PROCEEDINGS, 44th Workshop on Geothermal Reservoir Engineering
Stanford University, Stanford, California, February 11-13, 2019
SGP-TR-214

Performance Evaluation of a Physics-Based Multi-Stage Preconditioner in Numerical Simulation of Coupled Fluid and Heat Flow in Porous Media

Xiangyu Yu, Shihao Wang and Yu-Shu Wu

1911 Digger Dr, Unit 104, Golden, CO, 80401

xiyu@mymail.mines.edu

Keywords: physics-based preconditioner, iterative solver, coupled fluid and heat flow modeling, geothermal reservoir simulation

ABSTRACT

Modern reservoir simulation relies on iterative methods to solve the large sparse linear system. These efficient iterative methods are often built on the projection method and Krylov subspace which ensure the convergence behavior. However, iterative methods, applicable to general large sparse matrices, might encounter problems of instability or inefficiency in various simulation models. In addition to the commonly used preconditioners which are also employed for general sparse systems, such as Incomplete LU (ILU) factorization, specific physics-based preconditioners should also be developed for performance improvement of iterative methods. Constrained Pressure Residual (CPR) is proposed by Wallis (1983) to solve pressure equations first for damping the low-frequency error components before ILU preconditioning, which is proven to be effective in improving the convergence of iterative solver. The combination of CPR-ILU(0) is widely used in the black-oil reservoir simulator. However, this strategy could be less efficient for thermal models, due to the incorporation of energy equation into the linear system. In geothermal reservoir simulation, it is necessary to seek a more efficient physics-based preconditioner for large-scale thermal models.

Li et al. (2015) put forward a physics-based multi-stage preconditioner, named as SWIFT, aiming at the large magnitude terms generated by thermal energy equations. This preconditioner first scaled the matrix diagonally and applied row-column equilibration in order to transform the linear system into a near diagonally dominant one. Through this transformation, the Crout version of ILU (ILUC) can be used without partial pivoting and become capable of improving the accuracy of factorization by adaptively exploiting the sparsity and modifying fill-in terms. This preconditioner combining CPR, SWIFT and ILUC is observed to significantly improve the convergence performance of iterative methods in thermal models. In this paper, CPR-SWIFT-ILUC multi-stage preconditioner is implemented and applied into geothermal reservoir simulation. The effectiveness of CPR-SWIFT-ILUC is evaluated in several cases. The comparison of results shows that CPR-SWIFT-ILUC can help iterative methods achieve convergence efficiently in difficult thermal models. The overall efficiency with respect to computational time using CPR-SWIFT-ILUC is mainly dependent on problem complexity and should be determined by the simulator based on convergence conditions.

1. INTRODUCTION

Numerical modeling of coupled fluid and heat flow is a powerful tool to simulate the subsurface process that often occurs in geothermal energy development and thermal oil recovery. This technique can be applied widely for planning geothermal engineering projects or optimizing thermal oil recovery (Faust and Mercer, 1979; O'Sullivan et al., 2001). One difficulty that emerges in numerical simulation is impractical computation time for large or field scale cases, especially when it is necessary to incorporate heterogeneity, hydraulic and natural fractures and complex injection/production wells. Modern reservoir simulation usually adopts fully implicit method (FIM) for time stepping after space discretization, through which all primary unknowns of the linearized system are solved simultaneously by a certain linear solver. Consequently, a large sparse linear system, with a dimension of as high as millions, is to be solved at each Newton iteration, if a large-scale model with millions of grids is under consideration. Iterative methods, such as Generalized Minimum RESidual Method (GMRES) or BiConjugate Gradient Stabilized Method (BiCGStab), have become common options for solving a general large sparse linear system. Although direct solution techniques are advantageous for robustness and predictability, iterative methods have been improved on their stability due to the assistance of preconditioning. Nowadays, three-dimensional large-scale models cannot be solved efficiently without the acceleration of iterative methods (Saad, 2003). However, the overall solution of large sparse linear system still dominates the computational time in numerical simulation, leading to an increasing effort on the investigation of preconditioner improvement.

Traditional and basic iterative methods include Jacobi, Gauss-Seidel (GS) and Successive Over Relaxation (SOR) approaches, where the initial approximation was modified through each iteration, also known as a relaxation step, in order to gradually reduce the residual vector. Despite the fact that these basic methods are mathematically intuitive and easy to implement, the convergence performance differs for different matrices of realistic models. In such cases, an optimal acceleration parameter is needed for convergence rate, which could cost significant computational efforts, especially for large sparse linear systems (Saad, 2003). It was proposed that a projection method could be applied to approximate the solution in Krylov subspace of a smaller dimension, which is spanned by a series of sparse matrix polynomials, by imposing a certain amount of constraints (Saad, 1981). GMRES algorithms first adopted Arnoldi's approach (Arnoldi, 1951) that generates the orthonormal basis of Krylov subspace, followed by minimizing the residual vector. As the dimension of Krylov subspace increases with iterations, computation and memory cost grow rapidly beyond practical implementation. One of the remedies is to restart generating the Krylov subspace using the latest solution approximation until convergence.

The iterative methods, based on the projection method and Krylov subspace, are observed to suffer from slow convergence when applied in fluid dynamics simulation. Preconditioning is a technique used for resolving such issues and making iterative methods much more feasible for large sparse matrix. Generally speaking, any transformation of the large sparse linear system that reduces the difficulty of iterative solution can be regarded as a preconditioner, and hence, the iterative solution process becomes a preconditioned iteration. It is desired, in one case, that the preconditioner should close to the large matrix so that the preconditioned matrix for iteration methods approaches identity matrix. Incomplete LU (ILU) factorization, for example, was brought under the inspiration of the traditional approaches, such as Symmetric Successive Over Relaxation (SSOR) and Symmetric Gauss-Seidel (SGS). ILU process computes a lower and an upper triangular matrix satisfying a certain condition that leads to a specific type of preconditioner, such as, zero fill-in ILU (ILU(0)), level of fills-in ILU (ILU(p)), threshold based dropping ILU (ILUT(p,τ)) (Saad, 1994) and Crout version of ILU (ILUC(p,τ)) (Li et al., 2003). Based on various means of handling Gauss Elimination, these strategies of ILU provide different levels of factorization accuracy, hence different levels of acceleration for iterative solvers.

