2017

Incorporating Algebraic Multigrid when Solving Reservoir Simulation Equations

Brown, Geoffrey Lawrence

UNIVERSITY OF CALGARY

Incorporating Algebraic Multigrid when Solving Reservoir Simulation Equations

by

Geoffrey Lawrence Brown

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN CHEMICAL AND PETROLEUM ENGINEERING

CALGARY, ALBERTA

APRIL, 2017

Abstract

Adaptive-implicit petroleum reservoir simulations often result in huge, very ill-conditioned linear systems of equations. The convergence of single-level solvers, for these systems tends to deteriorate considerably as these systems become larger. The primary difficulty is the coupled nature of the model equations that exhibit characteristics of both hyperbolic and nearly elliptic systems. Multilevel methods provide a technique to efficiently solve nearly elliptic systems independent of size whereas preconditioned Krylov solvers are more robust for both. Therefore, multi-stage preconditioning methods are commonly used in practice to solve coupled systems.

In this thesis a two-stage combinative method, the constrained pressure residual (CPR) method, has been implemented in a commercial black-oil reservoir simulator. A decoupling preconditioner has been developed to approximately decouple and extract the nearly elliptic pressure sub-system. An algebraic multigrid (AMG) method is used to efficiently solve the pressure system followed by an incomplete LU (ILU) technique as a robust second stage solver. The primary objective is to generate systems that are suitable for an efficient AMG solution while simultaneously ensuring a fast overall convergence of the solver.

The combinative solver and the dynamic row-sum (DRS) preconditioner have been developed to include several optional variants. A study of these parameters has been performed to determine their importance and select reasonable default values. The accuracy of the new combinative method has been verified. It is generally able to produce small residual vectors with fewer solver iterations. Simulator efficiency has been improved for an assortment of test cases. Several simulations had clock times between 1.5 and 3 times faster with the combinative solver. Parameter choices leading to a small number of fast AMG cycles for the pressure solution at the sacrifice of some accuracy generally led to the best performance.

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Dr. Zhangxing (John) Chen, for giving me the opportunity to further my education. He has provided a unique research environment allowing me use my mathematics and fluids background in a new field of study for me. I have been able to broaden my field of knowledge and apply it to industrial applications.

I would like to thank my manager at the Computer Modelling Group, Dave Collins. He gave me the opportunity to combine a research project with a practical application. His guidance and direction were invaluable in shaping this research. The practical knowledge I have gained from his experience is innumerable.

I wish to thank the members of my advisory committee for their time, comments, and constructive criticism on my thesis.

I gratefully acknowledge the financial support from the following agencies that have provided funding that made this work possible: the Government of Alberta through the Queen Elizabeth II Graduate Scholarships program, the Dr. Zhangxing Chen NSERC/AIEES/Foundation CMG IRC in Reservoir Simulation, and the Dr. Zhangxing Chen iCore Chair in Reservoir Modelling.

I would also like to thank my family and friends who have supported me along the way. Especially, my parents, Kevin and Patti, who fostered a love of math and science early on and were there to nurture my passion as it grew.

Finally, and most importantly, I would like to express my gratitude to my wife, Melisa. She has helped me through the stressful times when the work seemed too difficult to handle on my own. She has taken on so many of the family responsibilities so I could proceed with my study late into the evening. Thank you so much for your love, support, and encouragement, without which this work would not have been accomplished.

To my wife, Melisa. Without you I could not have done this.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Throughout this thesis capitalized bold letters (*i.e.* $\boldsymbol{A}$) will denote matrices and lower case bold letters will denote vectors (*i.e.* $\boldsymbol{y}$). Below is a list of the specific meaning of symbols.

Symbol	Definition
B	Formation Volume Factor
g	Gravitational Acceleration
k_r	Relative Permeability
$\boldsymbol{K}$	Absolute Permeability
$\mathcal{K}$	Krylov Subspace
N_c	Number of Components
p	Pressure
q	Well Rate
R_s	Solution Gas-Oil Ratio
S	Saturation
t	Time
$\boldsymbol{T}$	Transmissibility
$\boldsymbol{u}$	Velocity
x	Length
z	Depth
δ	Change in Variable
μ	Viscosity
ρ	Density
ϕ	Porosity
ψ	Potential
Subscripts	
ℓ	Phase (o, w, g)
β	Component (O, W, G)

Below is a list of the specific meaning of frequently used abbreviations used in this document.

Abbreviation	Definition
ABF	Alternate Block Factorization
AIM	Adaptive-Implicit
AMG	Algebraic Multigrid
CMG	Computer Modelling Group
CPR	Constrained Pressure Residual
DRS	Dynamic Row-Sum
FI	Fully Implicit
GMRES	Generalized Minimum Residual Method
GOC	Gas Oil Contact
ILU	Incomplete LU Factorization
IMPES	Implicit Pressure, Explicit Saturation
PVT	Pressure Volume Temperature
WOC	Water Oil Contact

Chapter One: **Introduction**

Reservoir simulation is a tool used to aid decision making for reservoir development and management. Valuable knowledge about the processes occurring within the interior of a reservoir that cannot be observed can be obtained through numerical simulation. Reservoir processes are modelled using a system of differential equations describing primarily the flow of fluid through the reservoir and the physical behaviour of the fluid. These equations are discretized so as to be solvable numerically. This leads to a large, sparse system of highly coupled, non-linear, algebraic equations that determine the unknown state of the reservoir at a given time. Such systems necessarily become more complex and larger as complexity is added to the processes being modelled and as the model becomes more detailed with additional physical information. Solving these generated systems of equations usually requires the solution to a matrix problem of the type $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ for the unknown state of the reservoir $\boldsymbol{x}$ and is generally the most computationally intensive and therefore slowest task. Usually the solution of equations takes the majority of the total reservoir simulation time. Often this can be as large as 90% (Chen, Huan, & Ma, 2006, p. 3). Clearly, in order to solve more complex and larger models within the decision-making time frame, the computation time of reservoir simulation, and therefore the numerical solution to systems of equations, must be reduced. Moreover, decision-making now requires many successive simulations to produce ensembles used to quantify uncertainty and aid in determining optimal reservoir management making efficient numerical simulation even more important.

Computational time reduction can be realized through computer hardware improvements, software techniques to take advantage of hardware modifications, and general algorithmic changes amongst other software improvements. This research will focus on new algorithmic

techniques, in particular those involving the use of the Algebraic Multigrid (AMG) method, to improve solver efficiency.

1.1 Background

A brief description of the theoretical background most relevant to this research project follows. It is taken from (Brown, Collins, & Chen, 2015). A more thorough literature review is presented in Chapter Two:.

A source of the difficulty when solving the linear systems resulting from the discretization and linearization of the equations describing fluid flow in porous media is that they exhibit mixed parabolic-hyperbolic character (Aziz & Settari, 1979; Trangenstein & Bell, 1989). In reservoir simulation, the pressure field is essentially parabolic or nearly elliptic with long-range coupling, while the remaining equations (referred to as saturation equations) are hyperbolic or transport-dominated parabolic (Aziz & Settari, 1979; Peaceman, 1977; Trangenstein & Bell, 1989). For fully implicit (FI) time discretization the pressure field of the system is updated simultaneously with the saturation fields. This produces linear systems that are coupled. The coupling of pressure and saturation is reduced for IMPES (implicit pressure, explicit saturation) time discretization and adaptive-implicit (AIM) formulations (Forsyth & Sammon, 1986) but is not eliminated. In these formulations saturations remain coupled to pressure locally. Traditional solution techniques, including incomplete LU factorization (ILU) (Behie & Forsyth, 1984) and nested factorization (Appleyard & Cheshire, 1983) as preconditioners of Krylov subspace methods, have been popular because of their robustness. However, they are not the most efficient methods for solving parabolic or elliptic problems. The Constrained Pressure Residual (CPR) method (Wallis, Kendall, & Little, 1985; Wallis, 1983) can achieve improved convergence by targeting the parabolic portion of a linear system as a separate inner stage in a two-stage approach. A two-stage method exploits the fact that ILU techniques are well suited for

hyperbolic problems and multilevel techniques, in particular AMG solvers, are most efficient at solving elliptic and nearly elliptic problems (Cleary et al., 2000; Ruge & Stüben, 1987; Stüben, 2001a). Using AMG as part of a two-stage CPR preconditioner is a common approach in reservoir simulation (Cao, Tchelepi, Wallis, & Yardumian, 2005; Lacroix, Vassilevski, & Wheeler, 2001; Scheichl, Masson, & Wendebourg, 2003; Stüben, Clees, Klie, Lu, & Wheeler, 2007). An efficient application of a two-stage technique often uses a left preconditioner as a preprocessing step to weaken the coupling between pressure and saturation. Popular decoupling operators include alternate block factorization (ABF) (Bank, Chan, Coughran, & Smith, 1989) and quasi-IMPES (Lacroix et al., 2001; Scheichl et al., 2003). The primary goal of the decoupling processes is to ensure a fast overall convergence rate of the CPR method by reducing the impact of the outer global stage on the inner pressure stage. Indeed, decoupling preconditioners can also be applied as a global left preconditioner for a single-stage method to improve convergence rates. Decoupling preconditioners can aid the convergence of the inner pressure solution as well. They mix the governing equations locally ensuring the components most affected by pressure changes are included in the inner stage of CPR. An ideal decoupling preconditioner preserves the algebraic properties of the pressure system for AMG. However, decoupling changes the scaling of equations at different locations in the grid possibly destroying diagonal-dominance properties or producing a strongly non-symmetric pressure system that is not well suited for solution by AMG (Scheichl et al., 2003). AMG was originally designed to solve linear systems that correspond to discretized elliptic partial differential equations. However, the properties of the pressure fields considered in reservoir simulations are influenced by the physics of porous media flow. In particular, upstream weighting of flow terms may locally

modify the properties of nearly elliptic problems to such an extent that the convergence of AMG is adversely affected (Clees & Ganzer, 2010; Stüben et al., 2007).

1.2 Research Objectives

The aim of this research is to develop an algorithm to effectively incorporate an AMG preconditioner within an existing reservoir simulator currently utilizing an ILU preconditioner. Specifically, an AMG software package developed by Fraunhofer's SCAI institute, (SAMG) is incorporated into the Computer Modelling Group's (CMG) black-oil simulator (IMEX) that uses a variable degree ILU solver (PARASOL). The primary difficulty for AMG in reservoir simulation is due to the coupled hyperbolic-parabolic (elliptic) nature of the governing equations. The question is how to effectively decouple sections of the matrix so that AMG can be applied to the type of system it has success with. A suitable preconditioner for the surface volume based model equations (2.6) to (2.8) of IMEX that accomplishes the decoupling is developed in this research. The goal is to improve the efficiency of the numerical solver portion of the simulator for an assortment of test cases. The algorithm utilizes a collection of optional tuning parameters so that an additional goal is to determine reasonable default values for these parameters. The chosen defaults will likely not be optimal for any of test cases but should provide efficiency improvement for all the test cases. The idea is that for any new (unknown) reservoir simulation cases the default algorithm will give a reasonable running time.

1.3 Significance of Research

Clearly, to solve ever increasingly complex and larger models within a reasonable decision-making time frame requires a reduction in the computation time of reservoir simulation and therefore the numerical solution to associated systems of equations. Moreover, decision-making now requires many successive simulations to produce ensembles used to quantify uncertainty

and aid in determining optimal reservoir management making efficient numerical simulation even more important.

CPR methods are becoming popular and widely used within reservoir simulation; however it is not clear how to most efficiently apply the method. This research will provide new decoupling and weighting techniques, and quantify the usefulness of many of the options available within AMG. As this will be done within the framework of existing commercial software, large complex models will be used in the process rather than relying on only the simplest readily available data. Due to the involvement of CMG, a leading reservoir simulation software provider, the types of simulations that will be tested with this technique are current and relevant to the industry.

1.4 Outline

In Chapter Two: a review of reservoir simulation methods used in this study is presented. This includes a description of the model equations and linear systems to be solved as well as solution techniques. Chapter Three: discusses the combinative method implemented and the newly developed preconditioner. Variations to the combinative algorithm are described for the many options included. In Chapter Four: numerical analysis of the solver methods are presented including a description of the simulation environment used in this study and the models tested in §4.1 and §4.2. In §4.3 validation of the solver is presented with the establishment of linear solver convergence and full simulation results are verified. In §4.4 the effect of preconditioner method on the eigenvalue distribution for individual linear systems from the test cases is examined. In Chapter Five: performance results are given for both synthetic models and models based on real field data. The effectiveness of the combinative solver on simulation clock times is presented and the effect of parameter variations is examined. The cause of improvements to the efficiency of

the solver is also explored. Chapter Six: provides a summary of significant findings and describes some possibilities for further study.

Chapter Two: **Review of Reservoir Simulation Methodologies**

The following is an overview of methods used in reservoir simulation leading to the generation and solution of model equations. In particular, the review will focus on the methods to be used within this research project.

2.1 Model Formulation

Fluid flow in reservoir simulation is described mathematically as a system of PDEs governing subsurface flow. This study focuses on black-oil reservoir simulation as described first in Peaceman (1977) and later in Aziz and Settari (1979). The particular implementation used for this study is described in Forsyth and Sammon (1986). These equations result from the combination of the conservation of fluid mass and the conservation of fluid momentum for each of three components oil, water, and gas in three phases oil, water, and gas. The conservation of mass of fluid phase ℓ in a volume of rock V_r with porosity ϕ is

$$\frac{\partial(\phi\rho_\ell S_\ell)}{\partial t} = -\nabla\cdot(\rho_\ell \boldsymbol{u}_\ell) + q_\ell \tag{2.1}$$

where each phase has its own density ρ_ℓ, saturation S_ℓ (the fraction of reservoir fluid in phase ℓ), superficial fluid velocity $\boldsymbol{u}_\ell$, and source (well) term q_ℓ (mass rate per unit volume). This equation assumes no mass transfer between phases. For the black-oil model water is immiscible with oil and gas and that the gas component can be dissolved in oil. Because of mass transfer between the oil and gas phases, mass is not conserved within each phase, but rather the total mass of each component must be conserved. For each of three components oil, water, and gas equation (2.1) becomes

$$\frac{\partial(\phi\rho_{Oo}S_o)}{\partial t} = -\nabla\cdot(\rho_{Oo}\boldsymbol{u}_o) + q_O \tag{2.2}$$

$$\frac{\partial(\phi\rho_w S_w)}{\partial t} = -\nabla\cdot(\rho_w\boldsymbol{u}_w) + q_W \tag{2.3}$$

$$\frac{\partial}{\partial t}\left[\phi\left(\rho_{Go}S_o + \rho_g S_g\right)\right] = -\nabla\cdot\left(\rho_{Go}\boldsymbol{u}_o + \rho_g\boldsymbol{u}_g\right) + q_G \tag{2.4}$$

With ρ_{Oo} the partial density of oil in the oil phase (mass fraction times density), ρ_{Go} the partial density of gas in the oil phase and q_β the source term of component β.

