We Gra 11

Monotone Non-Galerkin Algebraic Multigrid Method Applied to Reservoir Simulations

T.B. Jonsthovel* (Schlumberger), A.A. Lukyanov (Schlumberger), E.D. Wobbes (Delft University of Technology) & C. Vuik (Delft University of Technology)

SUMMARY

Commercial reservoir simulators must be very robust and fast. Moreover, current hardware requires the simulators to scale over multiple number of computing nodes and for a fixed ('strong scalability') as well as an increasing problem size per computing node ('weak scalability'). In most current commercial reservoir simulators, due to the different geological structures and properties of hydrocarbon reservoirs and the use of enhanced oil recovery (EOR) techniques, the governing equations are strongly nonlinear and hard to solve. The Jacobian system is solved by FGMRES preconditioned by the two-level constrained pressure residual (CPR) preconditioner. The driving force of the CPR preconditioner is the solution of the pressure equation. The industry standard for solving the pressure equation is the algebraic multigrid (AMG) solver. AMG is well known for its 'weak scalability'. However, in these applications, AMG has unfavorable 'strong' scalability properties. This degradation in scalability is due to the increased level of inter-processor communication in the algorithm.

In this paper, a monotone non-Galerkin AMG (MNG-AMG) method is presented. The aim of the method is to reduce the overall communication in MNG-AMG by enforcing a predefined nonzero pattern and monotonicity property (i.e., M-matrices) on each multigrid level. This paper describes the application of the MNG-AMG method in the context of reservoir simulations. We will compare the parallel scalability of the default solver with the MNG-AMG solver and discuss the optimal values for the MNG-AMG solver for a variety of test cases based on full field reservoir simulations.

Introduction

Reservoir simulation is of paramount importance to predict reliable estimates of recoverable oil and gas reserves as well as minimizing risk of production strategies. Recent advances in computer memory as well as the introduction of affordable multi-core machines have pushed the physical and numerical reservoir model sizes to unparalleled complexity. Typically models have tens to hundred millions of grid cells entailing extremly complex geometries. The size and complexity of these problems require vast computational resources and parallel computation.

In this paper we consider a highly-scalable commercial reservoir simulator that models compositional multi-phase porous media fluid flow based on Darcy's law. The Krylov-subspace method is the driving force of the simulator and used to solve the large linear systems of equations originating from the linearization of the coupled mass-balance equations. The linear systems consist of an elliptic (or parabolic) pressure and hyperbolic transport part. As pressure is driving fluid flow, we use the two-stage Constrained Pressure Residual method introduced by Wallis (1985) to decouple the pressure field from the remaining unknowns. This allows for employing different solvers for the pressure and transport equations.

Multigrid methods are the methods of choice to solve elleptic equations in an efficient manner due to the coarse grid correction. In this paper we consider the Algebraic Multigrid (AMG) method as it is able to deal with complex structures, anisotropies and discontinuous or varying coefficients and does not require geometric information. Moreover, AMG is well known to have excellent weak-scalability properties. This means that for an increasing number of nodes with a fixed problem size per node runtime does not change. As such AMG is the perfect black box. However, the disadvantage of AMG is that it has poor strong-scalability properties. This means that for an increasing number of nodes with a fixed global problem size the runtime does not reduce accordingly. The underlying reason is an increasing unfavorable ratio between the amount of work spend on the coarser grids versus the amount of parallel communication involved. As typical reservoir simulation workflows involve running multiple realizations of the same model, good strong scalability of the simulator is crucial.

Currently two solution strategies exist to improve the strong scalability of AMG. The firsts strategy is aggresive coarsening, which was introduced in Krechel and Stüben (1998) and applied to AMG in Stüben (2000); Sterck et al. (2006), and reduces the ratio between the number of non-zero entries on the coarse and fine level. The second solution strategy is the non-Galerkin method presented in Ashby and Falgout (1996); Wienands and Yvneh (2009) and applied to AMG in Stüben (2000); Darwish et al. (2006); Falgout (2006); Vassilevski and Yang (2014). The non-Galerking method removes less important entries of the AMG coarse grid matrices while preserving the row sums and thus its robustness.

In this paper we investigate the effectiveness of the non-Galerkin method applied to multiphase flow in porous media. Moreover we present a modified non-Galerkin method which improves the robustness of the original method and is specifically tailored for the domain of reservoir simulation.

The outline of the paper is as follows. We provide a brief description of the governing equations in reservoir simulation and describe the problem with strong scalability. Next we provide a solution strategy based on the non-Galerkin method. We introduce the general ideas behind the method and present the modified non-Galerkin method. This follows by the results section where we discuss the numerical results in terms of the operator and grid complexities, number of non-linear and linear iterations, and linear solver and total execution time. We finish with conclusion, future work and references.

Reservoir simulation

Multiphase flow of hydrocarbons in porous media is governed by a system of time-dependent nonlinear partial differential equations. In this paper we consider flow of fluids comprising multiple components, which may partition into multiple phases. For a system with n_c components and n_p phases the generalized compositional flow equation can be written

$$\frac{\partial}{\partial t}\left(\phi\sum_{\alpha}C_{i\alpha}\rho_{\alpha}S_{\alpha}\right)=\nabla\cdot\left(\sum_{\alpha}\frac{C_{i\alpha}\rho_{\alpha}Kk_{r\alpha}}{\mu_{\alpha}}\left(\nabla p_{\alpha}-\rho_{\alpha}g\nabla d\right)\right)+q_i,\quad i=1,\ldots,n_c$$

$$\sum_{\alpha}S_{\alpha}=1 \tag{1}$$

$$\sum_{i=1}^{n_c}C_{i\alpha}=1,\quad\forall\alpha,$$

where ϕ the rock porosity, $C_{i,\alpha}$ is the mass fraction of component i in phase α, ρ_{α} is the phase fluid density, S_{α} is the phase saturation, K the absolute permeability, $k_{r\alpha}$ the phase relative permeability, μ_{α} the phase viscosity, p_{α} the phase pressure, g the gravity constant, d the cell depth, q_i the source term.

Additional equations define the thermodynamic equilibrium between components in different phases:

$$f_{i\alpha}=f_{i\beta},\ \beta\neq\alpha,\ i=1,\ldots,n_c, \tag{2}$$

where $f_{i\alpha}$ is the fugacity, which is a measure of the tendency of component i to escape from phase α.

The phase pressures p_{α} are related to each other through capillary pressures

$$P_{c\alpha\beta}=p_{\alpha}-p_{\beta},\qquad\beta\neq\alpha, \tag{3}$$

which are known functions of saturation. For a derivation of these equations see, for example, Aziz and Settari (1979), Peaceman (1977).

