure or contact simulations, for example, [49, 54–58]. In this case the matrix A can be written as

$$A = \begin{bmatrix} K & B_1 \\ B_2 & H \end{bmatrix}, \tag{2.11}$$

where the square diagonal blocks are responsible of the two clusters of eigenvalues and usually arise in a natural way from the original problem. Factorizing A in (2.11) according to Sylvester's theorem of inertia:

$$A = \begin{bmatrix} I & 0 \\ B_2K^{-1} & I \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & K^{-1}B_1 \\ 0 & I \end{bmatrix}, \tag{2.12}$$

where

$$S = H - B_2K^{-1}B_1 \tag{2.13}$$

is the Schur complement, a Generalized Jacobi (GJ) preconditioner [59], based on the seminal idea in [60], can be obtained as a diagonal approximation of the central block diagonal factor in (2.12):

$$M^{-1} = \begin{bmatrix} \operatorname{diag}(K) & 0 \\ 0 & \varphi \operatorname{diag}(\widetilde{S}) \end{bmatrix}^{-1}, \tag{2.14}$$

where φ is a user-specified parameter and $\widetilde{S}$ is an approximation of S in (2.13) with $\operatorname{diag}(K)$ used instead of K. Quite recently Bergamaschi and Martínez [61] have indirectly proved that an optimal value for φ is the ratio between the largest eigenvalues of K and S. Another very simple algorithm which has proved effective in similar applications is based on SOR and Symmetric SOR preconditioners, for example, [62–64].

Though the algorithms above cannot compete with more sophisticated tools in terms of global performance [58, 65], they are an example of how even somewhat naive ideas can work and be useful for practitioners. Generally speaking, it is always a good idea to develop an effective preconditioner starting from the specific problem at hand. Interesting strategies have been developed approximating directly the differential operator the matrix A arises from, by using a "nearby" PDE or a less accurate problem discretization, for example, [66–69]. For instance, popular preconditioners typically used within the transport community are often physically based, for example, [70–72].

For the developers of computer codes, however, using physically based preconditioners is not always desirable. In fact, it is usually simpler introducing the linear solver as a black box which is expected to work independently of the specific problem at hand, and especially so with less specialized codes such as the commercial ones. This is why a great interest has arisen also in the development of purely algebraic preconditioners that claim to be as much "general purpose" as possible. Algebraic preconditioners are usually robust algorithms which require the knowledge of the matrix A only, independently of the underlying physical problem. This paper will concentrate on such a class of algorithms. Subdividing algebraic

preconditioners into categories is not straightforward, as especially in recent years there have been several contaminations from adjacent fields with the development of "hybrid" approaches. Traditionally, and as already done by Benzi [2], algebraic preconditioners can be roughly divided into two classes, that is, incomplete factorizations and approximate inverses. The main distinction is that with the former approach M is actually computed and M^{-1} is applied but never formed explicitly, whereas with the latter M^{-1} is directly built and applied. Within each class, moreover, different strategies can be pursued according to a number of factors, such as whether A is SPD or not, its sparsity structure and size, the magnitude of its coefficients, and ultimately the computer hardware available to solve the problem, so that a large number of variants have been proposed by the researchers. As anticipated before, the unavoidable contamination from adjacent fields, for example, the direct solution methods, has made the boundaries between different groups of algorithms often vague and certainly questionable.

In this work, the traditional distinction between incomplete factorizations and approximate inverses is followed, describing the most successful algorithms belonging to each class and the most significant variants developed to address particular occurrences and increase efficiency. Then, two additional sections will be devoted to the most recent results obtained in the fields that currently appear to be the more active frontier of research in the area of algebraic preconditioning, that is, multigrid techniques and parallel algorithms. Finally, a few words will be spent on a special class of problems characterized by indefinite saddle-point matrices, which arise quite frequently in the applications and have attracted much interest in recent years.

3. Incomplete Factorizations

Given a nonsingular matrix A such that

$$A = LU, \tag{3.1}$$

the basic idea of incomplete factorizations is to approximate the triangular factors L and U in a cost-effective way. If L and U are computed so as to satisfy (3.1), for example, by a Gaussian elimination procedure, typically fill-in takes place, with L and U much less sparse than A. This is what limits the use of direct methods in large 3D problems. The approximations $\tilde{L}$ and $\tilde{U}$ of L and U, respectively, can be obtained by simply discarding a number of fill-in entries according to some rules. The resulting preconditioner

$$M^{-1} = \tilde{U}^{-1}\tilde{L}^{-1} \tag{3.2}$$

is generally denoted as Incomplete LU (ILU). Quite obviously, M^{-1} in (3.2) is never built explicitly as it is much denser than both $\tilde{L}$ and $\tilde{U}$. Its application to a vector is therefore performed by forward and backward substitutions. Moreover, the ILU factorized form (3.2) is well suited for split preconditioning. In case A is SPD, the upper incomplete factor $\tilde{U}$ is replaced by $\tilde{L}^T$ and the related preconditioner is known as Incomplete Cholesky (IC).

The native ILU algorithm runs as follows. Define a set $\mathcal{S}$ of position (i, j), with $1 \leq i, j \leq n$, where either $\tilde{L}$ if $i > j$ or $\tilde{U}$ if $j > i$ has a nonzero entry. The set $\mathcal{S}$ is also denoted as the *nonzero pattern* of the incomplete factors. To avoid breakdowns in the ILU computation, it is

Input: Matrix A, the nonzero pattern $\mathcal{S}$
Output: Matrix A containing $\widetilde{L}$ and $\widetilde{U}$
for each $(i, j) \notin \mathcal{S}$ **do**
  $a_{ij} = 0$
end
for $i = 2, \ldots, n$ **do**
  for $k = 1, \ldots, i-1$ and $(i, k) \in \mathcal{S}$ **do**
    $a_{ik} \leftarrow a_{ik} / a_{kk}$
    for $j = k+1, \ldots, n$ and $(i, j) \in \mathcal{S}$ **do**
      $a_{ij} \leftarrow a_{ij} - a_{ik} a_{kj}$
    end
  end
end

Algorithm 1: IKJ variant of ILU factorization with a static pattern $\mathcal{S}$.

generally recommended to include all the diagonal entries in $\mathcal{S}$. Then, make a copy of A over an auxiliary matrix which will contain $\widetilde{L}$ and $\widetilde{U}$ in the lower and upper parts, respectively, at the end of the computation, and set to zero all entries a_{ij} such that $(i, j) \notin \mathcal{S}$. If the main diagonal belongs to $\widetilde{L}$, then it is implicitly assumed that the diagonal of $\widetilde{U}$ is unitary and vice versa. For every position $(i, j) \in \mathcal{S}$, the following update is computed:

$$a_{ij} \longleftarrow a_{ij} - a_{ik} a_{kk}^{-1} a_{kj} \tag{3.3}$$

with $k = 1, \ldots, i-1$. At the end of the loop, the ILU factors are stored in the copy of A.

The procedure described above is clearly an incomplete Gauss elimination. Hence, no surprise that several implementation tricks developed to improve the efficiency of direct solution methods can be borrowed in order to reduce the computational cost of the ILU computation. For example, the algorithm can follow either the KIJ, or the IKJ, or the IJK elimination variants. To give an idea, Algorithm 1 provides a sketch of the sequence of operations required by the IKJ implementation, which is generally preferred whenever the sparse matrix A along with both $\widetilde{L}$ and $\widetilde{U}$ is stored in a compact form by rows. It can be easily observed that Algorithm 1 proceeds by rows, computing first the row of $\widetilde{L}$ and then that of $\widetilde{U}$ and accessing the previous rows only to gather the required information (Figure 1). Many other efficient variants have been developed, for example, storing A in a sparse skyline format or computing $\widetilde{L}$ by columns [73, 74].

3.1. Fill-In Strategies

The several existing versions of the ILU preconditioner basically differ for the rules followed to select the retained entries in $\widetilde{L}$ and $\widetilde{U}$. The simplest idea is to define $\mathcal{S}$ statically at the beginning of the algorithm and equal to the set of the nonzero positions in A. This idea was originally implemented by Meijerink and van der Vorst [31] in their 1977 seminal work, giving rise to what is traditionally referred to as ILU(0), or IC(0) for SPD matrices. This simple choice is easy to implement and very cost-effective, as the preconditioner construction is quite cheap and its application to a vector costs as much as a matrix-by-vector product with A. Moreover,

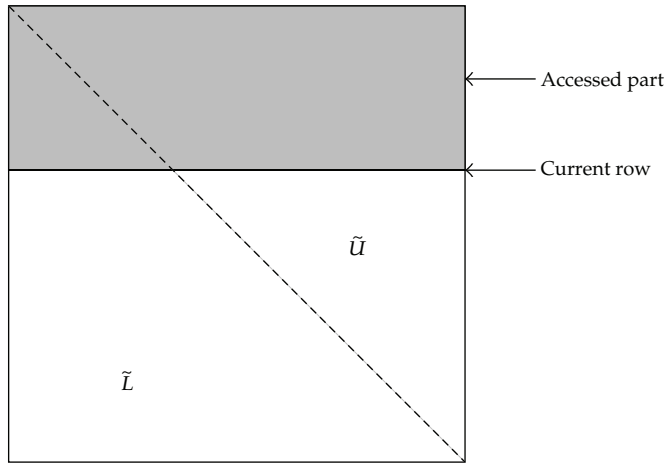

Figure 1: IKJ variant of ILU factorization.

ILU(0) can be quite efficient, especially for matrices arising from the discretization of elliptic problems, for example, [75–77].

Unfortunately, there are a number of problems where the ILU(0) preconditioner converges very slowly or even fails. There are several different reasons for this, depending on both the unstable computation and the unstable application of $\tilde{L}$ and $\tilde{U}$, as was experimentally found by Chow and Saad on a large number of test matrices [78]. For example, ILU(0) often exhibits a bad behavior with structural matrices and the highly nonsymmetric and indefinite matrices arising from computational fluid dynamics. In these cases the nonzero pattern of A proves a too crude approximation of the exact L and U structure, so that a possible remedy is to allow for a larger number of entries in $\tilde{L}$ and $\tilde{U}$ in order to make them closer to L and U. In the limiting case, the exact factors are computed and the solution to (1.1) is obtained in just one iteration. Obviously, enlarging the ILU pattern on one hand reduces the iteration count but on the other increases the cost of computation and application of the preconditioner, so an appropriate tradeoff must be found.

There are several different ways for enlarging the initial pattern $\mathcal{S}$, or computing it in a dynamic way. In any specific problem, the winning algorithm is the one that proves able to capture the most significant entries in $\tilde{L}$ and $\tilde{U}$ at the lowest cost. One of the first attempts for enlarging dynamically $\mathcal{S}$ is based on the so-called *level-of-fill* concept [79, 80]. The idea is to assign an integer value ℓ_{ij}, denoted as level-of-fill, to each entry in position (i, j) computed during the ILU construction process. The lower the level, the more important the entry. The initial level is set to

$$\ell_{ij} = \begin{cases} 0 & \text{if } a_{ij} \neq 0 \\ \infty & \text{if } a_{ij} = 0. \end{cases} \tag{3.4}$$

During the ILU construction, with reference for instance to the Algorithm 1, the level-of-fill of each entry is updated according to the level-of-fill of the entries used for its computation using the following rule:

$$\ell_{ij} \longleftarrow \min\left\{\ell_{ij}, \ell_{ik} + \ell_{kj} + 1\right\}. \tag{3.5}$$

In practice, any position corresponding to a nonzero entry in A remains with a zero level-of-fill. The terms computed using entries with a zero level-of-fill have a level equal to 1. Those built using one entry of the first level have level 2 and so on. The new level-of-fill of each entry is computed adding 1 to the sum of the levels of the entries used for it. This kind of algorithm creates a hierarchy among the potential entries that allows for selecting the "most important" nonzero terms only. The resulting preconditioner is denoted as ILU(p), where $p > 0$ is the integer user-specified level-of-fill of the entries retained into each row of $\widetilde{L}$ and $\widetilde{U}$. It follows immediately that ILU(0) coincides exactly with the zero fill-in ILU previously defined.

In most applications, ILU(1) is already a significant improvement to ILU(0). The experience shows that the fill-in increases quite rapidly with p, so that for $p \geq 2$ the cost for building and applying the preconditioner can become a price too high to pay. Moreover it can be observed that at each level there are also many terms which are very small in magnitude, thus providing a small contribution to the preconditioning of the original matrix A. Hence, a natural way to avoid such an occurrence is to add a dropping rule according to which the terms below a user-specified tolerance τ are neglected. Several criteria have been defined for selecting τ and the way the ILU entries are discarded. A usual choice is to compute the ith row of both $\widetilde{L}$ and $\widetilde{U}$ and drop an entry whenever smaller than $\tau\|\mathbf{a}_i\|$, being $\|\cdot\|$ an appropriate vector norm and $\mathbf{a}_i$ the ith row of A, but other strategies have been also attempted, such as comparing the entry l_{ij} of $\widetilde{L}$ with $\tau l_{ii} l_{jj}$ [81] or with an estimate of the norm of the jth column of $\widetilde{L}^{-1}$ [82, 83].

Similarly to the level-of-fill approach, a drawback of the drop tolerance strategy is that the amount of fill-in of $\widetilde{L}$ and $\widetilde{U}$ is strongly dependent on the selected user-specified parameter and the specific problem at hand and is unpredictable a priori. To address such an issue a second user-specified parameter ρ can be introduced, denoting the largest number of entries that can be retained per row. In some versions, ρ is also defined as the number of entries added at most per row in excess to the number of entries of the same row of A. The use of this *dual threshold* strategy, which may be quite simple and effective, has been proposed by Saad [84], and the related preconditioner is referred to as ILUT(ρ, τ). A sketch of the sequence of steps required for its computation is provided in Algorithm 2.

As the drop tolerance may be quite sensitive to the specific problem and so difficult to implement in a black box solver, ILU variants that use only the fill-in parameter ρ have been proposed [85, 86]. These versions are generally outperformed by ILUT(ρ, τ) but are much easier to set and generally less sensitive to the user-specified parameter, allowing the user for a thorough control of the memory occupation. More recent ILU variants developed with the aim of improving the preconditioner performance and better exploiting the hardware potential are based on multilevel techniques, dense block identification, and adaptive selections of the setup parameters [87–89], with significant contaminations from the methods developed for the latest multifrontal solvers.

3.2. Stabilization Techniques

Both Algorithms 1 and 2 require the execution of divisions where the denominator, that is, the pivot, is not generally guaranteed to be nonzero. It is well known that in finite arithmetics also small pivots can create numerical difficulties. If A is SPD, the pivot has to be also positive in order to produce a real incomplete factorization. These issues are very important because they can jeopardize the existence of the preconditioner itself and so the robustness of the overall iterative algorithm.

Input: Matrix A, the user-specified parameters ρ and τ
Output: Matrices $\widetilde{L}$ and $\widetilde{U}$
for $i = 1, \ldots, n$ **do**
  $\mathbf{w} = \mathbf{a}_i$
  for $k = 1, \ldots, i-1$ and $w_k \neq 0$ **do**
    $w_k \leftarrow w_k / a_{kk}$
    if $w_k \geq \tau \|\mathbf{a}_i\|_2$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} - w_k \mathbf{u}_k$
    else
      $w_k = 0$
    end
  end
  Retain the ρ largest entries in $\mathbf{w}$
  for $j = 1, \ldots, i-1$ **do**
    $l_{ij} = w_j$
  end
  for $j = i, \ldots, n$ **do**
    $u_{ij} = w_j$
  end
end

Algorithm 2: ILUT(ρ, τ) computation.

Meijerink and van der Vorst [31] demonstrated that for an M-matrix a zero fill-in IC is guaranteed to exist. However, if A is SPD but not of M-type zero or negative pivots may arise, thus theoretically preventing from the IC computation. Typically, such an inconvenience may be avoided by increasing properly the fill-in. In this way the incomplete factor tends to the exact factor, which is guaranteed to be SPD. However, this may generate a fill-in degree that is unacceptably high, lowering the performance of the preconditioned solver. To avoid this undesirable occurrence a number of tricks have been proposed which can greatly increase the IC robustness. The first, and maybe simplest, idea was advanced by Kershaw [34] who just suggested to replace zero or negative pivots with an arbitrary positive value, for example, the last valid pivot. The implementation of this trick is straightforward, but it seldom provides good results as the resulting IC often exhibits a very low quality. In fact, recall from Algorithm 1 that modifying arbitrarily a pivot in the kth row means influencing the computation of all the following rows. Another simple strategy recently advanced in [90] relies on avoiding the computation of the square root of the pivot, required in the IC algorithm, thus eliminating a source of breakdown. In [91] a diagonal compensated reduction of the positive off-diagonal entries is suggested in order to transform the native matrix into an M-matrix before the incomplete factorization process is performed. A different algorithm is proposed in [92] with the incomplete factors obtained through an A-orthogonalization instead of a Cholesky elimination.

A famous stabilization technique to ensure the IC existence is the one advanced by Ajiz and Jennings [93], that has proved quite effective especially in structural applications [50, 90]. Following this approach, we can write

$$A = LL^T = \widetilde{L}\widetilde{L}^T + R, \tag{3.6}$$

where R is a residual matrix collecting with all the entries discarded during the incomplete procedure. From (3.6), $\widetilde{L}\widetilde{L}^T$ is actually the exact factorization of $A - R$. As A is SPD, it follows immediately that a sufficient condition for $\widetilde{L}$ to be real is that R is negative semidefinite. Suppose that the first column of L has been computed and the term $l_{j1} = a_{j1}/a_{11}$ has to be dropped along with the corresponding l_{1j} term in L^T. Hence, the incomplete factorization is equivalent to the exact factorization of $\overline{A}$ where the entries a_{1j} and a_{j1} are replaced by zero, and the error matrix R contains a_{j1} in position $1j$ and $j1$. For example, in a 3×3 matrix with $j = 2$, we have

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & a_{13} \\ 0 & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & 0 \\ a_{21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \overline{A} + R. \tag{3.7}$$

The residual matrix R can be forced to be negative semidefinite by adding two coefficients α and β in position $(1,1)$ and (j,j), such that the submatrix

$$\begin{bmatrix} \alpha & a_{1j} \\ a_{j1} & \beta \end{bmatrix} \tag{3.8}$$

is negative semidefinite. The matrix $\overline{A}$ must be modified accordingly by subtracting α and β from the corresponding diagonal terms. Though different choices are possible, for example, [50, 93], the option $\alpha = \beta = -|a_{1j}|$ is particularly attractive because in this way the sum of the absolute values of the arbitrarily introduced entries $|\alpha| + |\beta|$ is minimized. This procedure ensures that $\widetilde{L}\widetilde{L}^T$ is the exact factorization of a positive definite matrix, hence its existence is guaranteed, but obviously such a matrix can be quite different from A. The quality of this stabilization strategy basically depends on the quality of $\overline{A}$ as an approximation of A.