Nonetheless, it is widely known that the ILU approach is a preconditioner that focuses on eliminating high-frequency error components of iterative process for general matrices. In the process of seeking numerical solutions for reservoir simulation, low-frequency errors are retained after ILU preconditioning and gradually slows down the convergence (Wallis et al., 1985; Cao et al., 2005; Li et al., 2015). It is deemed that pressure equations of elliptic character exhibits long-range global coupling and are responsible for the low-frequency errors (Cao et al., 2005). Constrained Pressure Residual (CPR) is applied to solve the pressure equations as a first stage preconditioner before a second stage ILU preconditioning, resulting in a better convergence performance (Wallis, 1983; Cao et al., 2005; Cusini et al., 2015; Li et al., 2015). CPR is identified as a physics-based preconditioner since it aims at the special character of pressure equation in reservoir simulation (Wang, 2015). Substantial techniques have been combined with CPR to pursue a better convergence performance, such as Algebraic Multi-Grid (AMG) and domain decomposition (Cao et al., 2005; Cusini et al., 2015; Li et al., 2015). Among these works, Li et al. (2015) proposed a methodology for multi-stage preconditioning of the iterative solver in large-scale thermal models, which also contains a physics-based preprocessing of the second-stage preconditioning, named as Strong Weak Incomplete Factorization with Threshold (SWIFT). This method implemented a matrix diagonal scaling and row-column equilibration before employing ILUT or ILUC, taking into account the character of Jacobian matrix generated from a thermal model. A parallel version of this algorithm is proved to be efficient in application of thermal oil recovery related simulation.

In this study, SWIFT as a preprocessing of the second stage preconditioner following the first stage preconditioner CPR, is implemented for GMRES and applied in geothermal reservoir simulation. The performance of an iterative solver is evaluated and compared for different geothermal reservoir simulation cases. It can be observed that the performance of GMRES is significantly enhanced by CPR-SWIFT, especially for those difficult problems. On the other hand, due to the increased computational time for ILUT and ILUC, SWIFT need to be adopted appropriately considering the overall computational time for linear solutions. The structure of this paper will be as follows: the algorithm of GMRES, ILUC, and CPR will be briefly introduced, followed by the explanation of SWIFT algorithm. Then, the performance of CPR-SWIFT-ILUC preconditioner will be compared for different simulation cases; The last part will be discussion and conclusion based on the evaluation and comparison.

2. AN ITERATIVE METHOD FOR LARGE SPARSE LINEAR SYSTEM: GMRES (RESTARTED)

The methodology of an iterative solver, GMRES, will be briefly introduced. As mentioned above, GMRES is based on the projection method which approximates solution in Krylov subspace. The projection method, Krylov subspace and GMRES (restarted) algorithm will be explained. Although this methodology was put forward decades ago (Saad and Schultz, 1986), it is among the most popular and powerful iterative approaches for solving non-symmetric large linear system nowadays, and has been implemented in most of linear solver software packages.. The principles behind GMRES lay the foundation for other iterative methodology as well.

2.1 Projection Methods

The main objective of the projection method is extracting solution approximation from a subspace $\mathcal{K}$ by imposing orthogonal constraints. Given a non-symmetric linear system of dimension n:

$$\mathrm{A}\mathrm{x} = \mathrm{b} \tag{1}$$

It is desired to approximate the solution from a subspace of much smaller dimension m, $\mathcal{K}_\mathrm{m}$. At the same time, the orthogonal constraints can be described as that the residual vector should be orthogonal to m linearly independent vectors, which defines a subspace $\mathcal{L}_\mathrm{m}$ of dimension m as well. If A is a $\mathrm{n} \times \mathrm{n}$ real matrix, and $\mathcal{K}_\mathrm{m}$ and $\mathcal{L}_\mathrm{m}$ are m dimensional subspace of $\mathbb{R}^\mathrm{n}$, a projection approach is the way of finding a solution $\overline{x}$ belonging to $\mathcal{K}_\mathrm{m}$ and the residual vector orthogonal to $\mathcal{L}_\mathrm{m}$:

$$\text{Find } \ \overline{\mathrm{x}} \in \mathrm{x}_0 + \mathcal{K}_\mathrm{m}, \text{such that } \ \mathrm{b} - \mathrm{A}\overline{\mathrm{x}} \perp \mathcal{L}_\mathrm{m} \tag{2}$$

in which x_0 is the initial guess and $\overline{\mathrm{x}}$ can be expressed as $\overline{\mathrm{x}} = \mathrm{x}_0 + \delta$. Therefore, equation (2) can be rewritten as $\mathrm{r}_0 - \mathrm{A}\delta \perp \mathcal{L}_\mathrm{m}$ where $\delta \in \mathcal{K}_\mathrm{m}$. It can be seen that the norm of the new residual is always smaller than the original residual due to that the projection is performed orthogonal to $\mathcal{L}_\mathrm{m}$, $\| \mathrm{r}_0 - \mathrm{A}\delta \|_2 \leq \| \mathrm{r}_0 \|_2$, indicating that projection method should reduce residual norm if iterations of projection are conducted. Saad (2003) proved that when A is nonsingular and $\mathcal{L}_\mathrm{m} = \mathrm{A}\mathcal{K}_\mathrm{m}$, approximate solution $\overline{\mathrm{x}}$ can be well defined. Under this circumstance, if and only if $\overline{\mathrm{x}}$ minimizes the 2-norm of the residual vector $\mathrm{b} - \mathrm{A}\overline{\mathrm{x}}$ $(= \mathrm{r}_0 - \mathrm{A}\delta)$ in the subspace $\overline{\mathrm{x}} \in \mathrm{x}_0 + \mathcal{K}_\mathrm{m}$ $(\delta \in \mathcal{K}_\mathrm{m})$, $\overline{\mathrm{x}}$ is resulted from the projection onto $\mathcal{K}_\mathrm{m}$ orthogonal to $\mathcal{L}_\mathrm{m}$, as shown in Figure 1.

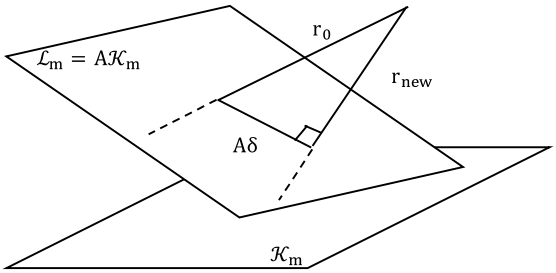

Figure 1 Illustration of the projection process orthogonal to a subspace $\mathcal{L}_m$, δ belongs to subspace $\mathcal{K}_m$ and $A\delta$ belongs to subspace $\mathcal{L}_m$ due to the orthogonal projection.