The flow equations are discretized using the upstream finite-volume method. The coupled reservoir-well linear system can be expressed as

$$\bar{\mathbf{A}}x=\begin{bmatrix}\mathbf{A}_{rr} & \mathbf{A}_{rw}\\ \mathbf{A}_{wr} & \mathbf{A}_{ww}\end{bmatrix}\begin{bmatrix}x_r\\ x_w\end{bmatrix}=\begin{bmatrix}b_r\\ b_w\end{bmatrix}=b. \tag{4}$$

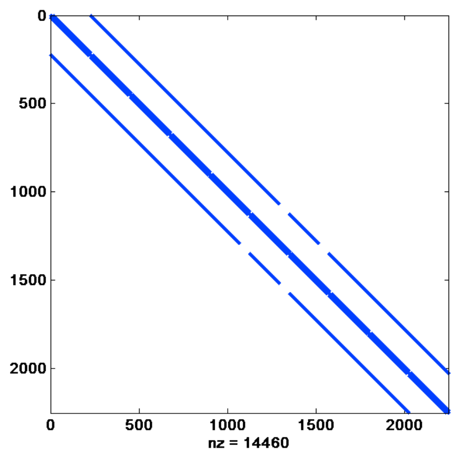
The subscript r refers to reservoir and w stands for well. We assume $\bar{\mathbf{A}}\in\mathbb{R}^{n\times n}$, $b_r,x_r\in\mathbb{R}^{nr}$, $b_w,x_w\in\mathbb{R}^{nw}$. The sub-matrices $\mathbf{A}_{rr}$ and $\mathbf{A}_{ww}$ are square $nr\times nr$ and $nw\times nw$ matrices respectively. Each sub-matrix of the Jacobian represents a derivative, e.g. A_{wr} is the derivative matrix of the well equations with respect to the reservoir variables. The matrix $\bar{\mathbf{A}}$ is typically very sparse. For example, some of examples of sparsity pattern of the matrix $\mathbf{A}_{rr}$ are illustrated in Figure 1. We assume that matrix $\mathbf{A}$ has the following properties:

- The computational grid consists of n_c cells or ‘points’, and each cell contains n_u unknowns. Hence, $n=n_c\times n_u$.
- $\bar{\mathbf{A}}\neq\bar{\mathbf{A}}^T$, i.e. $\bar{\mathbf{A}}$ is non-symmetric.
- $\lambda\neq0\ \forall\ \lambda\in\sigma(\bar{\mathbf{A}})$, i.e. all eigenvalues are non-zero.

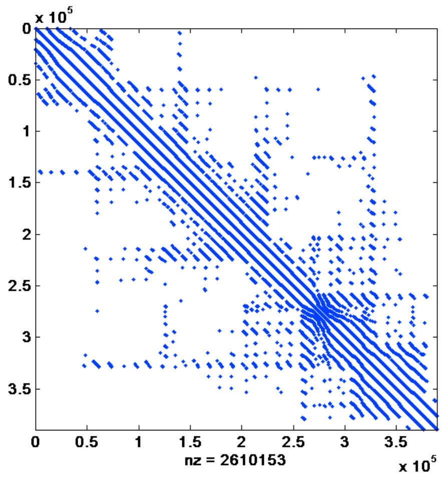
Traditionally, wells equations (variables) are eliminated using Schur complement step. This allows us to apply CPR preconditioner to the reservoir cells without focusing on wells placements which leads to decouple the pressure variables and the remaining unknowns. Below, the solution methods of the pressure system denoted by $\mathbf{A}$ are discussed.

Problem description

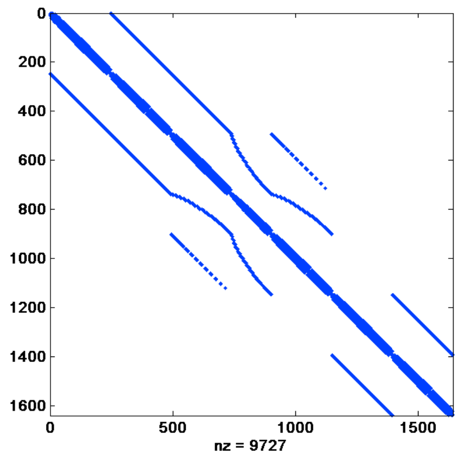
As it was pointed out above, the reservoir models have tens to hundred millions of grid cells entailing extremly complex geometries. The size and complexity of these problems require vast computational resources and parallel computation. Hence, the state-of-the-art solvers, for example, the AMG preconditoner is used for parallel computing of large problems. Below, some of the issues related to the parallel performance of AMG are discussed, which forms the problem description addressed in this paper.



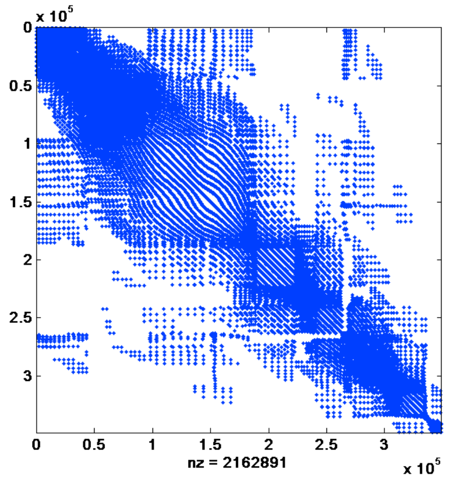
(a) 2250-case.



(c) 389559-case.



(b) 1639-case.



(d) 348809-case.

Figure 1 *Sparsity patterns of* $\boldsymbol{A}$ *the different selected cases.*

Scalability of AMG

Although, the greater part of the AMG algorithm, comprised of matrix and vector operations, can be parallelized in a straightforward way, it certainly requires communication and data exchange among processors. In addition, the parallelisation of the smoothing and coarsening processes is challenging Yang (2006). Therefore, the linear solver time is partially consumed by these processes. Figure 2 illustrates the total execution time of the simulations and the running time of the linear solver on an increasing number of processor cores, for a problem of approximately 4×10^6 variables. The total and linear solver time are efficiently reduced when the number of cores gradually rises from 16 to 128. However, the timings corresponding to 256 processors are approximately equal to those on 128 processor cores. This indicates the loss of strong scalability. Experiments have shown that a substantial amount of time within the linear solver is required for the communication within AMG. Obviously, for a more efficient functioning of the simulator in parallel, the data communication within the multigrid method should be reduced. Hence, AMG solver remains weakly scalable, independent of the number of cores.

Solution Strategies

The costs of AMG are strongly related to the algebraic complexity. The number of nonzero elements affects the number of operations per cycle. A denser operator requires more operations and, hence, a

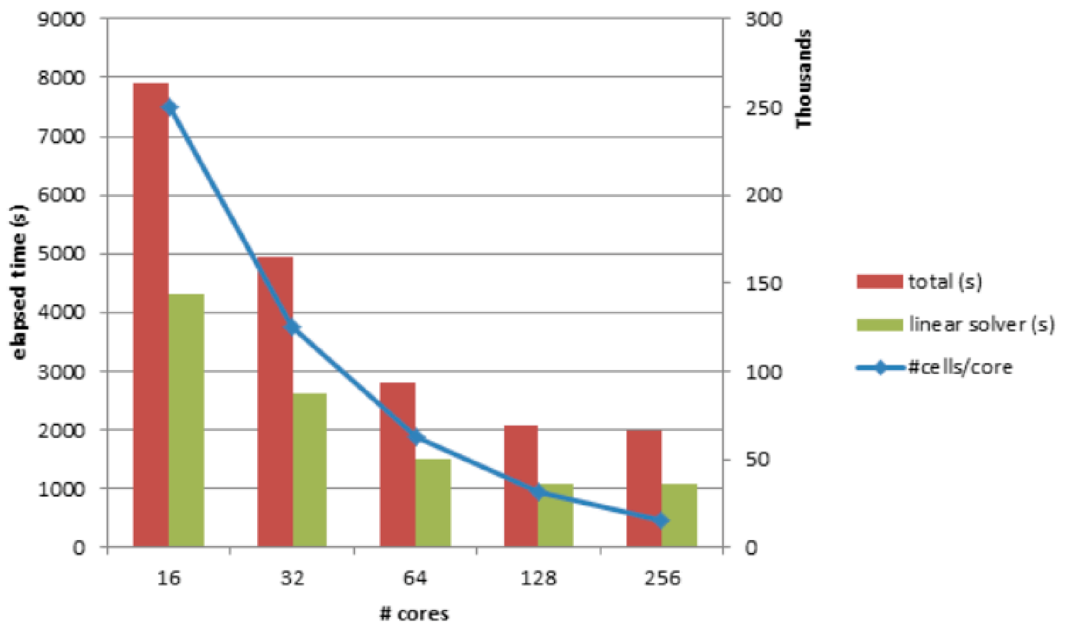


Figure 2 Strong scalability of AMG solver for a case with 4×10^6 variables.

denser communication pattern. At the same time, large number of variables may entail the exchange of larger sets of data, which also requires communication between processors. Thus, the appropriate minimization of the operator and grid complexities is crucial for the reduction of communication.