Generally, in reservoir simulation Darcy's law, which was first derived empirically in by Darcy in (1856), is used to remove velocity from the computation, as is the case here. Darcy's law indicates a linear relationship between the fluid velocity and the pressure head gradient. The superficial Darcy velocity is replaced with

$$\boldsymbol{u}_\ell = -\frac{k_{r_\ell}}{\mu_\ell}\boldsymbol{k}(\nabla p_\ell - \rho_\ell \boldsymbol{g}\nabla z) \tag{2.5}$$

where $\boldsymbol{k}$ is the absolute permeability tensor of the porous medium, k_{r_ℓ} is the relative permeability to phase ℓ, μ_ℓ is the fluid viscosity, ∇p_ℓ is the pressure gradient, $\boldsymbol{g}$ is the gravitational acceleration, and z is the depth.

The formation volume factor of each phase B_ℓ is the ratio of the volume of fluid at reservoir conditions to the volume of the same fluid at standard conditions. Since the oil phase contains both gas and oil B_o is the ratio of the volume of oil plus its dissolved gas to the volume of the oil component at standard conditions, which is the inverse of the mass fraction of oil in the oil phase multiplied by the ratio of the surface oil density to the oil phase density. The mass fraction of the gas component in the oil phase can be determined from the solution gas oil ratio R_s, which is the volume of gas at standard conditions dissolved in a unit volume of oil at standard conditions

within the reservoir. Thus R_s is the mass fraction of gas in the oil phase multiplied by the ratio of the surface gas density to the oil phase density divided by B_o.

Dividing (2.2)-(2.4) by the fluid densities at standard conditions and substituting (2.5) the resulting model equations for the conservation of the three components for a fixed volume of rock are written as:

$$\frac{\partial}{\partial t}\left(\phi \frac{S_o}{B_o}\right) = \nabla \cdot (\boldsymbol{T}_o \nabla \psi_o) + q_o \tag{2.6}$$

$$\frac{\partial}{\partial t}\left(\phi \frac{S_w}{B_w}\right) = \nabla \cdot (\boldsymbol{T}_w \nabla \psi_w) + q_w \tag{2.7}$$

$$\underbrace{\frac{\partial}{\partial t}\left[\phi\left(\frac{S_g}{B_g} + R_s \frac{S_o}{B_o}\right)\right]}_{\text{accumulation term}} = \underbrace{\nabla \cdot \left(\boldsymbol{T}_g \nabla \psi_g + R_s \boldsymbol{T}_o \nabla \psi_o\right)}_{\text{flow term}} + \underbrace{q_g + R_s q_o}_{\text{source term}} \tag{2.8}$$

where

$$\boldsymbol{T}_\ell = \frac{k_{r_\ell}}{\mu_\ell B_\ell}\boldsymbol{k} \tag{2.9}$$

is the transmissibility of phase ℓ and

$$\psi_\ell = p_\ell - \rho_\ell g z \tag{2.10}$$

is the potential of phase ℓ. The source terms are defined by

$$q_O = \rho_{Oo} q_o,\ q_W = \rho_W q_w,\ q_G = \rho_G q_g + \rho_G R_s q_o \tag{2.11}$$

where terms q_o, q_w, q_g represent volume sources of oil, water, and gas at standard conditions, per unit time per unit reservoir volume. Wells are modelled using the equation

$$q_\ell = \frac{2\pi h}{\ln(r_e/r_w) + S}\boldsymbol{T}_\ell (p_{\text{bh}} - \psi_\ell) \tag{2.12}$$

Where h is the perforated height of the well, r_e is the drainage radius of the well, r_w is the wellbore radius, S is a skin factor (for numerical adjustments), and p_{bh} is the bottom hole pressure of the well.

The model equations represent balances on "standard volumes" or surface volumes rather than mass or reservoir volumes. In this model there is no oil component in the gas phase though that could be allowed for in a volatile oil formulation using a similar structure for the oil equation (2.6) as in the gas equation (2.8). The derivatives with respect to time in the above equations are referred to as accumulation terms, the spatial derivatives as flow terms, and the others as source terms. An additional constraint equation is required to close the system. The reservoir fluid is assumed to fill the pore volume giving the saturation constraint

$$S_o + S_w + S_g = 1. \tag{2.13}$$

For a black-oil formulation the constraint equation is simple enough that it is substituted into the component equations to eliminate one of the saturations.

Pressure, volume, and temperature (PVT) dependent data such as formation volume factors and solution gas-oil ratio are inputs to the model. This is usually done via input tables giving the relevant fluid properties for a sequence of pressures. Reservoir properties such as porosity and permeability of the rock (matrix) are also model inputs.

The model formulation considered also includes extensions to volatile-oil and solvent injection. Both are used in this study. For solvent injection an additional conservation equation for a solvent component that can exist in a solvent phase or in the water phase is modelled. The solvent equation is

$$\frac{\partial}{\partial t}\left[\phi\left(\frac{S_s}{B_s} + R_{ss}\frac{S_w}{B_w}\right)\right] = \nabla\cdot(\boldsymbol{T}_s\nabla\psi_s + R_{ss}\boldsymbol{T}_w\nabla\psi_w) + q_s + R_{ss}q_w \tag{2.14}$$

The solution solvent-water ratio R_{ss} is the volume of solvent at standard conditions dissolved in a unit volume of water at standard conditions within the reservoir. The volatile-oil model modifies the structure for the oil equation (2.6) so that it is similar to the gas equation (2.8). For this model the oil component can exist in both the oil and the gas phases. The volatile-oil equation that replaces the black-oil the oil equation (2.6) is

$$\frac{\partial}{\partial t}\left[\phi\left(\frac{S_o}{B_o}+R_v\frac{S_g}{B_g}\right)\right] = \nabla\cdot\left(\boldsymbol{T}_o\nabla\psi_o+R_v\boldsymbol{T}_g\nabla\psi_g\right) + q_o+R_vq_g \tag{2.15}$$

The oil content of gas or oil volatility R_v is the volume of condensate at standard conditions dissolved in a unit volume of gas at standard conditions within the reservoir.

Another common reservoir model is a component model such as that used in Coats, Thomas, and Pierson (1998). In component models there are $2N_c+4$ model equations for N_c components. The equations consist of N_c+1 component (and water) conservation equations, N_c phase equilibrium constraints, and the summation to 1 of liquid mole fractions, gas molar fractions, and saturation. Such a model can be reduced to the black-oil model simply by restricting the number of components and imposing the same model assumptions.

In general, the equations (2.6)-(2.13) cannot be solved analytically and therefore must be done numerically.

2.2 Discretization

In contrast to analytical solutions, numerical solutions give the values of pressure and fluid saturations only in discrete cells on a meshed geometry of the reservoir at discrete times. Several numerical techniques, referred to as discretizations, can be used to convert the system of PDEs into algebraic equations. The finite volume, finite element, and finite difference methods are used in reservoir simulation.

In the finite volume method, volume integrals in a PDE that contain a divergence term are converted to surface integrals, via the divergence theorem. These terms are then evaluated as fluxes at the surfaces of each finite volume. Because the flux entering a given volume is identical to that leaving the adjacent volume, these methods are conservative. An advantage of the finite volume method is that it conserves mass and is easily formulated to allow for unstructured meshes.

Similar to the finite volume method, the finite element method, also subdivides the whole problem domain into simpler parts, called finite elements. The finite element method uses various functions, called basis functions, to express variables in the governing equation for each element. These functions lead to the development of an error function that is minimized in order to generate solutions to the global governing equation. An advantage of a finite element method is its ability to handle complicated geometries (and boundaries) with relative ease. The quality of a finite element approximation is often higher than in the corresponding finite volume approach, for instance, because the higher quality of the approximation between grid points. However, there are examples to the contrary.

A finite difference method results from discretization by a regular rectangular mesh and is the simplest discretization due to the ease of implementation. Finite difference methods use finite difference equations to approximate derivatives that essentially utilize a Taylor series expansion and neglects terms that are considered to be small for a small difference in space parameters. The finite difference method can be considered a special case of the finite volume and finite element approaches, i.e. a first order approximation with a regular rectangular mesh. This is the most common approach in the oil industry today since reservoir simulation problems usually require discretization of the problem into a large number of grid points (millions and more), and

therefore cost of the solution favours simpler, lower order approximation within each cell. This is especially true if there are discontinuous derivatives being modelled. Chen, Huan, and Ma (2006) describe the use of the finite difference and finite volume methods for reservoir simulation. The resulting discrete equations at points or volumes are highly nonlinear and stiff.

In this work, equations (2.6)-(2.13) are discretized into finite difference form using a control volume approach with central differences in space and mobilities determined using a single-point upstream or upwind method. For an upstream method information for the flux across the interface of a grid cell is obtained using the directions from which the flow is expected to come (LeVeque, 2002, p. 72). In this case the flow is from higher potential ψ to lower potential. The upstream method is used for stability reasons. Without upstreaming, the numerical solution may display oscillations, overshoots or undershoots (e.g., saturations less than zero or greater than one), or converge to an incorrect solution. As an example, the transport term in the x-direction from equation (2.6) using this spatial discretization becomes:

$$\frac{\partial}{\partial x}\left(T_o \frac{\partial \psi_o}{\partial x}\right) \rightarrow \frac{2}{\Delta x_i}\left(T_{o_{i+\frac{1}{2}}} k_{i+\frac{1}{2}} \frac{\psi_{o_{i+1}} - \psi_{o_i}}{x_{i+1} - x_i} - T_{o_{i-\frac{1}{2}}} k_{i-\frac{1}{2}} \frac{\psi_{o_i} - \psi_{o_{i-1}}}{x_i - x_{i-1}}\right) \tag{2.16}$$

where

$$k_{i\pm\frac{1}{2}} = \frac{\Delta x_i + \Delta x_{i\pm1}}{\frac{\Delta x_i}{k_i} + \frac{\Delta x_{i\pm1}}{k_{i\pm1}}} \tag{2.17}$$

is the absolute permeability at the interface defined using a harmonic mean and

$$T_{o_{i\pm\frac{1}{2}}} = \left(\frac{k_{ro}}{B_o \mu_o}\right)_{i\pm1 \text{ or } i} \tag{2.18}$$

is the transmissibility of oil in the x-direction with subscript i or $i \pm 1$ depending on which grid cell is upstream.

Advancement in time is done using an adaptive implicit (AIM) strategy. In an AIM scheme each grid cell is determined to use either a fully implicit (FI) time discretization or an implicit pressure, explicit saturation (IMPES) time discretization. The variables in the flow term are discretized using either the first order implicit backward Euler method

$$y'(t) = f\big(t, y(t)\big) \rightarrow y^{n+1} = y^n + \Delta t \cdot f(t^{n+1}, y^{n+1}) \tag{2.19}$$

or the explicit forward method

$$y'(t) = f\big(t, y(t)\big) \rightarrow y^{n+1} = y^n + \Delta t \cdot f(t^n, y^n). \tag{2.20}$$

The accumulation term is always discretized using the implicit backward Euler method. For the IMPES scheme the oil pressure is updated implicitly everywhere, while the saturations (or saturation pressures) are updated explicitly in the flow terms. For the FI scheme all primary variables are updated implicitly. Thus, since there is no spatial derivative (and hence no central differencing) in the accumulation term, saturation variables corresponding to IMPES grid cells are only connected to other variables from the same grid cell at the update time. Details of the switching criteria between the IMPES and FI schemes for this particular AIM method can be found in Forsyth and Sammon (1986).

This system of differential equations is considered a stiff system, since the system includes terms that can lead to rapid variation in the solution. The gas phase is considerably more mobile than the other phases due to differences in their viscosities. Furthermore, stiffness arises due to differences in how pressure and saturations vary. Saturation changes are slow, whereas pressure changes spread more rapidly across the field. Explicit time discretization methods are generally quite inefficient on stiff problems, since the time step has to be very small for the method to stay in its stability region.

2.3 Primary Variables

The black-oil formulation has three equations for each grid cell. Therefore three primary variables are chosen for each grid cell. If there is free gas present (when the pressure is below the bubble point pressure, i.e. $p < p_b$) the primary variables are oil pressure, water saturation, and oil saturation (p_o, S_w, S_o). When no free gas is present ($S_g = 0$), the bubble point pressure replaces the oil saturation (p_o, S_w, p_b) since the saturation constraint reduces to $S_w + S_o = 1$. The changes in primary variables, δp_o for example, represent a change in the state of the system over a change in time δt is what is being solved for. The pressure variable will often be referred to separately from the other primary variables that will jointly be referred to as saturation variables (even bubble point pressure where it is a primary variable). The reason for this distinction is that pressure is associated with the parabolic character of the system and the saturation variables are associated with the hyperbolic character (Trangenstein & Bell, 1989). Further details of the model can be found in the literature (Aziz & Settari, 1979; Chen et al., 2006; Forsyth & Sammon, 1986).