Unstable factors and negative pivots typically arise if the diagonal terms of A are remarkably smaller than the off-diagonal ones, or if there are significant contrasts in the coefficient magnitude. The latter occurrence is particularly frequent in heterogeneous problems or in models coupling different physical processes, such as multiphase flow in deformable media, coupled flow and transport, themomechanical models, fluid-structure interactions, or multibody systems, for example, [94–97]. All these problems give naturally rise to a *multilevel* matrix:

$$A = \begin{bmatrix} K_1 & B_{12} & B_{13} & \cdots & B_{1\ell} \\ B_{21} & K_2 & B_{23} & \cdots & B_{2\ell} \\ B_{31} & B_{32} & K_3 & \cdots & B_{3\ell} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ B_{\ell 1} & B_{\ell 2} & B_{\ell 3} & \cdots & K_\ell \end{bmatrix}, \tag{3.9}$$

where the system unknowns can be subdivided into contiguous groups. By distinction with multigrid methods, a multilevel approach assumes that the different levels do not necessarily correspond to model subgrids. Several strategies have been proposed to handle properly the matrix levels, for example, [98–102]. Generally speaking, all such algorithms can be viewed as approximate Schur complement methods that differ for the strategy used to perform the level

subdivision and the restriction and prolongation operators adopted [103, 104]. For a recent review, see, for instance, [105]. The basic idea of multilevel ILU proceeds as follows. Consider the partial factorization of A in (3.9), where A_1 is the submatrix obtained from A without the first level, and $\widehat{B}_{12}$ and $\widehat{B}_{21}$ are the rectangular submatrices collecting the blocks B_{1j} and B_{j1}, $j = 2, \ldots, \ell$:

$$A = \begin{bmatrix} K_1 & \widehat{B}_{12} \\ \widehat{B}_{21} & A_1 \end{bmatrix} = \begin{bmatrix} L_1 & 0 \\ \widehat{B}_{21}U_1^{-1} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & S_1 \end{bmatrix} \begin{bmatrix} U_1 & L_1^{-1}\widehat{B}_{12} \\ 0 & I \end{bmatrix}. \tag{3.10}$$

In (3.10) $K_1 = L_1U_1$ and S_1 is the Schur complement:

$$S_1 = A_1 - \widehat{B}_{21}U_1^{-1}L_1^{-1}\widehat{B}_{12}. \tag{3.11}$$

Replacing L_1 and U_1 with the ILU factors $\widetilde{L}_1$ and $\widetilde{U}_1$ of K_1 gives rise to a partial incomplete factorization of A that can be used as a preconditioner:

$$M^{-1} = \begin{bmatrix} \widetilde{U}_1 & H_{12} \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} I & 0 \\ 0 & \widetilde{S}_1^{-1} \end{bmatrix} \begin{bmatrix} \widetilde{L}_1 & 0 \\ H_{21} & I \end{bmatrix}^{-1}, \tag{3.12}$$

where

$$\begin{aligned} H_{12} &= \widetilde{L}_1^{-1}\widehat{B}_{12} \\ H_{21} &= \widehat{B}_{21}\widetilde{U}_1^{-1}, \end{aligned} \tag{3.13}$$

$$\widetilde{S}_1 = A_1 - H_{21}H_{12}. \tag{3.14}$$

As H_{12} and H_{21} tend to be much denser than $\widehat{B}_{12}$ and $\widehat{B}_{21}$, some degree of dropping can be conveniently enforced, thus replacing these blocks with $\widetilde{H}_{12}$ and $\widetilde{H}_{21}$, respectively. Recalling the level structure of A_1, the inverse of the approximated Schur complement $\widetilde{S}_1$ can be also replaced by a partial incomplete factorization with the same structure as M^{-1} in (3.12). Repeating recursively the procedure above starting from $\widetilde{S}_0 = A$, at the ith level, $\widetilde{S}_i^{-1}$ is replaced by

$$\widetilde{S}_i^{-1} \simeq \begin{bmatrix} \widetilde{U}_{i+1} & \widetilde{H}_{i+1,i+2} \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} I & 0 \\ 0 & \widetilde{S}_{i+1}^{-1} \end{bmatrix} \begin{bmatrix} \widetilde{L}_{i+1} & 0 \\ \widetilde{H}_{i+2,i+1} & I \end{bmatrix}^{-1}, \tag{3.15}$$

thus giving rise to an $(i+1)$th level Schur complement $\widetilde{S}_{i+1} = A_{i+1} - \widetilde{H}_{i+2,i+1}\widetilde{H}_{i+1,i+2}$. The algorithm stops when the size of $\widetilde{S}_{i+1}$ equals 0, that is, when $i+1$ equals the user specified number of levels ℓ. Multilevel ILU preconditioners are generally much more robust than standard ILU and can be very effective provided that a suitable level subdivision is defined. If A is SPD, however, it is necessary to enforce the positive definiteness of all approximate Schur complements, which is no longer ensured when incomplete decompositions and dropping

procedures are used. Recently, Janna et al. [106] have developed a stabilization strategy which can guarantee the positive definiteness of each $\widetilde{S}_i$. Basically, they extend to levels of the factorization procedure introduced by Tismenetsky [107] where the Schur complement is updated by a quadratic formula. The theoretical robustness of Tismenetsky's algorithm has been proved in [108]. Recalling (3.6), the matrix $\widetilde{S}_i$ can be additively decomposed as

$$\widetilde{S}_i = \begin{bmatrix} \widetilde{S}_{i,11} & \widetilde{S}_{i,12} \\ \widetilde{S}_{i,12}^T & \widetilde{S}_{i,22} \end{bmatrix} = \overline{S}_i + R_i = \begin{bmatrix} \overline{S}_{i,11} & \widetilde{S}_{i,12} \\ \widetilde{S}_{i,12}^T & \widetilde{S}_{i,22} \end{bmatrix} + \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix}, \tag{3.16}$$

where R is the residual matrix obtained from the IC decomposition of $\widetilde{S}_{i,11}$. Assuming that an appropriate stabilization technique has been used, for example, the one by Ajiz and Jennings [93], R_i is negative semidefinite, so $\overline{S}_i$ is positive definite. Let us collect all the entries dropped in the $\widetilde{H}_{i+1,i+2}$ computation in the error matrix E_H, thus writing $H_{i+1,i+2} = \widetilde{H}_{i+1,i+2} + E_H$. The Schur complement of $\widetilde{S}_i$ can be now rewritten as

$$S_{i+1} = \widetilde{S}_{i,22} - \widetilde{H}_{i+1,i+2}^T \widetilde{H}_{i+1,i+2} - E_H^T \widetilde{H}_{i+1,i+2} - \widetilde{H}_{i+1,i+2}^T E_H - E_H^T E_H. \tag{3.17}$$

Ignoring the last three terms of (3.17) we obtain

$$\widetilde{S}_{i+1} = \widetilde{S}_{i,22} - \widetilde{H}_{i+1,i+2}^T \widetilde{H}_{i+1,i+2}, \tag{3.18}$$

that is the standard Schur complement usually computed in a multilevel algorithm; see (3.14). As $E_H^T \widetilde{H}_{i+1,i+2}$ is generally indefinite, $\widetilde{S}_{i+1}$ in (3.18) may be indefinite too. As a remedy, add and subtract $\widetilde{H}_{i+1,i+2}^T \widetilde{H}_{i+1,i+2}$ to the right-hand side of (3.17):

$$S_{i+1} = \widetilde{S}_{i,22} + \widetilde{H}_{i+1,i+2}^T \widetilde{H}_{i+1,i+2} - \left(\widetilde{H}_{i+1,i+2} + E_H\right)^T \widetilde{H}_{i+1,i+2} - \widetilde{H}_{i+1,i+2}^T \left(\widetilde{H}_{i+1,i+2} + E_H\right) - E_H^T E_H. \tag{3.19}$$

Recalling that

$$\widetilde{H}_{i+1,i+2} + E_H = H_{i+1,i+2} = \widetilde{L}_{i+1}^{-1} \widetilde{S}_{i,12}, \tag{3.20}$$

(3.19) becomes

$$S_{i+1} = \widetilde{S}_{i,22} + \widetilde{H}_{i+1,i+2}^T \widetilde{H}_{i+1,i+2} - \widetilde{S}_{i,12}^T \widetilde{L}_{i+1}^{-T} \widetilde{H}_{i+1,i+2} - \widetilde{H}_{i+1,i+2}^T \widetilde{L}_{i+1}^{-1} \widetilde{S}_{i,12} - E_H^T E_H. \tag{3.21}$$

Neglecting the last term in (3.21) we obtain another expression for the Schur complement:

$$\widetilde{S}_{i+1} = \widetilde{S}_{i,22} + \widetilde{H}_{i+1,i+2}^T \widetilde{H}_{i+1,i+2} - \widetilde{S}_{i,12}^T \widetilde{L}_{i+1}^{-T} \widetilde{H}_{i+1,i+2} - \widetilde{H}_{i+1,i+2}^T \widetilde{L}_{i+1}^{-1} \widetilde{S}_{i,12}, \tag{3.22}$$

which is always SPD. In fact, (3.21) yields

$$\widetilde{S}_{i+1} = S_{i+1} + E_H^T E_H, \tag{3.23}$$

which is SPD independently of the degree of dropping enforced on $\widetilde{H}_{i+1,i+2}$. The procedure above is generally more expensive than the standard one but is also more robust.

If A is not SPD, enforcing the positivity of pivots or the positive definiteness of the Schur complements is not necessary. However, the experience shows that, in many problems, especially those with a high degree of nonsymmetry and a lack of diagonal dominance, ILU can be much more unstable in both the computation and the application, leading to frequent solver failures [78]. In this case no rigorous methods have been introduced to avoid numerical instabilities. The effect of a preliminary reordering and/or scaling of A has been debated for quite a long time, without achieving a shared conclusion. The fact that the matrix nonzero pattern has an effect on the quality of the resulting ILU preconditioner, similarly to what happens with direct methods, is quite well understood, but not all the methods performing well with direct techniques appear to be efficient for ILU, for example, [90, 109–112]. A similar result is obtained with a preliminary scaling. As proved in [113], scaling the native matrix does not change the spectral properties of an ILU preconditioned matrix; however it can greatly stabilize the ILU computation by reducing significantly the impact of the round-off errors. Useful scaling strategies can be taken from the GJ preconditioning of (2.14) or the Least Square Log algorithm introduced in [18].

4. Approximate Inverses

Incomplete factorizations can be very powerful preconditioners; however the issues concerning their existence and numerical stability can undermine their robustness in several examples. One reason for such a weakness can be understood using the following simple argument. Consider the incomplete factorization $\widetilde{L}\widetilde{U}$ along with the corresponding residual matrix for a general matrix A:

$$A = \widetilde{L}\widetilde{U} + R. \tag{4.1}$$

Left and right multiplying both sides of (4.1) by $\widetilde{L}^{-1}$ and $\widetilde{U}^{-1}$, respectively, yields

$$\widetilde{L}^{-1}A\widetilde{U}^{-1} = I + \widetilde{L}^{-1}R\widetilde{U}^{-1}. \tag{4.2}$$

The matrix controlling the convergence of a preconditioned Krylov subspace method is actually $\widetilde{L}^{-1}A\widetilde{U}^{-1}$ which differs from the identity by $\widetilde{L}^{-1}R\widetilde{U}^{-1}$. This means that a residual matrix close to 0 is not sufficient to guarantee a fast convergence. In fact, if the norms of either $\widetilde{L}^{-1}$ or $\widetilde{U}^{-1}$ are large, $\widetilde{L}^{-1}A\widetilde{U}^{-1}$ can be anyway far from the identity and yields a slow convergence, that is, even an accurate incomplete factorization can provide a poor outcome. This is why problems with strongly nonsymmetric matrices and a lack of diagonal dominance incomplete factorizations often have a bad reputation.

To cope with these drawbacks the researchers in the past have developed a second big class of preconditioners with the aim of computing an explicit form for M^{-1}, thus avoiding the solution of the linear system needed to apply the preconditioner. Generally speaking, an approximate inverse is an explicit approximation of A^{-1}. Quite obviously, it has to be sparse in order to maintain a workable cost for its computation and application. It is no surprise that several different methods have been proposed to build an approximate inverse. The most successful ones can be roughly classified according to whether M^{-1} is computed

monolithically or as the product of two matrices. In the first case, the approximate inverse is generally computed by minimizing the Frobenius norm:

$$\left\| I - AM^{-1} \right\|_F \quad \text{or} \quad \left\| I - M^{-1}A \right\|_F, \tag{4.3}$$

thus obtaining either a *right* or a *left* approximate inverse, which is generally different. In the second case, the basic idea is to find an explicit approximation of the inverse of the triangular factors of A:

$$M_1^{-1} \simeq L^{-1}, \qquad M_2^{-1} \simeq U^{-1}, \tag{4.4}$$

and then use M^{-1} in the factored form (2.3). Different approaches have been advanced within each class and also in between these two groups.

Early algorithms for computing an approximate inverse via the Frobenius norm minimization appeared as far back as in the 70s [114, 115]. The basic idea is to select a nonzero pattern $\mathcal{S}$ for M^{-1} including all the positions where nonzero terms may be expected. Denoting by $\mathcal{W}_S$ the set of matrices with nonzero pattern $\mathcal{S}$, any algorithm belonging to this class looks for a matrix $M^{-1} \in \mathcal{W}_S$ such that $\|I - AM^{-1}\|_F$ is minimum. Recalling the definition of Frobenius norm:

$$\|A\|_F = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}^2}, \tag{4.5}$$

it follows that

$$\left\| I - AM^{-1} \right\|_F^2 = \sum_{j=1}^{n} \left\| \mathbf{e}_j - A\mathbf{m}_j \right\|_2^2, \tag{4.6}$$

where $\mathbf{e}_j$ and $\mathbf{m}_j$ are the jth column of I and M^{-1}, respectively. The sum at the right-hand side of (4.6) is minimum only if each addendum is minimum. Therefore, the computation of M^{-1} is equivalent to the solution of n least-squares problems with the constraint that $\mathbf{m}_j$ has nonzero components only in the positions i such that $(i, j) \in \mathcal{S}$. This requirement reduces enormously the computational cost required to build M^{-1}. Denote by $\mathcal{J}$ the set

$$\mathcal{J} = \{ i : (i, j) \in \mathcal{S} \} \tag{4.7}$$

and $\mathbf{m}_j[\mathcal{J}]$ the subvector of $\mathbf{m}_j$ made of the components included in $\mathcal{J}$. In the product $A\mathbf{m}_j$ only the columns of A with index in $\mathcal{J}$ will provide a nonzero contribution. Moreover, as A is sparse, only a few rows will have a nonzero term in at least one of these columns (Figure 2). Denoting by $\mathcal{I}$ the set

$$\mathcal{I} = \{ i : a_{ij} \neq 0 \text{ with } j \in \mathcal{J} \} \tag{4.8}$$

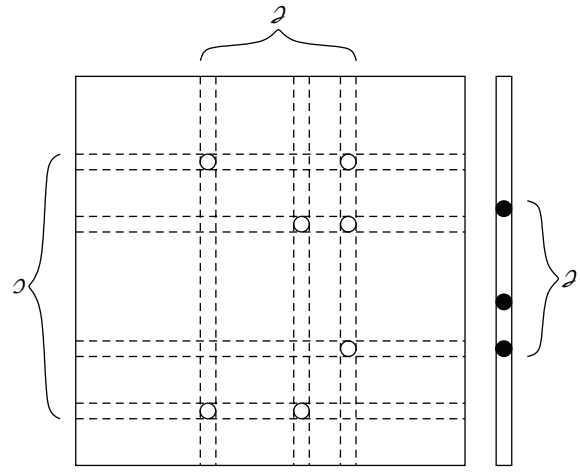

Figure 2: Schematic representation of a matrix-vector product subject to sparsity constraints.

and $A[\mathcal{I}, \mathcal{J}]$ the submatrix of A made of the coefficients a_{ij} such that $i \in \mathcal{I}$ and $j \in \mathcal{J}$, each least square problem at the right-hand side of (4.6) reads

$$\left\| e_j[\mathcal{I}] - A[\mathcal{I}, \mathcal{J}]\mathbf{m}_j[\mathcal{J}] \right\|_2^2 \longrightarrow \min \tag{4.9}$$

and is typically quite small. The procedure illustrated above allows for the computation of a right approximate inverse. To build a left approximate inverse just observe that

$$\left\| I - M^{-1}A \right\|_F = \left\| I - A^T M^{-T} \right\|_F. \tag{4.10}$$

Hence, (4.9) can still be used for any j replacing A with A^T and obtaining the rows of M^{-1}. It turns out that for strongly nonsymmetric matrices the right and left approximate inverse can be quite different.

The main difference in the algorithms developed within this class relies on how the nonzero pattern $\mathcal{S}$ is selected, and this is indeed the key factor for the overall quality of the approximate inverse. A more detailed discussion on this point will be provided in the next section. Some of the most popular algorithms in this class have been advanced in [116–121], each one with its pros and cons. Probably the most successful approach is the one proposed by Grote and Huckle [117] which is known as SPAI (SParse Approximate Inverse). Basically, the SPAI algorithm follows the lines described above dynamically enlarging $\mathcal{J}$ for each column of M^{-1} until the least square problem (4.9) is solved with a prescribed accuracy. SPAI has proved to be quite a robust method allowing for a good control of its quality by the user. Unfortunately, its computation can be quite expensive also on a parallel machine. A sketch of the SPAI construction is reported in Algorithm 3.