Lower complexities inevitably lead to the loss of accuracy within the AMG hierarchy and accordingly slower convergence of the multigrid method. Since most of reservoir simulators uses only one AMG V-cycle per linear iteration, the sacrifice of the convergence speed should lead to a poorer approximation of the cell pressure. Although this certainly has a negative influence on the second stage of the CPR preconitioner, ILU(0) may sufficiently improve the solution and, therefore, prevent the number of linear iterations from rising significantly. For this reason, the overall effect of the reduction of complexities should be positive with regard to the linear solver and total execution time.

Below, we will discuss two solution strategies used to resolve the communication issue. Although both approaches decrease the number of nonzero entries within the hierarchy of AMG, the theory behind them is entirely different. The first solution strategy is aggressive coarsening. The aggressive coarsening is a substitute for the standard coarsening, because it modifies the coarsening scheme. By contrast, the second solution strategy, the non-Galerkin method, decreases the density of the existing coarse level operators and should be seen as an extension of the AMG algorithm.

Aggressive Coarsening

Aggressive coarsening was introduced in 1996 in Krechel and Stüben (1998). In Stüben (2000), it is suggested to apply aggressive coarsening to the finer levels in order to reduce the complexities. On the one hand, the application of aggressive coarsening drastically reduces the setup and solution costs, the complexity of the operators and the memory requirement. On the other hand, it decreases the efficiency of the smoothing procedure and encumbers the interpolation, because the prolongation operator deals with longer distances between coarse grid variables. According to Stüben (2000), the benefits of the aggressive coarsening strategy certainly outweigh its disadvantages, at least when applied to Ruge-Stüben AMG. In Yang (2010) it is pointed out that aggressive coarsening can successfully be used with any coarsening algorithm. Similarly to the application of aggressive coarsening on classical Ruge-Stüben AMG, in order to construct sparser coarse grid operators than those generated by PMIS. The most efficient manner to implement aggressive coarsening is by applying the PMIS algorithm twice.

Non-Galerkin method

Non-Galerkin method has been successfully applied in settings where geometric information is used to aid the multigrid algorithms in constructing the sparsity patterns and choosing matrix coefficients Ashby and Falgout (1996); Wienands and Yvneh (2009). We focus on the non-Galerkin method described in Falgout and Schroeder (2014). This purely algebraic approach is based on the traditional AMG techniques.

As problem size increases, the number of levels in the AMG hierarchy grows and denser coarse grid operators are generated, see Falgout and Schroeder (2014). This leads to denser communication patterns than existed on the fine level, because the processors that were not coupled on the fine level become coupled. It is shown that in parallel the time spent on some coarse levels can actually be larger than the time required for the fine level due to high density of the coarse grid operators(see, Gahvari et al. (2011)). The increase in density is caused by the standard Galerkin operator $\mathbf{R}\cdot\mathbf{A}\cdot\mathbf{P}$ with $\mathbf{R}=\mathbf{P}^T$. The non-Galerkin algorithm replaces $\mathbf{R}\cdot\mathbf{A}\cdot\mathbf{P}$ with a sparser coarse grid matrix, which aims to improve parallel scalability and maintain the convergence rate of AMG. The algorithm consists of two phases. In the first phase the sparsity pattern of the non-Galerkin coarse grid operator is selected. While preserving the row sum, the second phase removes the entries in $\mathbf{R}\cdot\mathbf{A}\cdot\mathbf{P}$ that lie outside the non-Galerkin sparsity pattern. Below, non-galerkin algorithm is discussed in details.

Non-Galerkin Algorithm

The non-Galerkin method consists of two complementary parts: Compute sparsity algorithm and Lumping algorithm. The non-zero pattern N_{NG} for $\mathbf{A}_{NG}$ is found by the Compute sparsity algorithm (see, Falgout and Schroeder (2014)). The second part of the non-Galerkin method, the Lumping process, performs the elimination of entries in A_G based on the sparsity pattern N_{NG}.

Compute sparsity

To compute the sparsity pattern, two processes are combined. One of the processes initializes the minimal sparsity pattern, while the other process targets the heuristic for mid-range and high frequency eigenmodes. The minimal sparsity pattern is created using the prolongation operator $\mathbf{P}$ and the fine grid discretization matrix $\mathbf{A}$. It is defines as

$$\bar{N}_{NG} = \{(i,j) \text{ such that } \left(\mathbf{P}_I^T\cdot\mathbf{A}\cdot\mathbf{P}+\mathbf{P}^T\cdot\mathbf{A}\cdot\mathbf{P}_I\right)_{ij}\neq 0\}, \tag{5}$$

where $\mathbf{P}_I$ is the injection operator between the coarse and fine grids. The minimal sparsity pattern in Equation (5) preserves the important entries of $\mathbf{A}_G$, independent of their magnitude, see Falgout and Schroeder (2014). If it was based on the on the classical strength-of-connection operator the entries with small magnitude would be removed. The sparsity pattern $\bar{N}_{NG}$ is improved by the non-Galerkin method in order to target the heuristic. Let the set of neighbours of i in N_{NG} be given by

$$N_{NGi} = \{j \text{ such that } (i,j)\in N_{NG}\}.$$

After the pattern N_{NG} is initialized as the pattern of $\mathbf{A}_G$, the algorithm removes the entries from N_{NG} operating row by row. It start with entries with the smallest magnitude and proceeds until further elimination would violate

$$2\sum_{j\neq N_{NGi}}|a_{ij}^G| \leq \gamma\sum_j |a_{ij}^G|. \tag{6}$$

The factor of 2 compensates for the maximum change made to $\mathbf{A}_G$ when dropping an entry and lumping its value to the allowed neighbours (see, Falgout and Schroeder (2014)).

Algorithm 1 Compute sparsity.

```
1:  Data: $\mathbf{A}_G, \mathbf{P}, \mathbf{P}_I$
2:  $N_{NG} \leftarrow \emptyset$
3:  for $(i, j)$ such that $a^G_{ij} \neq 0$ do
4:      $N_{NG} \leftarrow N_{NG} \cup \{(i, j)\}$
5:  end for
6:  for $i$ to $nrows(\mathbf{A}_G)$ do
7:      Initialize set $K$: $K_m$ is index of $m$th smallest magnitude off-diagonal nonzero in row $i$
8:      for $m = 1$ to $|K|$ do
9:          $N_{NGi} \leftarrow N_{NGi} \setminus K_m$
10:         if $2\sum_{j \neq N_{NGi}} |a^G_{ij}| \leq \gamma \sum_j |a^G_{ij}|$ then
11:             continue
12:         else
13:             $N_{NG} \leftarrow N_{NG} \cup K_m$
14:             break
15:         end if
16:     end for
17: end for
18: for $(i, j)$ such that $\left(\mathbf{P}_I^T \cdot \mathbf{A} \cdot \mathbf{P} + \mathbf{P}^T \cdot \mathbf{A} \cdot \mathbf{P}_I\right)_{ij} \neq 0$ do
19:     $N_{NG} \leftarrow N_{NG} \cup \{(i, j)\}$
20: end for
21: return $N_{NG}$
```