2.4 Linear System

Pressure and saturation dependent properties including porosity, viscosity, density, formation volume factors, capillary pressures, and relative permeabilities make the system highly non-linear. Newton iteration is by far the most common method used to linearize the problem though there are other non-linear methods (Henson, 2002; Molenaar, 1995; Trottenberg, Oosterlee, & Schüller, 2001). Newton iteration requires the solution of associated Jacobian systems. For a general (non-linear) system of M equations of the form

$$\mathcal{L}\{F_m[\boldsymbol{y}(\boldsymbol{x})]\} = \boldsymbol{f}_m(\boldsymbol{x}), m = 1, 2, \dots, M \tag{2.21}$$

where $\mathcal{L}$ denotes linear differential operator, $F_m(\cdot)$ is a nonlinear function, $\boldsymbol{y}$ is a vector of unknowns, and $\boldsymbol{f}$ is a given vector that varies in space $\boldsymbol{x}$; Newton iteration generates the $(l+1)$th iterative solution vector, $\boldsymbol{y}^{l+1} = \boldsymbol{y}^l + \delta\boldsymbol{y}^l$, given a lth solution vector, $\boldsymbol{y}^l$. The correction vector $\delta\boldsymbol{y}^l$ is found by solving the system of equations

$$\mathcal{L}\{\nabla F_m(\boldsymbol{y}^l) \cdot \delta\boldsymbol{y}^l\} = f_m - \mathcal{L}\{F_m(\boldsymbol{y}^l)\} \tag{2.22}$$

known as a Jacobian system. The matrix defined $\boldsymbol{A}^l = \mathcal{L}\{\nabla F_m(\boldsymbol{y}^l)\}$ is called the Jacobian matrix. The linear Jacobian system for time step n and Newton iteration l (also called Newton cycle) is written simply as

$$\boldsymbol{A}^{n,l} \cdot (\delta\boldsymbol{y})^{n,l} = \boldsymbol{b}^{n,l} \tag{2.23}$$

however the superscripts are usually dropped during analysis for simplicity.

2.5 Jacobian Matrix

The structure and character of the Jacobian matrix are of crucial importance to the efficiency of the linear solver. Ordering the linear system by unknown type it can be described as a matrix problem given by:

$$\boldsymbol{A} \cdot \delta\boldsymbol{y} = \begin{pmatrix} \boldsymbol{A}_p & \boldsymbol{A}_{pS} \\ \boldsymbol{A}_{Sp} & \boldsymbol{A}_S \end{pmatrix} \cdot \begin{pmatrix} \delta\boldsymbol{y}_p \\ \delta\boldsymbol{y}_S \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_p \\ \boldsymbol{b}_S \end{pmatrix} \tag{2.24}$$

The sub-matrix $\boldsymbol{A}_p$ from is a referred to as the pressure matrix since it is aligned with the pressure unknowns, $\delta\boldsymbol{y}_p$, the matrix $\boldsymbol{A}_S$ is the saturation matrix applied to saturation or other non-pressure primary unknowns, $\delta\boldsymbol{y}_S$, and $\boldsymbol{A}_{Sp}$ and $\boldsymbol{A}_{pS}$ are the matrixes coupling pressure with the other primary unknowns. A second grid ordered notation of the matrix is useful when describing some matrix manipulations. With m the number of grid cells the Jacobian matrix is ordered into block matrices as:

$$\boldsymbol{A}\cdot\delta\boldsymbol{y}=\begin{pmatrix}[\boldsymbol{A}]_{(1,1)} & \cdots & [\boldsymbol{A}]_{(1,m)}\\ \vdots & \ddots & \vdots\\ [\boldsymbol{A}]_{(m,1)} & \cdots & [\boldsymbol{A}]_{(m,m)}\end{pmatrix}\cdot\begin{pmatrix}[\delta\boldsymbol{y}]_1\\ \vdots\\ [\delta\boldsymbol{y}]_m\end{pmatrix}=\begin{pmatrix}[\boldsymbol{b}]_1\\ \vdots\\ [\boldsymbol{b}]_m\end{pmatrix} \tag{2.25}$$

Block matrix $[\boldsymbol{A}]_{(i,j)}$ contains the coefficients relating the unknowns in cell j to those in cell i. Block-brackets $[\cdot]$ used herein will refer to this grid cell based ordering. Note that within each grid-cell ordered block-matrix the ordering is by unknown type as in (2.24). For the model formulation (2.6)-(2.8) each sub-matrix of the type $\left[\boldsymbol{A}_p\right]_{(i,j)}$ is a single numerical value and each sub-matrix of the type $[\boldsymbol{A}_S]_{(i,j)}$ is a (2×2) matrix.

Using the adaptive implicit method for time discretization means that some grid cells will be of the IMPES type with saturations that are determined explicitly in all but accumulation terms. This means that if grid cell k is IMPES then no other cells are coupled to the saturations of cell k and matrices of type $\left[\boldsymbol{A}_{pS}\right]_{(i,k)}$ and $[\boldsymbol{A}_S]_{(i,k)}$ where $i \neq k$ are zero matrices (in this section k is used as an index). For this formulation the pressures and saturations of grid cell k are fully coupled to one another so that $\left[\boldsymbol{A}_{pS}\right]_{(k,k)}$ and $[\boldsymbol{A}_S]_{(k,k)}$ are full.

2.6 Iterative Solution Strategies

There are two classes of methods to solve a linear system of equations; direct methods and iterative methods. In the absence of rounding errors, direct methods, such as Gaussian elimination, find an exact solution to a linear system of equations in a finite sequence of operations. For Gaussian elimination, these operations include multiplying, adding and switching rows. In contrast, iterative methods use an initial guess of the solution to iteratively generate better and better approximations to the solution.

For large systems of the order of millions of grid cells, direct methods for sparse linear systems are insufficient and outperformed by iterative solution techniques. An iterative linear solver is a procedure that, given an initial solution $\boldsymbol{y}^0$, finds the solution by successive approximations

$$\boldsymbol{y}^0 \to \boldsymbol{y}^1 \to \boldsymbol{y}^2 \to \cdots \to \boldsymbol{y}^k \to \cdots \tag{2.26}$$

where $\boldsymbol{y}^k \to \boldsymbol{y}$ as $k \to \infty$. The simplest iterative methods are known as stationary methods and include the Jacobi method, the Gauss-Seidel method, and the Successive Over-Relaxation method (SOR) (Saad, 2003). Trottenberg, Oosterlee, and Schüller (2001, p. 52) show these stationary methods are not efficient enough as stand-alone solvers for reservoir simulation.

The most popular linear solver methods for reservoir simulation are Krylov subspace methods. A Krylov subspace method is a projection method for which the solution, $\boldsymbol{y}^k$, is constructed of vectors from $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b})$ where

$$\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{b}) = \text{span}\{\boldsymbol{b}, \boldsymbol{A} \cdot \boldsymbol{b}, \boldsymbol{A}^2 \cdot \boldsymbol{b}, \dots, \boldsymbol{A}^{k-1} \cdot \boldsymbol{b}\} \tag{2.27}$$

is called a Krylov subspace. These methods accelerate the convergence of the solution requiring vastly fewer iterations at the cost of more work per iteration. The purpose of the projection is to extract an approximate solution from the subspace in way that minimizes some measure of the errors. Popular Krylov methods used in reservoir simulation include the Conjugate Gradient method applied to the normal equations (CGN), the Orthogonal Minimum Residual method (ORTHOMIN), the Generalized Minimal Residual method (GMRES), and the Biconjugate Gradient Stabilized method (BiCGSTAB) (Chen et al., 2006, p. 220). Studies have shown that GMRES performs better than ORTHOMIN, since ORTHOMIN in general requires more arithmetic operations and storage than GMRES (Chen et al., 2006, p. 224). GMRES is used in this study as the primary iterative technique. A detailed description of the GMRES and other Krylov methods can be found in Saad (2003).

2.7 Preconditioning

For linear systems that come from linearizing strongly heterogeneous and nonlinear systems, including those in reservoir simulation, applying iterative linear solvers without preconditioning results in slowly converging and possibly non-robust solvers (Chen et al., 2006). In practice, the choice of preconditioner has greater impact on performance than the particular Krylov method used (Saad, 2003). Preconditioning is a technique, which transforms the original linear system into an easier to solve linear system where the solution remains the same. There are three ways to precondition a linear system. Left preconditioning which multiplies the Jacobian matrix on the left as

$$\boldsymbol{M}_L^{-1} \cdot \boldsymbol{A} \cdot \boldsymbol{y} = \boldsymbol{M}_L^{-1} \cdot \boldsymbol{b}. \tag{2.28}$$

Right preconditioning multiplies the Jacobian matrix on the right as in

$$\boldsymbol{A} \cdot \boldsymbol{M}_R^{-1} \cdot (\boldsymbol{M}_R \cdot \boldsymbol{y}) = \boldsymbol{b}. \tag{2.29}$$

Finally, split-preconditioning is a combination of left and right

$$\boldsymbol{M}_L^{-1} \cdot \boldsymbol{A} \cdot \boldsymbol{M}_R^{-1} \cdot (\boldsymbol{M}_R \cdot \boldsymbol{y}) = \boldsymbol{M}_L^{-1} \cdot \boldsymbol{b}. \tag{2.30}$$

The matrices $\boldsymbol{M}$ are called preconditioning matrices. The goal of preconditioning is to produce a preconditioning matrix that can be efficiently inverted and represents a good approximation of $\boldsymbol{A}$. The right preconditioner is generally preferred because it does not change the right hand side of the linear system, and therefore keeps any aspect of the simulator dependent on the size (norm) of the right hand side (such as stopping criteria) consistent. Applying preconditioning to GMRES can be done with all three techniques above. A difference between left and right preconditioning is that right preconditioning allows for the Flexible Generalized Minimal Residual method (FGMRES) (Saad, 2003), where the preconditioner can be changed every iteration (Chen et al., 2006, p. 229). This is particularly useful for complex preconditioning methods that generate a

non-constant preconditioner. In this study the split-preconditioned GMRES and FGMRES algorithms are used. FGMRES requires one fewer right preconditioning steps at the additional cost of the extra memory required to save the "solution vectors" $\boldsymbol{z}_j$. The algorithm is presented in Figure 2.1.

Split-Preconditioned FGMRES Algorithm

1. Compute $\boldsymbol{r}_0 = M_L^{-1}(\boldsymbol{b} - A \cdot \boldsymbol{x}_0)$, $\beta = \|\boldsymbol{r}_0\|$, and $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$
2. For $j = 1, \ldots, m$ do:
3. $\quad \boldsymbol{z}_j = M_{R,j}^{-1} \cdot \boldsymbol{v}_j$
4. $\quad \boldsymbol{w}_j = M_L^{-1} A \cdot \boldsymbol{z}_j$
5. $\quad h_{i,j} = \langle \boldsymbol{w}_j, \boldsymbol{v}_i \rangle$ for $i = 1, \ldots, j$
6. $\quad \boldsymbol{v}_{j+1} = \boldsymbol{w}_j - \sum_{i=1}^{j} h_{i,j} \boldsymbol{v}_i$
7. $\quad h_{j+1,j} = \|\boldsymbol{v}_{j+1}\|$ and $\boldsymbol{v}_{j+1} = \boldsymbol{v}_{j+1}/h_{j+1,j}$
8. End do
9. Define $Z_m = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m]$, $\bar{H}_m = \{h_{i,j}\}$
10. Determine minimizer $\boldsymbol{y}_m$ of $\|\beta \boldsymbol{e}_1 - \bar{H}_m \cdot \boldsymbol{y}\|$
11. Solution $\boldsymbol{x}_m = \boldsymbol{x}_0 + Z_m \cdot \boldsymbol{y}_m$

Figure 2.1: Split-Preconditioned FGMRES algorithm

Some of the simplest methods used for preconditioning are the stationary iterative methods: Jacobi, the Gauss-Seidel and SOR. However, these can only be applied with limited success in reservoir simulation (Chen et al., 2006, p. 231). More powerful preconditioning techniques are usually used including those based on LU factorizations.

2.8 Incomplete LU factorization

Incomplete LU factorization (ILU) is a preconditioner that is formed by applying Gaussian elimination and dropping some elements in predetermined non-diagonal positions. ILU factorizations generate a sparse lower triangular matrix $\boldsymbol{L}$ and a sparse upper triangular matrix $\boldsymbol{U}$ such that the residual matrix

$$\boldsymbol{R} = \boldsymbol{L} \cdot \boldsymbol{U} - \boldsymbol{A} \tag{2.31}$$

satisfies certain conditions, such as having zero entries in some locations (Saad, 2003, p. 301). The simplest ILU factorization is ILU(0), meaning ILU with no fill-in. ILU(0) drops all entries that correspond to zero entries in the matrix $\boldsymbol{A}$. A more efficient and reliable ILU factorization can be obtained by allowing some fill-in. The method, ILU(k) where k is the level of fill-in, involves some challenges when implementing the method. Some of these challenges are addressed in Saad (2003, p. 311). Another method, ILUT(ε), avoids some of these challenges by dropping entries that exceed a threshold, ε. In this work an implementation of ILU(k) is used.

2.9 Algebraic Multigrid (AMG)

For reservoir simulation, simple iterative methods such as Jacobi and Gauss-Seidel as well as the more robust ILU methods reduce the high frequency error components of $\delta\boldsymbol{y}$ (relative to the Jacobian matrix $\boldsymbol{A}$) in many fewer iterations than equivalently sized low frequency error components (those that act over space). In contrast, multigrid methods improve solver efficiency by exploiting a hierarchy of grids or levels giving a fast reduction of both high and low frequency error components of the iterative solution. Multigrid methods consist of two phases. The first is the setup phase in which a hierarchy of grids or levels is constructed along with interpolation operators between the levels. The second phase is the solution phase where approximate solutions are found at each level and then interpolated to the next. At the coarsest level the solution is determined using either an iterative or direct method and then interpolated back to the finer levels. Usually the solvers used at intermediate levels are simple relaxation schemes and are referred to as smoothers. The process of recursive solution based on the hierarchy of levels is referred to as cycling. Several multigrid cycles are performed until the

solution at the finest level (the original matrix problem) is sufficient or a maximum number of cycles is reached.

Algebraic multigrid (AMG) is developed from the older Geometric multigrid where the hierarchy of *grids* is constructed based primarily on the discretization grid of the problem. In contrast, for AMG, the construction of the hierarchy of *levels* during a setup phase is entirely formed from the size and structure of the matrix elements and does not explicitly rely on any geometric information. AMG mimics geometric grids by interpreting the matrix as a graph and distinguishing between strong and weak couplings. Based on this graph, a maximally independent set of nodes is computed. The resulting set of nodes defines the coarse level operator using the Galerkin principle (Ruge & Stüben, 1987; Stüben, 2001b). The interpolation operators between different levels are defined by computing weights based on the matrix entries. There is no single fixed AMG strategy; at each stage of an AMG method various options are available. For example, different coarsening algorithms for the setup phase at different levels are possible as are the selection of different smoothers at different levels. In this way, the basic AMG strategy can be adapted taking into consideration the matrix properties common to a particular application. AMG was originally developed for matrices of certain types, especially for weakly diagonally dominant M-matrices as they arise in the discretization of elliptic partial differential equations. In practical applications, especially in industrial contexts, such ideal properties only rarely appear, although AMG has been shown to work quite well for matrices that are outside of the desired type. AMG in practice is known to work for asymmetric matrices as well as for systems with large heterogeneities and anisotropies. For reservoir simulation, AMG strategies can be efficient at solving pressure variables that exhibit parabolic (or nearly

elliptic) character. Detailed information on the AMG technique can be found in the literature (Falgout, 2006; Ruge & Stüben, 1987; Stüben, 2001b).