A different way to build an approximate inverse of A relies on approximating the inverse of its triangular factors. Assuming that all the leading principal minors of A are not

Input: Matrix A, the initial nonzero pattern $\mathcal{S}$ and the tolerance ε
Output: Matrix M^{-1}
for $j = 1, \ldots, n$ **do**
  Compute $\mathcal{J} = \{i : (i, j) \in \mathcal{S}\}$
  Compute $\mathcal{I} = \{i : a_{ik} \neq 0, k \in \mathcal{J}\}$
  Gather $A[\mathcal{I}, \mathcal{J}]$ from A
  Solve the least square problem $\|\mathbf{e}_j[\mathcal{I}] - A[\mathcal{I}, \mathcal{J}]\mathbf{m}_j[\mathcal{J}]\|_2 \rightarrow \min$
  $\mathbf{r}_j = \mathbf{e}_j[\mathcal{I}] - A[\mathcal{I}, \mathcal{J}]\mathbf{m}_j[\mathcal{J}]$
  while $\|\mathbf{r}_j\|_2 > \varepsilon$ **do**
    Enlarge $\mathcal{J}$ and update $\mathcal{I}$
    Gather $A[\mathcal{I}, \mathcal{J}]$ from A
    Solve the least square problem $\|\mathbf{e}_j[\mathcal{I}] - A[\mathcal{I}, \mathcal{J}]\mathbf{m}_j[\mathcal{J}]\|_2 \rightarrow \min$
    $\mathbf{r}_j = \mathbf{e}_j[\mathcal{I}] - A[\mathcal{I}, \mathcal{J}]\mathbf{m}_j[\mathcal{J}]$
  end
end

Algorithm 3: Computation of the SPAI preconditioner.

null, that is, A can be factorized as the product LDU where L is unit lower triangular, D is diagonal, and U is upper unit triangular, then

$$L^{-1}AU^{-1} = D \tag{4.11}$$

implying that the rows of L^{-1} and the columns of U^{-1} are two sets of A-biconjugate vectors. In other words, denoting by $\mathbf{w}_i$ and $\mathbf{z}_j$ the ith row of L^{-1} and the jth column of U^{-1}, respectively, (4.11) means that

$$\mathbf{w}_i^T A\mathbf{z}_j = 0 \quad \forall i, j = 1, \ldots, n, \ i \neq j. \tag{4.12}$$

If W and Z are the matrices collecting all vectors $\mathbf{w}_i$ and $\mathbf{z}_j$, $i, j = 1, \ldots, n$, satisfying the condition (4.12) and $D = W^T AZ$, then the inverse of A reads

$$A^{-1} = ZD^{-1}W^T = \sum_{k=1}^{n} \frac{\mathbf{z}_k \mathbf{w}_k^T}{d_k} \tag{4.13}$$

and can be computed explicitly. There are infinite sets of vectors satisfying (4.12), and they can be computed by means of a biconjugation process, for example, via a two-sided Gram-Schmidt algorithm, starting from two arbitrary initial sets. Quite obviously, these vectors are dense and their explicit computation is practically unfeasible. However, if the process is carried out incompletely, for instance, dropping the entries of $\mathbf{w}_k$ and $\mathbf{z}_k$, $k = 1, \ldots, n$, whenever smaller than a prescribed tolerance, the matrices W and Z can be acceptable approximations of L^{-T} and U^{-1} though preserving a workable sparsity. The resulting preconditioner is denoted as AINV (Approximate INVerse) and has been introduced in the late 90s by Benzi and coauthors [122–124]. In the following years the native algorithm has been further developed by other researchers as well, for example, [125–130]. A sketch of the basic procedure to build the AINV preconditioner is provided in Algorithm 4.

Input: Matrix A, the tolerance τ
Output: Matrices W, Z and D such that $M^{-1} = ZD^{-1}W^T$
Set $\mathbf{p} = \mathbf{q} = \mathbf{0}$, $Z = W = I$
for $k = 1, \ldots, n$ **do**
  $p_k = \mathbf{w}_k^T A\mathbf{z}_k$, $q_k = \mathbf{w}_k^T A\mathbf{z}_k$
  for $i = k+1, \ldots, n$ **do**
    $p_i = (\mathbf{w}_i^T A\mathbf{z}_k)/p_k$, $q_i = (\mathbf{w}_k^T A\mathbf{z}_i)/q_k$
    $\mathbf{w}_i \leftarrow \mathbf{w}_i - p_i\mathbf{w}_k$, $\mathbf{z}_i \leftarrow \mathbf{z}_i - q_i\mathbf{z}_k$
    for $j = 1, \ldots, i$ **do**
      Drop the jth component of $\mathbf{w}_i$ and $\mathbf{z}_i$ if smaller than τ
    end
  end
end
for $k = 1, \ldots, n$ **do**
  $d_k = p_k$
end

Algorithm 4: Computation of the AINV preconditioner.

Quite obviously, if A is SPD, then $W = Z$, and in Algorithm 4 only the update of $\mathbf{z}_i$ must be carried out. The resulting AINV preconditioner can be written as $M^{-1} = ZZ^T$, is SPD, and exists if $p_k \neq 0$ for all $k = 1, \ldots, n$. Unfortunately, the same does not hold true for non-factored sparse approximate inverses based on minimizing the Frobenius norm in (4.3). In general, using an algorithm developed along the lines of SPAI there is no guarantee that M^{-1} is SPD if A is so. This is quite a strong limitation, as it prevents from using the powerful CG method to solve an SPD linear system. Such a weak point motivated the development of the Factorized Sparse Approximate Inverse (FSAI) preconditioner by Kolotilina and Yeremin [131]. The FSAI algorithm lies somewhat between the two classes of approximate inverses previously identified, as it produces a factored preconditioner in the form

$$M^{-1} = G^T G, \tag{4.14}$$

where G is a sparse approximation of the inverse of the exact Cholesky factor L of an SPD matrix A obtained with a Frobenius norm minimization procedure. Select a lower triangular nonzero pattern $\mathcal{S}_L$ and find the matrix $G \in \mathcal{W}_{SL}$, where $\mathcal{W}_{SL}$ is the set of matrices with pattern $\mathcal{S}_L$ such that

$$\|I - GL\|_F \longrightarrow \min. \tag{4.15}$$

The minimization (4.15) can be performed recalling that

$$\|I - GL\|_F = \mathrm{Tr}\left[(I - GL)(I - GL)^T\right] = n - 2\ \mathrm{Tr}(GL) + \ \mathrm{Tr}\left(GAG^T\right). \tag{4.16}$$

Deriving the right-hand side of (4.16) with respect to the nonzero entries of G and setting to 0 yield

$$[GA]_{ij} = l_{ji} \quad \forall (i,j) \in \mathcal{S}_L, \tag{4.17}$$

where the symbol $[\cdot]_{ij}$ denotes the entry in position (i, j) of the matrix in the brackets. As $\mathcal{S}_L$ is a lower triangular pattern, the entry l_{ji} of L is nonzero only if $i = j$ and the off-diagonal terms of L are not required. Scaling (4.17) so that the right-hand side is either 1 or 0 yields

$$\left[\widehat{G}A\right]_{ij} = \delta_{ij} \quad \forall (i, j) \in \mathcal{S}_L, \tag{4.18}$$

where $\widehat{G}$ is the scaled factor G and δ_{ij} the Kronecker delta. Solving (4.18) by rows leads to a sequence of dense SPD linear systems with size equal to the number of nonzeroes set for each row of $\widehat{G}$. More precisely, denoting by $\mathcal{J}$ the set of column indices where the ith row $\mathbf{g}_i$ of $\widehat{G}$ has a nonzero entry, the following SPD systems have to be solved:

$$A[\mathcal{J}, \mathcal{J}]\, \mathbf{g}_i[\mathcal{J}] = \mathbf{i}_m, \tag{4.19}$$

where $\mathrm{i}_m \in \mathbb{R}^m$ has zero components except the mth one that is unitary, and m is the cardinality of $\mathcal{J}$. The FSAI factor G is finally restored from $\widehat{G}$ as

$$G = \left[\mathrm{diag}\left(\widehat{G}\right)\right]^{-1/2} \widehat{G} \tag{4.20}$$

such that the preconditioned matrix GAG^T has all diagonal entries equal to 1. It has been demonstrated that FSAI is optimal in the sense that it minimizes the Kaporin conditioning number of GAG^T for any $G \in \mathcal{W}_{SL}$ [131, 132]. Another remarkable property of FSAI is that the minimization in (4.15) does not require the explicit knowledge of L, which would make the algorithm unfeasible. A sketch of the procedure to build FSAI is reported in Algorithm 5. The FSAI preconditioner was later improved in several ways [133], with the aim of increasing its efficiency by dropping the smallest entries while preserving its optimal properties. Between the late 90s and early 00s further developments of FSAI were abandoned, as AINV proved generally superior. In recent years, however, a renewed interest for FSAI has started, mainly because of its high intrinsic parallel degree, for example, [134–136].

The native FSAI algorithm has been developed for SPD matrices. A generalization to nonsymmetric matrices is possible and quite straightforward [137, 138]; however the resulting preconditioner is much less robust because the solvability of all local linear systems and the nonsingularity of the approximate inverse are no longer theoretically guaranteed. This is why with general sparse matrices SPAI and AINV are generally preferred.

4.1. Nonzero Pattern Selection and Stability Issues

A key factor for the efficiency of approximate inverse preconditioning based on the Frobenius norm minimization is the choice of the nonzero pattern $\mathcal{S}$ or $\mathcal{S}_L$. If the nonzero pattern is not good for the problem at hand, the result is generally a failure in the Krylov solver convergence. This is due to the fact that approximate inverse techniques assume that a sparse matrix M^{-1} can be a good approximation, in some sense, of A^{-1}. This is not obvious at all, as the inverse of a sparse matrix is structurally dense. If A is diagonally dominant, the entries of A^{-1} decay exponentially away from the main diagonal [139], hence a banded pattern for M^{-1} typically provides good results. Other favorable situations include block tridiagonal or

Input: Matrix A and the nonzero pattern $\mathcal{S}_L$
Output: Matrix G
for $i = 1, \ldots, n$ **do**
  Compute $\mathcal{J} = \{j : (i, j) \in \mathcal{S}_L\}$
  Set $m = |\mathcal{J}|$
  Gather $A[\mathcal{J}, \mathcal{J}]$ from A
  Solve $A[\mathcal{J}, \mathcal{J}]\mathbf{g}_i[\mathcal{J}] = \mathbf{i}_m$
  $\mathbf{g}_i \leftarrow \mathbf{g}_i / \sqrt{\mathbf{g}_{i,m}}$
end

Algorithm 5: Computation of the FSAI preconditioner.

pentadiagonal matrices [140], but for a general sparse matrix choosing a good nonzero pattern is not trivial at all.

With respect to the nonzero pattern selection, approximate inverses can be defined as *static* or *dynamic*, according to whether $\mathcal{S}$ and $\mathcal{S}_L$ are initially chosen and kept unvaried during the computation, or $\mathcal{S}$ and $\mathcal{S}_L$ are generated by an adaptive algorithm which selects the position of the nonzero entries in an a priori unpredictable way. Typically, dynamic approximate inverses are more powerful than static ones, but their implementation especially on parallel computers can be much more complicated. The most popular and common strategy to define a static nonzero pattern for M^{-1} is to take the nonzero pattern of a power κ of A as is justified in terms of the Neumann series expansion of A^{-1}. In real problems, however, it is uncommon to set a κ value larger than 2 or 3, as $\mathcal{S}$ can soon become too dense, so it is often preferable working with sparsified matrices. The sparsification of A^κ can be done by either a prefiltration, or a postfiltration strategy, or both. For example, Chow [141] suggests dropping the entries of A smaller than a threshold, thus obtaining the sparsified matrix $\widehat{A}$ and using the pattern of $\widehat{A}^\kappa$. An a posteriori sparsification strategy is advanced for the FSAI algorithm by Kolotilina and Yeremin [133] with the aim of reducing the cost for the preconditioner application by dropping its smallest entries. Such a dropping, however, is nontrivial and is performed so as to preserve the property that the FSAI preconditioned matrix has unitary diagonal entries, with a possibly nonnegligible computational cost. Anyway, pre- and/or postfiltration of a static approximate inverse is generally to be recommended, as the cost of the preconditioner application can be dramatically reduced with a relatively small increase of the iteration count, for example, [135, 136, 142]. By the way, this suggests that the pattern of A^κ is often much larger than what would be actually required to get a good preconditioner, the difficulty relying on the a priori recognition of the most significant entries.

Dynamic approximate inverses are based on adaptive strategies which start from a simple initial guess, for example, a diagonal pattern, and progressively enlarge it until a certain criterion is satisfied. For instance, a typical dynamic algorithm is the one proposed by Grote and Huckle [117] for the SPAI preconditioner; see Algorithm 3. More specifically, they suggest to collect the indices of the nonzero components of the residual vector in a new set $\mathcal{L}$ and to enlarge the current set $\mathcal{J}$ by adding the new column indices that appear in the rows with index in $\mathcal{L}$. An improvement can be obtained by retaining only a few new column indices, specifically the indices j such that the new residual norm:

$$\rho_j = \|\mathbf{r}_j\|_2^2 - \frac{\left(\mathbf{r}_j^T A \mathbf{e}_j\right)^2}{\|A\mathbf{e}_j\|_2^2}, \tag{4.21}$$

is larger than a user-specified tolerance. Then, the set $\mathcal{J}$ is enlarged correspondingly, and a new least square problem is solved. Other dynamic techniques for generating $\mathcal{S}$ have been advanced by Huckle for both nonsymmetric [143] and SPD matrices [134]. More recently, Jia and Zhu [121] have advanced a novel dynamic approximate inverse denoted as PSAI (Power Sparse Approximate Inverse), and Janna and Ferronato [144] have developed a new adaptive strategy for Block FSAI that has proved quite efficient also for the native FSAI algorithm and will be presented in more detail in the sequel.

In contrast with approximate inverses based on the Frobenius norm minimization, the AINV algorithm does not need the selection of the nonzero pattern, so it can be classified as a purely dynamic preconditioner. However, its computation turns out to be pretty similar to that of an incomplete factorization, see Algorithms 2 and 4, thus raising concerns on its numerical stability. In particular, due to incompleteness a zero pivot may result, causing a breakdown, or small pivots can trigger unstable behaviors. To avoid such difficulties a stabilized AINV preconditioner has been developed independently in [124, 145] for SPD problems. For generally nonsymmetric problems, especially with large off-diagonal entries, the numerical stability in the AINV computation may still be an issue.

Theoretically, for an arbitrary nonzero pattern $\mathcal{S}$ it is not guaranteed that M^{-1} is nonsingular. This could also happen for a few combinations of the user-specified tolerances set for generating dynamically the approximate inverse pattern. However, such an occurrence is really unlikely in practice, and typically approximate inverses prove to be very robust in problems arising from a variety of applications. According to the specific problem at hand, the choice of the most efficient type of approximate inverse may vary. For a thorough comparative study of the available options as of 1999 see the work by Benzi and Tůma [146]. Generally, SPAI is a robust algorithm able to converge also in difficult problems where incomplete factorizations fail, for example, [147], allowing the user to have a complete control of the quality of the preconditioner through the parameter ε, see Algorithm 3. However, its computation is quite expensive and can be rarely competitive with AINV if only one system has to be solved so that the construction time cannot be amortized by recycling a few times the same preconditioner. For SPD problems, SPAI is not an option and FSAI is always well defined and a very stable preconditioner, even if its convergence rate may be lower than AINV. For unsymmetric matrices, however, FSAI is not reliable and may often fail. On the other hand, AINV seldom fails and generally outperforms both SPAI and FSAI.

Numerical experiments in different problems typically show that, whenever a stable ILU-type preconditioner can be computed, that is probably the most efficient preconditioner on a scalar machine [129, 148, 149]. However, approximate inverses play an important role in offering a stable and robust alternative to ILU in ill-conditioned problems. Moreover, they can become the preconditioner of choice for parallel computations, as it will be discussed in more detail in the sequence.

5. Algebraic Multigrid Methods

It is well known that the convergence rate of any Krylov subspace method preconditioned by either an incomplete factorization or an approximate inverse tends to slow down as the problem size increases. The convergence deterioration along with the increased number of operations per iteration may lead to an unacceptably high computational cost, thus limiting de facto the size of the simulated model even though large memory resources are available. This is why in recent years much work has been devoted to the so-called *scalable* algorithms,

that is, solvers where the iteration count is insensitive to the characteristic size of the discretization mesh.

Multigrid methods can provide an answer to such a demand. Pioneering works on multigrid are due to Brandt [150, 151] who promoted these techniques for the solution of regularly discretized elliptic PDEs. The matrices arising from this kind of problem have a regular structure for which all eigenvalues and eigenvectors are known. For example, let us consider the 1D model problem:

$$-u''(x) = f(x), \quad x \in [0,1], \tag{5.1}$$

with homogeneous Dirichlet boundary conditions discretized by centered differences over $n+2$ equally spaced points x_i, $i = 0, \ldots, n+1$. The grid characteristic size is therefore $h = 1/(n+1)$. This discretization results in the tridiagonal $n \times n$ linear system $A_h \mathbf{x} = \mathbf{b}$, where, as is well known, the eigenvalues of A_h are

$$\lambda_k = 4 \sin^2\left(\frac{kh\pi}{2}\right) \quad k = 1, \ldots, n, \tag{5.2}$$

with the associated eigenvectors

$$\mathbf{w}_k = [\sin(kh\pi), \sin(2kh\pi), \ldots, \sin(nkh\pi)]^T. \tag{5.3}$$

In other words, the ith component of $\mathbf{w}_k$ is actually

$$\mathbf{w}_{k,i} = \sin(k\pi x_i). \tag{5.4}$$

Let us solve the discretized model problem by a stationary method, for example, the Jacobi iteration. As the main diagonal of A_h has all components equal to 2, the Jacobi iteration matrix S_h is just

$$S_h = I - \frac{A_h}{2} \tag{5.5}$$

and the kth component of the error along the direction of the kth eigenvector of S_h is damped at each iteration by the reduction coefficient η_k:

$$\eta_k = 1 - 2 \sin^2\left(\frac{kh\pi}{2}\right). \tag{5.6}$$

The reduction coefficients with k around $n/2$ are very small, and the related error components are damped very rapidly in a few iterations. This part of the error is often referred to as the *oscillatory* part corresponding to *high frequency modes*. However, if h is small, there are also several reduction coefficients close to 1, related to both the smallest and largest values of k, that slow down significantly the stationary iteration. This part of the error is usually referred to as the *smooth* part corresponding to *low frequency modes*. Let us now consider a coarser grid

where the characteristic size is doubled, that is, we simply keep the nodes of the initial discretization with an even index. Recalling (5.4) it follows that a smooth mode on the fine grid, that is, $k < n/4$ and $k > 3n/4$, is now automatically injected into an oscillatory mode on the coarse grid. Building A_{2h} and S_{2h}, it is now possible to damp in a few iterations the error on the coarse grid and then project back the result on the fine grid. Similar conclusions can be drawn also using other stationary iterations, such as Gauss-Seidel, though in a much less straightforward way.