Lumping

Let the non-Galerkin operator be comprised of entries a^{NG}_{ij}. The lumping algorithm begins by initializing $\mathbf{A}_{NG}$ as a copy of $\mathbf{A}_G$. After that, each entry a^{NG}_{ij} that is not in N_{NG} is removed from $\mathbf{A}_{NG}$. A fraction of the value of a^{NG}_{ij} is added to each of j's strongly connected neighbours in row i. This is done to preserve the row sum of the Galerkin matrix as required by the heuristic. The lumping procedure uses as input the strength-of-connection matrix, S, the Galerkin coarse grid operator, $\mathbf{A}_G$ and the non-Galerkin sparsity pattern N_{NG}. The neighbours of j in S are defined as

$$N_{Sj} = \{k \text{ such that } s_{jk} \neq 0\},$$

where s_{jk} is an element of S. Subsequently, we find set U, which represents the strong connections of j shared by the nonzero pattern of row i. The neighbours of the eliminated a^{NG}_{ij}, to which its value should be lumped, are stored in U. If no strong neighbours are found, i.e. $U = \emptyset$, a^{NG}_{ij} is lumped to the diagonal. Finally, the algorithm symmetrizes $\mathbf{A}_{NG}$. Since this operation affects the row sums, it is followed by a row preserving procedure. The scheme is shown in Algorithm 2.

Algorithm 2 Lumping.

```
1: **Data**: $\mathbf{A}_G, S, N_{NG}$
2: $\mathbf{A}_{NG} \leftarrow \mathbf{A}_G$
3: **for** $i$ *to* $nrows(\mathbf{A}_{NG})$ **do**
4:   **for** $j$ *such that* $a_{ij}^{NG} \neq 0$ **do**
5:     **if** $j \notin N_{NGi}$ **then**
6:       $U \leftarrow N_{Sj} \cap N_{NGi}$
7:       **if** $U = \emptyset$ **then**
8:         $a_{ii}^{NG} \leftarrow a_{ii}^{NG} + a_{ij}^{NG}$
9:       **else**
10:        $U \leftarrow U \backslash \{i\}$
11:        $\sigma = \sum_{k \in U} |s_{jk}|$
12:        **for** $k \in U$ **do**
13:          $a_{ik}^{NG} \leftarrow a_{ik}^{NG} + (|s_{jk}|/\sigma) a_{ij}^{NG}$
14:        **end for**
15:      **end if**
16:      $a_{ik}^{NG} \leftarrow 0$
17:    **end if**
18:  **end for**
19: **end for**
20: $\mathbf{A}_{NG} \leftarrow 0.5 \left(\mathbf{A}_{NG}^T + \mathbf{A}_{NG}\right)$
21: **for** $i = 1$ *to* $nrows(\mathbf{A}_{NG})$ **do**
22:   $a_{ii}^{NG} \leftarrow a_{ii}^{NG} + \sum_j a_{ij}^G - \sum_j a_{ij}^{NG}$
23: **end for**
24: **return** $\mathbf{A}_{NG}$
```

Non-Galerkin for Reservoir Simulations Problems

In reservoir simulators, the resulting matrix is generally not symmetric and, hence, not symmetric positive-definite (SPD) matrix. There is no previous related work that considers the application of the non-Galerkin algorithm to not SPD matrices. Despite this fact, the non-Galerkin method was implemented to solve reservoir simulation problems. Since the simulator makes no assumptions about the symmetry of the coarse grid operators, the symmetrization step (line 20 to line 22) within the Lumping algorithm was not included in the code. Furthermore, based on some experiments, the value of γ in Algorithm 1 was set to 0.03.

Modified Non-Galerkin

The analysis of the coarse grid operators during the simulations reveals that the non-Galerkin matrices typically satisfy most of the M-matrix properties. More precisely, the non-Galerkin operators generally contain a number of positive off-diagonal entries, but frequently satisfy the remaining properties of the M-matrix definition.

Algorithm 3 Lumping.

```
1: Data: $A_G, S, N_{NG}$
2: $\mathbf{A}_{NG} \leftarrow \mathbf{A}_G$
3: for $i$ to $nrows(\mathbf{A}_{NG})$ do
4:   for $j$ such that $a_{ij}^{NG} \neq 0$ do
5:     if $j \notin N_{NGi}$ then
6:       $U \leftarrow N_{Sj} \cap N_{NGi}$
7:       if $U = \emptyset$ then
8:         $a_{ii}^{NG} \leftarrow a_{ii}^{NG} + a_{ij}^{NG}$
9:       else
10:        $U \leftarrow U \backslash \{i\}$
11:        $\sigma = \sum_{k \in U} |s_{jk}|$
12:        for $k \in U$ do
13:          $a_{ik}^{NG} \leftarrow a_{ik}^{NG} + (|s_{jk}|/\sigma) a_{ij}^{NG}$
14:        end for
15:      end if
16:      $a_{ik}^{NG} \leftarrow 0$
17:    end if
18:  end for
19:  for $l$ such that $l \neq i$ and $a_{il}^{NG} \neq 0$ do
20:    if $a_{il}^{NG} > 0$ then
21:      $a_{ii}^{NG} \leftarrow a_{ii}^{NG} + a_{il}^{NG}$
22:      $a_{il}^{NG} \leftarrow 0$
23:    end if
24:  end for
25: end for
26: return $\mathbf{A}_{NG}$
```

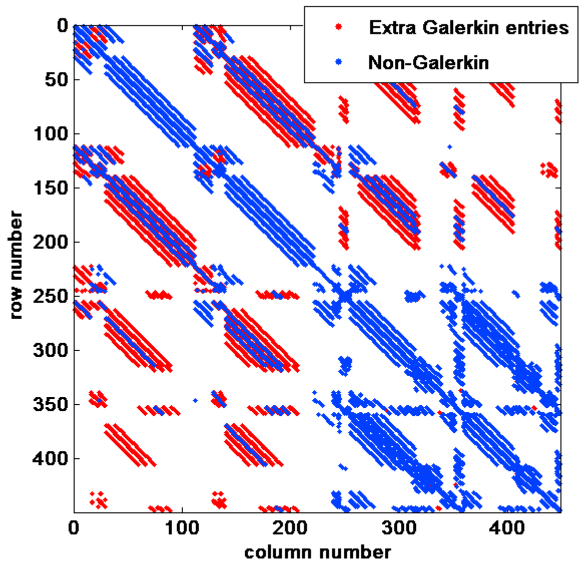
The importance of the M-matrix properties for the non-Galerkin algorithm is supported by several observations. Firstly, being an M-matrix guarantees the convergence of the basic iterative methods. Secondly, AMG was originally designed for the M-matrices Briggs et al. (2000). Therefore, the use of coarse grid operators with the M-matrix properties should be beneficial for the algorithm. Finally, M-matrices belong to the more general class of monotone matrices, i.e. matrices with nonnegative inverses. Usually, the use of monotone matrices considerably improves the performance of the Multiscale Finite Volume method, which is closely related to two-grid AMG.

In the light of provided arguments, we remove the positive off-diagonal entries in order to enforce as many M-matrix properties as possible. To preserve the row sums, as required by the non-Galerkin method, the values of eliminated entries are lumped to the diagonal. We note that this process can change the effect of the non-Galerkin method on the high and mid-range frequency modes, because it increases the sparsity of the operators. The elimination of the positive off-diagonal entries is implemented within reservoir simulator as a part of the Lumping algorithm. The modified lumping procedure used within INTERSECT is presented in Algorithm 3.

