COMBINED PRECONDITIONING WITH APPLICATIONS IN RESERVOIR SIMULATION[∗]

XIAOZHE HU[†], SHUHONG WU[‡], XIAO-HUI WU[§], JINCHAO XU[¶], CHEN-SONG ZHANG[‖], SHIQUAN ZHANG[∗∗], AND LUDMIL ZIKATANOV[¶]

Abstract. We develop a simple algorithmic framework to solve large-scale symmetric positive definite linear systems. At its core, the framework relies on two components: (1) a norm-convergent iterative method (i.e., smoother) and (2) a preconditioner. The resulting preconditioner, which we refer to as a combined preconditioner, is much more robust and efficient than the iterative method and preconditioner when used in Krylov subspace methods. We prove that the combined preconditioner is positive definite and show estimates on the condition number of the preconditioned system. We combine an algebraic multigrid method and an incomplete factorization preconditioner to test the proposed framework on problems in petroleum reservoir simulation. Our numerical experiments demonstrate noticeable speed-up when we compare our combined method with the stand-alone algebraic multigrid method or the incomplete factorization preconditioner.

Key words. combined preconditioning, multigrid method, incomplete LU factorization, reservoir simulation

AMS subject classifications. 65F08, 65F10, 65N55, 65Z05

DOI. 10.1137/120885188

1. Introduction. We consider a sparse linear system of equations that arises from the discretizations of the elliptic partial differential equations (PDEs) used to simulate the physical processes in petroleum reservoirs. Petroleum reservoir simulation (PRS) is a tool for predicting hydrocarbon reservoir performance under various operating regimes. PRS helps engineers to obtain information pertaining to the processes that take place within oil reservoirs—information that can be used to maximize recovery and minimize environmental damage. The crucial aspect of PRS is its ability to solve large-scale discretized PDEs, which are strongly coupled, indefinite, and often nonsymmetric. Solving these linear systems consumes most of the computational time in all modern reservoir simulators—more than 75% of the computational time in general. Furthermore, the demand for more accurate simulations has led to larger discrete reservoir models. This increase in model size results in larger linear systems that are more difficult, or even impossible, to solve in an acceptable amount of time using standard direct or standard iterative solvers.

Over the last 30 years, incomplete LU (ILU) factorization has become one of the most commonly used methods for solving large sparse linear equations arising in PRS. First developed in the 1960s [6, 27], ILU methods provide an approximation of the exact LU factorization (computed via Gaussian elimination) by specifying the sparsity of the L and U factors. These methods need not be convergent when used in linear iterative procedures, but they do provide preconditioners that are used in Krylov subspace methods. Thought to have introduced the word *preconditioning*, Evans [11] may also have been the first to use ILU as a preconditioner. As noted, due to their simplicity, ILU methods are of particular interest to researchers in the field of reservoir simulation. However, when ILU preconditioners are applied to problems in PRS, their performance usually deteriorates as the number of grid-blocks increases.

To solve discretized scalar elliptic PDEs (Poisson-like PDEs), multigrid (MG) methods are efficient and provide a solution in optimal time and memory complexity in cases that allow standard coarse spaces to be used (see [13, 31, 2, 25] and references therein for details). However, the MG methods with standard coarse spaces make extensive use of the analytic information from the discretized equation and geometric information explicitly related to the discretization. This makes such methods difficult to use, such that in practice more sophisticated methods, such as algebraic multigrid (AMG) methods are preferred. There are many different types of AMG methods, such as classical AMG [22, 4], smoothed aggregation AMG [26, 5], and AMGe [16, 18]. However, despite their differences, they generally do not require geometric information from the grids. Interested readers can refer to [20, 29, 3, 8, 12, 28] and references therein for details on AMG methods. Due to their efficiency, scalability, and applicability, AMG methods have become increasingly popular in practice (see, e.g., [24]), including in modern PRSs [23, 9, 10, 15, 14]. However, the performance and efficiency of MG methods may degenerate as the physical and geometric properties of the problems become more complex. In such circumstances, there is an increase in AMG setup time (the time needed to construct coarse spaces and operators) and superfluous fill-in in the coarse grid operators, which makes applying relaxation more expensive on coarser levels.

In this paper, we provide a simple and transparent framework for combining preconditioners like ILU, Jacobi, or additive Schwarz with AMG or another norm-convergent iterative method for large-scale symmetric positive definite linear systems of equations. In the combined preconditioners we propose, the component provided by the norm-convergent iterative method need not be very effective when used alone. However, as we shall see, when it is paired with an existing preconditioner (such as an ILU preconditioner), the result is an efficient method. We show that the combined preconditioner is symmetric positive definite (SPD) under some mild assumptions. This guarantees the convergence of the conjugate gradient (CG) method for the preconditioned system. We apply the combined preconditioner to PRSs that involve highly heterogeneous media and unstructured grids in three spatial dimensions (with faults and pinch-outs). The numerical results show an improved performance that justifies using the combined preconditioners in PRS.

The rest of the paper is organized as follows. The algorithmic framework is introduced and analyzed in section 2. As an example in section 3, we formally present the combined preconditioner by describing its essential components, e.g., an ILU preconditioner and an AMG method. In section 4, we recall the mixed-hybrid finite element method used to provide a discretization of an important component of reservoir simulation—an elliptic PDE with heterogeneous coefficients. Numerical results showing the improvement in performance are presented in section 5, and some con-

cluding remarks are given in section 6.

2. Definition of the combined preconditioner. Consider the linear system

$$(2.1) \qquad Au = f,$$

with an SPD coefficient matrix A. Let $(\cdot, \cdot)$ be an inner product on a finite-dimensional Hilbert space V; its induced norm is $\|\cdot\|$. The adjoint of A with respect to $(\cdot, \cdot)$, denoted by A^T, is defined by $(Au, v) = (u, A^T v)$ for all $u, v \in V$. A is SPD if $A^T = A$ and $(Av, v) > 0$ for all $v \in V \backslash \{0\}$. As A is SPD with respect to $(\cdot, \cdot)$, the bilinear form $(A\cdot, \cdot)$ defines an inner product on V, denoted by $(\cdot, \cdot)_A$, and the induced norm of A is denoted by $\|\cdot\|_A$.