Recalling the previous observations, the basic idea of multigrid proceeds as follows. The operator S_h, usually denoted as the *smoother* in the multigrid terminology, is applied to the fine grid discretization for ν_1 iterations. The resulting fine grid residual is computed and injected into the coarse grid through a *restriction* operator R_h^H. Recalling the relationship between error and residual, the coarse grid problem is solved using the residual as known term, thus providing the coarse grid error. Such an error is then projected back to the original fine grid using a *prolongation* operator P_H^h, and the fine grid solution, obtained after the initial ν_1 smoothing iterations, is corrected. Finally, the fine grid smoother is applied again until convergence. The algorithm described above is known as *two-grid cycle* and is the basis of multigrid techniques. Quite naturally, the two-grid cycle can be extended in a recursive way. At the coarse level, another two-grid cycle can be applied moving at a coarser grid, defining appropriate smoother, restriction, prolongation operators, and so on, until the coarsest problem is so small that it can be efficiently solved by a direct method. The recursive application of the two-grid cycle gives rise to the so-called *V-cycle*. Other more complex recursions can provide the so-called *W-cycle*; see, for instance, [152] for details.

Multigrid methods have been introduced as a solver for discretized PDEs of elliptic type, and indeed in such problems they have soon proved to be largely superior to existing algorithms. The first idea to extend their use to other applications was to look at the multigrid as a purely algebraic solver, where one has to define the smoother, restriction, and prolongation operators knowing the system matrix A only independently of the grid and the discretized PDE it actually comes from. This strategy, known as Algebraic Multigrid (AMG), was first introduced in the mid 80s [153–155] and became soon quite popular. The second idea was to regard AMG not as a competitor with Krylov subspace solvers, rather as a potentially effective preconditioner. In fact, to work as a preconditioner it is simply sufficient to fix the number ν_2 of smoothing iterations needed to reconstruct the system solution at the finest level, thus obtaining an inexact solution. The scheme of a V-cycle AMG application as a preconditioner is provided in Algorithm 6.

Basically, AMG preconditioners vary according to the choice of both the restriction operator and the smoother, while the prolongation operator is often defined as the transposed of the restrictor. The basic idea for defining a restriction is to coarsen the native pattern of A and prescribe an appropriate interpolation over the coarse nodes. Classical coarsening strategies have been introduced in [155, 156] which separate the variables into coarse (C-) and fine (F-) points according to the *strength* of each matrix connection. A point j strongly influences the point i if

$$-a_{ij} \geq \theta \max_{k=1,n,k\neq i} (-a_{ik}), \tag{5.7}$$

where θ is the user-defined *strength threshold*. For each point j that strongly influences an F-point i, j is either a C-point or is strongly influenced by another C-point k. Moreover, two C-points should not be connected to each other. An alternative successful coarsening strategy

Input: Matrix $A^{(k)}$, the known vector $\mathbf{v}^{(k)}$, the restriction and prolongation operators $R_h^{H\ (k)}$ and $P_H^{h\ (k)}$, the smoothers $S_h^{(k)}$, k, ν_1, ν_2 and μ
Output: Vector $\mathbf{u}^{(k)} = M^{-1}\mathbf{v}^{(k)}$
if $k = \mu$ **then**
  Solve $A^{(k)}\mathbf{u}^{(k)} = \mathbf{v}^{(k)}$
else
  Apply ν_1 iterations of the smoother $S_h^{(k)}$ to $A^{(k)}\mathbf{u}^{(k)} = \mathbf{v}^{(k)}$
  $\mathbf{r}^{(k)} = \mathbf{v}^{(k)} - A^{(k)}\mathbf{u}^{(k)}$
  $\mathbf{r}^{(k+1)} = R_h^{H\ (k)}\mathbf{r}^{(k)}$
  Call MGV$(A^{(k+1)}, R_h^{H\ (k+1)}, P_H^{h\ (k+1)}, S_h^{(k+1)}, \mathbf{r}^{(k+1)}, \mathbf{e}^{(k+1)})$
  $\mathbf{e}^{(k)} = P_H^{h\ (k)}\mathbf{e}^{(k+1)}$
  $\mathbf{u}^{(k)} \leftarrow \mathbf{u}^{(k)} + \mathbf{e}^{(k)}$
  Apply ν_2 iterations of the smoother $S_h^{(k)}$ to $A^{(k)}\mathbf{u}^{(k)} = \mathbf{v}^{(k)}$
end

Algorithm 6: Recursive application for μ times of the V-cycle AMG function MGV$(A, R_h^H, P_H^h, S_h, \mathbf{v}, \mathbf{u})$.

is based on the so-called *smoothed aggregation* (SA) technique [157], where a coarse node is defined by an aggregate of a root point i and all the neighboring nodes j such that

$$|a_{ij}| > \theta\sqrt{|a_{ii}a_{jj}|}. \tag{5.8}$$

Another possibility is to use a variant of the independent set ordering (ISO) technique which subdivides a matrix into a 2×2 block structure and projects the native system into the Schur complement space. Using ISO or SA can lead to an elegant relationship between AMG and multilevel ILU as exploited in [98, 101, 158, 159]. More recently, a number of both parallel and scalar coarsening strategies have been also advanced, for example, [160–164]. As far as the smoother is concerned, for many years the method of choice has been the standard Gauss-Seidel iteration. With the development of parallel computers other smoothers have been introduced with the aim of increasing the parallel degree of the resulting AMG algorithm. Multicoloring approaches [165] can be used in connection to standard Gauss-Seidel, or also polynomial and C-F Jacobi smoothers [166, 167]. It is no surprise that also approximate inverses can be efficiently used as a smoother, for example, [168, 169].

The last decade has witnessed an explosion of research on AMG techniques. The key factor leading to such a great interest is basically their potential scalability with the size of the problem to be solved, in the sense that the iteration count to converge for a given problem does not depend on the number of the mesh nodes. Several theoretical results have been obtained with the aim of generalizing as much as possible AMG to nonelliptic problems, for example, [158, 170–174]. AMG techniques have been used in several different applications with promising results, such as fluid-structure interactions, meshless methods, Maxwell and Helmoltz equations, structural mechanics, sedimentary basin reconstruction, and advection-convection and reactive systems, for example, [175–185]. The above list of references is just representative and of course largely incomplete. Nonetheless, much work is still to be done in order to make AMG the method of choice. In particular, difficulties can be experienced where the system matrix arises from highly distorted and irregular meshes, or in presence of strong heterogeneities. In these cases, even the most advanced AMG strategies can fail.

6. Parallel Preconditioners

Parallel computing is widely accepted as the only pathway toward the possibility of managing millions of unknowns [186]. At the same time, hardware companies are improving the available computational resources with an increasing degree of parallelism rather than accelerating each single computing core. This is why the interest in parallel computations is continuously growing in recent years. Krylov subspace methods are in principle almost ideally parallel, as their kernels are matrix-vector and scalar products along with vector updates. Unfortunately, the same is not true for most algebraic preconditioners. There are basically two approaches for developing parallel preconditioners: the first tries to extract as much parallelism as possible from existing algorithms with the aim of transferring them to high-performance platforms, while the second is based on developing novel techniques which would not make sense on a scalar computer. Quite obviously, the former approach is easier to understand from a conceptual point of view, as the native algorithm is not modified and the difficulties are mainly technological. The latter implies an additional effort to develop new explicitly parallel methods.

ILU-based preconditioners are highly sequential in both their computation and application. Anyway, some degree of parallelism can be achieved through graph coloring techniques. These strategies have been known from a long time by numerical analysts, for example, [6], and used in the context of relaxation methods. The aim is to build a partition of the matrix graph such that all vertices in the same group form an independent set, so that all unknowns in the same subset can be solved in parallel. Other useful orderings are based on domain decomposition strategies minimizing the edges connecting different subdomains, with a local ordering applied also to each unknown subset. Parallel ILU implementations based on these concepts have been developed in [187–190]; however it was soon made clear that generally such algorithms could not have a promising scalability.

In contrast, approximate inverses are intrinsically much more parallel than ILU factorizations, as they can be applied by a matrix-vector product instead of forward and backward substitutions. The construction of an approximate inverse, however, can be difficult to parallelize. For example, the AINV Algorithm 4 proceeds by columns and is conceptually pretty similar to an ILU factorization. A parallel AINV construction can be obtained by means of graph partitioning techniques which decompose the adjacency graph of A in a number of subsets of roughly the same size with a minimal number of edge cuts [191]. In such a way the inner variables of each subset can be managed independently by each processor, while communications occur only with the boundary variables. By distinction, the construction of an approximate inverse based on a Frobenius norm minimization is naturally parallel if a static nonzero pattern is defined a priori. For example, the computation of both static SPAI and FSAI consists of a number of independent least-square problems and dense linear systems, respectively, which can be trivially partitioned among the available processors. Hence, static SPAI and FSAI are almost perfectly parallel algorithms. However, when the nonzero pattern is defined dynamically parallelization is no longer straightforward, as the operation of gathering the required submatrix of A, see Algorithms 3 and 5, gives rise to unpredictable communications. For example, details of the nontrivial parallel implementation of the dynamic SPAI algorithm are provided in [147].

The theoretical scalability properties of AMG make it very attractive for parallel computations. This is why in the last few years the research on AMG techniques has concentrated on high-performance massively parallel implementations. The main difficulties may arise in parallelizing the Gauss-Seidel smoother and the coarsening stage. As mentioned before, these problems can be overcome by using naturally parallel smoothers, such as Jacobi, relaxed

Jacobi, polynomial or a static approximate inverse smoother, for example, [162, 166], and parallel coarsening strategies, for example, [160, 162, 192, 193]. Currently, efficient parallel implementations of AMG are already available in several software packages, for example, Hypre [194] and Trilinos [195].

An alternative strategy for developing parallel preconditioners relies on building new algorithms which should consist of matrix-vector products and local triangular solves only. Perhaps the earliest technique of this kind belongs to the class of the polynomial preconditioners which was first introduced by Cesari as back as in 1937 [4], even though obviously not in the context of Krylov subspace methods and supercomputing. Modern polynomial preconditioners have been developed in [196–198]. A polynomial preconditioner is defined as

$$M^{-1} = s(A), \tag{6.1}$$

where $s \in \mathbb{P}_k$ is a polynomial of degree not exceeding k. The preconditioned system becomes:

$$s(A)A\mathbf{x} = s(A)\mathbf{b}, \tag{6.2}$$

where $s(A)$ and A commute, so that right and left preconditionings coincide. Quite obviously, $s(A)A$ is never formed explicitly and its application to a vector is simply performed by a sequence of matrix-vector products. Therefore, polynomial preconditioners can be ideally implemented in a parallel environment. There are different polynomial preconditioners according to the choice of $s(A)$. One of the simplest options, advanced in [196], is using the Neumann expansion $I + N + N^2 + \cdots + N^l$, where

$$N = I - \omega A \tag{6.3}$$

and ω is a scaling parameter. The degree $l \leq k$ cannot be too large, and this can limit the effectiveness of this preconditioner. Another option, advanced in [197], is to select a somewhat "optimal" polynomial, where optimality is evaluated in some sense. A natural choice is prescribing that the eigenspectrum of the preconditioned matrix is as close as possible to that of the identity, that is, finding $s \in \mathbb{P}_k$ such that

$$\max_{\lambda \in \sigma(A)} |1 - \lambda s(\lambda)| \longrightarrow \min, \tag{6.4}$$

where $\sigma(A)$ is the eigenspectrum of A. The min-max problem above can be addressed by using Chebyshev polynomials but cannot be solved exactly, as $\sigma(A)$ is generally not known. An approximate minimization can be done replacing $\sigma(A)$ with a continuous set which encloses $\sigma(A)$. A third option is to compute s as the polynomial that minimizes

$$\|1 - \lambda s(\lambda)\|_w, \tag{6.5}$$

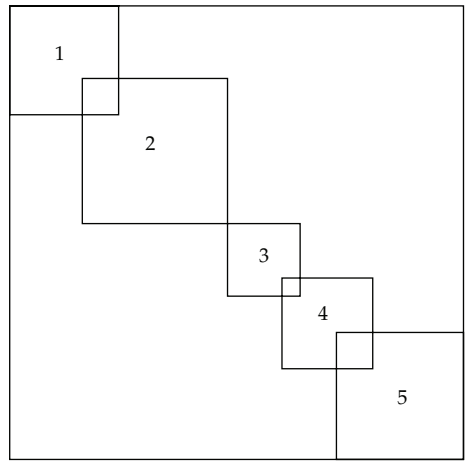

Figure 3: Matrix partitioning into possibly overlapping subdomains.

where the symbol $\|\cdot\|_w$ denotes the 2-norm induced by the inner product on the space $\mathbb{P}_k$ with respect to some nonnegative weight function defined in the interval $[\alpha, \beta]$:

$$(p, q) = \int_{\alpha}^{\beta} p(\lambda)\, q(\lambda)\, w(\lambda)\, d\lambda. \tag{6.6}$$

The resulting polynomial is known as *least-squares* and was actually introduced by Stiefel in the solution of eigenproblems [199]. According to Saad [198], least-squares polynomials tend to perform better than Chebyshev and Neumann. However, a major drawback for the use of this kind of preconditioners stems from the need at least for estimates of the smallest and largest eigenvalues of A and often their performance is quite poor in terms of convergence rate [200].

Another popular parallel preconditioner is based on the Additive Schwarz (AS) methods [201]. The idea is to use a domain decomposition approach for dividing the variables into a number of possibly overlapping subdomains and to address separately each subdomain at a local level (Figure 3). On the overlaps the contribution of each subdomain is added or treated in some special way by averaging techniques. AS preconditioners can be easily parallelized by assigning each subdomain to a single processor. For instance, a sequential ILU factorization can be computed for each subdomain, thus miming a kind of parallel ILU. If no overlaps are allowed for, AS preconditioners with inner ILU factorizations simply reduce to an incomplete Block Jacobi. This class of preconditioners has experienced some success in the 90s as it is conceptually perfectly parallel. Unfortunately, the quality of the preconditioner deteriorates as the number of processors, that is, subdomains, grows, because the size of each subset progressively decreases. This causes an increase of the iteration count which can also completely offset the advantage of using a larger number of processors.

A novel recent parallel preconditioner which tries to combine the positive features of both approximate inverses based on the Frobenius norm minimization and AS methods is Block FSAI [202]. The basic idea exploits the fact that in the Frobenius norm minimization

the preconditioned matrix can be actually forced to resemble any *target* matrix T, by computing a pseudoapproximate inverse M^{-1} such that

$$\left\| T - AM^{-1} \right\|_F \longrightarrow \min \tag{6.7}$$

for all matrices having a prescribed nonzero pattern $\mathcal{S}$. This concept, originally introduced in [120], can be used to choose a target matrix T that is particularly attractive for parallel computing. The idea of Janna et al. [202] is to generalize the FSAI approach using as a target a block diagonal matrix. As FSAI is a robust preconditioner for SPD problems, Block FSAI has been originally developed for this class of matrices, although a generalization to nonsymmetric indefinite linear systems has also been attempted [203]. So let A be SPD with $\mathcal{S}_L$ and $\mathcal{S}_{BD}$ a sparse lower triangular and a dense block diagonal nonzero $n \times n$ pattern, respectively. Denote by n_b the number of diagonal blocks and m_{i_b} the size of the i_bth block, and let D be an arbitrary full-rank matrix with nonzero pattern $\mathcal{S}_{BD}$. The Block FSAI preconditioner of A is defined as the product $F^T F$, where F is a lower block triangular factor with nonzero pattern $\mathcal{S}_{BL} = \mathcal{S}_{BD} \cup \mathcal{S}_L$ such that

$$\|D - FL\|_F \longrightarrow \min \tag{6.8}$$

with L the exact lower Cholesky factor of A. As D is arbitrary, it goes without saying that F as defined in (6.8) is also. Similarly to the FSAI algorithm, minimization of (6.8) yields a linear relationship for the ith row $\mathbf{f}_i$ of F:

$$A[\mathcal{J}, \mathcal{J}]\mathbf{f}_i[\mathcal{J}] = \mathbf{v}, \tag{6.9}$$

where $\mathcal{J}$ is the set of column indices in the ith row of F with nonzero entries and $\mathbf{v}$ is the null vector except for the last m_{i_b} components which are arbitrary, being i_b the block index the row i belongs to (Figure 4). For the system (6.9) to have a unique solution m_{i_b} components of $\mathbf{f}_i[\mathcal{J}]$ can be set arbitrarily because of the arbitrariness of D. According to [202] the most convenient choice is to prescribe all the components of $\mathbf{f}_i[\mathcal{J}]$ falling within the i_bth diagonal block to be null, with the exception of the diagonal entries set to 1. This implies that F is actually a unit lower triangular matrix with structure

$$F = \begin{bmatrix} I & 0 & \cdots & 0 \\ F_{21} & I & & \vdots \\ \vdots & & \ddots & \vdots \\ F_{n_b 1} & F_{n_b 2} & \cdots & I \end{bmatrix}. \tag{6.10}$$