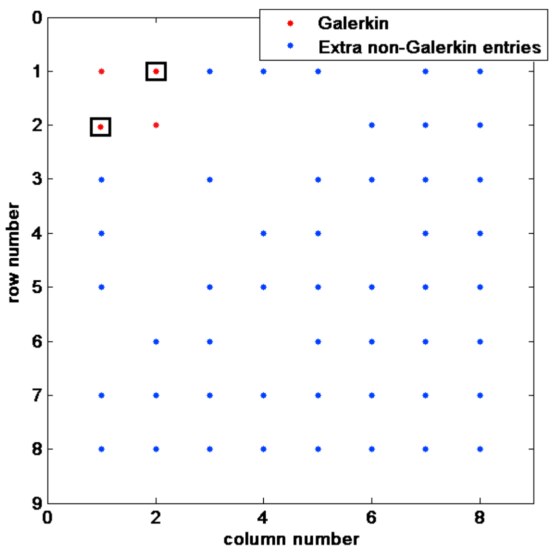
Numerical Results

In this section we present results for the non-Galerkin method and the modified non-Galerking method introduced in this paper. We have implemented both methods in a commercial reservoir simulator and we will compare against Galerkin AMG, which is the default solver.

We describe the test cases in terms of the number of grid cells, fluid models and time discretization. We evaluate the methods by comparing the characteristics provided below.



(a) Second coarse level.



(b) Fifth coarse level.

Figure 3 *Sparsity patterns. In Figure (a), the blue dots form sparsity pattern of the non-Galerkin operator, whereas the combination of the blue and red dots represents the sparsity of the Galerkin operator. In Figure (b), the red dots correspond to the Galerkin sparsity pattern, while the combination of the blue and red dots without squares shows the sparsity pattern of the non-Galerkin operator.*

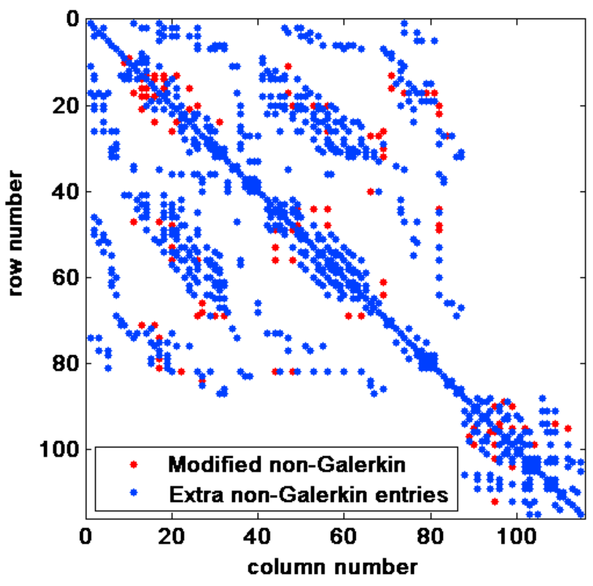
- *Algebraic complexity*
- *Geometric complexity*
- *Time steps:* the total number of time steps required to complete the simulation.
- *Non-linear iterations:* the number of non-linear iterations in the simulation.
- *Linear iterations:* the number of iterations used to solve the linear systems generated by the non-linear solver.
- *Linear solver time:* the amount of time needed to solve the linear systems.
- *CPU time:* the overall computational time required to complete the simulation.

Test Cases

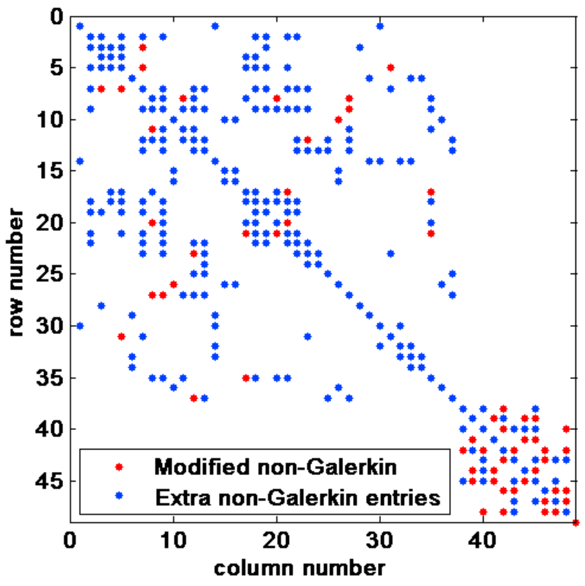
In Table 1 we present eleven test cases ranging from small toy models to larger real life test cases. We consider black oil, iso-thermal and thermal compositional models with varying degree of heterogeneity in the reservoir grid properties. We have named the cases based on the number of cells. Naturally, the number of cells is equal to the number of rows in the pressure matrix A^*_{pp} from Equation (**??**). The dimensions correspond to the total number of cells in the x-, y- and z-direction. Table 1 contains several examples where the number of active cells is lower than the number of cells suggested by the dimensions. In those cases, the domain includes a number of inactive cells and local grid refinements.

Non-Galerkin

Is this section we present results for the non-Galerkin method. For a fair comparison between the different methods we have disabled any heuristics to reduce the setup costs of AMG as well as varying the linear solver tolerances related to Newton forcing terms.



(a) Third coarse level.



(b) Fourth coarse level

Figure 4 *Sparsity patterns. The blue dots form sparsity pattern of the modified non-Galerkin operator, whereas the combination of the blue and red dots represents the sparsity of the standard non-Galerkin operator.*

I. Algebraic and Geometric Complexity

The effect of the non-Galerkin algorithm on the operator and grid complexity is demonstrated in Figure 6a and 6b, respectively. Delta illustrates the difference in the complexities. In Figure 6a the decrease in the algebraic complexity is demonstrated. The total bar height represents the operator complexity when the Galerkin method is applied. The blue part of a bar shows the complexity obtained with the non-Galerkin approach. In Figure 6b we demonstrate the increase in the geometric complexity due to the non-Galerkin method. The total bar heights denotes the grid complexity with the non-Galerkin algorithm. The red part shows the complexity obtained with the Galerkin method.

In general, while maintaining the geometric complexity, the non-Galerkin algorithm significantly reduces the algebraic complexity. The in-depths analysis of the non-Galerkin coarse grids reveals that the increase in the geometric complexity originates from the coarsest levels, where the non-Galerkin algorithm preserves more variables than the Galerkin approach. Moreover, the number of non-zero elements on the coarsest levels generated by non-Galerkin is usually higher than that of Galerkin. This implies that the decrease in the algebraic complexity, usually originating from the first three/four coarse levels, is slightly diminished by the coarser levels.

II. Small cases

For the small cases, the number of time steps and non-linear iterations are not changed when the non-Galerkin method is applied instead of the standard Galerkin technique. The total number of linear iterations required for these cases is illustrated in Figure 7a.