2.10 Combinative or Multistage Solution Methods

The reservoir system is known to have mixed hyperbolic and parabolic (or nearly elliptic) character (Trangenstein & Bell, 1989). The pressure field is essentially parabolic or nearly elliptic with long-range coupling, while the remaining equations (referred to as saturation equations) are hyperbolic or transport-dominated parabolic (Aziz & Settari, 1979; Peaceman, 1977; Trangenstein & Bell, 1989). Traditional single stage-solvers, such as the above-described ILU, are often not efficient at converging elliptic or nearly elliptic components. This gives rise to a combinative multi-stage method (Behie & Vinsome, 1982) where estimates of the solution of different variable types are found separately as well as an estimate of all unknowns of the whole system. These results are then combined prior to the typical Krylov acceleration process. If the system can be decoupled effectively then more efficient solvers can be applied to each constituent part. A variant of the Combinative method, known as the Constrained Pressure Residual (CPR) method, is a two-stage method (Wallis et al., 1985; Wallis, 1983) that is widely used in reservoir simulation (Cao et al., 2005; Lacroix et al., 2001; Scheichl et al., 2003; Stüben et al., 2007). In CPR, prior to extraction of a pressure system, usually saturations are approximately decoupled. Decoupling operators $\boldsymbol{D}_L$ and $\boldsymbol{D}_R$ are left and right preconditioners that transform the Jacobian matrix as:

$$\boldsymbol{D}_L^{-1} \cdot \boldsymbol{A} \cdot \boldsymbol{D}_R^{-1} = \begin{pmatrix} \widetilde{\boldsymbol{A}}_p & \widetilde{\boldsymbol{A}}_{pS} \\ \widetilde{\boldsymbol{A}}_{Sp} & \widetilde{\boldsymbol{A}}_S \end{pmatrix} = \widetilde{\boldsymbol{A}}. \tag{2.32}$$

The CPR method is then applied as follows:

- Restrict the full system to a pressure only system:

$$\widetilde{\boldsymbol{A}}_p \cdot \left(\delta \boldsymbol{y}_p\right)_1 = \tilde{\boldsymbol{r}}_p. \tag{2.33}$$

The remaining couplings of saturation to pressure ($\widetilde{\boldsymbol{A}}_{pS}$) are neglected.

- Approximately solve the pressure system with a suitable linear solver, here AMG is chosen, and expand the pressure solution to the full system:

$$\begin{pmatrix} \delta \boldsymbol{y}_p \\ \boldsymbol{0} \end{pmatrix}_1 = \boldsymbol{M}_1^{-1} \cdot \begin{pmatrix} \tilde{\boldsymbol{r}}_p \\ \boldsymbol{0} \end{pmatrix}. \tag{2.34}$$

  This forms the first stage of the preconditioner.

- Correct or update the full system residual using the first stage solution:

$$\hat{\boldsymbol{r}} = \tilde{\boldsymbol{r}} - \widetilde{\boldsymbol{A}} \begin{pmatrix} \delta \boldsymbol{y}_p \\ \boldsymbol{0} \end{pmatrix}. \tag{2.35}$$

  This is the constraint part of CPR.

- Solve the full system with a second stage preconditioner applied to the corrected full system residual:

$$(\delta \boldsymbol{y})_2 = \boldsymbol{M}_2^{-1} \cdot \hat{\boldsymbol{r}}. \tag{2.36}$$

  This is the second stage of the preconditioner.

- Finally, combine the first and second stage solutions:

$$\delta \boldsymbol{y} = \begin{pmatrix} \delta \boldsymbol{y}_p \\ \boldsymbol{0} \end{pmatrix}_1 + \left(\delta \boldsymbol{y}\right)_2. \tag{2.37}$$

The action of the complete two-stage preconditioner can be expressed as a right preconditioner

$$\boldsymbol{M}_R^{-1} = \boldsymbol{M}_2^{-1} \cdot (\boldsymbol{I} - \boldsymbol{A} \cdot \boldsymbol{M}_1^{-1}) + \boldsymbol{M}_1^{-1} \tag{2.38}$$

Iterative application of the CPR preconditioner is often accelerated within a Krylov method and will be done in this study. Right preconditioned FGMRES is a good choice because it has the flexibility for the preconditioners $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ to vary with Krylov iteration.

Because the pressure aligned sub-matrix, $\widetilde{\boldsymbol{A}}_p$, of the Jacobian matrix is expected to have properties similar to those of a matrix derived from the discretization of an elliptic PDE, multigrid methods are a common choice for the first-stage or inner preconditioner $\boldsymbol{M}_1$ (Cleary et al., 2000; Stüben, 2001a). Fraunhofer SCAI's SAMG software library will be used in this study. To reduce the error in saturation variables, simple iterative methods are usually sufficient. In this study variable degree incomplete LU factorization, ILU(k), will be used to construct the second-stage or outer preconditioner $\boldsymbol{M}_2$ (Collins, Grabenstetter, & Sammon, 2003).

2.11 Decoupling Preconditioners

The first step of CPR is to approximately decouple the pressure and saturation variables. In fact, prior to a decoupling step the equation aligned with pressure may be insignificant in some or many grid cells particularly when the equation for the phase with the dominant saturation is not aligned with pressure. Decoupling on the left with $\boldsymbol{D}_L$ combines the residual equations for a grid cell and can concentrate the elliptic properties of the resulting equation that is aligned with pressure. The primary purpose of $\boldsymbol{D}_L$ is minimizing $\left\|\widetilde{\boldsymbol{A}}_{pS}\right\|$ and consequently decoupling the computation of pressure components from the effect of any variation in the current saturation approximation. Usually the construction of the decoupling operator $\boldsymbol{D}_L$ is based on the assumption that the pressure-saturation couplings within a grid-cell, $\left[\boldsymbol{A}_{pS}\right]_{ii}$, are expected to be much stronger than pressure-saturation couplings between different grid-cells. Therefore, the diagonal grid-blocks of $\boldsymbol{A}$ are expected to dominate those in off-diagonal blocks (Lacroix et al., 2001; Scheichl et al., 2003). Thus, it is usually sufficient to decouple the diagonal blocks $[\boldsymbol{A}]_{ii}$ only. The resulting operator applied to the grid ordered matrix has a block-diagonal structure:

$$\boldsymbol{D}_L = \begin{pmatrix} [\boldsymbol{D}_L]_{1,1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & [\boldsymbol{D}_L]_{m,m} \end{pmatrix}. \tag{2.39}$$

Applying matrix operations on the small block matrices (in the black-oil formulation 3×3 matrices) is expected to be inexpensive. This gives a great flexibility in how to construct $\boldsymbol{D}_L$ and various approaches are reported in the literature.

One such technique is the alternate block factorization (Bank et al., 1989) used, for example, by (Cao et al., 2005). In ABF each row of block matrices of the Jacobian matrix is multiplied on the left by the inverse of the block diagonal matrix, $[\boldsymbol{D}_L]_{ii}^{-1} = [\boldsymbol{A}]_{ii}^{-1}$. This allows the full decoupling of primary variables locally, within a grid cell. However, this preconditioner also normalizes the resulting matrix that makes it unsuitable for AMG. This type of decoupling is crucial for IMPES grid cells where the saturation variables are only coupled locally through the accumulation term. Decoupling IMPES saturation variables locally decouples them entirely from the rest of the Jacobian and allows them to be updated directly at the end of the linear solution. ABF can also be used in conjunction with a Schur complement method to reduce the matrix size using a red-black ordering scheme. Grid cells on the boundaries of each class are labeled black and all grid cells that are only connected to black-grid cells are labeled red. The block rows associated with red cells can then be removed from the iterative matrix solution using a Schur complement. The reordering reduces the matrix size by increasing the matrix complexity.

Another popular decoupling method is the quasi-IMPES (Scheichl et al., 2003) where pressures and saturations are approximately decoupled without the normalization of the matrix. The diagonal block matrices of the preconditioner are constructed using a Schur complement to eliminate saturation variable coefficients as in

$$[\boldsymbol{D}_L]_{ii}^{-1} = \begin{pmatrix} [\boldsymbol{I}_p] & -[\boldsymbol{A}_{pS}]_{ii} \cdot [\boldsymbol{A}_S]_{ii}^{-1} \\ \boldsymbol{0} & [\boldsymbol{I}_S] \end{pmatrix}. \tag{2.40}$$

Similar to the quasi-IMPES method the true-IMPES (Scheichl et al., 2003) decoupling method constructs the preconditioner in the same way. Unlike quasi-IMPES however, the preconditioner is constructed using only saturation coefficients from the accumulation terms of the model equations. Saturation coefficients from the transport terms are neglected entirely. This method requires the construction of a decoupled pressure equation in addition to the generation of the Jacobian matrix since the accumulation and transport coefficients are already blended when the linear solver is applied.

The development of an effective decoupling preconditioner for the particular model equation considered will be a primary goal of this research.

Chapter Three: **Combinative Solver Development & Variations**

To develop a two-stage combinative solver an AMG solver has been coupled to the existing solver in CMG's black-oil simulator IMEX. A commercial software package, SAMG from Fraunhofer SCAI, was used for the AMG solver. The AMG solver is incorporated in the simulator as the first inner stage of a CPR method. Coupled this way the original ILU based solver, acts as the second stage of a CPR method and the outer Krylov iteration controller. This allows a direct comparison between the newly developed combinative solver and the original solver of IMEX (ILU only) using exactly the same numerical convergence criteria at the linear solver level.

3.1 Combinative Solver Variations

The combinative solver is developed to include optional parameters to vary the method. There are parameters to vary the AMG solver method, the outer ILU preconditioned method, and the combinative method itself. Parameters varied for this study are described in this chapter.

To make the combinative solver more effective a new preconditioning technique has been developed. The preconditioner and its optional variations are also described in this chapter.

3.1.1 AMG Options

The SAMG software package contains many optional parameters. The code has been written to be able to pass those parameters through the IMEX data input structure directly to SAMG. This allows the testing of many AMG options. A range of parameter studies searching for optimal values when using the combinative solver has been performed. The studies have been done while using the best preconditioning techniques.

3.1.1.1 Number of AMG Cycles

A parameter to absolutely set the number of AMG cycles (iterations) preformed for each combinative solver iteration has been tested for a range of values. This is in contrast to setting a relative residual tolerance parameter as the stopping criteria for the first stage (SAMG) of a combinative iteration. A relative residual tolerance stops the AMG solution when the magnitude of the residual vector of the current approximation $\|\boldsymbol{r}_{\mathrm{AMG}}^{(n)}\|$, is reduced by a factor greater than specified ϵ,

$$\left\|\boldsymbol{r}_{\mathrm{AMG}}^{(n)}\right\| < \epsilon * \|\boldsymbol{r}_{\mathrm{AMG}}^{(0)}\|. \tag{3.1}$$

In effect, this allows the number of AMG cycles per combinative iteration to vary. The GMRES Krylov method requires a non-varying AMG method making the FGMRES method necessary for varying AMG cycles. A range of residual tolerances and set number of AMG cycles has been tested. Results are in §5.3.1.1.

3.1.1.2 AMG Cycle Type

There are essentially three types of cycle, referred to as V-, F- and W-cycle, which differ only in the extent to which they approximate the corresponding two-level cycle. Each cycle type recursively applies the AMG solution method through to the coarsest level and back to the finest and they are increasingly more robust but also increasingly more expensive. The cycle types recursively use the following:

- V-cycle
    - Smooth ➔ Restriction ➔ V-cycle ➔ Prolongation ➔ Smooth
- F-cycle
    - Smooth ➔ Restriction ➔ F-cycle ➔ V-cycle ➔ Prolongation ➔ Smooth

- W-cycle
    - Smooth ➔ Restriction ➔ W-cycle ➔ W-cycle ➔ Prolongation ➔ Smooth

At the coarsest level each cycle type calculates a solution in place of the recursion. In practice, the V-cycle is generally the best and most efficient choice (Stüben, 2015). The cycle types are depicted in Figure 3.1.

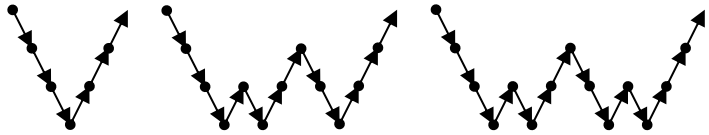


Figure 3.1: V-, F-, and W-cycle depictions

3.1.1.3 AMG Cycle Acceleration

The acceleration of AMG cycles when more than one is performed is examined. AMG can be used as a standalone method or as preconditioner for a Kryolv method. The methods tested are AMG preconditioned BiCGStab, AMG preconditioned *flexible* GMRES, and standalone AMG iteration. For very small numbers of iterations the overhead in using the Krylov methods is larger than the benefit of using them.

3.1.1.4 AMG Smoother Type

The AMG process typically uses very simple smoothers. The smoothers tested for this study are Jacobi relaxation, Gauss-Seidel relaxation, and ILU(0) smoothing. The Gauss-Seidel relaxation is performed in C/F order. That is, on each level, first all coarse variables are relaxed followed by all fine variables.

3.1.1.5 AMG Coarsening Strategy

SAMG includes both an aggressive coarsening strategy and a standard coarsening strategy. The difference is the aggressive strategy leaves more variables on the fine level reducing the number

of variables on next coarsest level. The effect is that fewer levels are needed to reach the coarsest level. The main purpose of the more aggressive coarsening strategies is to substantially reduce the memory requirements of SAMG. However, generally, this will be at the expense of a slower convergence. Nevertheless, aggressive coarsening often also reduces total computational time due to the reduced number of levels and therefore smoothing and interpolating steps per cycle. The aggressive strategy is most effective when applied to the first (or first few) levels only. After that, the gains in speed are reduced and do not compensate for the decline in performance.