2.2 Krylov Subspace and Arnoldi's Method

As shown in the previous section, the projection method ensures the minimization of residual norm in the whole subspace $x_0 + \mathcal{K}_m$. An appropriate approximation subspace should be picked and the constraint subspace $\mathcal{L}_m$ will be determined accordingly. Cayley-Hamilton theorem demonstrates that the inverse of a matrix can be expressed as a polynomial of the matrix itself:

$$A^{-1} = \frac{(-1)^{n-1}}{\det(A)}(A^{n-1} + c_{n-1}A^{n-2} + \cdots + c_1 I_n) \tag{3}$$

in which $\det(A)$ is the determinant of matrix A, A^k is k^{th} polynomial of matrix A, and c_k is the coefficient that is to be determined, which could cost significant computational time, if the dimension n is large. Nonetheless, equation (3) motivates us to construct a subspace to which A^{-1} belongs to: $\mathrm{span}\{I_n, A, A^2, \dots, A^{n-1}\}$ so that the solution of linear system of equation (1) should belong to a subspace called Krylov subspace (Yang, 2015):

$$\mathcal{K}_n(A, r_0) = \mathrm{span}\{r_0, Ar_0,\ A^2 r_0, \dots, A^{n-1} r_0\} \tag{4}$$

The dimension of this subspace must be reduced for practical implementation, so that the solution should be approximated in a much smaller dimension subspace, with the dimension of m:

$$\mathcal{K}_m(A, r_0) = \mathrm{span}\{r_0, Ar_0,\ A^2 r_0, \dots, A^{m-1} r_0\} \tag{5}$$

The dimension of the above subspace increases as the approximation process (projection step) proceeds. It can be proven that the vectors shown in equation (5) forms a basis of Krylov subspace, since they are linearly independent vectors. Despite the fact that equation (5) has already formed a basis, Arnoldi's method, based on the methodology of Gram-Schmidt procedure, is employed to build an orthogonal basis for the Krylov subspace $\mathcal{K}_m$. This is mainly because Arnoldi's algorithm would produce a Hessenberg matrix which can be directly used for minimizing the residual vector norm as mentioned in the section of projection method, $\| b - Ax \|_2 = \| r_0 - A\delta \|_2$. Detailed algorithms and implementation of different versions of orthogonalization for Arnoldi's approaches are shown in Saad (2003). One version will be explained here in the next section.

2.3 GMRES-restarted Algorithms

GMRES is essentially a projection method, extracting the approximate solution from Krylov subspace $\mathcal{K}_m$ by imposing the constraints $\mathcal{L}_m = A\mathcal{K}_m$, due to its intent of minimizing the residual vector norm (as mentioned in section 2.1). The algorithm includes Arnoldi's method for generation of an orthonormal basis for Krylov subspace and Hessenberg matrix for minimizing the norm of the residual vector. A vector in subspace $x_0 + \mathcal{K}_m$ can be expressed as $x = x_0 + V_m y_m$, in which V_m is the orthonormal basis generated by Arnoldi's iteration and y_m is a vector of m dimension. Consequently, GMRES approximation is the vector $x_0 + V_m y_m$ that minimizes:

$$\| b - Ax \|_2 = \| b - A(x_0 + V_m y) \|_2 = \| r_0 - AV_m y \|_2 = \| \beta e_1 - \overline{H}_m y \|_2 \tag{6}$$

where $\overline{H}_m$ is the $(m+1) \times m$ Hessenberg matrix, $e_1 = (1, 0, \dots, 0)_{(m+1)\times 1}$ and $\beta = \| r_0 \|_2$. As can be seen, the minimization problem is transformed into finding a y_m for a minimum of $\| \beta e_1 - \overline{H}_m y \|_2$, which can be regarded as a least-square problem of small dimensions.

As the dimension of Krylov subspace increases, the computational and memory cost both increase rapidly beyond practical algorithm implementation, especially when n is large. One of the means of resolving this issue is assigning the latest x_m to x_0 and restarting to construct the Krylov subspace. This approach is known as restarted GMRES, where maximum dimension of Krylov subspace can be specified as restarting criteria. As shown in the algorithm below, line 2-10 implements Arnoldi's iteration for orthonormal basis V_m, and computes the Hessenberg matrix elements; Line 11-12 performs minimization calculation and updates approximation, x_m.

Combining Arnoldi's method and minimization procedure, restarted GMRES algorithm can be summarized as below (Saad, 2003):

1. *Initialize* $r_0 = b - Ax_0$, $\beta := \| r_0 \|_2$ *and* $v_1 = r_0/\beta$

2. *For j=1,2, …, m Do:*

3. $w_j \coloneqq Av_1$

4. *For i=1, …, j Do:*

5. $h_{ij} \coloneqq (w_j, v_i)$

6. $w_j \coloneqq w_j - h_{ij}v_i$

7. *EndDo*

8. $h_{j+1,j} = \| w_j \|_2$.

9. $v_{j+1} = \frac{w_j}{h_{j+1,j}}$

10. *EndDo*

11. *Apply* $(m+1) \times m$ *Hessenberg matrix* $\overline{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$

12. *Compute* y_m *that minimizes* $\| \beta e_1 - \overline{H}_m y \|_2$ *and* $x_m = x_0 + V_m y_m$

13. *If satisfy the convergence, then stop*

14. *Else* $x_0 \coloneqq x_m$ *and Restart from 1*

Restarted GMRES is available in MATLAB and this embedded iterative solver is used in this study for performance evaluation.

3. PRECONDITIONER TECHNIQUES FOR ITERATIVE METHODS

Preconditioner could be any operator that improves the convergence performance of an iterative solver. A general expression for a left preconditioner, which is operated on the left of the original matrix, can be written as:

$$\mathrm{M^{-1}Ax = M^{-1}b} \tag{7}$$

in which $\mathrm{M^{-1}}$ is a left preconditioner, multiplied on both sides of the equation without changing the results of the linear system. The preconditioned iterative methods are similar to the original ones. If the GMRES-restarted method in section 2 is considered, the Krylov subspace becomes:

$$\mathcal{K}_{\mathrm{m}}(\mathrm{M^{-1}A}, \mathrm{r_0}) = \mathrm{span}\{\mathrm{r_0}, \mathrm{M^{-1}Ar_0}, \ (\mathrm{M^{-1}A})^2\mathrm{r_0}, \dots, (\mathrm{M^{-1}A})^{\mathrm{m-1}}\mathrm{r_0}\} \tag{8}$$

in which the initial residual, $\mathrm{r_0 = M^{-1}(b - Ax_0)}$. The algorithm in section 2.3 should be modified at line 1 for initial residual and at line 3 for replacing A by $\mathrm{M^{-1}A}$. The new system is named as preconditioned GMRES-restarted (Saad, 2003). Yang (2015) mentioned that preconditioner can be deemed as two extreme situations: (1) $\mathrm{M^{-1} = I}$ indicates that preconditioner does not affect the linear system; (2) $\mathrm{M^{-1} = A^{-1}}$ indicates that solution can be obtained immediately by $\mathrm{x = M^{-1}b}$. As can be demonstrated, preconditioner M should be as close as A so that $\mathrm{M^{-1}A}$ is close to identity matrix which is much easier for an iterative solver to handle. Several preconditioners widely used are of this type: ILU(0), ILU(k), ILUT and ILUC. These preconditioners are based on incomplete LU factorization through different strategies, but the principle behind is similar: factorization LU should be close to A. ILU is nothing but performing Gaussian Elimination (GE) by dropping elements according to various strategies, during which a lower and an upper triangular matrix will be generated. Several variants of GE approaches could result in various practical implementations of ILU.