II. Large cases

In Table 2 we observe that the number of time steps differs between the Galerkin and non-Galerkin algorithms. The number of non-linear iterations of the simulations with the Galerkin and non-Galerkin algorithms can be found in Table 3. We observe that the non-Galerkin method increases the number of

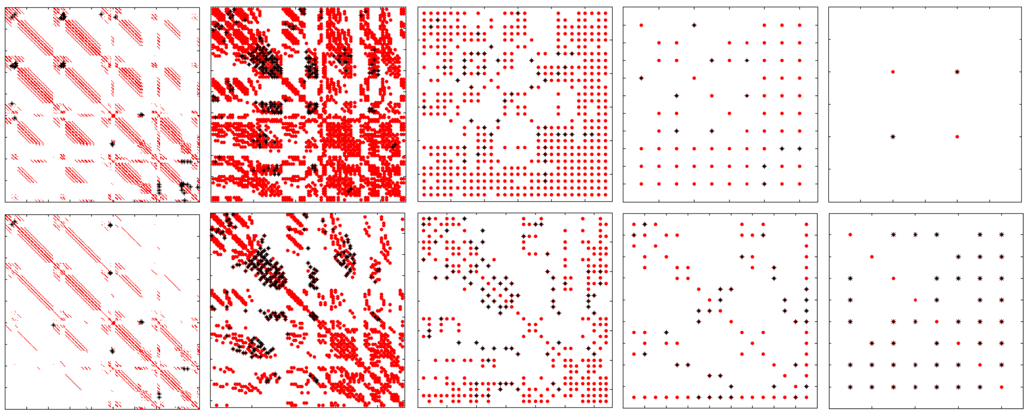
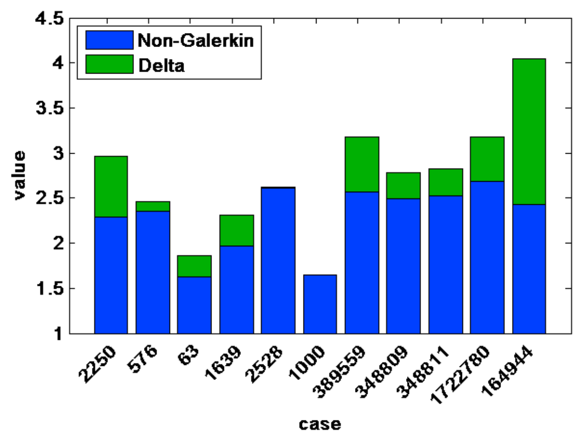
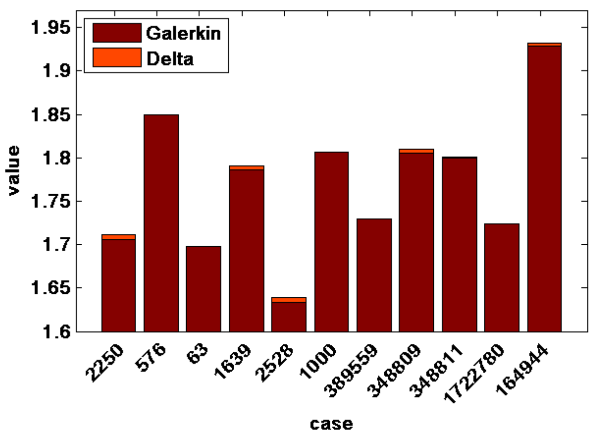


Figure 5 *The sparsity patterns of the Galerkin and non-Galerkin operators. The negative and positive off-diagonal entries are shown in red and black, respectively. The first row is formed by the Galerkin operators starting from the second coarse level. The second row shows the non Galerkin operators on the same levels.*



(a) Algebraic.



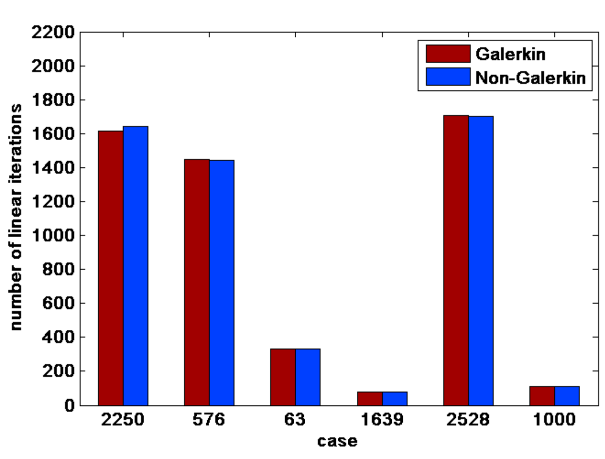
(b) Geometric.

Figure 6 *The influence of the non-Galerkin algorithm on the algebraic and geometric complexity with respect to the Galerkin method.*

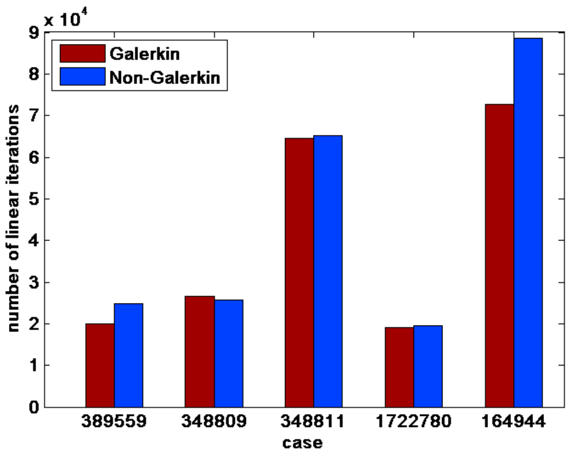
non-linear iterations for all the large cases except the 348809-case. The maximal increase is 28.8%.

The number of linear iterations is demonstrated in Figure 7b. Clearly the number of linear iterations increases for the non-Galerkin method. In Figure 8 we present the cumulative number of linear iterations for the 389559-case which is representative for the other large test cases. We can observe that linear solver is slightly less robust over the whole simulation and not just for a single patch. This indicates that the convergence rate of the non-Galerkin method is worse than default AMG as other components of the linear solver have not changed. However, the increase in linear iterations is acceptable given the fact that the non-Galerkin method has a more favorable sparsity pattern that will affect strong scalability.

In Figure 9 we observe that the total simulation time is increased when the non-Galerkin method is applied. As mentioned above, the increase in time arises mainly from the setup phase, that consists of the construction of the injection matrix, the sparsity pattern calculation and the lumping procedure.



(a) Small cases.



(b) Large cases.

Figure 7 Number of linear iterations with the Galerkin and non-Galerkin coarse grids.

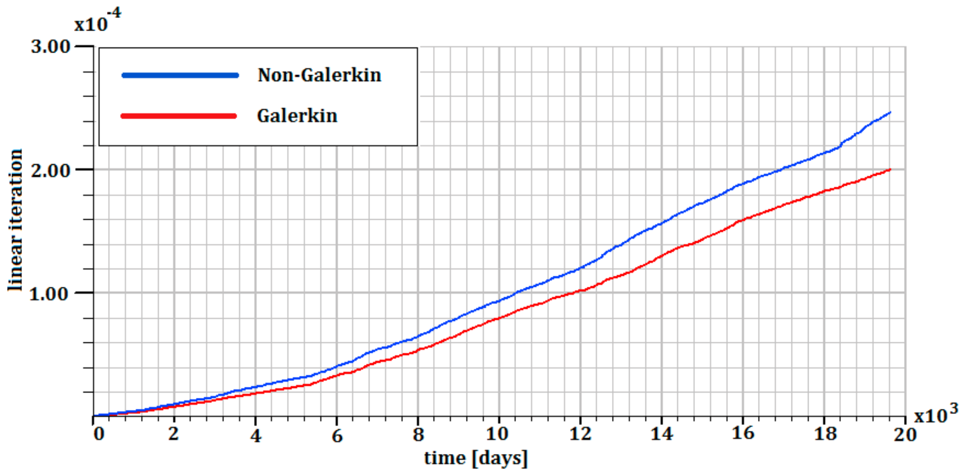


Figure 8 Cumulative number of linear iterations of the 389559-case with Galerkin and non-Galerkin methods.