3.1.2 GMRES Accelerated ILU Solver Options

The ILU preconditioned outer CPR stage controls the convergence criteria of the combinative solver. A relative residual tolerance, or precision, stops the combinative solution when the magnitude of the residual vector of the current iteration $\|\boldsymbol{r}^{(n)}\|$, is reduced by a factor greater than specified ε,

$$\left\|\boldsymbol{r}^{(n)}\right\| < \varepsilon * \|\boldsymbol{r}^{(0)}\|. \tag{3.2}$$

The combinative method is also stopped if a maximum number of iterations is reached prior to convergence. These criteria are identically used for the single-level ILU only preconditioned comparison simulations. The precision is kept consistent for compared runs.

The particular ILU method of this study is variable degree ILU(k) with fill-in parameter k. Domain decomposition for parallel runs is accomplished using vertex separators (Collins et al., 2003) rather than edge separators with a block-Jacobi method applied between domains as is used for most traditional parallel ILU. The treatment of the level of fill-in for ILU between domains is of interest for this study. For parallel runs the reservoir is partitioned into disjoint sets of grid cells referred to as classes. These classes are then organized into levels where there must

be no flow between grid cells of different classes at the same level. For example, grid cells of two different level-one classes cannot connect directly; there must be an additional grid cell between them. Grid cells in level-two classes are in effect planes that separate level-one classes. Higher-level classes are necessary for intersecting level-two planes or cells with additional connections. Although the effect of parallelism is not considered in this study this level and class structure does play a role. This is because the degree of ILU fill-in can differ for intra-class terms and inter-class terms. By default the level of fill-in is greater for inter-class terms. In this study the degree of ILU factorization for inter-class terms is varied.

3.1.3 Combinative Solver Options

The implementation of the combinative solver allows some flexibility in how the solutions of the ILU and AMG methods are both calculated and combined. Several options for the combinative solver have been implemented.

3.1.3.1 Setup Frequency

The setup phase of AMG is quite expensive numerically, requiring roughly the same amount of computation time as the solution phase of AMG. Consequently, an option to perform a limited setup phase, only recalculating transfer operators, is available in SAMG. The combinative solver has been developed with the option to limit the number of full AMG setup operations, instead using only a partial setup. The full AMG setup can be limited to once per Newton cycle (and therefore every new matrix), time step, well opening (or shut-in), or only for the initial setup. Reducing the number of full setups leads to poorer convergence by the combinative solver but can reduce the running time if the convergence remains good enough. The frequency of partial setups can also be limited to these frequencies. Partial setups can only be performed on a more frequent basis than full setups. For example, full setups may be done each new time step and

partial setups done every other Newton cycle. If neither a full setup nor a partial setup is done for a matrix the solution phase is calculated using the previous setup information. In this case only matrix values are updated and the transfer operators remain the same. Tests have been run for several data sets showing that doing full setups for each time step and partial setups for subsequent Newton cycles generally gives the best performance.

3.1.3.2 Well Inclusion

The combinative solver has been implemented with the option to either include or exclude the well equations (2.12) from the AMG stage of the solver. Tests indicate including the wells give the best performance. When using the weighted versions of the preconditioners described below an option to weight the well equations has been created. This allows the weighting of well terms, based on the total flow reservoir fluid. The results of this option are inconclusive.

3.1.3.3 Stage Weights

Another option modifies the coupling of first and second stage solutions of the combinative solver by weighting vectors. This option can weigh the sum of the two-stage solution and can also weight the updated vector for the right hand side of the second stage linear system. The normal updated vector for a 2-stage combinative method is

$$\boldsymbol{v}' = \boldsymbol{v} - \boldsymbol{A} \cdot \boldsymbol{z}_p \tag{3.3}$$

and the sum of the stage solutions is not weighted.

The first possibility for the option weights both update vector and the solution vector from the first stage. Effectively, it modifies the solution given by AMG. The solution is multiplied by the scalar value $\left[\frac{\boldsymbol{v}^T(\boldsymbol{A}\cdot\boldsymbol{z}_p)}{\|\boldsymbol{A}\cdot\boldsymbol{z}_p\|^2}\right]$ in order to minimize the square of the first stage residual

$$\|\boldsymbol{r}_{\mathrm{AMG}}\|^2 = \left\|\boldsymbol{v} - \boldsymbol{A} \cdot \boldsymbol{z}_p\right\|^2. \tag{3.4}$$

The minimization of the residual will reduce the effect of the first stage if it is a poor solution for the full system. This is especially true if the saturations depend strongly on pressure and the pressure depends strongly on the saturations.

Two choices for a weighted update vector are possible. The first weights the pressure solution as above but the solution from AMG is not modified only the sum is weighted. The update vector becomes

$$\boldsymbol{v}' = \boldsymbol{v} - \left[\frac{\boldsymbol{v}^T\left(\boldsymbol{A}\cdot\boldsymbol{z}_p\right)}{\left\|\boldsymbol{A}\cdot\boldsymbol{z}_p\right\|^2}\right]\boldsymbol{A}\cdot\boldsymbol{z}_p \tag{3.5}$$

which makes the update vector orthogonal to the product, i.e. $\boldsymbol{v}' \perp \boldsymbol{A}\cdot\boldsymbol{z}_p$, and minimizes the norm of the update vector. The second update weighting is

$$\boldsymbol{v}' = \left[\frac{\boldsymbol{v}^T\left(\boldsymbol{A}\cdot\boldsymbol{z}_p\right)}{\|\boldsymbol{v}\|^2}\right]\boldsymbol{v} - \boldsymbol{A}\cdot\boldsymbol{z}_p \tag{3.6}$$

which makes the update vector orthogonal to the original right hand vector of the iteration, $\boldsymbol{v}' \perp \boldsymbol{v}$.

An option to weight the sum of the two stages is possible,

$$\boldsymbol{z} = \alpha_I\cdot\boldsymbol{z}_I + \alpha_p\cdot\boldsymbol{z}_p. \tag{3.7}$$

The weights are chosen in order to minimize the square of the total residual

$$\|\boldsymbol{r}\|^2 = \|\boldsymbol{v} - \boldsymbol{A}\cdot\boldsymbol{z}\|^2. \tag{3.8}$$

The weights are

$$\alpha_I = \left[\frac{[\boldsymbol{v}^T(\boldsymbol{A}\boldsymbol{z}_I)]\left\|\boldsymbol{A}\boldsymbol{z}_p\right\|^2 - \left[\boldsymbol{v}^T\left(\boldsymbol{A}\boldsymbol{z}_p\right)\right]\left[(\boldsymbol{A}\boldsymbol{z}_I)^T\left(\boldsymbol{A}\boldsymbol{z}_p\right)\right]}{\|\boldsymbol{A}\boldsymbol{z}_I\|^2\left\|\boldsymbol{A}\boldsymbol{z}_p\right\|^2 - \left[(\boldsymbol{A}\boldsymbol{z}_I)^T\left(\boldsymbol{A}\boldsymbol{z}_p\right)\right]^2}\right] \tag{3.9}$$

and

$$\alpha_p = \left[\frac{\left[\boldsymbol{v}^T\left(\boldsymbol{Az}_p\right)\right]\|\boldsymbol{Az}_I\|^2 - \left[\boldsymbol{v}^T(\boldsymbol{Az}_I)\right]\left[(\boldsymbol{Az}_I)^T\left(\boldsymbol{Az}_p\right)\right]}{\|\boldsymbol{Az}_I\|^2\left\|\boldsymbol{Az}_p\right\|^2 - \left[(\boldsymbol{Az}_I)^T\left(\boldsymbol{Az}_p\right)\right]^2}\right] \tag{3.10}$$

The above combinative solver options have been implemented and tested.

3.1.3.4 Combinative Stopping Criteria

The 2-stage combinative method can be stopped prematurely, continuing using only the ILU stage of preconditioning for the remainder of the Newton cycle.

The first stopping criterion occurs when the magnitude of the first stage pressure residual is less than a small threshold. At that point the pressure solution is assumed to have converged.

The second stopping criterion is simply to set a maximum number of combinative iterations per Newton cycle.

A second stopping criterion uses the magnitude of the residual of the first stage pressure only residual. If the current pressure residual

$$\left\|\boldsymbol{v}_{j_p} - \boldsymbol{A}_p \cdot \boldsymbol{z}_{j_p}\right\| \tag{3.11}$$

is too much larger than the previous pressure residual then combinative iteration is stopped. A relative residual tolerance is an input variable. For example the combinative iteration may be stopped if the pressure residual is 1000 times larger than it was previously. A growth is the magnitude of the pressure residual may indicate a diverging solution.

A third stopping criterion uses the magnitude of the contribution the pressure solution vector from the current combinative iteration. The magnitude of contribution is calculated using the first stage pressure solution in place of the full combinative solution. The orthogonal projection of the pressure component to the current Krylov space is

$$\boldsymbol{v}_p = \boldsymbol{A} \cdot \boldsymbol{z}_p - \sum_{i=1}^{j} \left[\frac{\boldsymbol{v}_i^T \left(\boldsymbol{A} \cdot \boldsymbol{z}_p \right)}{\boldsymbol{v}_i^T \boldsymbol{v}_i} \right] \boldsymbol{v}_i \tag{3.12}$$

The minimization coefficient y_p is calculated using the GMRES algorithm (see Figure 2.1) with $\boldsymbol{v}_p$ used in place of most recent Krylov vector.

If the magnitude of $y_p \left\| \boldsymbol{A} \cdot \boldsymbol{z}_p \right\|$ is small relative to the initial residual, combinative iterations are stopped. A small relative magnitude implies the contribution of the pressure has such a small significance it is assumed that it is not worth the computational effort to calculate additional pressure solutions.

Another stopping criterion is to stop when SAMG reports a non-positive diagonal entry was found on a coarse-level Galerkin matrix during the setup phase. A non-positive diagonal entry is a sign of a potentially bad matrix for AMG. Positive definite matrices cannot have negative diagonal entries in the Galerkin operators. Hence, the original matrix was not positive definite. AMG is known to work for some non-positive definite matrices however there is no theoretical backing (Stüben, 2015).

3.1.3.5 Threshold Checking

One additional combinative solver option has been developed. That is an off-diagonal threshold check. The purpose is to use a simple metric (row sums) to compare the size of the diagonal pressure matrix $\widetilde{\boldsymbol{A}}_p$ and the off-diagonal matrices corresponding to the other variables $\widetilde{\boldsymbol{A}}_{pS}$. The goal is to determine if threshold values can be found for each of the off-diagonal variable types that predict the likelihood of poor (or slow) combinative solver performance for a given matrix. If such values can be determined then the simulation would adaptively be able to choose to use

either the combinative solver or the ILU only method. It is unclear at this time values can be found that are valid for a range of problem data sets.

3.2 Preconditioner Development

The initial version of the combinative solver uses the Alternate Block Factorization decoupling method as a preconditioner (Bank et al., 1989). Applying ABF by itself can lead to a less favourable matrix for an efficient AMG treatment. Algebraic Multigrid is not a solver that can handle arbitrary matrices; rather it seeks to exploit properties of certain classes of matrices such as those generated for pure diffusion problems. Since the modelled equations are of a coupled hyperbolic-parabolic nature influences from compressibility, wells, and phase changes may lead to a poor decoupled pressure matrix for AMG. In fact, the pressure matrix may be highly indefinite with no guarantee of convergence with AMG.

As a first improvement to the ABF preconditioner, an additional preconditioning step has been implemented. Each row of the extracted pressure matrix is multiplied by the scalar value of the weighted sum of the original matrix entries associated with the pressure component of the corresponding grid cell. The resulting row-scaling method can use each components mass, surface volume, or reservoir volume for the weighted sum. Scaling rows (or columns) after applying ABF is essential for AMG in many cases because the setup phase of AMG relies on the relative size of matrix entries when constructing the hierarchy of levels. ABF on its own scales the resulting matrix to unit diagonal entries.

Another preconditioner, referred to as the dynamic row-sum (DRS) preconditioner, has been developed and published jointly with Fraunhofer's SCAI institute (Gries, Stüben, Brown, Chen, & Collins, 2014). For DRS preconditioning pressure derivatives from each equation in a block-row of the Jacobian matrix are summed prior to the first solver stage of CPR. Consequently, a

pressure derivative approximating one from a total fluid pressure is generated. The purpose is to include the pressure related behaviour from all three phases within each grid cell. The DRS preconditioner modifies this to dynamically select which equation to include in the sum. If the matrix entries from any of the phase equations of a particular grid cell appear to be problematic for solution by AMG they are skipped. In particular, if an equation produces matrix entries that are far from diagonally-dominant they are not likely to be solved well using the AMG method and are excluded from the sum.

Finally, a modification to the DRS preconditioner using weighted sums has been developed independently of Fraunhofer (Brown et al., 2015). Using the surface volume based equations (2.6)-(2.8) of IMEX an un-weighted summation of terms will not eliminate saturation from the governing equations due to the presence of formation volume factors B_ℓ. Reservoir volume dimensioned equations are in the canonical form and their summation approximates the behaviour of an averaged single-phase fluid at reservoir conditions as in Scheichl, Masson, and Wendebourg (2003). To approximate a single-phase fluid for surface volume based equations, physical weights of each equation are necessary. The resulting weighted summation will approximately eliminate saturations from the accumulation term.

3.2.1 DRS Preconditioner Description

The following is a description of the dynamic row-sum (DRS) preconditioner. When using the DRS preconditioner the block sub-matrices of the pressure matrix to be solved with AMG are given as

$$\left\{\widetilde{\boldsymbol{A}_p}\right\}_{i,j} = \boldsymbol{\xi}_i^T \cdot \left\{\boldsymbol{A}_{\ell,p}\right\}_{i,j} \tag{3.13}$$

where

$$\boldsymbol{\xi}_i^T = \left(\delta_o \alpha_o, \delta_w \alpha_w, \delta_g \alpha_g\right)_i \tag{3.14}$$

is the dynamically chosen summation vector for each grid cell. For each phase ℓ of each block i the inclusion term $\delta_{\ell,i}$ is computed as:

$$\delta_{\ell,i} = \begin{cases} 0 & \text{if } \dfrac{\left|\left\{\boldsymbol{A}_{\ell,p}\right\}_{i,i}\right|}{\sum_{j\neq i}^{n_p}\left|\left\{\boldsymbol{A}_{\ell,p}\right\}_{i,j}\right|} < \varepsilon_{dd} \\ 1 & \text{otherwise} \end{cases} \tag{3.15}$$

for a diagonal dominance threshold value ε_{dd}. This threshold check determines which rows are included in the sum.