In what follows, we also need an operator S, referred to as a smoother. In some of the results, we distinguish two cases: (1) a *norm-convergent smoother*, i.e., $(I - SA)$ is a contraction in $\|\cdot\|_A$-norm; and (2) more generally, a *nonexpansive smoother*, i.e., $(I - SA)$ is a nonexpansive operator in $\|\cdot\|_A$ norm. Note that *norm-convergent* implies *nonexpansive*.

Next, in Algorithm 2.1, we combine a nonexpansive iterative method S with an existing SPD preconditioner B, and as a result we get a preconditioner B_{co}.

Algorithm 2.1

Given S, B, and $u^{k,0} = u^k$, the new iterate u^{k+1} is obtained by the following steps:

$$(1) \quad u^{k,1} = u^{k,0} + S(f - Au^{k,0}); \quad (2) \quad u^{k,2} = u^{k,1} + B(f - Au^{k,1});$$
$$(3) \quad u^{k+1} = u^{k,2} + S^T(f - Au^{k,2}).$$

It is easy to see that Algorithm 2.1 defines a linear operator B_{co} and that

$$(2.2) \qquad I - B_{\mathrm{co}}A = (I - S^T A)(I - BA)(I - SA).$$

From (2.2), it follows that

$$B_{\mathrm{co}} = \widetilde{S} + (B - S^T AB - BAS + S^T ABAS)$$
$$(2.3) \qquad = \widetilde{S} + (I - S^T A)B(I - AS).$$

Here $\widetilde{S}$ is the symmetrization of S, defined as

$$I - \widetilde{S}A = (I - S^T A)(I - SA), \qquad \widetilde{S} = S + S^T - S^T AS.$$

We now explain the properties pertaining to the notions of *norm-convergent* and *nonexpansive* smoothers, which we refer to in the subsequent sections.

- *Norm-convergent smoother.* In this case, $(I - SA)$ is a contraction in the $\|\cdot\|_A$-norm; i.e., there exists $\rho \in [0, 1)$, such that for all $v \in V$ we have

$$(2.4) \qquad \|(I - SA)v\|_A^2 \le \rho\|v\|_A^2.$$

Note that under this assumption, S is invertible. Indeed, we have

$$\|SAv\|_A = \|(I - (I - SA))v\|_A \ge \|v\|_A - \|(I - SA)v\|_A \ge (1 - \sqrt{\rho})\|v\|_A,$$

which means SA is coercive and implies that S is invertible. Recall also that (2.4) holds if and only if the symmetrization $\widetilde{S}$ is SPD; i.e.,

$$\|I - SA\|_A < \rho \quad \text{if and only if} \quad (\widetilde{S}v, v) > 0 \quad \forall \quad v \in V \setminus \{0\}.$$

- *Nonexpansive smoother.* This is a more general case in which we assume that

$$(2.5) \qquad \|(I - SA)v\|_A^2 \le \|v\|_A^2 \quad \forall v \in V.$$

One important difference between these two cases is this: with the nonexpansive case, $\widetilde{S}$ can be positive *semi*definite, whereas in the norm-convergent case, $\widetilde{S}$ is positive definite.

The following result shows that the operator B_{co} is SPD and, therefore, can be employed as a preconditioner for the preconditioned conjugate gradient (PCG) method. A preliminary version of the following theorem can be found in [33].

THEOREM 2.1. *Assume that $S : V \to V$ is a nonexpansive smoother. Moreover, the operator $B : V \to V$ is SPD. Then, B_{co} defined in (2.2) is SPD.*

Proof. For any $x \in V$, $x \ne 0$, we define $\tilde{x} := (I - SA)x$. Thus, we obtain

$$((I - B_{\mathrm{co}}A)x, x)_A = ((I - BA)(I - SA)x, (I - SA)x)_A$$
$$= ((I - BA)\tilde{x}, \tilde{x})_A = (\tilde{x}, \tilde{x})_A - (A\tilde{x}, A\tilde{x})_B,$$

where $(\cdot, \cdot)_B$ is an inner product on V defined as $(\cdot, \cdot)_B := (B\cdot, \cdot)$ since B is SPD.

Further, if $\tilde{x} = 0$, then $((I - B_{\mathrm{co}}A)x, x)_A = 0 < \|x\|_A^2$. On the other hand, for $\tilde{x} \ne 0$, we obtain from $(A\tilde{x}, A\tilde{x})_B > 0$ that

$$\|x\|_A^2 - (B_{\mathrm{co}}Ax, x)_A = ((I - B_{\mathrm{co}}A)x, x)_A < \|\tilde{x}\|_A^2 \le \|x\|_A^2 \quad \implies \quad (B_{\mathrm{co}}Ax, Ax) > 0.$$

As $x \in V$ is an arbitrary nonzero element, and A is nonsingular, we conclude that B_{co} is SPD. Finally, in view of (2.3) and because both A and B are symmetric, it is easy to see that B_{co} is symmetric. This completes the proof. $\square$

Remark 2.1. Note that the assumption in the theorem is that the error transfer for the convergent iteration $I - SA$ is nonexpansive in the A norm and does not require $I - SA$ to be a contraction.

Remark 2.2. Theorem 2.1 shows that the operator B_{co} is SPD. Hence, we can use the PCG method to solve (2.1) with B_{co} as a preconditioner, and in exact arithmetic, the PCG method will always be convergent. Algorithm 2.1 is relatively simple, but the order in which S and B are applied is crucial to the positive definiteness property of B_{co}. For example, consider the operator $\overline{B}$ defined by

$$(2.6) \qquad I - \overline{B}A = (I - BA)(I - \widetilde{S}A)(I - BA).$$

In this case, it is not true in general that $\overline{B}$ is positive definite. For example, if we use the ILU method to define a preconditioner B, finding the right scaling to ensure the positive definiteness of $\overline{B}$ could be a difficult task, and with indefinite $\overline{B}$ the PCG will not converge.

Remark 2.3. Note that directly using Algorithm 2.1 as an iterative method may not result in a convergent method. This is because though we assume that B is only SPD, we do not assume that $I - BA$ is a contraction. The contraction property of $I - BA$ (and also of $I - B_{\mathrm{co}}A$) is that any eigenvalue of BA satisfies $0 < \lambda(BA) \le \omega < 2$. However, we do not assume that such a contraction property holds for B.