Similarly to the static FSAI algorithm, the factor F can be efficiently built in parallel as all systems (6.9) are independent. According to (6.8), the preconditioned matrix FAF^T should

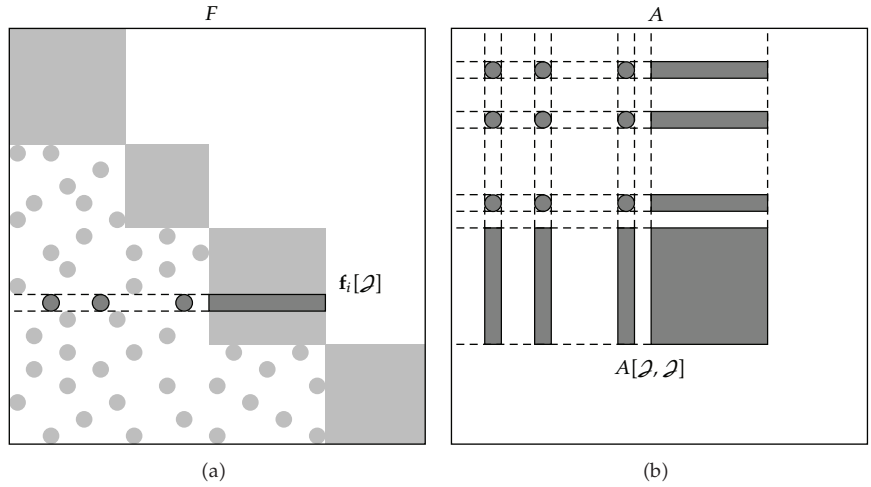

Figure 4: Elements belonging to $\mathbf{f}_i[\mathcal{J}]$ (a) and $A[\mathcal{J},\mathcal{J}]$ (b).

resemble DD^T, that is, a block diagonal matrix for any arbitrary D. In other words, FAF^T has the block structure

$$FAF^T = \begin{bmatrix} B_1 & R_{1,2} & \cdots & R_{1n_b} \\ R_{12}^T & B_2 & \cdots & R_{2n_b} \\ \vdots & & \ddots & \vdots \\ R_{1n_b}^T & R_{2n_b}^T & \cdots & B_{n_b} \end{bmatrix}, \tag{6.11}$$

where R_{ij} are residual blocks and B_{i_b} tend to the diagonal blocks of DD^T. As D is arbitrary, there is no reason for FAF^T to be better than A in a CG iteration, so it is necessary to precondition FAF^T again. Assuming that the residual off-diagonal blocks in (6.11) are close to 0, an effective choice could be using as "outer" preconditioner a nonoverlapped AS method containing a local preconditioner for each block B_{i_b}. Writing the "outer" preconditioner in the factored block diagonal form $J^{-T}J^{-1}$, the resulting preconditioned matrix reads

$$J^{-1}FAF^TJ^{-T} = WAW^T \tag{6.12}$$

with the final Block FSAI preconditioner:

$$M^{-1} = W^TW = F^TJ^{-T}J^{-1}F. \tag{6.13}$$

The Block FSAI preconditioner described above can be improved by defining $\mathcal{S}_{BL}$ dynamically. An adaptive algorithm for the $\mathcal{S}_{BL}$ pattern identification can be implemented starting from the theoretical optimal properties of Block FSAI. Janna and Ferronato [144] have demonstrated that under the hypothesis that the $J^{-T}J^{-1}$ contains the exact inverse of each

diagonal block B_{i_b} the Kaporin conditioning number β of the preconditioned matrix WAW^T [132]:

$$\beta\left(WAW^T\right) = \frac{\operatorname{tr}\left(WAW^T\right)}{n \det\left(WAW^T\right)^{1/n}}, \tag{6.14}$$

satisfies the following bound:

$$1 \leq \beta\left(WAW^T\right) \leq C\psi(F), \tag{6.15}$$

where C is a constant scalar independent of W and

$$\psi(F) = \left(\prod_{i=1}^{n}\left[FAF^T\right]_{ii}\right)^{1/n}. \tag{6.16}$$

Block FSAI has the theoretical property of minimizing $\psi(F)$ for any given pattern $\mathcal{S}_{BL}$. The basic idea of the adaptive algorithm is to select the off-block diagonal nonzero entries in any row $\mathbf{f}_i$ of F so as to reduce $\psi(F)$ as much as possible. This is feasible because each factor $[FAF^T]_{ii}$ turns out to be a quadratic form depending on $\mathbf{f}_i$ only. Denoting by $\widetilde{\mathbf{f}}_i$ the subvector of $\mathbf{f}_i$ including the off-block diagonal entries only, $\widetilde{A}_{i_b}$ the square submatrix of A built from the first to the mth row/column, with m the sum of size of the first $(i_b - 1)$ diagonal blocks of $\mathcal{S}_{BD}$, and $\mathbf{a}_i$ the subrow of A with the first m elements of the ith row (Figure 5), each factor $[FAF^T]_{ii}$ in (6.16) reads

$$\left[FAF^T\right]_{ii} = \widetilde{\mathbf{f}}_i^T \widetilde{A}_{i_b} \widetilde{\mathbf{f}}_i + 2\widetilde{\mathbf{f}}_i^T \widetilde{\mathbf{a}}_i + a_{ii}. \tag{6.17}$$

Minimizing every $[FAF^T]_{ii}$, $i = 1, \ldots, n$, is equivalent to minimizing $\psi(F)$. The adaptive pattern search for F can be therefore implemented as follows. Start from an initial guess $\mathcal{S}_F^0$ for the nonzero pattern of the off-block diagonal part of F, for example, the empty pattern, and compute the gradient $\mathbf{g}_i$ of each quadratic form $[FAF^T]_{ii}$:

$$\mathbf{g}_i = 2\left(\widetilde{A}_{i_b} \widetilde{\mathbf{f}}_i + \widetilde{\mathbf{a}}_i\right). \tag{6.18}$$

Then, add to $\mathcal{S}_F^0$ the position j into each row i corresponding to the largest component of $\mathbf{g}_i$ computed in (6.18), so as to obtain an augmented pattern $\mathcal{S}_F^1$. After computing the new factor F over $\mathcal{S}_F^1$, the procedure above can be iterated in order to build $\mathcal{S}_F^2$, $\mathcal{S}_F^3$, and so on. The search into each row can be stopped when either a maximum number $k_{\max}$ of entries are added to the initial pattern or the relative variation of $[FAF^T]_{ii}$ in two consecutive steps is smaller than a prescribed tolerance ϵ_F. The construction of the dynamic Block FSAI is summarized in Algorithm 7. The Block FSAI computed with the dynamic algorithm described above is known as Adaptive Block FSAI (ABF) and has proved very effective even if the outer preconditioner does not contain the exact inverse of the diagonal blocks B_{i_b}. As any dynamic algorithm, the parallel construction of ABF is not straightforward. However, an almost ideal

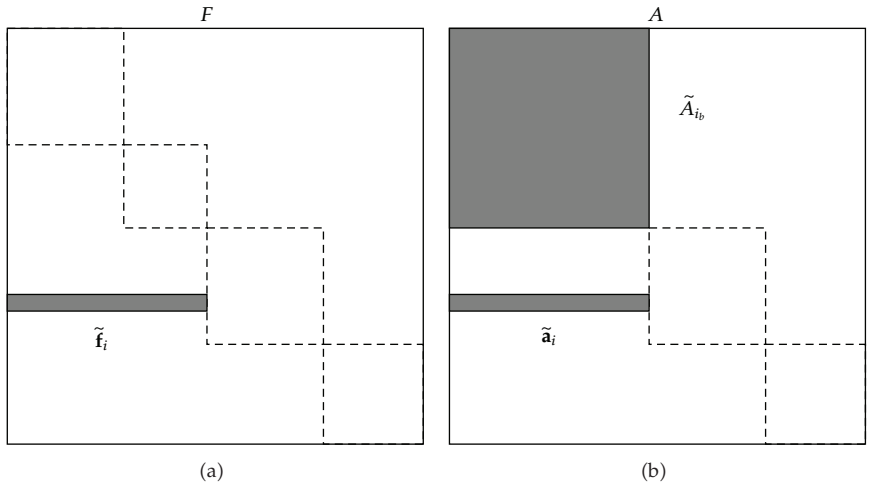

Figure 5: Matrix $\widetilde{A}_{ib}$ and vectors $\widetilde{\mathbf{f}}_i$ and $\widetilde{\mathbf{a}}_i$ used in the $[FAF^T]_{ii}$ and $\mathbf{g}_i$ computation.

Input: Matrix A, the initial pattern $\mathcal{S}_F^0$
Output: Matrix F
for $i = 1, \ldots, n$ **do**
  Extract $\mathcal{J}^0$ from $\mathcal{S}_F^0$
  if $\mathcal{J}^0 \neq \emptyset$ **then**
    Gather $A[\mathcal{J}^0, \mathcal{J}^0]$ and $A[\mathcal{J}^0, i]$ from A
    Solve $A[\mathcal{J}^0, \mathcal{J}^0]\ \ \mathbf{f}_i[\mathcal{J}^0] = -A[\mathcal{J}^0, i]$
  end
  Set $k = 0$ and $\epsilon_i > \epsilon_F$
  while $(k \leq k_{\max})$ and $(\epsilon_i < \epsilon_F)$ **do**
    $k \leftarrow k + 1$
    Compute $\mathbf{g}_i = 2(\widetilde{A}_{i_b}\widetilde{\mathbf{f}}_i + \widetilde{\mathbf{a}}_i)$
    Update $\mathcal{J}^{k-1}$ by adding the index of the largest component of $\mathbf{g}_i$
    Gather $A[\mathcal{J}^k, \mathcal{J}^k]$ and $A[\mathcal{J}^k, i]$ from A
    Solve $A[\mathcal{J}^k, \mathcal{J}^k]\ \ \mathbf{f}_i[\mathcal{J}^k] = -A[\mathcal{J}^k, i]$
    Compute the diagonal term $[FAF^T]_{ii}^k$ at the current step
    Compute $\epsilon_i = ([FAF^T]_{ii}^k - [FAF^T]_{ii}^{k-1})/[FAF^T]_{ii}^{k-1}$
  end
end

Algorithm 7: Dynamic construction of the Block FSAI preconditioner.

efficiency can be obtained in massively parallel computations as well. For more details, see [144].

Block FSAI coupled with a block diagonal IC factorization, for example, as implemented in [74, 84, 86, 90], has proved to be a very efficient parallel preconditioner for both SPD linear systems [144, 202] and eigenproblems [204, 205] arising from different areas, such as fluid dynamics, structural mechanics, and contact problems. The combination of FSAI with block diagonal IC joins the parallelization degree of the former with the effectiveness of the latter. Similarly to standard incomplete Block Jacobi algorithms, the iteration count tends to

grow increasing the number of computing cores. However, the convergence deterioration is much smoothed by the Block FSAI effect which always outperforms the standard FSAI algorithm. To improve the preconditioner scalability for massively parallel simulations the use of graph partitioning and/or domain decomposition strategies can be of great help [206, 207].

7. Saddle-Point Problems

In recent years the interest in the solution of saddle-point problems has grown, with many contributions from different fields. The main reason is the great variety of applications requiring the solution of linear systems with a saddle-point matrix. For example, such problems arise in the discretization of compressible and incompressible Navier-Stokes equations in computational fluid dynamics [208, 209], constrained optimization [210, 211], electromagnetism [212, 213], mixed FE approximations in fluid and solid mechanics [208, 214], and coupled consolidation problems in geomechanics [58].

A saddle-point problem is defined as a linear system where the matrix A has the structure

$$A = \begin{bmatrix} K & B_1 \\ B_2 & -C \end{bmatrix}, \tag{7.1}$$

where $K \in \mathbb{R}^{n_1 \times n_1}$, $B_1 \in \mathbb{R}^{n_1 \times n_2}$, $B_2 \in \mathbb{R}^{n_2 \times n_1}$, $C \in \mathbb{R}^{n_2 \times n_2}$, and K, B_1, and B_2 are nonzero. Typically, but not necessarily, $n_1 \geq n_2$. Moreover, C is symmetric and positive semidefinite, and often $C = 0$, while in most applications K is SPD and $B_2 = B_1^T$, that is, A is symmetric indefinite. For example, a saddle-point matrix A arises naturally in an equality-constrained quadratic problem:

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{f}^T\mathbf{x} \longrightarrow \min \quad \text{subject to } B\mathbf{x} = \mathbf{g} \tag{7.2}$$

with $C = 0$ and $B_2 = B_1^T = B$, solved with the aid of Lagrangian multipliers. A nonzero C may arise in some interior point methods [210, 211], compressible Navier-Stokes equations [215], or coupled consolidation problems [65]. Frequently the norm of C is much smaller than that of K. Because of the great variety of applications leading to saddle-point matrices, it is no surprise that several different solution methods have been developed. For an exhaustive, review, see Benzi et al. [216]. In the context of the present paper, the attention will be obviously restricted to preconditioned iterative methods.

A natural way to solve the problem

$$\begin{bmatrix} K & B^T \\ B & -C \end{bmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \tag{7.3}$$

is to reduce the system (7.3) to the Schur complement form. Solve for $\mathbf{x}_1$ in the upper set of equations:

$$\mathbf{x}_1 = K^{-1}\left(\mathbf{b}_1 - B^T\mathbf{x}_2\right), \tag{7.4}$$

and substitute (7.4) in the lower set:

$$\left(BK^{-1}B^T + C\right)\mathbf{x}_2 = BK^{-1}\mathbf{b}_1 - \mathbf{b}_2. \tag{7.5}$$

The matrix $S = BK^{-1}B^T + C$ at the left-hand side of (7.5) is the Schur complement of system (7.3). Hence, the global solution $\mathbf{x}$ can be obtained by solving the Schur complement system (7.5) and substituting the result back in (7.4) which implies solving another system with the matrix K. This pair of SPD linear systems can be elegantly solved by partitioned schemes, as those proposed in [217, 218] in the context of coupled diffusion and consolidation problems, resulting in a double CG iteration where two distinct preconditioners for K and S are needed. A usually more efficient alternative is to solve the systems (7.4) and (7.5) inexactly and use the procedure described above as a preconditioner in a Krylov subspace method. This is what is traditionally referred to as *constraint preconditioning* because the idea was first introduced in the solution of equality-constrained optimization problems such as (7.2) [219, 220].

Quite obviously, there are several different constraint preconditioners according to how K^{-1} and S^{-1} in (7.4) and (7.5) are approximated. The most effective choice turns out to be strongly dependent on the specific problem at hand. A popular strategy consists of replacing K^{-1} with M_K^{-1} and solving exactly the Schur complement system (7.5) [219–222]. Such an approach can be convenient if n_2 is much smaller than n_1, so that the system (7.5) can be efficiently solved by a direct method. Otherwise, an inner CG iteration can be set up exiting the procedure even at a relatively large residual [130]. In this case the resulting preconditioner reads

$$M_{\mathrm{ECP}}^{-1} = \begin{bmatrix} M_K & B^T \\ B & -C \end{bmatrix}^{-1} \tag{7.6}$$

and is usually denoted as *Exact Constraint Preconditioner* (ECP). ECP has several interesting theoretical properties which have been studied in detail by several authors. For instance, an exhaustive eigenanalysis of the preconditioned matrix can be found in [223]. In particular, denoting with α_K and β_K the smallest and the largest eigenvalue, respectively, of $M_K^{-1}K$, Durazzi and Ruggiero [224] have shown that the eigenvalues λ of $M_{\mathrm{ECP}}^{-1}A$ are either one, with multiplicity at least n_2, or real positive and bounded by $\alpha_K \leq \lambda \leq \beta_K$. Quite interestingly, this result can lead the classical CG algorithm to converge even with the indefinite matrix (7.3), provided that the last n_2 components of the initial residual are zero. This can be simply accomplished by choosing as initial guess $M_{\mathrm{ECP}}^{-1}\mathbf{b}$.

Despite its remarkable properties, ECP has at least two severe limitations. First, M_K^{-1} must be known explicitly to allow for the computation of S and must be very sparse in order to preserve a workable sparsity for S as well. Second, although an accurate solution of (7.5) is not really needed, solving it at each preconditioner application can represent a significant computational burden for the overall scheme. This is why ECP is often used with $M_K^{-1} = [\mathrm{diag}(K)]^{-1}$, which can be quite detrimental for the solver convergence in case K is not diagonally dominant or is ill-conditioned, such as in coupled consolidation problems [130].

To make the preconditioner application cheaper, a substitute for (7.5) can be solved replacing S with another approximation M_S as easy to invert as possible. Moreover, in some cases it may also be possible to approximate S without computing it explicitly, thus further reducing the constraint preconditioner cost. Obviously the theoretical properties of ECP no

longer hold true. Recalling the factorization (2.12) by Sylvester's theorem of inertia, the new preconditioner is

$$\begin{aligned} M_{\mathrm{ICP}}^{-1} &= \begin{bmatrix} I & -M_K^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} M_K^{-1} & 0 \\ 0 & -M_S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -BM_K^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} M_K & B^T \\ B & BM_K^{-1}B^T - M_S \end{bmatrix}^{-1} \end{aligned} \tag{7.7}$$

and is denoted as *Inexact Constraint Preconditioner* (ICP). This class of preconditioners has been introduced in [130, 225, 226] using $[\mathrm{diag}(K)]^{-1}$ or AINV for M_K^{-1} and a zero fill-in IC factorization for M_S^{-1}. The choice for M_S^{-1} is justified by the fact that usually S is well conditioned. Moreover, this allows for splitting the central block diagonal factor in (7.7) into the product of two matrices, thus providing a factorized form for M_{ICP}^{-1}. A block triangular variant of ICP can be easily deduced from (7.7) by neglecting either the upper or the lower outer factors:

$$M_{\mathrm{TICP}}^{-1} = \begin{bmatrix} I & -M_K^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} M_K^{-1} & 0 \\ 0 & -M_S^{-1} \end{bmatrix} = \begin{bmatrix} M_k & B^T \\ 0 & -M_S \end{bmatrix}^{-1}. \tag{7.8}$$

The theoretical properties of block triangular ICP have been investigated by Simoncini [227] and effectively used in the solution of the discretized Navier-Stokes equations [228–230].