3.2.2 Formulation Dependent Row Weighting

The weighting parameters α_ℓ are determined at the time of Jacobian building from the reservoir conditions at the end of the previous Newton cycle. Three weighting options have been implemented, surface volume weighting, reservoir volume weighting, and mass weighting. The model formulations used in IMEX are surface volume weighted so the weighting parameters are simply set to 1.

For reservoir weighting the weights required include formation volume factors B_ℓ, and the solution gas/oil ratio R_s. A reservoir volume weighted DRS vector becomes

$$\boldsymbol{\xi} = \left(\delta_o\left[B_o - R_s B_g\right], \delta_w B_w, \delta_g B_g\right)^T \tag{3.16}$$

This weighting will include pressure behaviour from each phase as well as approximately decoupling pressure from saturation. The resulting summed row approximates the discretization of a single-phase reservoir fluid balance.

Row weights have been determined and implemented for each of the IMEX fluid models in addition to black-oil. This allows the combinative solver option to be used for all IMEX fluid model simulations. For a fluid model including water-soluble solvent the weight vector is

$$\boldsymbol{\xi_s} = \left(\delta_o\left[B_o - R_s B_g\right], \delta_w[B_w - R_{ss}B_s], \delta_g B_g, \delta_s B_s\right)^T \tag{3.17}$$

For a fluid model allowing volatile oil the weight vector is

$$\boldsymbol{\xi_v} = \left(\delta_o\left[\frac{B_o - R_s B_g}{1 - R_s R_v}\right], \delta_w B_w, \delta_g\left[\frac{B_g - R_v B_o}{1 - R_s R_v}\right]\right)^T \tag{3.18}$$

The final weighting option uses density weights resulting in a mass weighted summation. However, this will not approximately decouple pressure from saturation.

3.2.3 DRS for IMPES Grid Cells

The weighted DRS method is not necessarily applied in the same manner to both FI grid cells and IMPES grid cells. The rows aligned with saturation variables are decoupled with ABF in both the FI and IMPES cases to ensure numerical stability of the ILU preconditioner in the outer stage of the combinative solver. For FI cells the weighted DRS method is applied directly with no ABF component for pressure aligned rows. Assuming that pressure is aligned with the first row of each block-matrix, the resulting weighted DRS preconditioner is a diagonal block matrix G, with block diagonals given by:

$$G_{ii}^{\mathrm{FI}} = [\boldsymbol{e}_1 \boldsymbol{\xi}_i^T + (I - \boldsymbol{e}_1 \boldsymbol{e}_1^T)\{A\}_{ii}^{-1}] \tag{3.19}$$

For IMPES cells the desired goal is to completely decouple local saturations from pressure to remove them from the matrix calculation. To accomplish this decoupling is applied to IMPES grid cells. The DRS preconditioner can modify ABF to be applied to IMPES grid cells in two ways. The first is to apply DRS to inner pressure solve of the combinative method and ABF to the outer full solve. This is possible since the saturation variables for IMPES grid cell can be

explicitly found. One detail of this option is that the DRS vector needs to be applied every solver iteration. In particular, $\boldsymbol{g}_i^T = \boldsymbol{\xi}_i^T\{A\}_{ii}$ must be multiplied to the right hand side vector first before solving for pressure with AMG. The second method uses the ABF preconditioner for IMPES grid cells but also multiplies the resulting row aligned with pressure by a scale factor. To maintain the same scaling throughout the matrix IMPES diagonal blocks are first multiplied on the left by the weighted DRS vector. The resulting pressure $\{\widetilde{A_p}\}_{i,i}$ value is then used as a scalar multiplier of the pressure aligned row after ABF is applied. Again, assuming that pressure is aligned with the first row of each block-matrix, the resulting weighted DRS preconditioner in this case is a diagonal block matrix $\boldsymbol{G}$, with block diagonals given by:

$$\boldsymbol{G}_{ii}^{\text{IMPES}} = [\boldsymbol{e}_1\boldsymbol{\xi}_i^T\{\boldsymbol{A}\}_{ii}\boldsymbol{e}_1\boldsymbol{e}_1^T + (\boldsymbol{I} - \boldsymbol{e}_1\boldsymbol{e}_1^T)]\{\boldsymbol{A}\}_{ii}^{-1} \tag{3.20}$$

Both IMPES preconditioning methods give the same result for diagonal block of the pressure matrix $\{\widetilde{\boldsymbol{A}_p}\}_{i,i}$. They differ in the result for off diagonal blocks $\{\widetilde{\boldsymbol{A}_p}\}_{i,j},\ j \neq i$. The second method is the default method for this study unless otherwise stated.

3.2.4 Well Inclusion and Weight

The well equations (2.12) can be included in the matrix solution. They contain only one variable, bottom-hole pressure. There is an option to weight the well terms in which they are multiplied by a scalar term that equals the sum of the total reservoir volume of fluid from the source terms in the model equations. Therefore the diagonal element of a well row will be the total flow of fluid volume for that time step rather than a normalized row. The weighting of the wells seems to have little effect.

Chapter Four: **Numerical Analysis**

The complex nature of the solver methods investigated in this study precludes the use of strictly analytic methods for analysis. Instead, experimental analysis is used. Verification and analysis of the solver methods are performed using full field reservoir simulations. Both synthetic and real field industrial data are used in the test cases. Since solver methods are built into the commercially available IMEX software, results using real field data correspond to those performed commercially.

4.1 Simulation Environment

The 2016.10 general release of CMG's adaptive-implicit black-oil simulator, IMEX, was used as the base reservoir simulator on which the solver methods are modified. The 28a1p2 version of SAMG (Stüben, 2015) was incorporated as the algebraic multigrid solver in the simulator as described in Chapter Three:.

All simulations were run on a Dell Power Edge M820 Blade Server with two 2.7 GHz Intel Xeon E5-2697 v2 12 core processors and 128GB of 1866MHz DDR3 RAM. Simulations were run using between 1 and all 24 CPUs under Red Hat Enterprise Linux 6 update 5. The number of cores used was kept consistent for each data set tested. When fewer than the maximum 24 cores were used simulations were ‘locked’ to cores.

4.2 Test Cases

The test cases used for analysis in this study are described below. Some of the test cases are based on publically available data while others are based on proprietary real field data. Simulations are run utilizing parallelization. The number of threads and CPUs used for each simulation is chosen based on the model size and is kept consistent for each model for the entirety of this study.

4.2.1 SPE09

This model is based on the benchmark problem of the Ninth SPE Comparative Solution Project (Killough, 1995). It has 24 × 25 × 15 = 9000 cells, all active in the simulation. The grid dimensions are 7200 × 7500 × 359 ft^3. The depth to the center of the first grid block is 9010 ft and the reservoir dips 10 degrees. The GOC and WOC are located at 8800 ft and 9950 ft respectively. The porosity and thickness of each layer, the oil and gas PVT property data, and the gas-oil saturation functions are based on the Second SPE Comparative Solution Project (Weinstein, Chappelear, & Nolen, 1986). The reservoir has a geostatistically generated permeability distribution, which leads to a strong heterogeneity. Difficulties for simulation can arise because the water-oil capillary pressure has a discontinuity at a water saturation of 0.35 and the capillary pressure has a tail that does not extend to the water saturation 1.0. The model consists of one water injection well and 25 production wells and is a water injection problem. It is run for 900 days of simulated time.

All simulations of this data set are run serially (single thread on a single CPU).

4.2.2 MX53

This model consists of a single five spot patterns divided into 20 vertical layers. It has 53 × 53 × 20 = 56,180 cells aligned on a Cartesian grid. Its dimensions are 780 × 780 × 50 m^3. Depth is 1000m. The GOC and WOC are located at 1049 m and 800 m respectively. The permeabilities and porosities are constant. The model is a three-phase oil-water-gas water injection problem and is simulated for 15 years. This simulation is designed so that free gas appears and then increases throughout the simulation.

All simulations of this data set are run serially (single thread on a single CPU).

4.2.3 CPUTEST

This model is based on a real field data. It has $26 \times 81 \times 16 = 33.7 \times 10^3$ grid cells with 29.5×10^3 of them active in the simulation. The grid is a corner point approximation to a radial-like grid. The simulation is of is a three-phase, oil-water-gas water and gas injection problem. It is run for approximately 32 years simulated time. It has 47 wells. Plots of the vertical permeability in 3D and aerially for the fifth layer in Figure 4.1c and Figure 4.1d give an impression of this test case's heterogeneity and variation of cell sizes.

All simulations of this data set are run using 6 CPUs.

4.2.4 PCTST

This model is based on real field data characterized by an irregular region. It is discretized with $123 \times 40 \times 44 = 216.5 \times 10^3$ grid cells with 195.1×10^3 of them active in the simulation. The model is a three-phase, oil-water-gas primary production problem run for 3440 days simulated time and contains eleven production wells.

All simulations of this data set are run using 12 CPUs.

4.2.5 TM2T

This model is a fine grid version of PCTST. Each cell from the original data set was split equally, aerially into 3×3×1 cells. It has $369 \times 1200 \times 44 = 1.95$ million cells with 1.64 million of them active in the simulation. The model is a three-phase, oil-water-gas primary production problem run for 3440 days simulated time and contains eleven production wells. A plot of the vertical permeability is presented in Figure 4.1b.

All simulations of this data set are run using 24 CPUs.

4.2.6 SPE10

This model is based on the fine grid case for Model 2 from the Tenth SPE Comparative Solution Project (Christie & Blunt, 2001). It has 60 × 220 × 85 = 1.12 million cells, with 1.09 million of them active in the simulation. Its dimensions are 1200 × 2200 × 170 ft^3. The depth to the top this rectangular grid is 12,000 ft. The WOC is located at 14,000 ft. The reservoir has a geostatistically generated permeability distribution, which leads to a strong heterogeneity. The well pattern consists of a single five spot. The model is a two-phase, oil-water water injection problem run for 2000 days simulated time. A plot of the vertical permeability, showing its heterogeneity, is given in Figure 4.1a.

All simulations of this data set are done run 24 CPUs.

4.2.7 MX521

This model consists of 90 five spot patterns in a 10 x 9 areal arrangement divided into 20 vertical layers. It has 521 x 469 x 20 = 4.88 million cells aligned on a Cartesian grid. Its dimensions are 7800 × 7020 × 50 m^3. It is a three-phase oil-water-gas water injection problem and is simulated for 15 years. It has 200 wells. The five spot patterns are identical to that of MX53.

All simulations of this data set are run using 24 CPUs.

4.2.8 SINC

This model is based on a real field data characterized by an irregular region. It is discretized with 275 × 235 × 133 = 8.6 million grid cells with 2.6 million of them active in the simulation. The model is a three-phase, oil-water-gas water injection problem run for approximately 32 years simulated time. The model consists of six water injection wells and 997 production wells.

All simulations of this data set are run using 24 CPUs.

4.2.9 B3MM

This model is based on a real field data characterized by an irregular region. It is discretized with 230 × 273 × 209 = 13.1 million grid cells with 2.2 million of them active in the simulation. The model is a three-phase, API tracking, oil-water-gas primary production and water injection problem run for approximately 69 years simulated time. The oil is assumed to be comprised of two components, a light component and a heavy component. Properties of the oil are interpolated between the components. The model contains 32 injection wells and 2781 production wells.

All simulations of this data set are run using 24 CPUs.

4.2.10 MAUR

This model is based on a real field data of a reservoir which is 76% water. The reservoir has an irregular region discretized with 372 × 218 × 109 = 8.8 million grid cells with 3.5 million of them active in the simulation. The model is a three-phase, oil-water-gas water injection problem that with a water-cut above 95% for the entire 14.3 years simulated time. It contains 2 injection wells and 25 production wells.

All simulations of this data set are run using 24 CPUs.

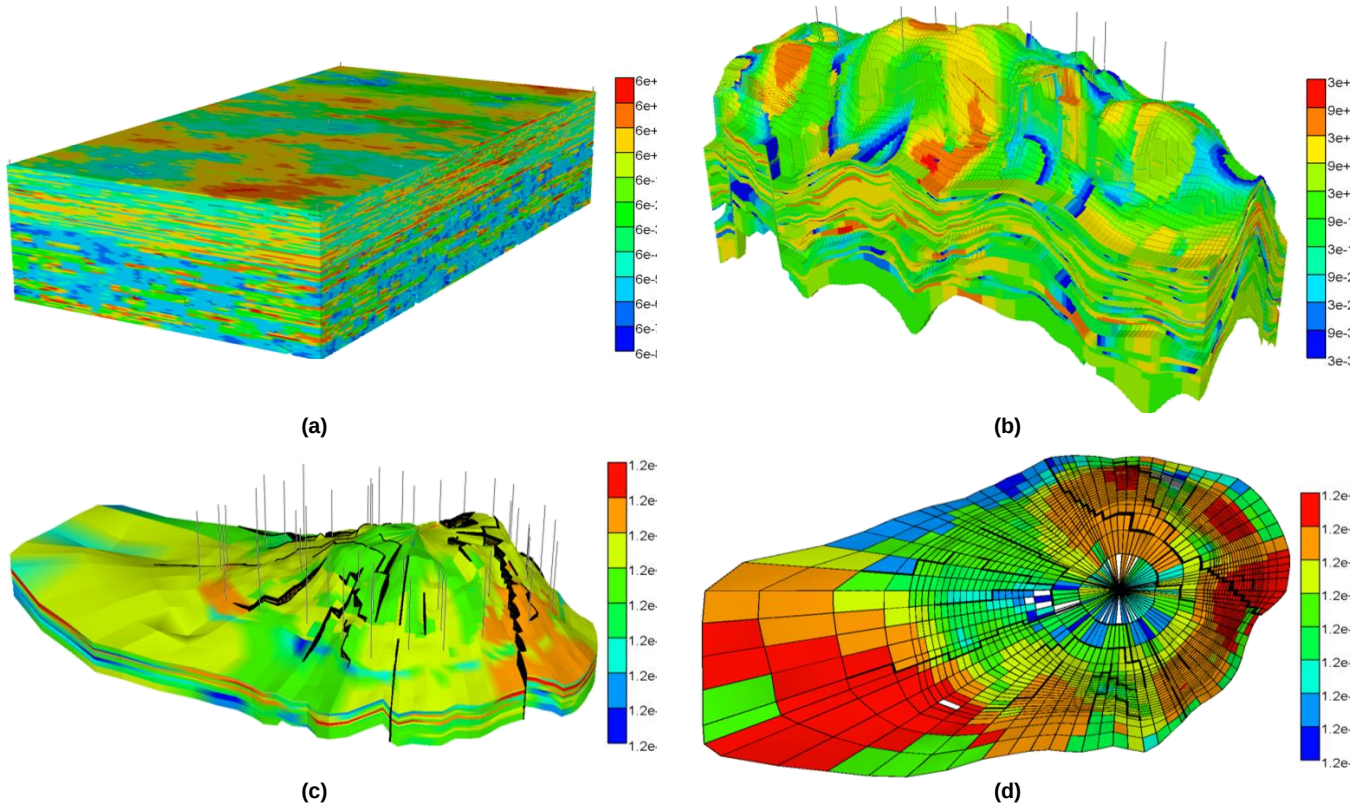


Figure 4.1: Vertical permeability plots of (a) SPE10, (b) TM2T, (c) CPUTEST in 3D and (d) vertical layer 5 of CPUTEST from (Gries et al., 2014).