Beyond the incomplete LU factorization, physics-based preconditioners designed specifically for different applications of iterative solvers are drawing attention in various areas of high-performance computing. In reservoir simulation, CPR is a powerful tool as a first stage preconditioner before the second stage ILU preconditioner. Multi-stage preconditioning can be expressed as (Cao et al., 2005):

$$\mathrm{M^{-1}_{1,2,\dots,n}} = \mathrm{M_n^{-1}[I - AM_{n-1}^{-1}]} \dots \mathrm{[I - AM_1^{-1}]} + \cdots + \mathrm{M_3^{-1}[I - AM_2^{-1}][I - AM_1^{-1}]} + \mathrm{M_2^{-1}[I - AM_1^{-1}]} + \mathrm{M_1^{-1}} \tag{9}$$

where I is an identity matrix and $\mathrm{M_n^{-1}}$ is the $\mathrm{n^{th}}$ stage preconditioner. Such as, a first stage preconditioner is applied: $\mathrm{M_1^{-1}Ax = M_1^{-1}b = M_1^{-1}(r_0 + Ax_0)}$, so that $\mathrm{M_1^{-1}Ax = M_1^{-1}(r_0 + Ax_0) \approx x_1}$ since $\mathrm{M_1^{-1}A}$ is close to I. Then, it can be obtained that $\mathrm{\delta_1 = M_1^{-1}r_0}$, which can be deduced that:

$$\mathrm{\delta_i = M_i^{-1}r_{i-1}} \tag{10}$$

in which $\mathrm{\delta_i}$ represents the solution shift after at $\mathrm{i^{th}}$ stage preconditioning $\mathrm{M_i^{-1}}$ with the previous stage residual $\mathrm{r_{i-1}}$. Equation (10) can lead to:

$$\sum_{i=1}^{n} \delta_i = M_1^{-1} r_0 + \sum_{i=2}^{n} M_i^{-1}\left[I - AM_{i-1}^{-1}\right] r^{i-2} \tag{11}$$

where the residual of i^{th} stage can also be expressed as:

$$r^i = \left[I - AM_i^{-1}\right] r^{i-1} \tag{12}$$

In two-stage preconditioning, the overall preconditioner can be written as:

$$M_{1,2}^{-1} = M_2^{-1}[I - AM_1^{-1}] + M_1^{-1} \tag{13}$$

in which, in CPR preconditioned system, M_1^{-1} refers to CPR and M_2^{-1} refers to ILU preconditioner. CPR basically provides a better initial guess through δ_i for ILU preconditioned iterative solver, damping the low-frequency error component as mentioned above.

In this section, the ILUC method and algorithm, which is applied in this study, will be the first to be explained. Then, CPR and SWIFT preconditioning will be discussed for their implementation.

3.1 Crout Version of Incomplete LU Factorization (ILUC(p,τ))

ILU(0) and ILU(K) are two factorization preconditioners that drop elements based on the sparsity pattern of original matrix A, without considering the numerical values of the elements. This dropping strategy could give an inaccurate LU factorization which might impact the performance of an iterative solver. To resolve this drawback, ILUT and ILUC are developed to dynamically drop the negligible terms based on their values during the Gaussian Elimination procedure (Saad, 1994; Li et al., 2003). ILUT algorithm is derived from the IKJ version of Gaussian Elimination during which the elements that are below the user-defined threshold are dropped and a certain number of largest elements are retained. However, there are still some drawbacks or difficulties from the perspective of implementation, such as: (1) obtaining a certain number of largest elements; (2) accessing the elements of lower triangular matrix in an ascending order of columns; (3) encountering a zero pivot. The first two could result in an expensive extra computation in factorization and the third one is ideally handled by partial pivoting which is also computationally complicated. Saad (2003) suggested that applying standard reordering techniques, such as Reverse Cuthill-McKee (RCM) ordering or Red Black reordering should improve the performance of preconditioning in some applications. It is also suggested a scaling of rows or columns and a small drop tolerance might be helpful to solve difficult matrices.

The Crout version of ILU contains the key solution to the difficulties mentioned above: in the step of updating lower and upper factorization, the Gaussian Elimination process computes the columns of L and rows of U at the same time, which makes it convenient to store L by columns and U by rows. Through this means of algorithm design, sparsity is taken into account dynamically and modified adaptively according to the numerical values of the fill-in. Accuracy of factorization is hence enhanced (Saad, 2003; Li et al., 2015). On the other hand, ILUC faces similar problems of zero pivot as ILUT, which might cost extra computational effort. In a thermal related reservoir simulation case where ILU(0) and ILU(K) hardly accelerate the iterative convergence, the thermal energy equation usually generates large terms in Jacobian matrix, which also brings difficulties for the application of ILUT or ILUC. In order to avoid pivoting in ILUC, CPR-SWIFT is considered to precondition the GMRES-restarted before ILUC.

3.2 First stage preconditioner: CPR

As has been stated, CPR is a first stage preconditioner that removes the low-frequency error components and provides an initial pressure value for the second stage preconditioning and iterative solver. The approach is nothing but constraining the complete matrix, A, to a pressure equation linear system:

$$A_{cpr}\delta = (C^T WAC)\delta = C^T W r \tag{14}$$

in which $C^T W$ is a restriction operator and C is a prolongation operator applied on matrix A to obtain the pressure equation. Equation (14) can be solved first by an appropriate iterative solver, such as GMRES-restarted. There are several ways of obtaining the constrained matrix of pressure equation, such as quasi-IMPES, true-IMPES and Alternate-Block Factorization method (Cusini et al., 2015). Two implementations of quasi-IMPES are tested for this study: (1) the first one is referred as block diagonal scaling in (Lacroix et al., 2001; Cao et al., 2005; Li et al., 2015); (2) the second one is referred as quasi-IMPES reduction approach. These two approaches lead to exactly the same solution for the first CPR stage preconditioning.