Similar to ECP, ICP as well requires an explicit approximation M_K^{-1} to build the Schur complement. This is still a limitation in some problems, such as coupled consolidation [130], where even AINV can be a poor approximation of K^{-1} leading to an unacceptably high iteration count to converge. Actually, this is only partially connected with the quality of AINV itself, rather it is the need for the explicit construction of the Schur complement matrix that calls for a reduced fill-in of M_K^{-1}. In order to remove this inconvenience an implicit approximation of K^{-1} can be used, such as a stabilized IC factorization with partial fill-in:

$$M_K^{-1} = \tilde{L}_K^{-T}\tilde{L}_K^{-1}. \tag{7.9}$$

The new Schur complement reads

$$S = B\tilde{L}_K^{-T}\tilde{L}_K^{-1}B^T + C \tag{7.10}$$

but cannot be computed explicitly. Although the S implicit form (7.10) can still be used to perform a product between S and a vector, and consequently one may think of solving the related inner system by an iterative method, a suitable preconditioner for S is not easily available. Alternatively, an explicit approximation of S can be computed, for example, using the AINV of K^{-1}, namely

$$\widehat{S} = BZZ^TB^T + C, \tag{7.11}$$

where Z is the upper triangular factor resulting from the AINV Algorithm 4, and then an IC factorization of $\widehat{S}$ is performed:

$$M_S^{-1} = \widetilde{L}_S^{-T}\widetilde{L}_S^{-1}. \tag{7.12}$$

In this way the quality of M_K^{-1} is improved, but generally an additional approximation in M_S^{-1} is introduced. However, in several problems, especially those arising from coupled consolidation where such a constraint preconditioning variant has been introduced [148, 231], the overall algorithm has a beneficial effect. This ICP variant blending two different approximations for K^{-1} in the same scheme reads

$$\begin{aligned} M_{\mathrm{MCP}}^{-1} &= \begin{bmatrix} I & -\widetilde{L}_K^{-T}\widetilde{L}_K^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} \widetilde{L}_K^{-T}\widetilde{L}_K^{-1} & 0 \\ 0 & -\widetilde{L}_S^{-T}\widetilde{L}_S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -B\widetilde{L}_K^{-T}\widetilde{L}_K^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{L}_K^{-T} & -\widetilde{L}_K^{-T}G^T \\ 0 & \widetilde{L}_S^{-T} \end{bmatrix} \begin{bmatrix} \widetilde{L}_K^{-1} & 0 \\ G\widetilde{L}_K^{-1} & -\widetilde{L}_S^{-1} \end{bmatrix}, \end{aligned} \tag{7.13}$$

where $G = \widetilde{L}_S^{-1}B\widetilde{L}_K^{-T}$, and is denoted as *Mixed Constraint Preconditioner* (MCP). Recently an improved MCP version has been advanced by properly relaxing the Schur complement approximation [61].

It is quite well accepted that constraint preconditioners outperform any other "global" preconditioning approach applied to saddle-point matrices. This remarkable property has attracted the theoretical interest of several researchers who have investigated the spectral properties of constraint-preconditioned saddle-point matrices [225, 227, 231–233], obtaining quite accurate bounds for the eigenvalues of $M^{-1}A$. For instance, in the case of MCP in (7.13) Ferronato et al. [231] have shown that the eigenvalues λ of M_{MCP}^{-1} satisfy the following inequality:

$$|\lambda - 1| \le e_k\left(1 + g^2\right) + e_s, \tag{7.14}$$

where

$$\begin{aligned} e_k &= \left\| I - \widetilde{L}_K^{-1}K\widetilde{L}_K^{-T} \right\|, \\ e_s &= \left\| I - \widetilde{L}_S^{-1}S\widetilde{L}_S^{-T} \right\|, \\ g &= \|G\|. \end{aligned} \tag{7.15}$$

In other words, e_k and e_s are a measure of the error made when using the approximations M_K^{-1} and M_S^{-1} in place of K^{-1} and S^{-1}, respectively, while g provides an estimate of the strength of coupling between the upper and lower set of equations in (7.3). Inequality (7.14) provides a useful and relatively cheap indication on how the quality of M_K^{-1} and M_S^{-1} affects the overall convergence rate of MCP. In particular, note that the quality of M_K^{-1} is generally more important than that of M_S^{-1}, with the difference becoming larger as the coupling becomes stronger.

Quite naturally, there is the distinct interest to implement constraint preconditioners on parallel computers. The main difficulties for developing efficient parallel variants of constraint preconditioners are twofold. First, the constraint preconditioning concept is inherently sequential, as it involves the preliminary computation of the Schur complement and then the subsequent approximate solution to two inner systems. Thus, a standard parallel approach where the system matrix is distributed among the processors as stripes of contiguous rows is not feasible, because all the processors owning the second block of unknowns are idle when the other processors perform the operation on the first block, and viceversa. Second, efficient parallel approximations of K^{-1} and S^{-1} have to replace incomplete factorizations. For example, in incompressible Navier-Stokes problems parallel AMG techniques have been efficiently used to approximate the K block which arises from a discretized Laplace operator [234]. In coupled consolidation better results have been obtained by using a double FSAI [235], parallel multilevel variants [236], and Block FSAI [237], which appears to be currently the most promising approach in the context of geomechanical problems.

8. Conclusions and Future Perspectives

The development and implementation of efficient preconditioners are the key factors for improving the performance and robustness of Krylov subspace methods in the solution of large sparse linear systems. Such as issue is a central task in a great number of different applications, not all necessarily related to PDEs, so it is no surprise that much research has been devoted to it. After a time where most efforts were focused on direct and iterative solvers, the last two decades can be denoted as the "preconditioning era." The number of papers, projects, and international conferences addressing this topic has largely exceeded those aimed at the development of new solvers, and this trend is not likely to fade in the next few years. This is mainly due to the fact that an optimal general-purpose preconditioner does not exist and that any specific problem can afford the opportunity to develop an ad hoc strategy.

The present paper is intended to offer an overview of the most important algebraic preconditioners available today for the solution of sparse linear systems. Hence, it cannot exhaust the subject of preconditioning as the entire class of problem-specific algorithms has been omitted. Algebraic preconditioners have been chosen for their "universality," in that they can be effectively used as black-box tools in general-purpose codes requiring the knowledge of the system matrix only. A classification of algebraic preconditioners has been attempted based on the most significant representatives of each group along with their prominent features:

(i) *incomplete factorizations* have been the earliest algebraic preconditioners massively used to accelerate Krylov subspace solvers. The several weak points related to robustness and numerical stability have stimulated the development of many different variants meant to both improve their performance and extend the application area. Although some advances are still under way, this class of preconditioners appears to be quite mature with the related strong and weak points very well known. Incomplete factorizations can be very effective and in many cases are still the most performant preconditioners, but their intrinsic lack of robustness and especially the very reduced parallel degree are probably going to bound their role in the next future;

(ii) *approximate inverses* were originally developed with the aim of resolving the main difficulties of ILU-based preconditioners. Between the late 90s and early 00s they looked as a very promising approach especially in view of their robustness and potentially high parallel degree. Indeed, these preconditioners have proved very robust and so well suited for black-box solvers and in some cases almost perfectly parallel. In a sequential run, however, their performance can be lower than that of an ILU-based preconditioner. At present, the class of approximate inverses appears to be quite a mature field of research, with the most significant breakthroughs advanced more than 10 years ago. Nonetheless, their influence is probably going to be still very important in the next few years;

(iii) the potentially high scalability of *algebraic multigrid* techniques has promoted much research in this area. In contrast with both incomplete factorizations and approximate inverses, AMG can allow for a stable iteration count to converge independently of the characteristic mesh size, that is, the overall system dimension, thus making AMG a potentially ideally scalable tool. Currently, AMG appears as one of the more active research areas, with possible contaminations from incomplete factorizations and approximate inverses in the choice of the smoother. In several applications arising from discretized PDEs with severe mesh distortions, however, AMG can still lack robustness, and more research is still needed to make it the algorithm of choice in the next future;

(iv) the current hardware development is leading to a much more pronounced parallel degree of any computer. Besides the parallelization of existing algorithms, this trend is going to enforce the development of totally new techniques which can make sense in a parallel computing environment only. This is why *parallel preconditioners* are now emerging as a class of algorithms which are likely to expand in the next future. These tools can take advantage of important contributions from all the previous classes and can also relaunch some techniques, such as polynomial preconditioners or FSAI-based approximate inverses, which had been somewhat forgotten over the last years.

Acknowledgments

The author is very thankful to Giuseppe Gambolati and Carlo Janna for their careful reading of the paper and helpful comments.

References

[1] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, Pa, USA, 1997.
[2] M. Benzi, "Preconditioning techniques for large linear systems: a survey," *Journal of Computational Physics*, vol. 182, no. 2, pp. 418–477, 2002.
[3] D. J. Evans, "The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices," *Journal of the Institute of Mathematics and Its Applications*, vol. 4, pp. 295–314, 1968.
[4] L. Cesari, "Sulla risoluzione dei sistemi di equazioni lineari per approssimazioni successive," *Atti dell'Accademia Nazionale dei Lincei*, vol. 25, pp. 422–428, 1937.
[5] Y. Saad and H. A. van der Vorst, "Iterative solution of linear systems in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 1-2, pp. 1–33, 2000.
[6] R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1962.

[7] L.F. Richardson, "The approximate arithmetical solution by finite differences of physical problems involving differential equations with an application to the stresses to a masonry dam," *Philosophical Transactions of the Royal Society of London A*, vol. 210, pp. 307–357, 1910.
[8] G. Cimmino, "Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari," *Ricerca Scientifica e Programma Tecnico Economico Nazionale*, vol. 9, pp. 326–333, 1938 (Italian).
[9] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, UK, 1965.
[10] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proceedings of the 24th National Conference (ACM '69)*, pp. 157–172, 1969.
[11] A. George, "Nested dissection of a regular finite element mesh," *SIAM Journal on Numerical Analysis*, vol. 10, pp. 345–363, 1973.
[12] R. J. Lipton, D. J. Rose, and R. E. Tarjan, "Generalized nested dissection," *SIAM Journal on Numerical Analysis*, vol. 16, no. 2, pp. 346–358, 1979.
[13] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.
[14] M. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proceedings of IEEE*, vol. 55, no. 11, pp. 1801–1809, 1967.
[15] J. W. H. Liu, "Modification of the minimum-degree algorithm by multiple elimination," *ACM Transactions on Mathematical Software*, vol. 11, no. 2, pp. 141–153, 1985.
[16] A. George and J. W. H. Liu, "The evolution of the minimum degree ordering algorithm," *SIAM Review*, vol. 31, no. 1, pp. 1–19, 1989.
[17] D. R. Fulkerson and P. Wolfe, "An algorithm for scaling matrices," *SIAM Review*, vol. 4, pp. 142–146, 1962.
[18] A. R. Curtis and J. K. Reid, "On the automatic scaling of matrices for gaussian elimination," *IMA Journal of Applied Mathematics*, vol. 10, no. 1, pp. 118–124, 1972.
[19] M. R. Hestenes and E. L. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards B*, vol. 49, pp. 409–436, 1952.
[20] C. Lanczos, "Solution of systems of linear equations by minimized-iterations," *Journal of Research of the National Bureau of Standards B*, vol. 49, pp. 33–53, 1952.
[21] M. Engeli, T. Ginsburg, H. Rutishauser, and E. L. Stiefel, *Refined Iteratie Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems*, Birkhäuser, Basel, Switzerland, 1959.
[22] D. M. Young, *Iterative methods for solving partial differential equations of elliptic type [Ph.D. thesis]*, Harvard University, Cambridge, Mass, USA, 1950.
[23] S. P. Frankel, "Convergence rates of iterative treatments of partial differential equations," *Mathematical Tables and Other Aids to Computation*, vol. 4, pp. 65–75, 1950.
[24] I. S. Duff and J. K. Reid, "The multifrontal solution of indefinite sparse symmetric linear system," *Association for Computing Machinery*, vol. 9, no. 3, pp. 302–325, 1983.
[25] I. S. Duff, "Parallel implementation of multifrontal schemes," *Parallel Computing. Theory and Applications*, vol. 3, no. 3, pp. 193–204, 1986.
[26] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, UK, 1986.
[27] T. A. Davis, *Direct Methods for Sparse Linear Systems*, vol. 2, SIAM, Philadelphia, Pa, USA, 2006.
[28] J. K. Reid, "On the method of conjugate gradients for the solution of large sparse systems of linear equations," in *Large Sparse Sets of Linear Equations*, J. K. Reid, Ed., pp. 231–254, Academic Press, New York, NY, USA, 1971.
[29] C. C. Paige and M. A. Saunders, "Solutions of sparse indefinite systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 12, no. 4, pp. 617–629, 1975.
[30] R. Fletcher, "Conjugate gradient methods for indefinite systems," in *Proceedings of the Dundee Biennal Conference on Numerical Analysis*, G. A. Watson, Ed., pp. 73–89, Springer, New York, NY, USA, 1976.
[31] J. A. Meijerink and H. A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," *Mathematics of Computation*, vol. 31, no. 137, pp. 148–162, 1977.
[32] N. I. Buleev, "A numerical method for the solution of two-dimensional and three-dimensional equations of diffusion," *Mathematics Sbornik*, vol. 51, pp. 227–238, 1960 (Russian).
[33] R. S. Varga, "Factorization and normalized iterative methods," in *Boundary Problems in Differential Equations*, R. E. Langer, Ed., pp. 121–142, University of Wisconsin Press, Madison, Wis, USA, 1960.
[34] D. S. Kershaw, "The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations," *Journal of Computational Physics*, vol. 26, no. 1, pp. 43–65, 1978.

[35] Y. Saad and M. H. Schultz, "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 3, pp. 856–869, 1986.

[36] P. Sonneveld, "CGS, a fast Lanczos-type solver for nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 10, no. 1, pp. 36–52, 1989.

[37] H. A. van der Vorst, "Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 2, pp. 631–644, 1992.

[38] V. Faber and T. Manteuffel, "Necessary and sufficient conditions for the existence of a conjugate gradient method," *SIAM Journal on Numerical Analysis*, vol. 21, no. 2, pp. 352–362, 1984.

[39] R. W. Freund and N. M. Nachtigal, "QMR: a quasi-minimal residual method for non-hermitian linear systems," *Numerische Mathematik*, vol. 60, no. 3, pp. 315–339, 1991.

[40] R. W. Freund, "A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems," *SIAM Journal on Scientific Computing*, vol. 14, no. 2, pp. 470–482, 1993.

[41] R. W. Freund and N. M. Nachtigal, "A new Krylov-subspace method for symmetric indefinite linear systems," in *Proceedings of the 14th IMACS World Congress on Computational and Applied Mathematics*, pp. 1253–1256, 1994.

[42] H. A. van der Vorst and C. Vuik, "GMRESR: a family of nested GMRES methods," *Numerical Linear Algebra with Applications*, vol. 1, no. 4, pp. 369–386, 1994.

[43] E. de Sturler, "Nested Krylov methods based on GCR," *Journal of Computational and Applied Mathematics*, vol. 67, no. 1, pp. 15–41, 1996.

[44] V. Simoncini and D. B. Szyld, "Recent computational developments in Krylov subspace methods for linear systems," *Numerical Linear Algebra with Applications*, vol. 14, no. 1, pp. 1–59, 2007.

[45] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, Pa, USA, 2nd edition, 2003.

[46] S. Dupont and J. M. Marchal, "Preconditioned conjugate gradients for solving the transient Boussinesq equations in three-dimensional geometries," *International Journal for Numerical Methods in Fluids*, vol. 8, no. 3, pp. 283–303, 1988.

[47] J. K. Dickinson and P. A. Forsyth, "Preconditioned conjugate gradient methods for three-dimensional linear elasticity," *International Journal for Numerical Methods in Engineering*, vol. 37, no. 13, pp. 2211–2234, 1994.

[48] G. F. Carey, Y. Shen, and R. T. McLay, "Parallel conjugate gradient performance for least-squares finite elements and transport problems," *International Journal for Numerical Methods in Fluids*, vol. 28, pp. 1421–1440, 1998.

[49] H. Mroueh and I. Shahrour, "Use of sparse iterative methods for the resolution of three-dimensional soil/structure interaction problems," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 23, no. 15, pp. 1961–1975, 1999.

[50] P. Saint-George, G. Warzee, Y. Notay, and R. Beauwens, "Problem-dependent preconditioners for iterative solvers in FE elastostatics," *Computers and Structures*, vol. 73, pp. 33–43, 1999.

[51] G. Gambolati, G. Pini, and M. Ferronato, "Numerical performance of projection methods in finite element consolidation models," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 25, no. 14, pp. 1429–1447, 2001.

[52] O. Axelsson, "A class of iterative methods for finite element equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 9, no. 2, pp. 123–127, 1976.

[53] A. Greenbaum, V. Pták, and Z. Strakoš, "Any nonincreasing convergence curve is possible for GMRES," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 3, pp. 465–469, 1996.

[54] H. Mroueh and I. Shahrour, "A full 3-D finite element analysis of tunneling-adjacent structures interaction," *Computers and Geotechnics*, vol. 30, no. 3, pp. 245–253, 2003.

[55] K. K. Phoon, S. H. Chan, K. C. Toh, and F. H. Lee, "Fast iterative solution of large undrained soil-structure interaction problems," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 27, no. 3, pp. 159–181, 2003.

[56] M. Ferronato, C. Janna, and G. Gambolati, "Mixed constraint preconditioning in computational contact mechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 45–48, pp. 3922–3931, 2008.

[57] J. B. Haga, H. Osnes, and H. P. Langtangen, "Efficient block preconditioners for the coupled equations of pressure and deformation in highly discontinuous media," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 35, no. 13, pp. 1466–1482, 2011.