4.3 Solver Verification

Verification of reservoir simulation using the combinative solver is done in two ways. The first and most important is to verify the linear solver produces the correct result. The second is to confirm the results of the simulation.

The linear solver is used to find the solution to $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$ for unknown $\boldsymbol{x}$. The solver is working properly if it finds correct solutions to given problems. The linear solver uses finite arithmetic and an iterative method. Therefore, the solution is considered correct and said to have converged if the residual $\|\boldsymbol{r}\| = \|\boldsymbol{b} - \boldsymbol{A} \cdot \boldsymbol{x}\|$ is smaller than a prescribed tolerance. The solver must be able to converge for the type of problems typically assigned to it. There are solver methods that are guaranteed to converge for every well-posed problem but that is not the case for the combinative solver. The solution methods making up the combinative solver sacrifices some robustness for improved speed. In particular, the combinative solver must be able to converge to the given tolerance for the vast majority of matrices produced by reservoir simulation problems.

Confirming the results of a reservoir simulation are correct provides an additional measure to ensure the correctness of the linear solver. Additional routines of reservoir simulator follow the linear solver portion to ensure the reservoir simulation results are correct numerically. When the nonlinear time stepping routine converges all that is required of the linear solver is the convergence of the linear problem. However, when a reservoir simulation is not converging to the correct solution it is more likely to create difficult matrix problems for the linear solver due to the unphysical nature of the solution. Also, a systematically non-converging linear solver will certainly lead to incorrect reservoir simulation results. Therefore a check of simulation results will indicate the linear solver is finding solutions that are “good enough” to reach the desired simulation result. This is especially important because the convergence tolerance of the linear

solver is arbitrary. The linear solver may not be able to converge to a prescribed tolerance that is set too tight but still leads to correct results. Quite often there is quite a lot f uncertainty in the equations an properties leading to the linear system and so requiring very tight solutions does not necessarily mean the simulation result will be "correct".

For this study the results of simulations using the combinative solver are compared to those using the commercially available ILU only solver. Though these results are not guaranteed to be the "real" solution the method and simulator are widely accepted by the industry to produce correct results. That is the standard by which the combinative solver is measured. In some cases, correct solutions are known and are used for comparison as well.

4.3.1 Combinative Method Analysis

The primary goal of the combinative method is reducing simulation time. However, it is important to examine the convergence behaviour of the method to help ensure its robustness. Unfortunately, an analytic result for AMG and the combinative method does not exist. Therefore empirical evidence must suffice.

In §4.3.3 a comparison of the average number of solver iterations per Newton cycle for seven models is made. This result provides evidence that linear systems from this class of problems converge well with the combinative method. In this section, a comparison is made using representative matrices from the same models. Results of the preconditioned combinative method are compared to results of the ILU preconditioned GMRES method that is standard in the IMEX solver measured by the root mean square (RMS) of the residual vectors. All methods use the same matrix and initial guess. The stopping criterion of the (f-)GMRES iteration is a relative tolerance of 10^{-9} meaning that iteration stops if the RMS of the final residual is reduced to less than that factor of the initial residual. To ensure the combinative method has stable

convergence behaviour these linear systems are solved to a higher accuracy than is normally used in simulation.

4.3.2 Single System Convergence

Results showing the behaviour of the relative residual of select linear systems from eight test cases are shown in Figure 4.2 to Figure 4.9. The magnitude of the reduction of the residual is plotted on a logarithmic scale against the number of iterations used. The linear systems shown were chosen from early, middle, and late simulation times from the various test cases. An effort was made to choose linear systems that were representative of each model.

For the SPE09 case results are shown in Figure 4.2. The linear system is taken at simulation day 177.223 which has a time step of 16.617 days. The first Newton cycle is shown.

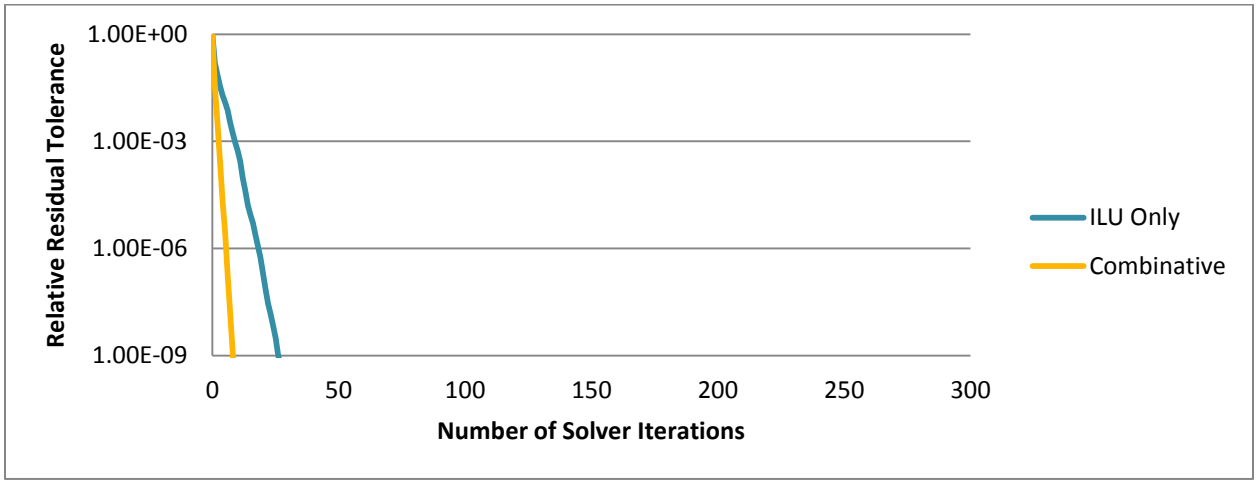


Figure 4.2: Linear solver convergence for time step 18 from case SPE09.

Both solver methods are able to continuously reduce the residual logarithmically. The combinative method converges in 9 iterations while the ILU only method takes 27. In this case both methods work extremely well and the ILU only method is more efficient because of the relatively low numerical cost of iteration.

For the SPE10 case results are shown in Figure 4.3. The linear system is taken at simulation day 50 which has a time step of 0.9 days. The first Newton cycle is shown.

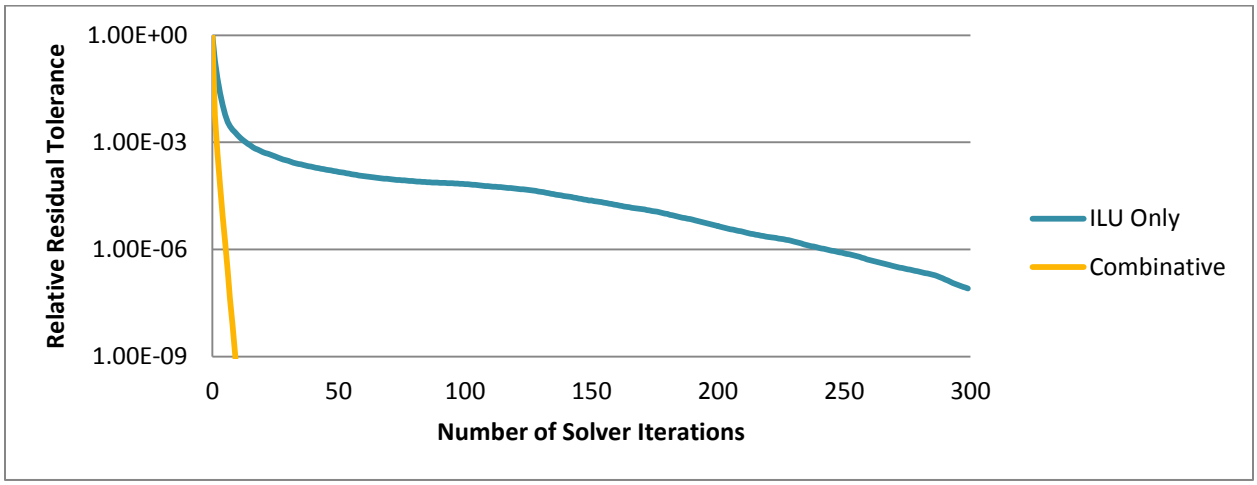


Figure 4.3: Linear solver convergence for time step 214 from case SPE10.

Both methods are able to continuously reduce the residual but the ILU only method slows significantly after reducing by 10^{-3}. The combinative method converges in 10 iterations and the ILU only method ends with a relative residual of 8.12×10^{-8} in the maximum 300 iterations. The combinative method is clearly better for this system.

For the MX53 case results are shown in Figure 4.4. The linear system is taken at simulation day 5420 which has a time step of 32 days. The first Newton cycle is shown.

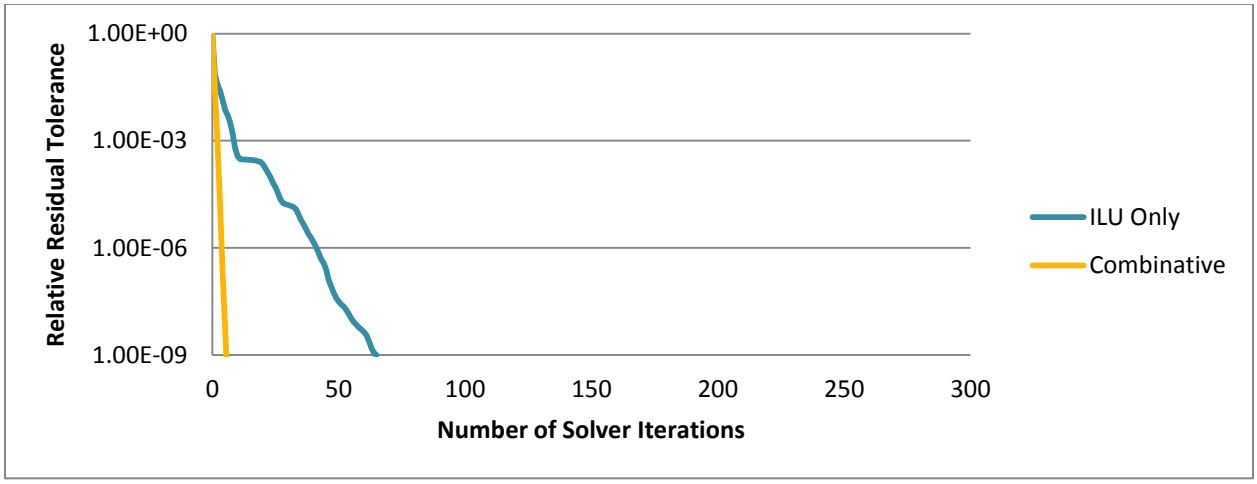


Figure 4.4: Linear solver convergence for time step 211 from case MX53.

Both methods are able to continuously reduce the residual. The combinative method converges in 6 iterations while the ILU only method takes 65. In this case both methods converge with the combinative method taking about half the time. The MX521 case is the same as the MX53 case except the single 5 spot pattern is repeated 90 times in a 10×9 grid. Results are shown in Figure 4.5. The linear system corresponds to that of MX53.

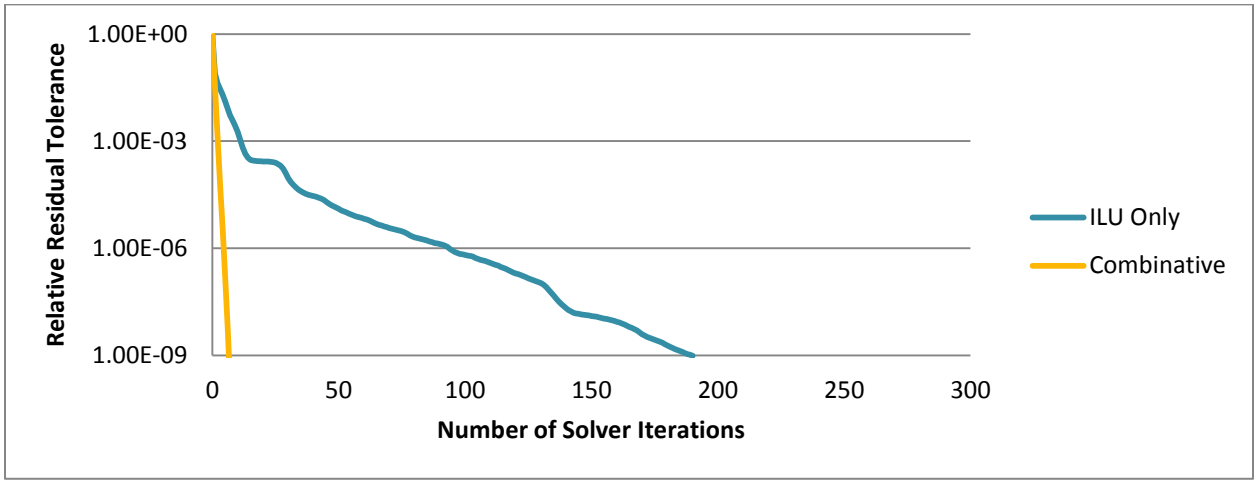


Figure 4.5: Linear solver convergence for time step 211 from case MX521.

The combinative method converges in 7 iterations, just one more than for MX53. The ILU only method convergence is similar up to iteration 27 with a reduction of 9.3×10^{-6}. Then the convergence is slower than with MX53 taking 190 iterations to converge. The evidence indicates the pressure solution on the larger grid is more difficult to solve using the ILU only method but not with the combinative method.

For the PCTST case results are shown in Figure 4.6. The first linear system is taken which has a time step of 5 days. The first Newton cycle is shown.