The first implementation of quasi-IMPES treats the restriction operator as a block diagonal scaling process:

$$W = \mathrm{bdiag}^{-1}(A) \tag{15}$$

in which $\mathrm{bdiag}(A)$ is main block diagonal of matrix A. Consider the unknowns as a full vector which is arranged in the order of pressure-saturation-temperature for each grid block, then the prolongation operator should be in the form of:

$$C = \begin{bmatrix} e_p & 0 & 0 & 0 \\ 0 & e_p & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & e_p \end{bmatrix} \tag{16}$$

where $e_p = [1,0,0]^T$ indicating that the first equation (pressure equation) is extracted and the dimension of the prolongation operator should be $(n_{cell} \times n_{eqn}) \times n_{cell}$. n_{cell} is the number of grid blocks and n_{eqn} is the number of unknowns for each grid block. Thus, pressure can be solved in the first stage as:

$$\delta_p = (C^T W A C)^{-1} C^T M r_0 \tag{17}$$

then expand the first stage solution back to the full system:

$$x_1 = x_0 + C\delta_p \tag{18}$$

as the initial value for the second stage preconditioning. According to equation (10) and (12),

$$r_1 = r_0 - AC\delta_p \tag{19}$$

Equation (19) should be used as the updated residual vector for second stage preconditioning.

The second implementation of quasi-IMPES divides the Jacobian matrix into four blocks of pressure and saturation equations:

$$A\delta = \begin{bmatrix} A_{pp} & A_{ps} \\ A_{sp} & A_{ss} \end{bmatrix} \begin{bmatrix} \delta_p \\ \delta_s \end{bmatrix} = \begin{bmatrix} r_p \\ r_s \end{bmatrix} \tag{20}$$

In this case, prolongation operator C will be $C = [I, O]^T$, W in restriction operator will be:

$$W = \begin{bmatrix} I & \left(-\mathrm{diag}(A_{ps})\mathrm{diag}^{-1}(A_{ss})\right) \\ O & I \end{bmatrix} \tag{21}$$

leading to a pressure solution, according to equation (17): $[A_{pp} - \mathrm{diag}(A_{ps})\mathrm{diag}^{-1}(A_{ss})A_{sp}]\delta_p = r_p - \mathrm{diag}(A_{ps})\mathrm{diag}^{-1}(A_{ss})r_s$. The same procedure should be performed for expanding the first stage pressure solution and correcting the residual as mentioned above.

As can be seen, the first stage preconditioner in these two implementations can be written as: $M_1^{-1} = C(C^T W A C)^{-1} C^T W$ for CPR method.

3.3 Second stage preconditioner: SWIFT-ILUC(τ)

SWIFT, proposed by Li et al. (2015), scaling the matrix by row-column equilibration after block diagonal scaling, in order to reduce the 1-norm of rows and columns to approximately 1. The objective of conducting block diagonal scaling and row-column equilibration is to transform the matrix into one close to be diagonally dominant. The thermal energy equation produces large elements in matrix which will be taken care of by the scaling and equilibration. These two steps are very effective since ILUC can be adopted as a second stage without the computationally expensive pivoting after these two steps. In this section, the algorithm and implementation of SWIFT will be illustrated in detail.

First step is to perform block diagonal scaling for the matrix:

$$\overline{A} = \mathrm{bdiag}^{-1}(A)A \tag{22}$$

and the second step is to conduct Sinkhorn-Knopp Row-Column Equilibration (Knight, 2008; Takapoui and Javadi, 2016) on $\hat{A}$ ($\hat{A} = \overline{A}$), as shown in the following algorithm, given $D_c = I$ and $D_r = I$:

1. *For* $\ell = 1, \ell_{max}$, *Do:*
2. $f_i = \left\|\hat{A}_{\star,i}\right\|_1^{-1}, i = 1, \dots, n$
3. $\hat{A} = \hat{A} \cdot diag(f)$
4. $D_c = D_c \cdot diag(f)$
5. $d_i = \left\|\hat{A}_{i,\star}\right\|_1^{-1}, i = 1, \dots, n$
6. $\hat{A} = diag(d) \cdot \hat{A}$
7. $D_r = D_r \cdot diag(d)$
8. *EndDo*

In this algorithm, $l_{max} = 2$. The modified matrix after row-column equilibration can be expressed as:

$$\hat{A} = D_r \overline{A} D_c \tag{23}$$

and the right-hand-side residual should be rewritten as:

$$\hat{r} = D_r b \mathrm{diag}^{-1}(A) r \tag{24}$$

The new linear system $\hat{A}\hat{\delta} = \hat{r}$ can be input into the iterative solver GMRES-restarted which provides the solution $\hat{\delta}$. This solution should be transformed back to obtain the real solution: $\delta = D_c \hat{\delta}$. Therefore, the second stage preconditioner can be expressed as: $M_2^{-1} = D_c U^{-1} L^{-1} D_r$ in which L and U are given by the ILUC. The overall two-stage preconditioner operated on the diagonally scaled matrix, $\overline{A}$, can be expressed as:

$$M^{-1} = D_c U^{-1} L^{-1} D_r \left[I - \overline{A} C \left(C^T \overline{A} C \right)^{-1} C^T W \right] + C \left(C^T \overline{A} C \right)^{-1} C^T W \tag{25}$$

It has been observed that the matrix after block diagonal scaling and row-column equilibration has diagonal elements close to 1 and the off-diagonal terms are relatively small in comparison. ILUC should be effective enough, without pivoting, to precondition GMRES-restarted.

Li et al. (2015) also suggested to employ cell reordering, such as RCM or Red-Black reordering, after two-stage preconditioning before the iterative solver. The reordering was tested in this study but not much convergence improvement was observed. It is believed that this negligible effect is mainly due to the fact that the mesh grids adopted are structured Cartesian grids.

4. EVALUATION OF CPR-SWIFT-ILUC PRECONDITIONER AND DISCUSSION OF RESULTS

In this section, the results of applying CPR-SWIFT-ILUC to several reservoir simulation cases will be displayed and compared for demonstrating the effectiveness of this physics-based two-stage preconditioner. In this study, TOUGH2-CSM is the geothermal reservoir simulation program from which Jacobian matrix can be output. In a reservoir simulation case, a time step is completed when equation residuals are reduced to a negligible tolerance. Within a single time step, multiple Newton iterations are updating primary variables to achieve such convergence, during which the Jacobian linear system will be solved for corresponding times. The Jacobian matrix for one iteration will be selected based upon time step size and convergence condition.

Jacobian matrix and right-hand-side vector are output from the TOUGH2-CSM program into two data files. MATLAB reads in the data files and reassembles the sparse Jacobian matrix. CPR-SWIFT has been implemented in MATLAB using the algorithm discussed in section 3 before invoking ILU preconditioner embedded in MATLAB. Preconditioned GMRES (10)-restarted is used as the iterative solver. Final solution out of the iterative solver is compared with MATLAB intrinsic linear solver and with solution given by TOUGH2-CSM as well, for the purpose of validation.

As stated in Cao et al. (2005), heterogeneity, long time step and complex well injection/production would affect the convergence performance of iterative solver. Five cases were computed using CPR-SWIFT-ILUC preconditioned GMRES in this study. The case conditions are listed in the Table 1. In the heterogeneous case, permeability and porosity modifiers are generated randomly for each grid block using an internal seed. The maximum permeability/porosity could be as high as 148 times of the minimum permeability/porosity. Total number of grids for all cases are 30000 (60 (x-direction) *50 (y-direction) *10 (z-direction)). The Cartesian mesh grids are three-dimensional, of dimension 500*480*20 m^3.