Modified Non-Galerkin

In this section we present the results of the Modified non-Galerkin method and compare to the default AMG solver and non-Galerkin method. All benchmarks are without any heuristics to reduce setup costs and linear iterations.

I. Algebraic and Geometric Complexity

The replacement of the undesired off-diagonal values decreases the algebraic complexity of the Modified non-Galerkin method compared to the standard non-Galerkin metod by approximately 1.5%. This implies that the complexity is substantially improved in comparison with the Galerkin method. For the small 576-, 63- and 1000-case, the complexities are unchanged, because the coarse grid operators of these cases contain no positive off-diagonal entries. But for the 1722780-case, the decrease is exceptionally large, almost 25%. The effect of the modifications on the geometric complexity of standard non-Galerkin seems arbitrary. The geometric complexity can be slightly higher, lower or remain completely unchanged. The differences in the geometric complexity are not higher than 0.5%.

II. Small cases

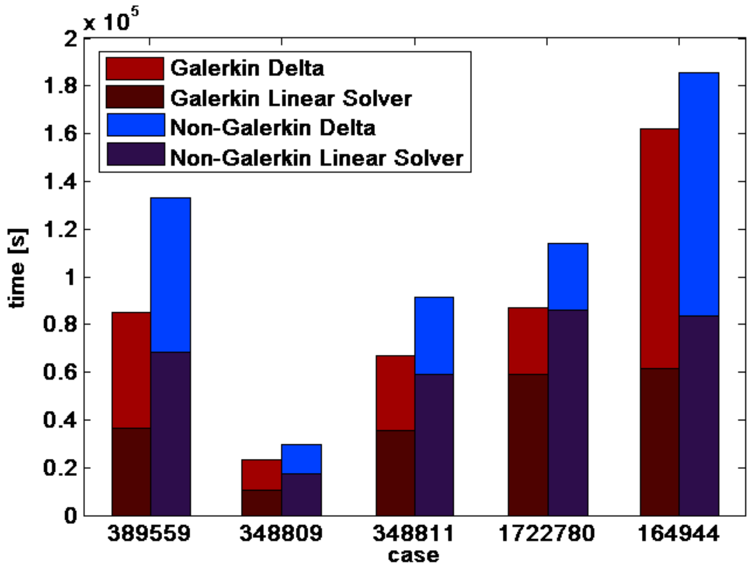
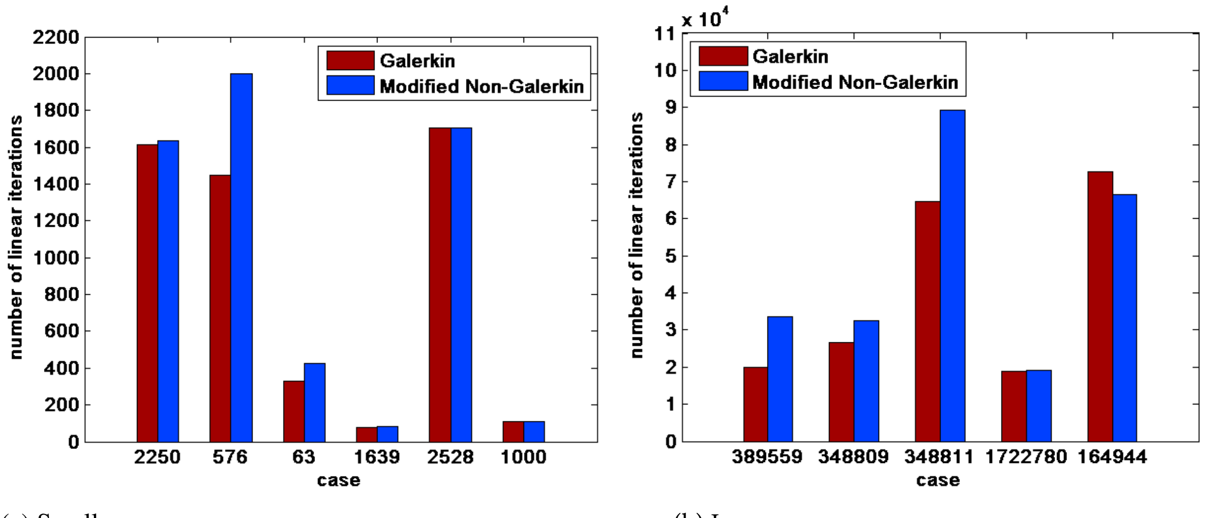


Figure 9 *Total simulation time and linear solver time with Galerkin and non-Galerkin methods. The total bar length is equal to the total simulation time, whereas the bottom part denotes the linear solver time.*

The Modified non-Galerkin method does not change the number of time steps and non-linear iterations compared to the Galerkin method. However, in some cases there is a significant effect on the number of linear iterations as presented in Figure 10a. There are no changes for the 1000-case and the increase for the 1639- and 2528-case is negligible. For the remaining three cases the difference is noticeable: the increase for the 576-case is 40%. From the previous results follows that for the 2250-, 576- and 63-case the modified version also significantly increases the number of linear iterations compared to the standard non-Galerkin approach.



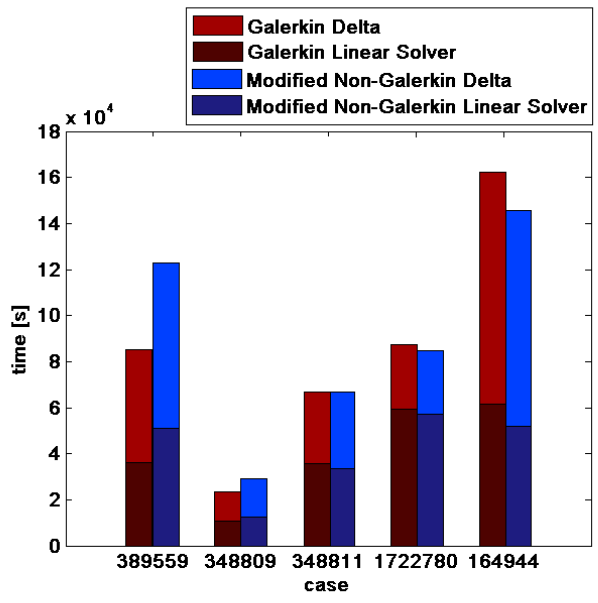
(a) Small cases.

(b) Large cases.

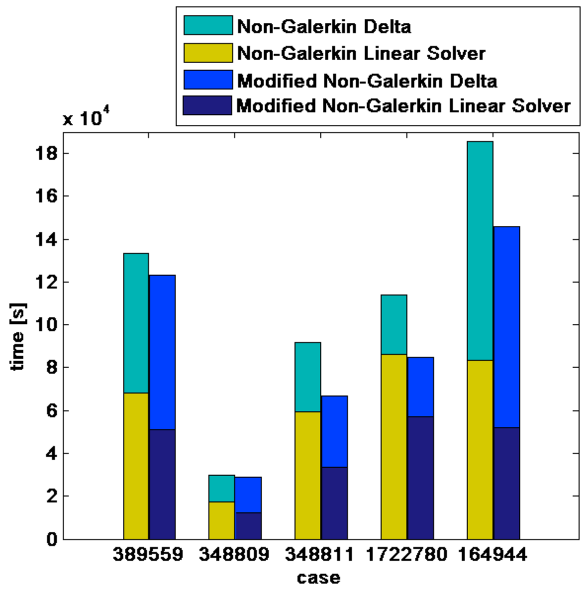
Figure 10 *Number of linear iterations with the Galerkin and modified non-Galerkin coarse grids.*

III. Large cases

In Table 2 and Table 3 we present the number of time steps and non-linear iterations of the modified non-Galerkin and Galerkin method respectively. The Modified non-Galerkin method requries more non-linear iterations than the Galerkin method and the number of linear iterations increases for all cases except the 164944-case. However, the duration of a linear iteration is reduced when the Galerkin operators are replaced by the non-Galerkin operators with negative off-diagonal entries. Therefore, when the difference in the number of linear iterations is not too large, the linear solver time of the modified non-Galerkin is lower than the Galerkin linear solver time. This is shown in Figure 11a.