2.1. Other derivations of the combined preconditioner. Here, we present two distinct points of view that can lead to algorithms such as Algorithm 2.1. One is from the point of view of the fictitious or auxiliary space techniques, developed in [19] and [32], and another is from the point of view of the block factorization

techniques, developed in [28, Chapter 5]. Below, we derive Algorithm 2.1 using these techniques. Of course, all three derivations lead to one and the same method, and they are equivalent to each other. However, they present different points of view and help us understand different aspects of the method.

2.1.1. Derivation via the auxiliary space method. An additive version of Algorithm 2.1 was derived via fictitious or auxiliary space techniques developed in [19] and [32]. Equivalently, Algorithm 2.1 can be viewed as a successive or multiplicative version of the auxiliary space additive preconditioner described below. Here, we use only one auxiliary space, which will turn out to be the same as V, but with a different inner product. We define

$$(2.7) \qquad \overline{V} = V \times W_1,$$

where W_1 is an auxiliary (Hilbert) space with an inner product

$$\overline{a}_1(\cdot, \cdot) = (\cdot, \cdot)_{A_1}.$$

We also introduce an operator $\Pi_1 : W_1 \mapsto V$ in order to define the additive preconditioner, and we define the latter as follows:

$$(2.8) \qquad \widehat{B} = \widetilde{S} + \Pi_1 A_1^{-1} \Pi_1^T.$$

A distinctive feature of the auxiliary space method is the presence of V in (2.7) as a component of $\overline{V}$ and the presence of the SPD operator $\widetilde{S} : V \mapsto V$. $\widetilde{S}$ is assumed to be SPD in order to guarantee that $\widehat{B}$ is SPD and can be applied as a preconditioner for the PCG method. It can be proved that the condition number of the preconditioned system can be bounded as follows (see [32]):

$$(2.9) \qquad \kappa(\widehat{B}A) \le c_0^2(c_s^2 + c_1^2)$$

if

$$\|\Pi_1 w_1\|_A^2 \le c_1^2 \|w_1\|_{A_1}^2, \qquad w_1 \in W_1,$$

and

$$(\widetilde{S}Av, v)_A \le c_s^2(v, v)_A \qquad \forall v \in V.$$

Moreover, for each $v \in V$, there exists $w_1 \in W_1$ such that

$$v = v_0 + \Pi_1 w_1, \quad v_0 = v - \Pi_1 w_1, \quad \text{and} \quad (\widetilde{S}Av_0, v_0)_A + (w_1, w_1)_{A_1} \le c_0^2(v, v)_A.$$

Taking $W_1 = V$ but with the inner product defined by B, $A_1^{-1} = B$, and $\Pi = I$ gives the additive version $\widehat{B}$ of the preconditioner B_{co}. Note that when $W_1 = V$, B is SPD and $\widetilde{S}$ is the symmetrization of the smoother S; the preconditioner $\widehat{B}$ is SPD for both the norm-convergent smoother and the nonexpansive smoother.

2.1.2. Derivation via block factorization. In this section, we present a derivation following the lines in [28, Chapter 5]. Let us introduce the operator $\overline{\overline{B}} : V \times V \mapsto V \times V$ in block factored form:

$$\overline{\overline{B}} = \begin{pmatrix} I & -S^T A \\ 0 & I \end{pmatrix} \begin{pmatrix} \widetilde{S} & 0 \\ 0 & B \end{pmatrix} \begin{pmatrix} I & 0 \\ -AS & I \end{pmatrix}.$$

We then set

$$\tag{2.10} \widetilde{\widetilde{B}} = (I, \quad I)\bar{\bar{B}} \begin{pmatrix} I \\ I \end{pmatrix}, \qquad \widetilde{\widetilde{B}} : V \mapsto V.$$

A straightforward calculation shows that B_{co} and $\widetilde{\widetilde{B}}$ are the same. Regarding the positive definiteness of $\widetilde{\widetilde{B}}$, note that for the norm-convergent smoother, it is immediately evident that $\widetilde{\widetilde{B}}$ is SPD, because both B and $\widetilde{S}$ are SPD. For the nonexpansive smoother, the positive definiteness of $\widetilde{\widetilde{B}}$ does not immediately follow from the form of the preconditioner given in (2.10). The arguments of Theorem 2.1 (or similar arguments) are needed to conclude that $\widetilde{\widetilde{B}}$ is SPD.

2.2. Effectiveness of B_{co}. In this section, we show that the combined preconditioner, under suitable scaling assumptions, performs no worse than its components. As the numerical tests show (see section 5), the combination of ILU (for B) and AMG (for S) performs significantly better than its components. Let us set

$$\tag{2.11} m_1 = \lambda_{\max}(BA) \quad \text{and} \quad m_0 = \lambda_{\min}(BA).$$

Without loss of generality (with proper scaling), we can assume that the preconditioner B is such that the following inequalities hold:

$$\tag{2.12} m_1 > 1 \geq m_0 > 0.$$

We now prove a result that compares $\kappa(B_{\mathrm{co}}A)$ with $\kappa(\widetilde{S}A)$ and $\kappa(BA)$ under the assumptions that B and S are such that both (2.12) and (2.4) are satisfied.

THEOREM 2.2. *If S is a norm-convergent smoother and B_{co} is defined as in* (2.2), *then*

$$\tag{2.13} \kappa(B_{co}A) \leq \frac{(1 - m_1)(1 - \rho) + m_1}{(1 - m_0)(1 - \rho) + m_0},$$

and

$$\tag{2.14} \kappa(B_{co}A) < \kappa(BA).$$

Furthermore, if S is such that (2.4) *holds with $\rho \geq 1 - \frac{m_0}{m_1 - 1}$, then*

$$\tag{2.15} \kappa(B_{co}A) \leq \kappa(\widetilde{S}A).$$

Proof. From the assumption stated in (2.12), we immediately conclude that B is SPD and $\kappa(BA) = m_1/m_0$. By the definition of $\widetilde{S}$, we have

$$0 \leq ((I - \widetilde{S}A)w, w)_A = ((I - SA)w, (I - SA)w)_A = \|(I - SA)w\|_A^2 \leq \rho\|w\|_A^2,$$

where we have used the assumption of the convergence of S in the last inequality. By choosing $v = Aw$, we can obtain

$$\tag{2.16} (1 - \rho)(A^{-1}v, v) \leq (\widetilde{S}v, v) \leq (A^{-1}v, v).$$