Table 1 Related parameters of cases for comparison

	Heterogeneity	Time Step	Number of Well	Fluid Phase
Case 1	Homogeneous	0.3 days	1 injection/1 production	Water
Case 2	Homogeneous	7.41 days	1 injection/1 production	Water
Case 3	Heterogeneous	5000 seconds	1 injection/1 production	Water
Case 4	Homogeneous	1 day	4 injection/1 production	Water
Case 5	Homogeneous	5000 seconds	1 injection/1 production	Water/Air

The performance of GMRES (10) is shown in Figure 2-Figure 6. The preconditioner compared in each case might be slightly different based on the convergence performance, which will be discussed later in this section.

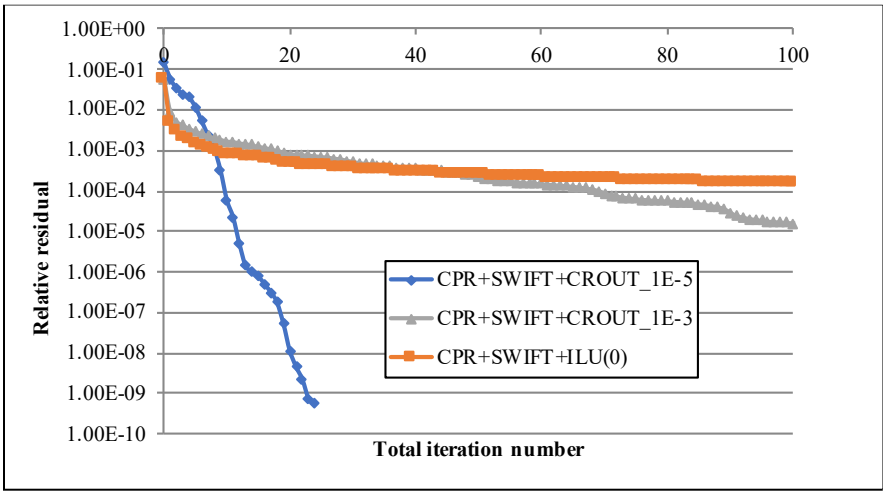

Figure 2 Performance of preconditioned iterative solver for Case #1: comparing CPR+SWIFT with ILUC or ILU(0).

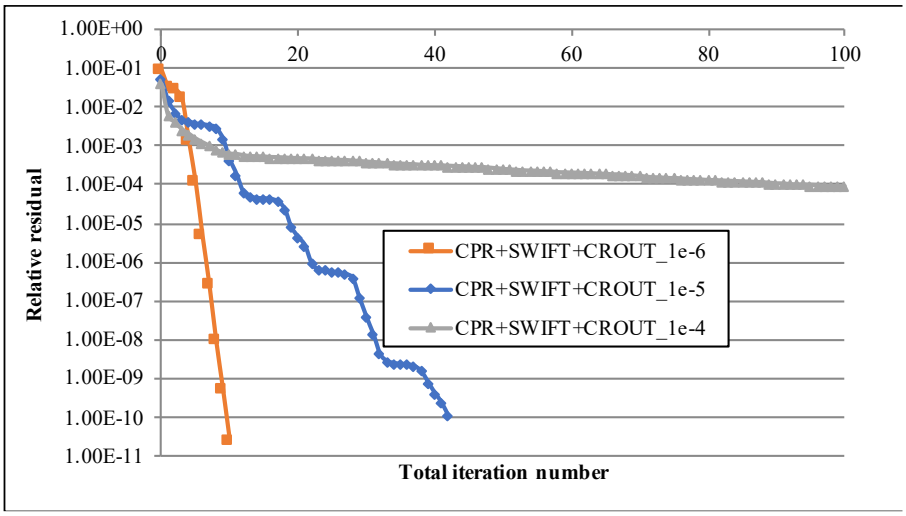

Figure 3 Performance of preconditioned iterative solver for Case #2: comparing CPR+SWIFT+ILUC with different dropping tolerance.

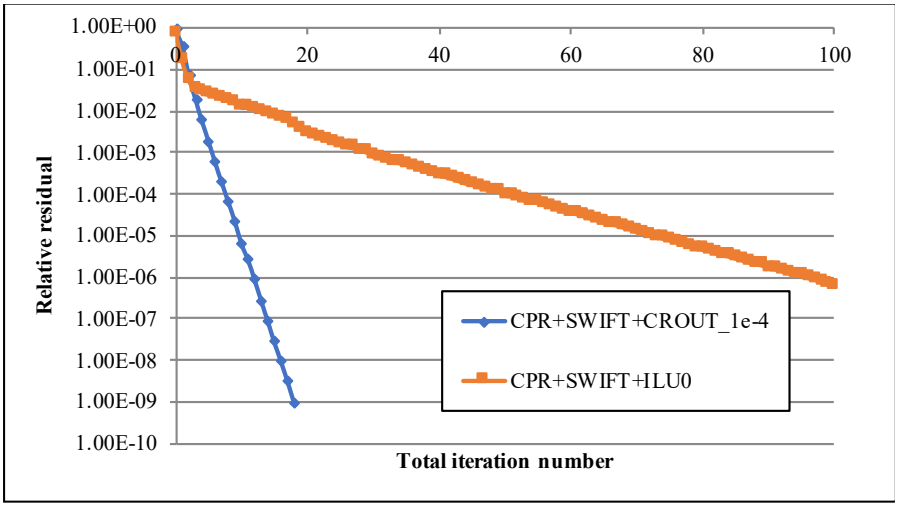

Figure 4 Performance of preconditioned iterative solver for Case #3: comparing CPR+SWIFT with ILUC or ILU(0).