[58] G. Gambolati, M. Ferronato, and C. Janna, "Preconditioners in computational geomechanics: a survey," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 35, no. 9, pp. 980–996, 2011.
[59] K. K. Phoon, K. C. Toh, S. H. Chan, and F. H. Lee, "An efficient diagonal preconditioner for finite element solution of Biot's consolidation equations," *International Journal for Numerical Methods in Engineering*, vol. 55, no. 4, pp. 377–400, 2002.
[60] M. F. Murphy, G. H. Golub, and A. J. Wathen, "A note on preconditioning for indefinite linear systems," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 1969–1972, 2000.
[61] L. Bergamaschi and A.. Martínez, "RMCP: relaxed mixed constraint preconditioners for saddle point linear systems arising in geomechanics," *Computer Methods in Applied Mechanics and Engineering*, vol. 221, pp. 54–62, 2012.
[62] X. Chen, K. C. Toh, and K. K. Phoon, "A modified SSOR preconditioner for sparse symmetric indefinite linear systems of equations," *International Journal for Numerical Methods in Engineering*, vol. 65, no. 6, pp. 785–807, 2006.
[63] X. Chen and K. K. Phoon, "Applications of symmetric and nonsymmetric MSSOR preconditioners to large-scale Biot's consolidation problems with nonassociated plasticity," *Journal of Applied Mathematics*, vol. 2012, Article ID 352081, 15 pages, 2012.
[64] S.-L. Wu and C.-X. Li, "A modified SSOR preconditioning strategy for Helmholtz equations," *Journal of Applied Mathematics*, vol. 2012, Article ID 365124, 9 pages, 2012.
[65] M. Ferronato, G. Pini, and G. Gambolati, "The role of preconditioning in the solution to FE coupled consolidation equations by Krylov subspace methods," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 33, no. 3, pp. 405–423, 2009.
[66] S. V. Parter and E. E. Rothman, "Preconditioning Legendre spectral collocation approximations to elliptic problems," *SIAM Journal on Numerical Analysis*, vol. 32, no. 2, pp. 333–385, 1995.
[67] S. D. Kim and S. V. Parter, "Preconditioning Chebyshev spectral collocation by finite-difference operators," *SIAM Journal on Numerical Analysis*, vol. 34, no. 3, pp. 939–958, 1997.
[68] S. D. Kim, "Piecewise bilinear preconditioning of high-order finite element methods," *Electronic Transactions on Numerical Analysis*, vol. 26, pp. 228–242, 2007.
[69] J. K. Kwon, S. Ryu, P. Kim, and S. D. Kim, "Finite element preconditioning on spectral element discretizations for coupled elliptic equations," *Journal of Applied Mathematics*, vol. 2012, Article ID 245051, 16 pages, 2012.
[70] S. F. Ashby, P. N. Brown, M. R. Dorr, and D. C. Hindmarsh, "A linear algebraic development of diffusion synthetic acceleration for for the Boltzmann transport equation," *SIAM Journal on Numerical Analysis*, vol. 32, no. 1, pp. 179–214, 1995.
[71] V. A. Mousseau, D. A. Knoll, and W. J. Rider, "Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion," *Journal of Computational Physics*, vol. 160, no. 2, pp. 743–765, 2000.
[72] K. J. Lange, C. Misra, P.-C. Sui, and N. Djilali, "A numerical study on preconditioning and partitioning schemes for reactive transport in a PEMFC catalyst layer," *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 9–12, pp. 905–916, 2011.
[73] E. Chow and Y. Saad, "ILUS: an incomplete LU preconditioner in sparse skyline format," *International Journal for Numerical Methods in Fluids*, vol. 25, no. 7, pp. 739–748, 1997.
[74] N. Li, Y. Saad, and E. Chow, "Crout versions of ILU for general sparse matrices," *SIAM Journal on Scientific Computing*, vol. 25, no. 2, pp. 716–728, 2003.
[75] G. Gambolati, "Fast solution to finite element flow equations by Newton iteration and modified conjugate gradient method," *International Journal for Numerical Methods in Engineering*, vol. 15, no. 5, pp. 661–675, 1980.
[76] G. Gambolati, G. Pini, and G. Zilli, "Numerical comparison of preconditionings for large sparse finite element problems," *Numerical Methods for Partial Differential Equations*, vol. 4, no. 2, pp. 139–157, 1988.
[77] X. Chen, K. K. Phoon, and K. C. Toh, "Performance of zero fill-in preconditioning techniques for iterative solutions with geotechnical applications," *International Journal of Geomechanics*, vol. 12, no. 5, pp. 596–605, 2012.
[78] E. Chow and Y. Saad, "Experimental study of ILU preconditioners for indefinite matrices," *Journal of Computational and Applied Mathematics*, vol. 86, no. 2, pp. 387–414, 1997.
[79] I. Gustafsson, "A class of first order factorization methods," *BIT*, vol. 18, no. 2, pp. 142–156, 1978.
[80] J. W. Watts III, "A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation," *Society of Petroleum Engineers Journal*, vol. 21, no. 3, pp. 345–353, 1981.

[81] N. Munksgaard, "Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients," *ACM Transactions on Mathematical Software*, vol. 6, pp. 206–219, 1980.
[82] M. Bollhöfer, "A robust ILU with pivoting based on monitoring the growth of the inverse factors," *Linear Algebra and Its Applications*, vol. 338, pp. 201–218, 2001.
[83] M. Bollhöfer, "A robust and efficient ILU that incorporates the growth of the inverse triangular factors," *SIAM Journal on Scientific Computing*, vol. 25, no. 1, pp. 86–103, 2003.
[84] Y. Saad, "ILUT: a dual threshold incomplete ILU factorization," *Numerical Linear Algebra with Applications*, vol. 1, no. 4, pp. 387–402, 1994.
[85] M. T. Jones and P. E. Plassmann, "An improved incomplete Cholesky factorization," *ACM Transactions on Mathematical Software*, vol. 21, no. 1, pp. 5–17, 1995.
[86] C.-J. Lin and J. J. Moré, "Incomplete Cholesky factorizations with limited memory," *SIAM Journal on Scientific Computing*, vol. 21, no. 1, pp. 24–45, 1999.
[87] M. Bollhöfer and Y. Saad, "Multilevel preconditioners constructed from inverse-based ILUs," *SIAM Journal on Scientific Computing*, vol. 27, no. 5, pp. 1627–1650, 2006.
[88] P. Hénon, P. Ramet, and J. Roman, "On finding approximate supernodes for an efficient block-ILU(k) factorization," *Parallel Computing*, vol. 34, no. 6–8, pp. 345–362, 2008.
[89] A. Gupta and T. George, "Adaptive techniques for improving the performance of incomplete factorization preconditioning," *SIAM Journal on Scientific Computing*, vol. 32, no. 1, pp. 84–110, 2010.
[90] C. Janna, A. Comerlati, and G. Gambolati, "A comparison of projective and direct solvers for finite elements in elastostatics," *Advances in Engineering Software*, vol. 40, no. 8, pp. 675–685, 2009.
[91] O. Axelsson and L. Yu. Kolotilina, "Diagonally compensated reduction and related preconditioning methods," *Numerical Linear Algebra with Applications*, vol. 1, no. 2, pp. 155–177, 1994.
[92] M. Benzi and M. Tůma, "A robust incomplete factorization preconditioner for positive definite matrices," *Numerical Linear Algebra with Applications*, vol. 10, no. 5-6, pp. 385–400, 2003.
[93] M. A. Ajiz and A. Jennings, "A robust incomplete Choleski-conjugate gradient algorithm," *International Journal for Numerical Methods in Engineering*, vol. 20, no. 5, pp. 949–966, 1984.
[94] B. A. Schrefler and R. Scotta, "A fully coupled dynamic model for two-phase fluid flow in deformable porous media," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 24-25, pp. 3223–3246, 2001.
[95] A. Mazzia, L. Bergamaschi, and M. Putti, "On the reliability of numerical solutions of brine transport in groundwater," *Transport in Porous Media*, vol. 43, no. 1, pp. 65–86, 2001.
[96] T. I. Zohdi, "Modeling and simulation of a class of coupled thermo-chemo-mechanical processes in multiphase solids," *Computer Methods in Applied Mechanics and Engineering*, vol. 193, no. 6–8, pp. 679–699, 2004.
[97] N. Castelletto, M. Ferronato, and G. Gambolati, "Thermo-hydro-mechanical modeling of fluid geological storage by Godunov-mixed methods," *International Journal for Numerical Methods in Engineering*, vol. 90, no. 8, pp. 988–1009, 2012.
[98] R. E. Bank and C. Wagner, "Multilevel ILU decomposition," *Numerische Mathematik*, vol. 82, no. 4, pp. 543–576, 1999.
[99] Y. Saad and J. Zhang, "BILUM: block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems," *SIAM Journal on Scientific Computing*, vol. 20, no. 6, pp. 2103–2121, 1999.
[100] Y. Saad and J. Zhang, "BILUTM: a domain-based multilevel block ILUT preconditioner for general sparse matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 1, pp. 279–299, 1999.
[101] Y. Saad and B. Suchomel, "ARMS: an algebraic recursive multilevel solver for general sparse linear systems," *Numerical Linear Algebra with Applications*, vol. 9, no. 5, pp. 359–378, 2002.
[102] J. K. Kraus, "Algebraic multilevel preconditioning of finite element matrices using local Schur complements," *Numerical Linear Algebra with Applications*, vol. 13, no. 1, pp. 49–70, 2006.
[103] O. Axelsson and P. S. Vassilevski, "Algebraic multilevel preconditioning methods. I," *Numerische Mathematik*, vol. 56, no. 2-3, pp. 157–177, 1989.
[104] O. Axelsson and P. S. Vassilevski, "Algebraic multilevel preconditioning methods. II," *SIAM Journal on Numerical Analysis*, vol. 27, no. 6, pp. 1569–1590, 1990.
[105] P. S. Vassilevski, *Multilevel Block Factorization Preconditioners: Matrix-based Analysis and Algorithms for Solving Finite Element Equations*, Springer, Berlin, Germany, 2008.
[106] C. Janna, M. Ferronato, and G. Gambolati, "Multi-level incomplete factorizations for the iterative solution of non-linear finite element problems," *International Journal for Numerical Methods in Engineering*, vol. 80, no. 5, pp. 651–670, 2009.

[107] M. Tismenetsky, "A new preconditioning technique for solving large sparse linear systems," *Linear Algebra and Its Applications*, vol. 154–156, pp. 331–353, 1991.

[108] I. E. Kaporin, "High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU+U^TR+R^TU$-decomposition," *Numerical Linear Algebra with Applications*, vol. 5, no. 6, pp. 483–509, 1998.

[109] I. S. Duff and G. A. Meurant, "The effect of ordering on preconditioned conjugate gradients," *BIT Numerical Mathematics*, vol. 29, no. 4, pp. 635–657, 1989.

[110] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, "Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 3, pp. 944–961, 1992.

[111] L. C. Dutto, "The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Engineering*, vol. 36, no. 3, pp. 457–497, 1993.

[112] M. Benzi, D. B. Szyld, and A. van Duin, "Orderings for incomplete factorization preconditioning of nonsymmetric problems," *SIAM Journal on Scientific Computing*, vol. 20, no. 5, pp. 1652–1670, 1999.

[113] G. Gambolati, G. Pini, and M. Ferronato, "Scaling improves stability of preconditioned CG-like solvers for FE consolidation equations," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 27, no. 12, pp. 1043–1056, 2003.

[114] M. W. Benson, *Iterative solution of large scale linear systems [M.S. thesis]*, Lakehead University, Ontario, Canada, 1973.

[115] P. O. Frederickson, "Fast approximate inversion of large sparse linear systems," Mathematical Report 7, Lakehead University, Thunder Bay, Ontario, Canada, 1975.

[116] J. D. F. Cosgrove, J. C. Diaz, and A. Griewank, "Approximate inverse preconditioning for sparse linear systems," *International Journal of Computational Mathematics*, vol. 44, pp. 91–110, 1992.

[117] M. J. Grote and T. Huckle, "Parallel preconditioning with sparse approximate inverses," *SIAM Journal on Scientific Computing*, vol. 18, no. 3, pp. 838–853, 1997.

[118] N. I. M. Gould and J. A. Scott, "Sparse approximate-inverse preconditioners using norm-minimization techniques," *SIAM Journal on Scientific Computing*, vol. 19, no. 2, pp. 605–625, 1998.

[119] E. Chow and Y. Saad, "Approximate inverse preconditioners via sparse-sparse iterations," *SIAM Journal on Scientific Computing*, vol. 19, no. 3, pp. 995–1023, 1998.

[120] R. M. Holland, A. J. Wathen, and G. J. Shaw, "Sparse approximate inverses and target matrices," *SIAM Journal on Scientific Computing*, vol. 26, no. 3, pp. 1000–1011, 2005.

[121] Z. Jia and B. Zhu, "A power sparse approximate inverse preconditioning procedure for large sparse linear systems," *Numerical Linear Algebra with Applications*, vol. 16, no. 4, pp. 259–299, 2009.

[122] M. Benzi, C. D. Meyer, and M. Tůma, "A sparse approximate inverse preconditioner for the conjugate gradient method," *SIAM Journal on Scientific Computing*, vol. 17, no. 5, pp. 1135–1149, 1996.

[123] M. Benzi and M. Tůma, "A sparse approximate inverse preconditioner for nonsymmetric linear systems," *SIAM Journal on Scientific Computing*, vol. 19, no. 3, pp. 968–994, 1998.

[124] M. Benzi, J. K. Cullum, and M. Tůma, "Robust approximate inverse preconditioning for the conjugate gradient method," *SIAM Journal on Scientific Computing*, vol. 22, no. 4, pp. 1318–1332, 2000.

[125] L. Bergamaschi, G. Pini, and F. Sartoretto, "Approximate inverse preconditioning in the parallel solution of sparse eigenproblems," *Numerical Linear Algebra with Applications*, vol. 7, no. 3, pp. 99–116, 2000.

[126] R. Bridson and W.-P. Tang, "Multiresolution approximate inverse preconditioners," *SIAM Journal on Scientific Computing*, vol. 23, no. 2, pp. 463–479, 2001.

[127] G. A. Meurant, "A multilevel AINV preconditioner," *Numerical Algorithms*, vol. 29, no. 1–3, pp. 107–129, 2002.

[128] M. Bollhöfer and Y. Saad, "A factored approximate inverse preconditioner with pivoting," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 3, pp. 692–705, 2002.

[129] L. Bergamaschi, G. Pini, and F. Sartoretto, "Computational experience with sequential and parallel, preconditioned Jacobi-Davidson for large, sparse symmetric matrices," *Journal of Computational Physics*, vol. 188, no. 1, pp. 318–331, 2003.

[130] L. Bergamaschi, M. Ferronato, and G. Gambolati, "Novel preconditioners for the iterative solution to FE-discretized coupled consolidation equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 25-28, pp. 2647–2656, 2007.

[131] L. Yu. Kolotilina and A. Yu. Yeremin, "Factorized sparse approximate inverse preconditionings. I. Theory," *SIAM Journal on Matrix Analysis and Applications*, vol. 14, no. 1, pp. 45–58, 1993.

[132] I. E. Kaporin, "New convergence results and preconditioning strategies for the conjugate gradient method," *Numerical Linear Algebra with Applications*, vol. 1, no. 2, pp. 179–210, 1994.
[133] L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, "Factorized sparse approximate inverse preconditionings. IV. Simple approaches to rising efficiency," *Numerical Linear Algebra with Applications*, vol. 6, no. 7, pp. 515–531, 1999.
[134] T. Huckle, "Factorized sparse approximate inverses for preconditioning," *Journal of Supercomputing*, vol. 25, no. 2, pp. 109–117, 2003.
[135] L. Bergamaschi, G. Gambolati, and G. Pini, "A numerical experimental study of inverse preconditioning for the parallel iterative solution to 3D finite element flow equations," *Journal of Computational and Applied Mathematics*, vol. 210, no. 1-2, pp. 64–70, 2007.
[136] M. Ferronato, C. Janna, and G. Pini, "Shifted FSAI preconditioners for the efficient parallel solution of non-linear groundwater flow models," *International Journal for Numerical Methods in Engineering*, vol. 89, no. 13, pp. 1707–1719, 2012.
[137] A. Yu. Yeremin and A. A. Nikishin, "Factorized sparse approximate inverse preconditioning of linear systems with nonsymmetric matrices," *Journal of Mathematical Sciences*, vol. 121, pp. 2448–2457, 2004.
[138] L. Bergamaschi and A. Martínez, "Parallel acceleration of Krylov solvers by factorized approximate inverse preconditioners," in *Proceedings of the Conference on High Performance Computing for Computational Science (VECPAR '04)*, vol. 3402 of *Lecture Notes in Computer Science*, pp. 623–636, June 2005.
[139] S. Demko, W. F. Moss, and P. W. Smith, "Decay rates for inverses of band matrices," *Mathematics of Computation*, vol. 43, no. 168, pp. 491–499, 1984.
[140] M. H. Koulaei and F. Toutounian, "Factored sparse approximate inverse of block tridiagonal and block pentadiagonal matrices," *Applied Mathematics and Computation*, vol. 184, no. 2, pp. 223–234, 2007.
[141] E. Chow, "A priori sparsity patterns for parallel sparse approximate inverse preconditioners," *SIAM Journal on Scientific Computing*, vol. 21, no. 5, pp. 1804–1822, 2000.
[142] L. Bergamaschi, A. Martínez, and G. Pini, "Parallel preconditioned conjugate gradient optimization of the Rayleigh quotient for the solution of sparse eigenproblems," *Applied Mathematics and Computation*, vol. 175, no. 2, pp. 1694–1715, 2006.
[143] T. Huckle, "Approximate sparsity patterns for the inverse of a matrix and preconditioning," *Applied Numerical Mathematics*, vol. 30, no. 2-3, pp. 291–303, 1999.
[144] C. Janna and M. Ferronato, "Adaptive pattern research for block FSAI preconditioning," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3357–3380, 2011.
[145] A. A. Kharchenko, L. Yu. Kolotilina, A. A. Nikishin, and A. Yu. Yeremin, "A robust AINV-type method for constructing sparse approximate inverse preconditioners in factored form," *Numerical Linear Algebra with Applications*, vol. 8, no. 3, pp. 165–179, 2001.
[146] M. Benzi and M. Tůma, "A comparative study of sparse approximate inverse preconditioners," *Applied Numerical Mathematics*, vol. 30, no. 2-3, pp. 305–340, 1999.
[147] S. T. Barnard, L. M. Bernardo, and H. D. Simon, "MPI implementation of the SPAI preconditioner on the T3E," *International Journal of High Performance Computing Applications*, vol. 13, no. 2, pp. 107–123, 1999.
[148] L. Bergamaschi, M. Ferronato, and G. Gambolati, "Mixed constraint preconditioners for the iterative solution of FE coupled consolidation equations," *Journal of Computational Physics*, vol. 227, no. 23, pp. 9885–9897, 2008.
[149] M. Ferronato, C. Janna, and G. Pini, "Parallel solution to ill-conditioned FE geomechanical problems," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 36, no. 4, pp. 422–437, 2012.
[150] A. Brandt, "Multi-level adaptive solutions to boundary-value problems," *Mathematics of Computation*, vol. 31, no. 138, pp. 333–390, 1977.
[151] A. Brandt, "A guide to multigrid development," in *Multigrid Methods*, W. Hackbusch and U. Trottenberg, Eds., vol. 960, pp. 220–312, Springer, Berlin, Germany, 1982.
[152] W. L. Briggs, V. E. Henson, and S. F. Mccormick, *A Multigrid Tutorial*, SIAM, Philadelphia, Pa, USA, 2nd edition, 2000.
[153] A. Brandt, S. F. Mccormick, and J. W. Ruge, "Algebraic multigrid (AMG) for sparse matrix equations," in *Sparsity and Its Applications*, D. J. Evans and U. Trottenberg, Eds., pp. 257–284, Cambridge University Press, Cambridge, Mass, USA, 1984.
[154] A. Brandt, "Algebraic multigrid theory: the symmetric case," *Applied Mathematics and Computation*, vol. 19, no. 1–4, pp. 23–56, 1986.