On the other hand, as $m_1 = \lambda_{\max}(BA)$ and $m_0 = \lambda_{\min}(BA)$, we have

$$m_0(A^{-1}v, v) \leq (Bv, v) \leq m_1(A^{-1}v, v).$$

By the definition of B_{co} (2.3), we have

$$
\begin{aligned}
(B_{\mathrm{co}}v, v) &= (\widetilde{S}v, v) + (B(I - AS)v, (I - AS)v) \\
&\le (\widetilde{S}v, v) + m_1(A^{-1}(I - AS)v, (I - AS)v) \\
&= (1 - m_1)(\widetilde{S}v, v) + m_1(A^{-1}v, v).
\end{aligned}
$$

Similarly, we can derive that

$$
\begin{aligned}
(B_{\mathrm{co}}v, v) &= (\widetilde{S}v, v) + (B(I - AS)v, (I - AS)v) \\
&\ge (\widetilde{S}v, v) + m_0(A^{-1}(I - AS)v, (I - AS)v) \\
&= (1 - m_0)(\widetilde{S}v, v) + m_0(A^{-1}v, v).
\end{aligned}
$$

As $m_1 > 1 \ge m_0$, we have

$$
[(1 - m_0)(1 - \rho) + m_0](A^{-1}v, v) \le (B_{\mathrm{co}}v, v) \le [(1 - m_1)(1 - \rho) + m_1](A^{-1}v, v);
$$

then the condition number of $B_{\mathrm{co}}A$ is bounded by

$$
\kappa(B_{\mathrm{co}}A) \le \frac{(1 - m_1)(1 - \rho) + m_1}{(1 - m_0)(1 - \rho) + m_0}.
$$

If $m_1 > 1$, then

$$
(1 - m_1)(1 - \rho) + m_1 < m_1,
$$

and if $1 \ge m_0 > 0$, then

$$
(1 - m_0)(1 - \rho) + m_0 \ge m_0.
$$

Note that

$$
\kappa(B_{\mathrm{co}}A) \le \frac{(1 - m_1)(1 - \rho) + m_1}{(1 - m_0)(1 - \rho) + m_0} < \frac{m_1}{m_0} = \kappa(BA).
$$

Hence, the inequality (2.14) holds if $m_1 > 1 \ge m_0 > 0$.

On the other hand, (2.15) follows from

(2.17) $$
\frac{(1 - m_1)(1 - \rho) + m_1}{(1 - m_0)(1 - \rho) + m_0} \le \frac{1}{1 - \rho} = \kappa(\widetilde{S}A),
$$

where the last equality comes from (2.16). Note that

$$
\frac{(1 - m_1)(1 - \rho) + m_1}{(1 - m_0)(1 - \rho) + m_0} = \frac{1 + \rho(m_1 - 1)}{(1 - \rho) + \rho m_0}.
$$

Note that $\frac{a+c}{b+d} \le \frac{a}{b}$ if $\frac{c}{d} \le \frac{a}{b}$ and $a, b, c, d > 0$. Therefore, (2.17) holds if $\frac{m_1 - 1}{m_0} \le \frac{1}{1 - \rho}$, i.e., $\rho \ge 1 - \frac{m_0}{m_1 - 1}$. □

Remark 2.4. If $\frac{m_1 - 1}{m_0} < \frac{1}{1 - \rho}$, then (2.15) becomes $\kappa(B_{\mathrm{co}}A) < \kappa(\widetilde{S}A)$.

Remark 2.5. If either $\widetilde{S}$ or B works well as a preconditioner itself, there is no need to use the more complicated B_{co}. However, we are interested in cases in which neither $\widetilde{S}$ nor B works effectively alone. In these cases, we use B_{co} to combine these

two, and we use Theorem 2.2 to guarantee that B_{co} will be a better preconditioner than either B or $\widetilde{S}$ under reasonable conditions. The condition

$$\rho \geq 1 - m_0/(m_1 - 1)$$

means that both $\widetilde{S}$ and B yield slow convergence as

$$\rho \geq 1 - m_0/(m_1 - 1) \approx 1 - \kappa(BA)^{-1} \quad \text{if } m_1 >> 1.$$

Therefore, if B is not a good preconditioner, i.e., $\kappa(BA)$ is large, then $\rho \approx 1$, which implies that $\widetilde{S}$ does not work well either. The new preconditioner $\widetilde{B}$ is no worse than B or $\widetilde{S}$ alone. In fact, the new preconditioner may be capable of performing much better than either B or $\widetilde{S}$ based on our numerical experiments (see section 5).

3. A simple example: ILU+AMG. Algorithm 2.1 provides an approach to combining ILU and AMG methods. In the combined use of ILU and AMG for the linear system (2.1), AMG serves as S and ILU serves as B. Next, we specify our choice of ILU and AMG for problems specific to reservoir simulation. However, we would like to emphasize that the combined preconditioner works for a wide range of iterative methods and preconditioners as long as they satisfy the assumptions in Theorem 2.1.

3.1. ILU factorization. ILU factorizations compute a sparse lower triangular matrix L and a sparse upper triangular matrix U so that the residual matrix $R = A - LU$ satisfies certain conditions. A general algorithm of ILU can be obtained by Gaussian elimination and dropping some of the elements in the off-diagonal positions. There are many variants of ILU preconditioners. They differ in terms of the rules that govern the prescribed fill-in of the factors during the ILU factorization procedure. For example, discarding the fill-in based on position gives the ILU(k) method; discarding the fill-in based on the values of the corresponding entries in the L or U factors gives the *threshold* ILU method. There are also ILU methods for which the fill-in is managed based on a combination of positions and values or based on other dropping strategies. For details about the different ILU methods, we refer the reader to [1, 7] and Saad [21]. Here we use the notation and terminology from Saad [21].

We consider ILU(k) (Algorithm 3.1), which to the best of our knowledge was originally introduced for reservoir simulations in [30]. The ILU is based on the *level of fill* to determine the off-diagonal positions in the L and U factors where the entries' fill-in will not be introduced (or, as is often said, off-diagonal positions for which the fill-in entries are *dropped*). Next, we define *level of fill*, and the detailed algorithm is given in Algorithm 3.1.