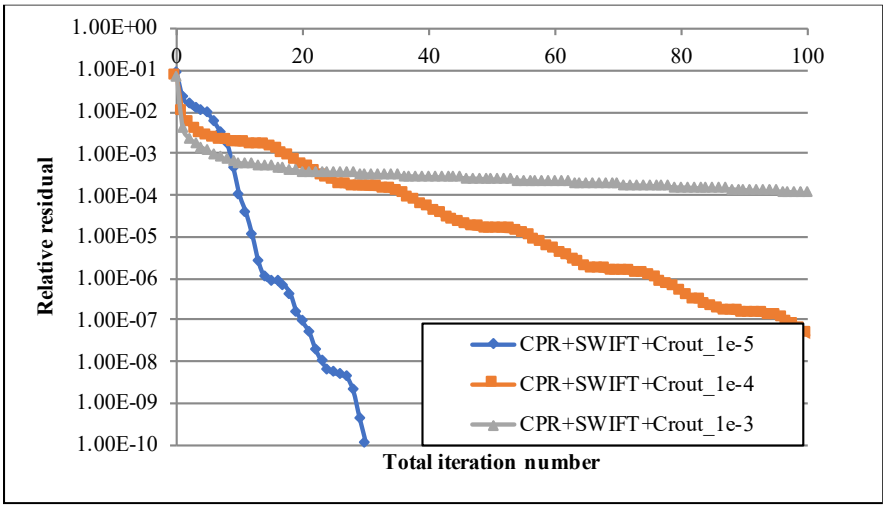

Figure 5 Performance of preconditioned iterative solver for Case #4: comparing CPR+SWIFT+ILUC with different dropping tolerance

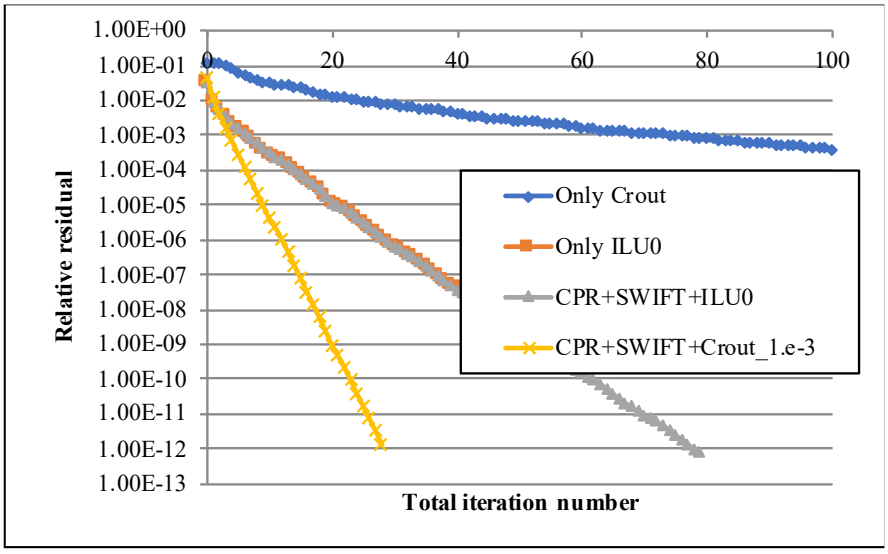

Figure 6 Performance of preconditioned iterative solver for Case #5: comparing CPR+SWIFT with ILUC or ILU(0) and single stage ILU preconditioning.

The above results for 5 cases show a common conclusion: two-stage preconditioner CPR+SWIFT+ILUC, with an appropriate dropping tolerance, always improves the convergence performance of GMRES-restarted. This observation proves the effectiveness of this physics-based multi-stage preconditioner. It is desired to discuss case by case because only limited test preconditioners are displayed in the above figures and model parameters are varied as well.

(1) For case #1, the time step is not too large but already beyond the capability of ILU(0) and even ILUC(1e-3) as the second stage preconditioner. In this case, neither CPR+SWIFT+ILU(0) or CPR+SWIFT+ILU(1e-3) could provide GMRES with a converged solution. ILU(0) and ILUC, as a single stage preconditioner cannot help GMRES achieve convergence in this case. Only CPR+SWIFT+ILUC(1e-5) leads to a convergence with about 20 iterations; **(2)** For case #2, similar results are observed as case #1, a time step of 7.41 days is too large for ILU(0) to handle. In this case, the convergence performance is not excellent even for CPR+SWIFT+ILUC(1e-5). This proves that long time step is a challenge for iterative solver which need CPR+SWIFT as a preconditioner; **(3)** For case #3, due to the heterogeneity, it is hard for the iterative solver to reach convergence and ILU(0) or ILUC as a single stage preconditioner cannot assist GMRES to achieve convergence. It is shown that they need pivoting process which could lead to an expensive computation. Timestep is not very large in this case, which proves again that heterogeneity is a significant impact on GMRES performance. CPR+SWIFT+ILUC(1e-4) provides a stable and effective preconditioning that makes GMRES to converge within 20 iterations; **(4)** For case #4, the complex injection well deteriorates the convergence performance. No converged results are obtained for CPR+SWIFT+ILU(0), ILU(0) or ILUC. A dropping tolerance, as low as 1e-5 is needed for a good convergence performance; **(5)** For case #5, this is a two-phase flow including water and air. It can be seen that ILU(0) is not behaving too badly due to a small time step. CPR+SWIFT+ILUC still provides the best performance for this case. And what is surprising is that ILUC as a

single stage would not help due to the fact that this preconditioner might not be as effective as ILU(0) when thermal energy equation produces large terms in Jacobian matrix.

It can be inferred, according to the results, that time step size, complex well injection/production, heterogeneity and multi-phase fluid flow are challenging for iterative solution process. CPR-SWIFT-ILUC is capable of improving convergence performance of GMRES for all these difficult situations. When the time step is not large, ILU(0) is also a good candidate for the second stage preconditioner. On the other hand, the convergence performance should not be the only factor to be considered. Reduction of overall computational cost of the linear solver is the ultimate goal for reservoir simulation. For example, ILUC is more computationally expensive than ILU(0), especially in a non-parallelized program. The trade-off between better convergence performance and longer ILU factorization time should be determined by the program automatically, which is also suggested by Li et al. (2015). In this study, the implementation of multi-stage preconditioner in TOUGH2-CSM hasn't been accomplished yet, therefore, the evaluation of overall computational cost for a preconditioned linear solver cannot be performed. A simple comparison has been conducted for the above five cases to illustrate the overall computation time for singe iterations. The comparison is shown in Table 2 below. It can be observed that ILUC is the most time-consuming part and the lower dropping tolerance leads to a higher factorization time. ILU(0) can be completed in no time but it may not help GMRES to converge properly. The computation time of ILUC can be significantly reduced if parallelization is adopted. A practical implementation is to make the program adaptively determine when to use CPR+SWIFT+ILUC instead of ILU(0), such as, if slow convergence or large number of iterations are encountered during Newton's method.