[155] J. W. Ruge and K. Stüben, "Algebraic multigrid (AMG)," in *Multigrid Methods*, vol. 3, pp. 73–130, SIAM, Philadelphia, Pa, USA, 1987.
[156] K. Stüben, "A review of algebraic multigrid," *Journal of Computational and Applied Mathematics*, vol. 128, no. 1-2, pp. 281–309, 2001.
[157] P. Vaněk, J. Mandel, and M. Brezina, "Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems," *Computing*, vol. 56, no. 3, pp. 179–196, 1996.
[158] Y. Notay, "Algebraic multigrid and algebraic multilevel methods: a theoretical comparison," *Numerical Linear Algebra with Applications*, vol. 12, no. 5-6, pp. 419–451, 2005.
[159] Y. Notay, "Aggregation-based algebraic multilevel preconditioning," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 998–1018, 2006.
[160] R. S. Tuminaro and C. Tong, "Parallel smoothed aggregation multigrid: aggregation strategies on massively parallel machines," in *Proceedings of the Supercomputing ACM/IEEE Conference*, J. Donnelley, Ed., 2000.
[161] A. Krechel and K. Stüben, "Parallel algebraic multigrid based on subdomain blocking," *Parallel Computing*, vol. 27, no. 8, pp. 1009–1031, 2001.
[162] V. E. Henson and U. M. Yang, "BoomerAMG: a parallel algebraic multigrid solver and preconditioner," *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 155–177, 2002.
[163] F. H. Pereira, S. L. L. Verardi, and S. I. Nabeta, "A wavelet-based algebraic multigrid preconditioner for sparse linear systems," *Applied Mathematics and Computation*, vol. 182, no. 2, pp. 1098–1107, 2006.
[164] C. R. Dohrmann, "Interpolation operators for algebraic multigrid by local optimization," *SIAM Journal on Scientific Computing*, vol. 29, no. 5, pp. 2045–2058, 2007.
[165] M. T. Jones and P. E. Plassmann, "A parallel graph coloring heuristic," *SIAM Journal on Scientific Computing*, vol. 14, no. 3, pp. 654–669, 1993.
[166] M. Adams, M. Brezina, J. Hu, and R. Tuminaro, "Parallel multigrid smoothing: polynomial versus Gauss-Seidel," *Journal of Computational Physics*, vol. 188, no. 2, pp. 593–610, 2003.
[167] A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang, "Multigrid smoothers for ultraparallel computing," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2864–2887, 2011.
[168] W.-P. Tang and W. L. Wan, "Sparse approximate inverse smoother for multigrid," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1236–1252, 2000.
[169] O. Bröker and M. J. Grote, "Sparse approximate inverse smoothers for geometric and algebraic multigrid," *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 61–80, 2002.
[170] R. D. Falgout and P. S. Vassilevski, "On generalizing the algebraic multigrid framework," *SIAM Journal on Numerical Analysis*, vol. 42, no. 4, pp. 1669–1693, 2004.
[171] M. Brezina, R. D. Falgout, S. P. Maclachlan, T. A. Manteuffel, S. F. Mccormick, and J. W. Ruge, "Adaptive algebraic multigrid," *SIAM Journal on Scientific Computing*, vol. 27, no. 4, pp. 1261–1286, 2006.
[172] E. Virnik, "An algebraic multigrid preconditioner for a class of singular M-matrices," *SIAM Journal on Scientific Computing*, vol. 29, no. 5, pp. 1982–1991, 2007.
[173] S. P. Maclachlan and C. W. Oosterlee, "Algebraic multigrid solvers for complex-valued matrices," *SIAM Journal on Scientific Computing*, vol. 30, no. 3, pp. 1548–1571, 2008.
[174] L. N. Olson, J. Schroder, and R. S. Tuminaro, "A new perspective on strength measures in algebraic multigrid," *Numerical Linear Algebra with Applications*, vol. 17, no. 4, pp. 713–733, 2010.
[175] K. H. Leem, S. Oliveira, and D. E. Stewart, "Algebraic multigrid (AMG) for saddle point systems from meshfree discretizations," *Numerical Linear Algebra with Applications*, vol. 11, no. 2-3, pp. 293–308, 2004.
[176] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq, and R. S. Tuminaro, "A comparison of eigensolvers for large-scale 3D modal analysis using AMG-preconditioned iterative methods," *International Journal for Numerical Methods in Engineering*, vol. 64, no. 2, pp. 204–236, 2005.
[177] J. J. Heys, T. A. Manteuffel, S. F. Mccormick, and L. N. Olson, "Algebraic multigrid for higher-order finite elements," *Journal of Computational Physics*, vol. 204, no. 2, pp. 520–532, 2005.
[178] J. J. Hu, R. S. Tuminaro, P. B. Bochev, C. J. Garasi, and A. C. Robinson, "Toward an h-independent algebraic multigrid method for Maxwell's equations," *SIAM Journal on Scientific Computing*, vol. 27, no. 5, pp. 1669–1688, 2006.
[179] M. Brezina, C. Tong, and R. Becker, "Parallel algebraic multigrids for structural mechanics," *SIAM Journal on Scientific Computing*, vol. 27, no. 5, pp. 1534–1554, 2006.
[180] C. D. Riyanti, A. Kononov, Y. A. Erlangga et al., "A parallel multigrid-based preconditioner for the 3D heterogeneous high-frequency Helmholtz equation," *Journal of Computational Physics*, vol. 224, no. 1, pp. 431–448, 2007.

[181] M. Pennacchio and V. Simoncini, "Algebraic multigrid preconditioners for the bidomain reaction-diffusion system," *Applied Numerical Mathematics*, vol. 59, no. 12, pp. 3033–3050, 2009.
[182] F. Willien, I. Chetvchenko, R. Masson, P. Quandalle, L. Agelas, and S. Requena, "AMG preconditioning for sedimentary basin simulations in Temis calculator," *Marine and Petroleum Geology*, vol. 26, no. 4, pp. 519–524, 2009.
[183] P. Thum, T. Clees, G. Weyns, G. Nelissen, and J. Deconinck, "Efficient algebraic multigrid for migration-diffusion-convection-reaction systems arising in electrochemical simulations," *Journal of Computational Physics*, vol. 229, no. 19, pp. 7260–7276, 2010.
[184] M. W. Gee, U. Küttler, and W. A. Wall, "Truly monolithic algebraic multigrid for fluid-structure interaction," *International Journal for Numerical Methods in Engineering*, vol. 85, no. 8, pp. 987–1016, 2011.
[185] H. Yang and W. Zulehner, "Numerical simulation of fluid-structure interaction problems on hybrid meshes with algebraic multigrid methods," *Journal of Computational and Applied Mathematics*, vol. 18, pp. 5367–5379, 2011.
[186] M. M. Resch, "High performance computer simulation for engineering: a review," in *Trends in Parallel, Distributed, Grid and Cloud Computing for Engineering*, P. Iványi and B. H. V. Topping, Eds., pp. 177–186, Saxe-Coburg Publications, Stirlingshire, UK, 2011.
[187] Y. Saad, "ILUM: a parallel multielimination ILU preconditioner for general sparse matrices," Tech. Rep. CS-92-241, Department of Computer Science, University of Minnesota, 1992.
[188] G. Karypis and V. Kumar, "Parallel threshold-based ILU factorization," in *Proceedings of the Supercomputing ACM/IEEE Conference*, pp. 1–24, 1997.
[189] D. Hysom and A. Pothen, "A scalable parallel algorithm for incomplete factor preconditioning," *SIAM Journal on Scientific Computing*, vol. 22, no. 6, pp. 2194–2215, 2001.
[190] M. Magolu monga Made and H. A. van der Vorst, "Parallel incomplete factorizations with pseudo-overlapped subdomains," *Parallel Computing*, vol. 27, no. 8, pp. 989–1008, 2001.
[191] M. Benzi and M. Tůma, "A parallel solver for large-scale Markov chains," *Applied Numerical Mathematics*, vol. 41, no. 1, pp. 135–153, 2002.
[192] G. Haase, M. Kuhn, and S. Reitzinger, "Parallel algebraic multigrid methods on distributed memory computers," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 410–427, 2002.
[193] H. de Sterck, U. M. Yang, and J. J. Heys, "Reducing complexity in parallel algebraic multigrid preconditioners," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 1019–1039, 2006.
[194] R. D. Falgout and U. M. Yang, "Hypre: a library of high performance preconditioners," in *Computational Science ICCS 2002 part III*, vol. 2331 of *Lecture Notes in Computer Science*, pp. 632–641, Springer, Berlin, Germany, 2002.
[195] M. A. Heroux, R. A. Bartlett, V. E. Howle et al., "An overview of the Trilinos project," *ACM Transactions on Mathematical Software*, vol. 31, no. 3, pp. 397–423, 2005.
[196] P. F. Dubois, A. Greenbaum, and G. H. Rodrigue, "Approximating the inverse of a matrix for use in iterative algorithms on vector processors," *Computing*, vol. 22, no. 3, pp. 257–268, 1979.
[197] O. G. Johnson, C. A. Micchelli, and G. Paul, "Polynomial preconditioners for conjugate gradient calculations," *SIAM Journal on Numerical Analysis*, vol. 20, no. 2, pp. 362–376, 1983.
[198] Y. Saad, "Practical use of polynomial preconditionings for the conjugate gradient method," *SIAM Journal on Scientific and Statistical Computing*, vol. 6, no. 4, pp. 865–881, 1985.
[199] E. L. Stiefel, "Kernel polynomials in linear algebra and their numerical applications," *US National Bureau of Standards Applied Mathematics Series*, no. 49, pp. 1–24, 1958.
[200] O. Axelsson, "A survey of preconditioned iterative methods for linear systems of algebraic equations," *BIT Numerical Mathematics*, vol. 25, no. 1, pp. 165–187, 1985.
[201] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, Mass, USA, 1996.
[202] C. Janna, M. Ferronato, and G. Gambolati, "A block FSAI-ILU parallel preconditioner for symmetric positive definite linear systems," *SIAM Journal on Scientific Computing*, vol. 32, no. 5, pp. 2468–2484, 2010.
[203] M. Ferronato, C. Janna, and G. Pini, "A generalized block FSAI preconditioner for nonsymmetric linear systems," *Journal of Computational and Applied Mathematics*. In Press.
[204] M. Ferronato, C. Janna, and G. Pini, "Efficient parallel solution to large-size sparse eigenproblems with block FSAI preconditioning," *Numerical Linear Algebra with Applications*, vol. 19, no. 5, pp. 797–815, 2012.
[205] L. Bergamaschi, A. Martínez, and G. Pini, "Parallel Rayleigh quotient optimization with FSAI-based preconditioning," *Journal of Applied Mathematics*, vol. 2012, Article ID 872901, 14 pages, 2012.

[206] C. Janna, N. Castelletto, and M. Ferronato, "Efficient coupling of graph partitioning techniques with the adaptive block FSAI parallel preconditioner," *ACM Transactions on Mathematical Software*. In Press.
[207] C. Janna, M. Ferronato, and G. Gambolati, "Enhanced Block FSAI preconditioning using domain decomposition techniques," *SIAM Journal on Scientific Computing*. In Press.
[208] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, vol. 23, Springer, Berlin, Germany, 1994.
[209] P. Wesseling, *Principles of Computational Fluid Dynamics*, vol. 29, Springer, Berlin, Germany, 2001.
[210] M. H. Wright, "Interior methods for constrained optimization," *Acta Numerica*, vol. 1, pp. 341–407, 1992.
[211] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, Pa, USA, 1997.
[212] I. Perugia, "A field-based mixed formulation for the two-dimensional magnetostatic problem," *SIAM Journal on Numerical Analysis*, vol. 34, no. 6, pp. 2382–2391, 1997.
[213] I. Perugia, V. Simoncini, and M. Arioli, "Linear algebra methods in a mixed approximation of magnetostatic problems," *SIAM Journal on Scientific Computing*, vol. 21, no. 3, pp. 1085–1101, 1999.
[214] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, vol. 15, Springer, New York, NY, USA, 1991.
[215] D. Braess, *Finite Elements. Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University Press, Cambridge, Mass, USA, 2nd edition, 2001.
[216] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta Numerica*, vol. 14, pp. 1–137, 2005.
[217] J. H. Prevost, "Partitioned solution procedure for simultaneous integration of coupled-field problems," *Communications in Numerical Methods in Engineering*, vol. 13, no. 4, pp. 239–247, 1998.
[218] G. Gambolati, G. Pini, and M. Ferronato, "Direct, partitioned and projected solution to finite element consolidation models," *International Journal for Numerical and Analytical Methods in Geomechanics*, vol. 26, no. 14, pp. 1371–1383, 2002.
[219] L. Lukšan and J. Vlček, "Indefinitely preconditioned inexact Newton method for large sparse equality constrained non-linear programming problems," *Numerical Linear Algebra with Applications*, vol. 5, no. 3, pp. 219–247, 1998.
[220] C. Keller, N. I. M. Gould, and A. J. Wathen, "Constraint preconditioning for indefinite linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1300–1317, 2000.
[221] I. Perugia and V. Simoncini, "Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations," *Numerical Linear Algebra with Applications*, vol. 7, no. 7-8, pp. 585–616, 2000.
[222] L. Bergamaschi, J. Gondzio, and G. Zilli, "Preconditioning indefinite systems in interior point methods for optimization," *Computational Optimization and Applications*, vol. 28, no. 2, pp. 149–171, 2004.
[223] H. S. Dollar, "Constraint-style preconditioners for regularized saddle point problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 2, pp. 672–684, 2007.
[224] C. Durazzi and V. Ruggiero, "Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems," *Numerical Linear Algebra with Applications*, vol. 10, no. 8, pp. 673–688, 2003.
[225] M. Benzi and V. Simoncini, "On the eigenvalues of a class of saddle point matrices," *Numerische Mathematik*, vol. 103, no. 2, pp. 173–196, 2006.
[226] L. Bergamaschi, J. Gondzio, M. Venturin, and G. Zilli, "Inexact constraint preconditioners for linear systems arising in interior point methods," *Computational Optimization and Applications*, vol. 36, no. 2-3, pp. 137–147, 2007.
[227] V. Simoncini, "Block triangular preconditioners for symmetric saddle-point problems," *Applied Numerical Mathematics*, vol. 49, no. 1, pp. 63–80, 2004.
[228] D. Silvester and A. Wathen, "Fast iterative solution of stabilised Stokes systems. II. Using general block preconditioners," *SIAM Journal on Numerical Analysis*, vol. 31, no. 5, pp. 1352–1367, 1994.
[229] D. J. Silvester, H. C. Elman, D. Kay, and A. J. Wathen, "Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow," *Journal of Computational and Applied Mathematics*, vol. 128, no. 1-2, pp. 261–279, 2001.
[230] H. C. Elman, D. J. Silvester, and A. J. Wathen, "Performance and analysis of saddle point preconditioners for the discrete steady-state Navier-Stokes equations," *Numerische Mathematik*, vol. 90, no. 4, pp. 665–688, 2002.

[231] M. Ferronato, L. Bergamaschi, and G. Gambolati, "Performance and robustness of block constraint preconditioners in finite element coupled consolidation problems," *International Journal for Numerical Methods in Engineering*, vol. 81, no. 3, pp. 381–402, 2010.
[232] O. Axelsson and M. Neytcheva, "Eigenvalue estimates for preconditioned saddle point matrices," *Numerical Linear Algebra with Applications*, vol. 13, no. 4, pp. 339–360, 2006.
[233] L. Bergamaschi, "On eigenvalue distribution of constraint-preconditioned symmetric saddle point matrices," *Numerical Linear Algebra with Applications*, vol. 19, no. 4, pp. 754–772, 2012.
[234] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, UK, 2005.
[235] L. Bergamaschi and A. Martínez, "FSAI-based parallel mixed constraint preconditioners for saddle point problems arising in geomechanics," *Journal of Computational and Applied Mathematics*, vol. 236, no. 3, pp. 308–318, 2011.
[236] J. B. Haga, H. Osnes, and H. P. Langtangen, "A parallel block preconditioner for large-scale poroelasticity with highly heterogeneous material parameters," *Computational Geosciences*, vol. 16, no. 3, pp. 723–734, 2012.
[237] C. Janna, M. Ferronato, and G. Gambolati, "Parallel inexact constraint preconditioning for ill-conditioned consolidation problems," *Computational Geosciences*, vol. 16, no. 3, pp. 661–675, 2012.