DEFINITION 3.1 (level of fill). *The initial level of fill of the elements of a sparse matrix A is defined as*

$$\mathcal{L}_{ij} = 0 \text{ if } a_{ij} \neq 0 \text{ or } i = j; \text{ otherwise } \mathcal{L}_{ij} = \infty.$$

When an entry a_{ij} is updated in the factorization procedure $a_{ij} := a_{ij} - a_{ik} * a_{kj}$, its level of fill is also updated by

$$\mathcal{L}_{ij} = \min\{\mathcal{L}_{ij}, \mathcal{L}_{ik} + \mathcal{L}_{kj} + 1\}.$$

Remark 3.1. Although the level of fill depends on the location of the element, the rationale is that the level of fill should indicate the magnitude of the element. The higher the level of fill, the smaller the element. For example, when the level of fill of a_{ij} is k this means that the size is $|a_{ij}| = O(\epsilon^k)$ for some $\epsilon < 1$.

Algorithm 3.1 ILU(k) method

---

1. For all nonzero elements a_{ij}, define $\mathcal{L}_{ij} = 0$
2. For $i = 1, \ldots, n$, Do
   For $m = 1, \ldots, i-1$ and $\mathcal{L}_{im} \leq k$, Do
      Compute $a_{im} := a_{im}/a_{mm}$
      Compute $a_{i*} := a_{i*} - a_{im}a_{m*}$, where a_{i*} is the ith row of A
      Update the levels of fill of nonzero by $\mathcal{L}_{ij} = \min\{\mathcal{L}_{ij}, \mathcal{L}_{ik} + \mathcal{L}_{kj} + 1\}$
   End
   Replace any element in row i with $\mathcal{L}_{ij} > k$ by zero
End

3.2. AMG methods. As an algebraic variant of MG methods, AMG methods are widely applicable and the focus of current intensive development. AMG methods have mesh-independent convergence rates and optimal computational complexity for a wide range of problems.

Any AMG method consists of two phases: the setup phase and the solve phase. In the setup phase, the intergrid operator P_l is constructed, and the coarse grid matrix is defined as

$$A_l = P_l^T A_{l+1} P_l, \quad l = L - 1, \ldots, 0,$$

and $A_L = A$. In the solve phase, the smoother S_l and coarse-grid correction are applied recursively, as shown in the following general V-cycle MG Algorithm 3.2.

Algorithm 3.2 V-cycle for solving $A_l u_l = f_l$, with an initial guess u_l^0

---

1. **Presmoothing:** $u_l^1 = u_l^0 + S_l(f_l - A_l u_l^0)$
2. **Coarse-grid correction:**
   (a) $f_{l-1} = P_{l-1}^T(f_l - A_l u_l^1)$
   (b) If $l = 1$, $e_0 = A_0^{-1} f_0$; else, apply Algorithm 3.2 for $A_{l-1} e_{l-1} = f_{l-1}$ with zero initial guess
   (c) $u_l^2 = u_l^1 + P_{l-1} e_{l-1}$
3. **Postsmoothing:** $u_l^3 = u_l^2 + S_l^T(f_l - A_l u_l^2)$

In AMG methods, P_l must be constructed based on algebraic principles, which presents certain challenges. The key to fast convergence is the complementary nature of operators S_l and P_l. That is, errors not reduced by S_l must be interpolated well by P_l. We choose the classical AMG as the iterative method S in our Algorithm 2.1. This method constructs the intergrid operator P_l in two steps. First, the classical C-F splitting method is used, and then the operator P_l is constructed by classical Ruge–Stüben interpolation; see [20] for details.

3.3. Application of Algorithm 2.1. In this paper, we consider SPD coefficient matrices A only. In this case, ILU(k) is replaced by an incomplete Cholesky (IC) factorization. It is easy to see that IC(k) is SPD, which satisfies the assumption on B in Theorem 2.1. For AMG methods, we have $A_l = P_l^T A_{l+1} P_l$, $l = L - 1, \ldots, 0$. A is SPD and P_l is constructed by the classical AMG method; therefore, it is guaranteed that A_l is also SPD for $l = 0, 1, \ldots, L$. Hence, it is easy to see that using the standard Gauss–Seidel (GS) smoother on each level gives a convergent iterative method. According to Algorithm 3.2, B_l, which stands for the V-cycle MG on level l

$(l = 1, 2, \ldots, L)$, can be defined recursively as

$$I - B_l A_l = (I - S_l^T A_l)(I - P_{l-1} B_{l-1} P_{l-1}^T A_l)(I - S_l A_l),$$

with $B_0 = A_0^{-1}$. As

$$\|I - S_l A_l\|_{A_l} < 1 \quad \text{and} \quad \|I - P_{l-1} B_{l-1} P_{l-1}^T A_l\|_{A_l} \leq 1,$$

by the induction assumption, we can show that

$$\|I - B_l A_l\|_{A_l} < 1.$$

Thus the classical AMG is convergent for SPD problems by mathematical induction. Hence, ILU(k) and the classical AMG can be combined using Algorithm 2.1 to yield the following corollary.

COROLLARY 3.2 (symmetric positive definiteness of ILU-AMG). *If we choose S from the classical AMG methods and B from ILU(k) in Algorithm* 2.1*, then the operator B_{co} defined in* (2.2) *is SPD.*

4. A model problem in reservoir simulation. PRS provides information about processes that take place within oil reservoirs. It is, therefore, of great assistance in efforts to achieve optimal recovery. Modern reservoir simulation faces increasingly complex physical models and uses highly unstructured grids. This results in Jacobian systems that are more difficult at each step. In this section, we describe a model problem in PRS and a discretization method that has great promise for efficient resource recovery, which is used for numerical tests in the next section.