Table 2 Comparison of overall computational time (in seconds) for five cases of iterations

	Combination of preconditioning process	ILUC/ILU(0)	GMRES
Case #1	CPR+SWIFT+ILUC(1e-5)	31.35	1.92
	CPR+SWIFT+ILUC(1e-3)	29.28	8.52
	CPR+SWIFT+ILU(0)	0.03	8.24
Case #2	CPR+SWIFT+ILUC(1e-6)	43.52	1.248
	CPR+SWIFT+ILUC(1e-5)	31.57	2.84
	CPR+SWIFT+ILUC(1e-4)	29.28	12.03
Case #3	CPR+SWIFT+ILUC(1e-4)	41.79	1.26
	CPR+SWIFT+ILU(0)	0.05	5.35
Case #4	CPR+SWIFT+ILUC(1e-5)	31.55	2.17
	CPR+SWIFT+ILUC(1e-4)	29.23	5.64
	CPR+SWIFT+ILUC(1e-3)	28.80	9.42
Case #5	CPR+SWIFT+ILUC(1e-3)	36.59	1.78
	CPR+SWIFT+ILU(0)	0.05	4.35
	ILU(0)	0.05	2.26
	ILUC(1e-3)	2.62	12.49

CONCLUSION

A two-stage preconditioner, named as CPR-SWIFT-ILUC, is implemented and evaluated for its effect on iterative methods, GMRES-restarted. It is observed that CPR-SWIFT-ILUC is capable of improving convergence performance of GMRES-restarted significantly for difficult problems: heterogeneity, large time step, complex injection/production well and multi-phase flow. The common preconditioner, ILU(0), often fails to help GMRES-restarted achieve convergence in these cases. However, the computational cost of ILUC is an important factor that should be taken into account in reservoir simulation, seeking the reduction of overall linear solution time. Parallelization and automatic preconditioner selection are suggested for reservoir simulation implementations.

In this paper, the basic methodology of an iterative method, GMRES-restarted, is introduced first, followed by the explanation of the dynamic incomplete LU factorization approach. The advantage of this preconditioner, compared to the common choice, ILU(0) or ILU(K), is that it is capable of exploiting the sparsity of the matrix and adaptively modify the fill-in terms to improve factorization

accuracy. Secondly, CPR as a first stage preconditioner, and SWIFT, as the preprocessing of second stage preconditioner, are demonstrated for their principles and implementation methods. SWIFT transforms the matrix into a near diagonally dominant one by block diagonal scaling and row-column equilibration, aiming at the large terms produced by the thermal energy equations, which reduces the difficulty for ILUC and GMRES to handle. Jacobian matrix from a single Newton's iteration of several geothermal reservoir simulation cases are output into a MATLAB program where CPR-SWIFT-ILUC is implemented. ILUC preconditioner is an embedded function in MATLAB, which is already available without coding implementation. At last, the effectiveness of CPR-SWIFT-ILUC is evaluated in different reservoir simulation cases.

The effectiveness of this physics-based multi-stage preconditioner has been proven through comparison. On the other hand, the overall efficiency of this approach needs to be further investigated by taking into consideration the computational time of ILUC factorization. In the future, since TOUGH2-CSM is a simulator built on a parallel framework, it is desired that parallelization should be implemented for this scalable CPR-SWIFT-ILUC algorithm so that the overall computational time of linear solution can be evaluated for complete simulation cases. It is also planned to determine by the program itself when this preconditioner should be triggered through the convergence and iteration condition.

ACKNOWLEDGEMENT

This research is supported by Energy Modeling Group of Colorado School of Mines and Energi Simulation.

REFERENCES

Arnoldi, W. E., 1951, The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem: Brown University, p. 17–29, doi:10.2307/43633863.

Cao, H., H. A. Tchelepi, J. R. Wallis, and H. E. Yardumian, 2005, Parallel Scalable Unstructured CPR-Type Linear Solver for Reservoir Simulation, *in* SPE Annual Technical Conference and Exhibition: Society of Petroleum Engineers, doi:10.2118/96809-MS.

Cusini, M., A. A. Lukyanov, J. Natvig, and H. Hajibeygi, 2015, Constrained pressure residual multiscale (CPR-MS) method for fully implicit simulation of multiphase flow in porous media: Journal of Computational Physics, v. 299, p. 472–486, doi:10.1016/J.JCP.2015.07.019.

Faust, C. R., and J. W. Mercer, 1979, Geothermal reservoir simulation: 1. Mathematical models for liquid- and vapor-dominated hydrothermal systems: Water Resources Research, v. 15, no. 1, p. 23–30, doi:10.1029/WR015i001p00023.

Knight, P. A., 2008, The Sinkhorn–Knopp Algorithm: Convergence and Applications: SIAM Journal on Matrix Analysis and Applications, v. 30, no. 1, p. 261–275, doi:10.1137/060659624.

Lacroix, S., Y. V. Vassilevski, and M. F. Wheeler, 2001, Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS): Numerical Linear Algebra with Applications, v. 8, no. 8, p. 537–549, doi:10.1002/nla.264.

Li, N., Y. Saad, and E. Chow, 2003, Crout Versions of ILU for General Sparse Matrices: SIAM Journal on Scientific Computing, v. 25, no. 2, p. 716–728, doi:10.1137/S1064827502405094.

Li, G., J. Wallis, and G. Shaw, 2015, A Parallel Linear Solver Algorithm for Solving Difficult Large Scale Thermal Models, *in* SPE Reservoir Simulation Symposium: Society of Petroleum Engineers, doi:10.2118/173207-MS.

O'Sullivan, M. J., K. Pruess, and M. J. Lippmann, 2001, State of the art of geothermal reservoir simulation: Geothermics, v. 30, no. 4, p. 395–429, doi:10.1016/S0375-6505(01)00005-0.

Saad, Y., 1994, ILUT: A dual threshold incomplete LU factorization: Numerical Linear Algebra with Applications, v. 1, no. 4, p. 387–402, doi:10.1002/nla.1680010405.

Saad, Y., 2003, Iterative Methods for Sparse Linear Systems: Second Edition - Yousef Saad - Google Books: SIAM.

Saad, Y., 1981, Krylov subspace methods for solving large unsymmetric linear systems: Mathematics of Computation, v. 37, no. 155, p. 105–105, doi:10.1090/S0025-5718-1981-0616364-6.

Saad, Y., and M. H. Schultz, 1986, GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems: SIAM Journal on Scientific and Statistical Computing, v. 7, no. 3, p. 856–869, doi:10.1137/0907058.

Takapoui, R., and H. Javadi, 2016, Preconditioning via Diagonal Scaling.

Wallis, J. R., 1983, Incomplete Gaussian Elimination as a Preconditioning for Generalized Conjugate Gradient Acceleration, *in* SPE Reservoir Simulation Symposium: Society of Petroleum Engineers, doi:10.2118/12265-MS.

Wallis, J. R., R. P. Kendall, and T. E. Little, 1985, Constrained Residual Acceleration of Conjugate Residual Methods, *in* SPE Reservoir Simulation Symposium: Society of Petroleum Engineers, doi:10.2118/13536-MS.

Wang, S., 2015, Numerical study of thermal-hydraulic-mechanical behavior of fractured geothermal reservoirs: Colorado School of Mines.

Yang, B., 2015, A GMRES Solver with ILU(k) Preconditioner for Large-Scale Sparse Linear Systems on Multiple GPUs: University of Calgary.