(a) Galerkin.



(b) Standard non-Galerkin.

Figure 11 *Total simulation time and linear solver time of the modified non-Galerkin method compared to the Galerkin and standard non-Galerkin method. The total bar length is equal to the total simulation time, whereas the bottom part denotes the linear solver time.*

The modified non-Galerkin algorithm improves the results of the standard non-Galerkin method in terms of linear solver and total time as illustrated in Figure 11b. The total simulation time is significantly re-duced for the thermal 1722780- and 164944-case. In addition, a small decrease is observed for the 348811-case. On the other hand, the CPU time of the 389559- and 348809-case increases when the modified non-Galerkin algorithm is used instead of the Galerkin method. The increase originates from the linear solver and linearization time. Since the number of non-linear iterations increases considerably for the 389559-case, the contribution of the linearization time exceeds the difference caused by the linear solver. However, we should point out that linearization is embarressingly parallel, and thus shifting work from the linear solver to the non-linear solver will benefit the strong scalability.

Conclusions

In this paper we reviewed the non-Galerkin method for reservoir simulation and introduced the modified non-Galerkin method which improves the M-matrix properties of the non-Galerkin coarse grid operators. We have showed that the non-Galerkin method can be a viable alternative to the Galerkin method whilst reducing the sparsity pattern of the AMG coarse grid to allow for an improved strong scalability of the linear solver. Moreover, we have showed that the Modified non-Galerkin method is more efficient and robust than the non-Galerkin method and even outperforms the Galerkin method in terms of CPU time due to the reduced sparsity pattern. Future work will focus on the introduction of heuristics to reduce the setup costs of the Modified non-Galerkin method. Furthermore, the obtained results reveal that on the coarsest levels the Galerkin algorithm can be more efficient than the non-Galerkin approach. Thus, the restriction of the number of levels, to which the non-Galerkin method is applied, may be used to optimize the structure of the AMG hierarchy.

Acknowledgements

The authors would like to thank Lisa Wobbes for her excellent research and prototype of the (modified) Non-Galerkin method in our reservoir simulator.

Appendix

Active cells	Dimensions	Fluid model	Implicitness	Number of phases	Number of active components
2,250	$15 \times 15 \times 10$	Black Oil Isothermal	Fully Implicit	4	4
576	$24 \times 1 \times 24$	Black Oil Isothermal	Fully Implicit	4	4
63	$1 \times 7 \times 9$	Black Oil Isothermal	Fully Implicit	3	3
1,639	$20 \times 15 \times 8$	Compositional Isothermal	AIM IMPES	3	9
2,528	$9 \times 9 \times 4$	Compositional Isothermal	AIM IMPES	3	10
1,000	$10 \times 10 \times 10$	Black Oil Isothermal	Fully Implicit	3	3
389,559	$154 \times 91 \times 35$	Compositional Isothermal	AIM IMPES	3	13
348,809	$238 \times 192 \times 114$	Black Oil Isothermal	Fully Implicit	3	3
348,811	$238 \times 192 \times 114$	Compositional Isothermal	AIM IMPES	3	8
1,722,780	$18 \times 1126 \times 85$	Compositional Thermal with steam permitted	Fully Implicit	3	3
164,944	not a ‘box’	Compositional Thermal with steam permitted	Fully Implicit	3	3

__Table 1__ General properties of the test cases.

Algorithm	Case				
	389559	348809	348811	1722780	164944
Galerkin	1165	2337	3036	447	28418
Non-Galerkin	1312	2276	3090	456	28487
Modified non-Galerkin	1582	2319	3208	463	28403

__Table 2__ Number of time steps with the Galerkin and non-Galerkin coarse grids.

Algorithm	Case				
	389559	348809	348811	1722780	164944
Galerkin	7296	5783	12701	1689	30820
Non-Galerkin	9394	5626	13609	1778	31118
Modified non-Galerkin	12678	5834	14695	1750	30779

__Table 3__ Number of non-linear iterations with the Galerkin and non-Galerkin coarse grids.

References

Ashby, S. and Falgout, R. [1996] A Parallel Multigrid Preconditioned Conjugate Gradient Algorithm for Groundwater Flow Simulations. *Nuclear Science and Engineering*, **124**(1), 145–159.

Aziz, K. and Settari, A. [1979] *Petroleum Reservoir Simulation*. Elsevier Applied Science Publishers, London, 1979.

Briggs, W., Henson, V. and McCormick, S. [2000] *A Multigrid Tutorial*. Second Edition, SIAM, Philadelphia.

Darwish, M., Saad, T. and Hamdan, Z. [2006] A High Scalability Parallel Algebraic Multigrid Solver. *European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2006, P. Wesseling, E. Oñate, J. Périaux (Eds), TU Delft, The Netherlands*.

Falgout, R. [2006] An Introduction to Algebraic Multigrid. *Computing in Science and Engineering*, **8**(6), 24–33.

Falgout, R. and Schroeder, J. [2014] Non-Galerkin Coarse Grids for Algebraic Mutigrid. *SIAM Journal on Scientific Computing*, **36**(3), 309–334.

Gahvari, H., Baker, A., Schulz, M., Yang, U., Jordan, K. and Gropp, W. [2011] Modeling the Performance of an Algebraic Multigrid Cycle on HPC Patforms. *Proceedings of the international conference on Supercomputing, ICS '11, New York, NY, USA*, **21**(2), 172–181.

Krechel, A. and Stüben, K. [1998] Operator Dependent Interpolation in Algebraic Multigrid. *Proceedings of the Fifth European Multigrid Conference, Stuttgart, October 1-4, 1996, Lecture Notes in Computational Science and Engineering, Springer, Berlin*, (3).

Peaceman, D. [1977] *Fundamentals of Numerical Reservoir Simulation*. Elsevier Scientific Publising Company, New York.

Sterck, H.D., Yang, U. and Heys, J. [2006] Reducing Complexity in Parallel Algebraic Multigrid Preconditioners. *SIAM Journal on Matrix Analysis and Applications*, **27**(4), 1019–1039.

Stüben, K. [2000] *Multigrid*, chap. Algebraic Multigrid (AMG) An Introduction with Applications. Academic Press, San Diego, 209–236.

Vassilevski, P. and Yang, U. [2014] Reducing Communication in Algebraic Multigrid Using Additive Variants. *Numerical Linear Algebra with Applications*, **21**(2), 275–296.

Wallis, J. [1985] Constrained residual acceleration of conjugate residual methods. *SPE Reservoir Simulation Symposium*, (1), doi:10.2118/13536–MS.

Wienands, R. and Yvneh, I. [2009] Collocation Coarse Approximation (CCA) in Multigrid. *SIAM Journal in Scientific Computing*, **31**, 3643–3660.

Yang, U. [2006] *Parallel Algebraic Multigrid Methods - High Performance Preconditioners*, chap. Numerical Solutions of Partial Differential Equations on Parallel Computers. Spinger-Verlag, 209–236.

Yang, U. [2010] On Long-range Interpolation Operators for Aggressive Coarsening. *Numerical Linear Algebra with Applications*, **17**, 453–472.