Let Ω be a bounded domain in $\mathbb{R}^3$ in our consideration of the following second-order elliptic problem:

$$(4.1) \qquad\qquad -\nabla \cdot (a \nabla p) + cp = f \quad \text{in } \Omega.$$

In reservoir simulation, the model problem (4.1) usually occurs after temporal semi-discretization of the mathematical models that describe the multiphase flow in porous media. Such scalar problems also occur in more sophisticated preconditioning techniques for coupled systems of PDEs (see [33] for further details). The unknown function p in (4.1) is the pressure; $a \in [L^\infty(\Omega)]^{3\times3}$ is the diffusion tensor, and it usually depends on the permeability and viscosity, etc.; $c \in L^\infty(\Omega)$ is nonnegative and proportional to the inverse of the time step size in an implicit temporal scheme (such as the backward Euler method). We assume that $\partial\Omega$ has two nonoverlapping parts, Γ_D and Γ_N, such that $\overline{\Gamma_D \cup \Gamma_N} = \partial\Omega$. The model problem (4.1) is completed by the boundary conditions $p = g_D$ on Γ_D and $(a\nabla p)\cdot \boldsymbol{n} = g_N$ on Γ_N, where $\boldsymbol{n}$ is the outward unit normal vector of $\partial\Omega$, and where g_D and g_N are given.

The difficulties that arise from (4.1) in regard to solving the linear system (2.1) are mainly due to the complex geometry and complicated physical properties of the reservoir, which often result in a heterogeneous diffusion tensor with large jumps, and in distorted, degenerated, and/or nonmatching meshes with faults and pinch-outs. Figure 4.1 shows an example of the computational domain and the permeability of this reservoir in one horizontal layer.

We use the mixed-hybrid finite element method to discretize (4.1). Due to its local conservation property and intrinsic and accurate approximation of the flux $\boldsymbol{u} := -a\nabla p$, the mixed-hybrid method is preferred in reservoir simulations. Let $\Omega_h = \{T\}$
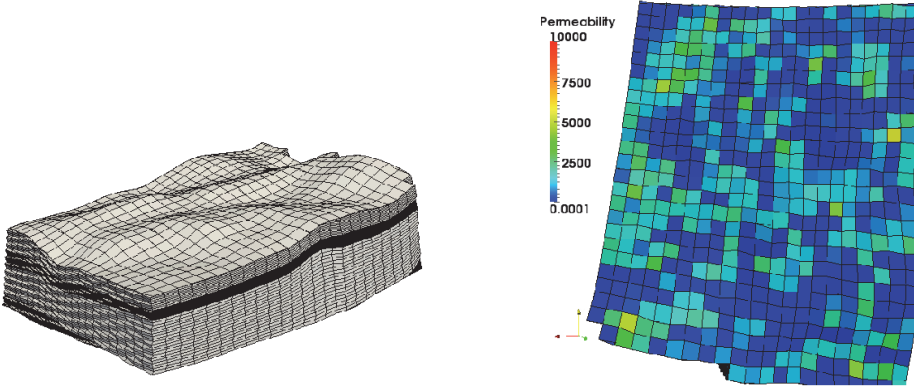
FIG. 4.1. *Left: A sample computational domain and mesh. Right: Permeabilities of the porous media in one horizontal layer (the magnitude of the variation of permeability is about 10^8).*

be a triangulation of Ω, and let Γ_h be the set of boundaries of T in Ω_h with the decomposition

$$\Gamma_h^\partial = \{\Gamma \in \Gamma_h : \Gamma \in \partial\Omega\}, \quad \Gamma_h^0 = \Gamma_h \setminus \Gamma_h^\partial.$$

We define the finite-dimensional spaces as

$$V_h := \{\boldsymbol{v} \in [L^2(\Omega)]^d : \boldsymbol{v}|_T \in V_h(T) \ \forall T \in \Omega_h\},$$
$$Q_h := \{q \in L^2(\Omega) : q|_T \in P_0(T) \ \forall T \in \Omega_h\},$$
$$\Lambda_h := \{\mu \in L^2(\Gamma_h) : \mu|_\Gamma \in P_0(\Gamma) \ \forall \Gamma \in \Gamma_h^0; \ \mu|_\Gamma = 0 \ \forall \Gamma \in \Gamma_h^\partial\},$$

where $V_h(T)$ is the Kuznetsov–Repin element on the polyhedral elements [17]. This element is based on the Raviart–Thomas element on the local conforming tetrahedral partitioning of T. $P_k(T)$ denotes the set of polynomials of a degree not greater than k, $k \geq 0$. V_h, Q_h, and Λ_h are used to approximate the flux $\boldsymbol{u}$; the pressure p is associated with elements in Ω_h; and the Lagrange multipliers λ_h are associated with faces in Ω_h.

The mixed-hybrid finite element formulation for (4.1) can be written as follows: Find $(\boldsymbol{u}_h, p_h, \lambda_h) \in V_h \times Q_h \times \Lambda_h$, $\boldsymbol{u}_h \cdot \boldsymbol{n} = -g_{N,h}$ on Γ_h, where $g_{N,h}$ is an appropriate approximation of g_N, such that

$$(a^{-1}\boldsymbol{u}_h, \boldsymbol{v}) - \sum_T \{(p_h, \text{div } \boldsymbol{v})_T - (\lambda_h, \boldsymbol{v} \cdot \boldsymbol{n}_T)_{\partial T}\} = -\int_{\Gamma_D} g_D(\boldsymbol{v} \cdot \boldsymbol{n})\mathrm{d}s \quad \forall \boldsymbol{v} \in V_h,$$

$$-\sum_T (\text{div } \boldsymbol{u}_h, q)_T - (cp_h, q) = -\int_\Omega fq\mathrm{d}x \qquad \forall q \in Q_h,$$

$$\sum_T (\mu, \boldsymbol{u}_h \cdot \boldsymbol{n}_T)_{\partial T} = 0 \qquad \forall \mu \in \Lambda_h,$$

where $\boldsymbol{n}_T$ denotes the outward unit normal to ∂T.

5. Numerical experiments. It is easy to see that the resulting discrete linear system of the mixed-hybrid finite element method in the previous section has the following matrix form:

$$\begin{pmatrix} \mathsf{D} & \mathsf{B}^T & \mathsf{C}^T \\ \mathsf{B} & -\mathsf{M} & 0 \\ \mathsf{C} & 0 & 0 \end{pmatrix} \begin{pmatrix} \widetilde{u} \\ \widetilde{p} \\ \widetilde{\lambda} \end{pmatrix} = \begin{pmatrix} \widetilde{f}_1 \\ \widetilde{f}_2 \\ 0 \end{pmatrix}.$$

Notice that the matrix D is block diagonal with each block corresponding to the flux unknowns in one element. Hence, we can easily invert D and obtain (2.1) with

$$(5.1) \quad A = \begin{pmatrix} \mathsf{BD}^{-1}\mathsf{B}^T + \mathsf{M} & \mathsf{BD}^{-1}\mathsf{C}^T \\ \mathsf{CD}^{-1}\mathsf{B}^T & \mathsf{CD}^{-1}\mathsf{C}^T \end{pmatrix}, \quad u = \begin{pmatrix} \widetilde{p} \\ \widetilde{\lambda} \end{pmatrix}, \quad \text{and } f = \begin{pmatrix} \mathsf{BD}^{-1}\widetilde{f}_1 - \widetilde{f}_2 \\ \mathsf{CD}^{-1}\widetilde{f}_1 \end{pmatrix},$$

where it is well known that A is SPD.

In the numerical experiments, performed on a Dell Precision desktop computer, we solve the above linear system by PCG with different preconditioners. We pick eight problems from the real petroleum reservoir data. Table 5.1 gives the degrees of freedom (DOFs) and the number of nonzeros (NNZ) for the test problems. Here, the difference between Models 1 (3) and 2 (4) is that the permeabilities in Model 1 (3) are homogeneous, whereas the permeabilities in Model 2 (4) have large jumps. Figure 4.1 shows the computational domain of Models 3 and 4 and the highly heterogeneous permeabilities used in Model 4. Models 5–8 are test problems that are relatively large in size. The computational domains of Models 5 and 6 are the same, but the physical properties, such as porosity and permeability, are different. Models 7 and 8 share the same computational domain, but the permeabilities used in Model 7 are from real reservoir data, and the permeabilities in Model 8 are artificially adjusted in order to make the resulting linear system very difficult to solve.

TABLE 5.1
Degree of freedom (DOF) and number of nonzeros (NNZ) of the model problems.

Test problem set 1	Model 1	Model 2	Model 3	Model 4
DOF	156,036	156,036	287,553	287,553
NNZ	1,620,356	1,620,356	3,055,173	3,055,173
Test problem set 2	Model 5	Model 6	Model 7	Model 8
DOF	1,291,672	1,291,672	4,047,283	4,047,283
NNZ	13,930,202	13,930,202	45,067,789	45,067,789

TABLE 5.2
Comparison of preconditioners for model problems (stopping criteria: relative residual less than 10^{-10}). In the table, — means the method does not converge within 10000 iterations, #Iter is the number of iterations, and the unit for CPU time is seconds.

Test problem set 1	Model 1		Model 2		Model 3		Model 4	
Preconditioner	#Iter	CPU	#Iter	CPU	#Iter	CPU	#Iter	CPU
ILU(0)	234	2.85	3034	32.91	291	6.56	4939	102.13
AMG(GS)	135	13.16	136	13.48	154	31.70	138	25.78
AMG(ILU(0))	5364	631.72	1058	115.71	—	—	4849	1024.21
$\widehat{B}$ (additive)	44	6.18	43	5.53	44	11.40	49	11.33
B_{co} (multiplicative)	23	4.04	25	3.92	22	7.36	28	7.78
Test problem set 2	Model 5		Model 6		Model 7		Model 8	
Preconditioner	#Iter	CPU	#Iter	CPU	#Iter	CPU	#Iter	CPU
ILU(0)	—	—	—	—	838	271.46	2039	646.77
AMG(GS)	30	36.70	30	37.07	19	96.03	22	112.79
AMG(ILU(0))	15	32.16	15	31.58	11	96.72	11	103.52
$\widehat{B}$ (additive)	32	39.72	32	39.87	26	100.78	25	104.29
B_{co} (multiplicative)	19	25.33	19	25.54	14	76.92	14	77.88

Table 5.2 compares different preconditioners for the model problems. The CPU time listed in Table 5.2 is the total CPU time, which includes both the setup and

solver times. We consider only the ILU(0) preconditioner due to memory-usage considerations. As we use highly unstructured meshes, the amount of fill-in for obtaining ILU(k) factorizations is not predictable for $k > 0$. Here, the AMG(GS) refers to the V-cycle MG method with standard GS smoother on each level. The AMG(ILU(0)) refers to the V-cycle MG method with an ILU(0) smoother on the finest level and a GS smoother on the other levels. In both cases, two cycles are applied in order to define the preconditioner. B_{co} is defined in (2.2) with one V-cycle MG; a GS smoother on each level serves as the smoother S, and ILU(0) serves as the preconditioner B. $\widehat{B}$ is its additive version defined in (2.8) with the special choices of $J = 1$, $W_1 = V$, and $\Pi_1 = I$, and A_1^{-1} is substituted by the preconditioner B.

The ILU(0) preconditioner works well for Models 1 and 3 because these two problems are relatively small in size and their permeabilities are homogeneous. We can see that the ILU(0) preconditioner does not work well for problems that are highly heterogeneous (Models 2 and 4). It is also not efficient for large-size problems (Models 5–8). Its performance is not predictable, and sometimes it may even break down (Models 5 and 6). AMG with a standard GS smoother works better than ILU(0) but still requires more than 100 iterations for Models 1–4. If we use ILU(0) as the smoother at the finest level, the resulting AMG preconditioner may not be SPD because ILU(0) might not converge. From the numerical results, we can see that for Models 1–4, using AMG with the ILU(0) smoother does not work as quickly as using AMG with the GS smoother. The former method may break down due to the lack of an SPD property (Model 3). However, if we combine AMG and ILU(0) as in Algorithm 2.1, the newly defined preconditioner B_{co} with AMG and ILU(0) gives the best performance in terms of CPU time, and it performs efficiently and robustly with respect to the problem size and heterogeneity. The additive algorithm $\widehat{B}$ also works efficiently and robustly. The numerical tests confirm our theoretical results and show the efficiency of our new approach for practical reservoir simulation problems.

6. Concluding remarks. In this paper, we discussed practical and efficient solvers for large sparse linear systems and their applications in petroleum reservoir simulation (PRS). We propose a simple and efficient framework for obtaining a new preconditioner by combining a convergent iterative method (such as an AMG method with a standard smoother) with an existing preconditioner (such as an ILU method). We used this approach to solve large linear systems arising from PRS (with highly heterogeneous media and three-dimensional unstructured grids). Our numerical results show that the new preconditioners significantly improve the efficiency of the linear solvers and are robust with respect to the size and heterogeneity of practical problems. This method has been implemented and employed in real reservoir simulation by ExxonMobil Upstream Company and RIPED, PetroChina.

REFERENCES

[1] M. BENZI, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.
[2] J. BRAMBLE, *Multigrid Methods*, Chapman & Hall/CRC, Boca Raton, FL, 1993.
[3] A. BRANDT, *General highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–20.

[4] A. BRANDT, S. MCCORMICK, AND J. RUGE, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications (Loughborough, 1983), Cambridge University Press, Cambridge, UK, 1985, pp. 257–284.

[5] M. BREZINA, R. FALGOUT, S. MACLACHLAN, T. MANTEUFFEL, S. MCCORMICK, AND J. RUGE, *Adaptive smoothed aggregation (αSA) multigrid*, SIAM Rev., 47 (2005), pp. 317–346.

[6] N. I. BULEEV, *A numerical method for the solution of two-dimensional and three-dimensional equations of diffusion*, Math. Sb., 51 (1960), pp. 227–238.

[7] T. F. CHAN AND H. A. VAN DER VORST, *Approximate and incomplete factorizations*, in Parallel Numerical Algorithms (Hampton, VA, 1994), ICASE/LaRC Interdiscip. Ser. Sci. Eng. 4, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997, pp. 167–202.

[8] A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, J. E. JONES, T. A. MANTEUFFEL, S. F. MCCORMICK, G. N. MIRANDA, AND J. W. RUGE, *Robustness and scalability of algebraic multigrid*, SIAM J. Sci. Comput., 21 (2000), pp. 1886–1908.

[9] T. CLEES AND L. GANZER, *An efficient algebraic multigrid solver strategy for adaptive implicit methods in oil-reservoir simulation*, SPE J., 15 (2010), pp. 670–681.

[10] O. DUBOIS, I. MISHEV, AND L. ZIKATANOV, *Energy minimizing bases for efficient multiscale modeling and linear solvers in reservoir simulation*, in Proceedings of the SPE Reservoir Simulation Symposium, SPE International, The Woodlands, TX, 2009, 119137.

[11] D. J. EVANS, *The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices*, J. Inst. Math. Appl., 4 (1968), pp. 295–314.

[12] R. FALGOUT, *An introduction to algebraic multigrid*, Comput. Sci. Engrg., 8 (2006), pp. 24–33.

[13] W. HACKBUSCH, *Multi-grid Methods and Applications, Vol.* 4, Springer-Verlag, New York, 1985.

[14] X. HU, W. LIU, G. QIN, J. XU, Y. YAN, AND C. ZHANG, *Development of a fast auxiliary subspace pre-conditioner for numerical reservoir simulators*, in Proceedings of the SPE Reservoir Characterization and Simulation Conference, SPE International, Abu Dhabi, United Arab Emirates, 2011, 148388.

[15] Y. JIANG AND H. TCHELEPI, *Scalable multi-stage linear solver for coupled systems of multi-segment wells and complex reservoir models*, in Proceedings of the SPE Reservoir Simulation Symposium, SPE International, The Woodlands, TX, 2009, 119175.

[16] J. E. JONES AND P. S. VASSILEVSKI, *AMGe based on element agglomeration*, SIAM J. Sci. Comput., 23 (2001), pp. 109–133.

[17] Y. KUZNETSOV AND S. REPIN, *New mixed finite element method on polygonal and polyhedral meshes*, Russian J. Numer. Anal. Math. Modelling, 18 (2003), pp. 261–278.

[18] I. LASHUK AND P. S. VASSILEVSKI, *On some versions of the element agglomeration AMGe method*, Numer. Linear Algebra Appl., 15 (2008), pp. 595–620.

[19] S. V. NEPOMNYASCHIKH, *Decomposition and fictitious domain methods for elliptic boundary value problems*, in Proceedings of the 5th International Symposium on Domain Decomposition Methods for Partial Differential Equations (Norfolk, VA, 1991), SIAM, Philadelphia, 1992, pp. 62–72.

[20] J. W. RUGE AND K. STÜBEN, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.

[21] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[22] K. STÜBEN, *Algebraic multigrid (AMG): Experiences and comparisons*, Appl. Math. Comput., 13 (1983), pp. 419–451.

[23] K. STÜBEN, P. DELANEY, AND S. CHMAKOV, *Algebraic multigrid (AMG) for ground water flow and oil reservoir simulation*, in Proceedings of the Conference MODFLOW and More, Integrated GroundWater Modeling Center, Colorado School of Mines, Golden, CO, 2003, pp. 17–19.

[24] U. TROTTENBERG AND T. CLEES, *Multigrid software for industrial applications—from MG*00 *to SAMG*, in Notes on Numerical Fluid Mechanics, E. Hirschel and E. Krause, eds., Notes on Numerical Fluid Mechanics and Multidisciplinary Design 100, Springer, Berlin, Heidelberg, 2009, pp. 423–436.

[25] U. TROTTENBERG, C. OOSTERLEE, AND A. SCHÜLLER, *Multigrid*, Academic Press, New York, 2001.

[26] P. VANĚK, J. MANDEL, AND M. BREZINA, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[27] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[28] P. S. VASSILEVSKI, *Multilevel Block Factorization Preconditioners*, Springer, New York, 2008.

[29] C. WAGNER, *Introduction to Algebraic Multigrid*, Course Notes of an Algebraic Multigrid Course, University of Heidelberg, Heidelberg, Germany, 1999.

[30] J. WATTS, III, *A conjugate gradient-truncated direct method for the iterative solution of the reservoir simulation pressure equation*, Old SPE J., 21 (1981), pp. 345–353.

[31] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

[32] J. Xu, *The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids*, Computing, 56 (1996), pp. 215–235.

[33] J. Xu, *Fast Poisson-based solvers for linear and nonlinear PDEs*, in Proceedings of the International Congress of Mathematics, Vol. 4, R. Bhatia, ed., World Scientific, Singapore, 2010, pp. 2886–2912.