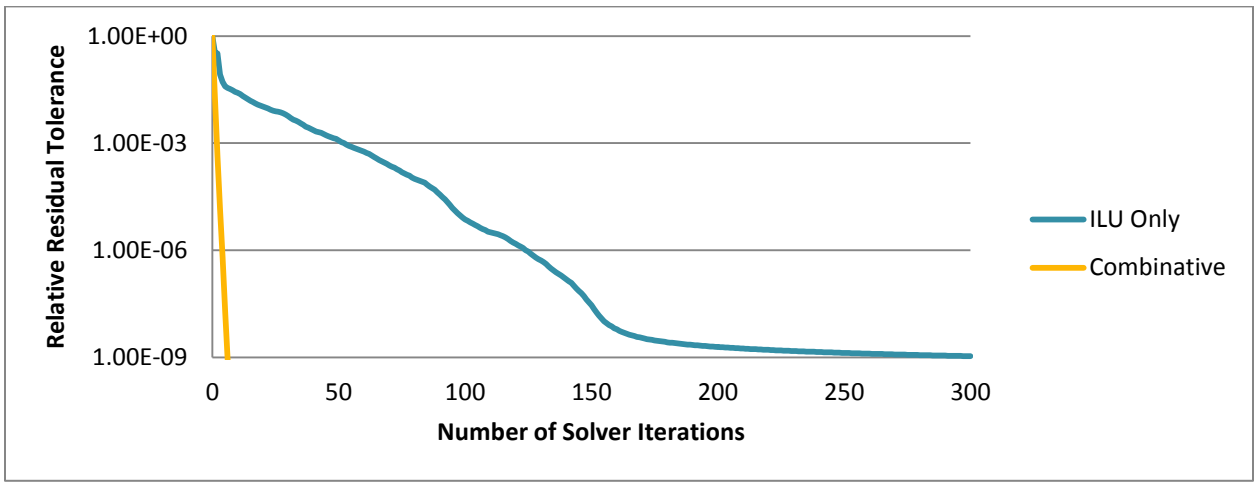


Figure 4.6: Linear solver convergence for time step 1 from case PCTST.

The combinative method converges in 6 iterations. The ILU only method fails to achieve the required reduction reaching 1.1×10^{-9} in the maximum 300 iterations. The TM2T case is the same as the PCTST case except each grid cell has been divided into 3×3 cells areally. Results are shown in Figure 4.7. The linear system corresponds to that of PCTST.

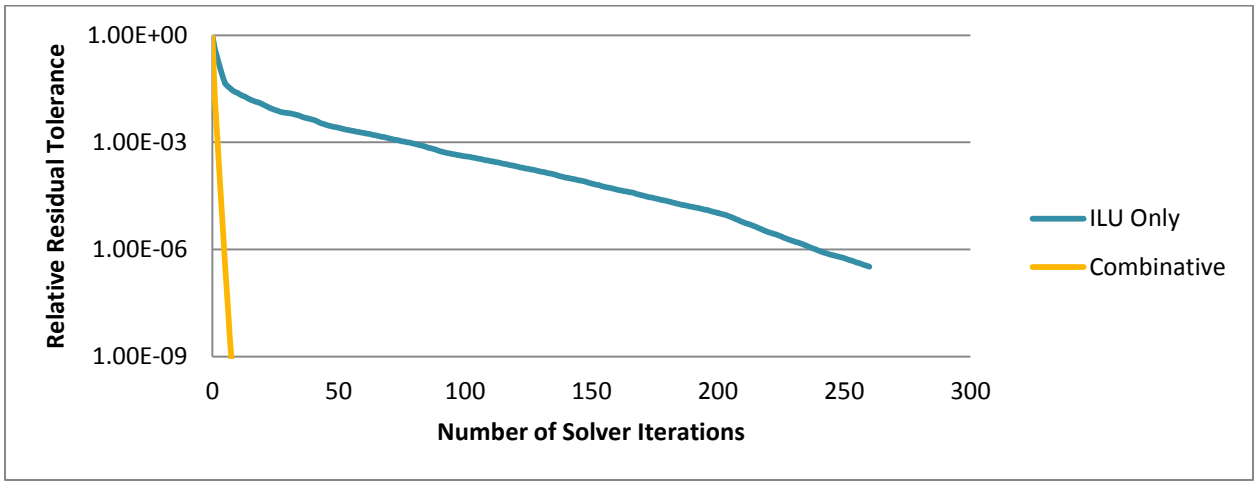


Figure 4.7: Linear solver convergence for time step 1 from case TM2T.

The combinative method converges in 8 iterations, just two more than for PCTST. The ILU only method fails to achieve the required reduction reaching 3.3×10^{-7} in 260 iterations. The linear solver stalls at this point due to linear dependence of the Krylov space. Again, the evidence indicates the pressure solution on the larger grid is more difficult to solve using the ILU only

method but not with the combinative method. The combinative method is clearly better for this system.

For the CPUTEST case results are shown in Figure 4.8. The linear system is taken at simulation day 10783 which has a time step of 32 days. The second Newton cycle is shown.

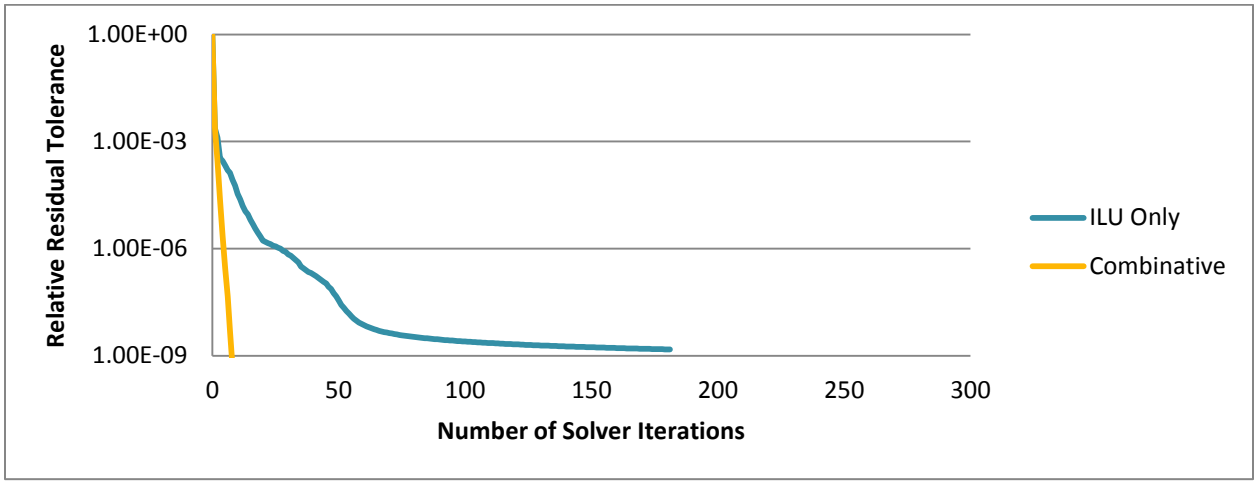


Figure 4.8: Linear solver convergence for time step 534, Newton cycle 2 from case CPUTEST.

Both methods are able to continuously reduce the residual to nearly 3×10^{-4} at which point the ILU only method slows considerably. The combinative method converges in 8 iterations while the ILU only method achieves a 1.5×10^{-9} reduction in 181 iterations before stalling.

Finally, for the SINC case results are shown in Figure 4.9. The linear system is taken at simulation day 5205.482 which has a time step of 4.815 days. The first Newton cycle is shown.

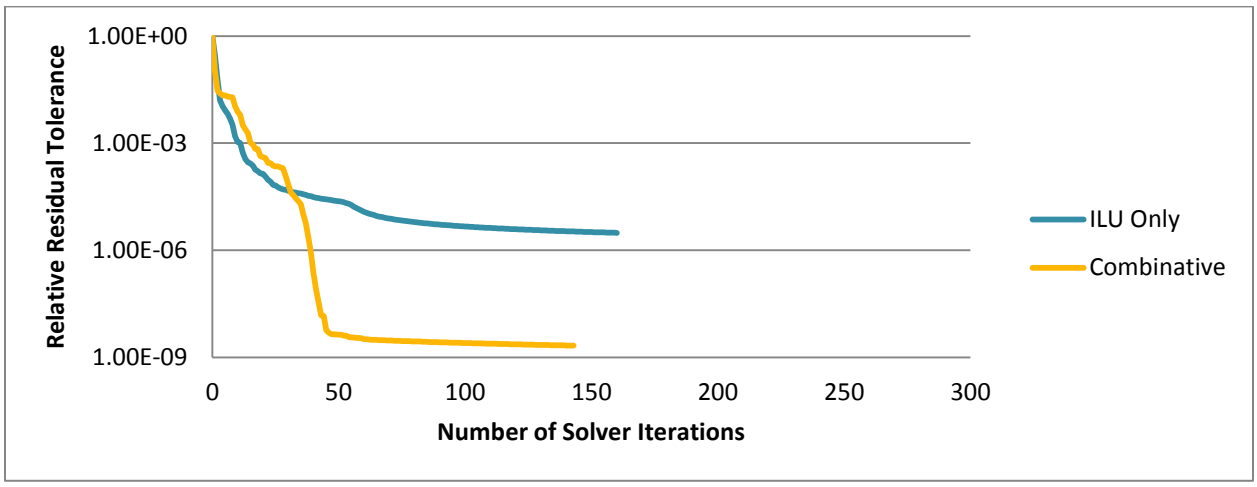


Figure 4.9: Linear solver convergence for time step 1539 from case SINC.

Of all the linear systems examined in this chapter this is the only one where the combinative method does not exhibit logarithmic convergence behaviour. Both methods are still able to continuously reduce the residual but the ILU only method actually outperforms the combinative method between iteration 3 and iteration 30 at which point the reduction is 4.6×10^{-5} with ILU only and 7.1×10^{-5} with the combinative method. The combinative method further reduces the residual to 5.8×10^{-9} at iteration 45 before slowing and finally stalling out with 2.1×10^{-9} at iteration 143. Interestingly, the combinative method switches to an ILU only method at iteration 43 because the magnitude of the pressure update from the first stage (AMG solution) is extremely small. The ILU only method performs worse at higher precision achieving a residual reduction of 3.1×10^{-6} in 160 iterations before stalling.

For each of the linear systems tested the combinative method was able to converge to a tighter precision is fewer iterations than the ILU only method. Although this does not confirm the robustness of the method it does provide compelling evidence. In full reservoir simulation models using tight precisions are often not necessary. The question of whether the combinative method is more efficient than the ILU only method is better assessed by examining computation time.

4.3.3 Linear Solution Convergence

A set of seven test cases was run for a range of relative residual tolerances tightening from 10^{-4} to 10^{-9}. The maximum number of solver iterations per Newton cycle was set to 300. If the relative residual is not less than the given tolerance within the maximum number of solver iterations the linear solver is considered to have failed. The linear solver stalls if the new basis vector generated in the current step of the Krylov subspace iteration (Saad, 2003) is in or nearly in the linear span of the preceding basis vectors. Linear dependence in the basis generation process often occurs when the preconditioned system matrix is ill conditioned and therefore has a wide range of eigenvalue magnitudes. The stall test does not directly assess linear dependence of the basis, but does detect when the norm of the current vector has little contribution from vectors outside the span of the basis.

For the seven test cases run for each relative residual tolerance, no solver failures or stalls occurred. Therefore the combinative solver is converging well for this set of cases. The combinative solver actually outperformed the ILU solver, which did have a small number of solver failures for tight tolerances. This implies the combinative solver is working well; it was able to converge to the desired tolerance for all linear systems generated in these test cases. Additionally, the average number of solver iterations required to reach the given relative residual tolerance is tracked and shown in Figure 4.10 when using the combinative preconditioner and in Figure 4.11 when using the ILU only solver.

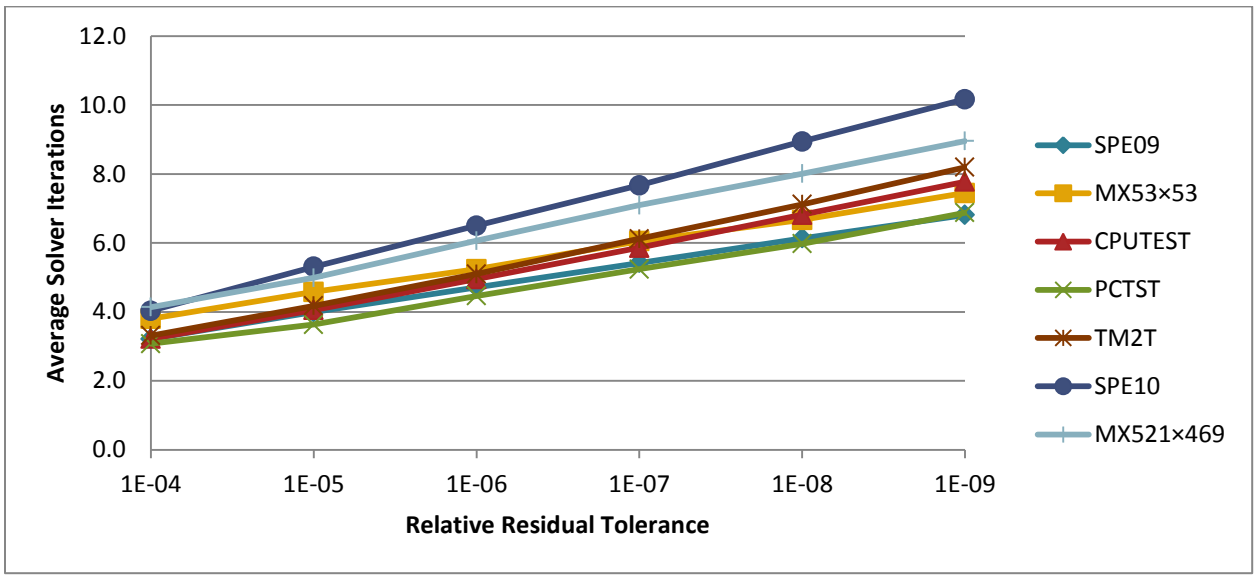


Figure 4.10: The average number of solver iterations per Newton cycle required to converge to a given relative residual tolerance for 7 test cases. Results are shown for the combinative solver.

The results show a linear trend on the log plot meaning the average number of solver iterations goes up by a constant (~ 0.8–1.2) for each order of magnitude decrease in the residual. The fewest average solver iterations required is 3.1 and the most is 10.2.

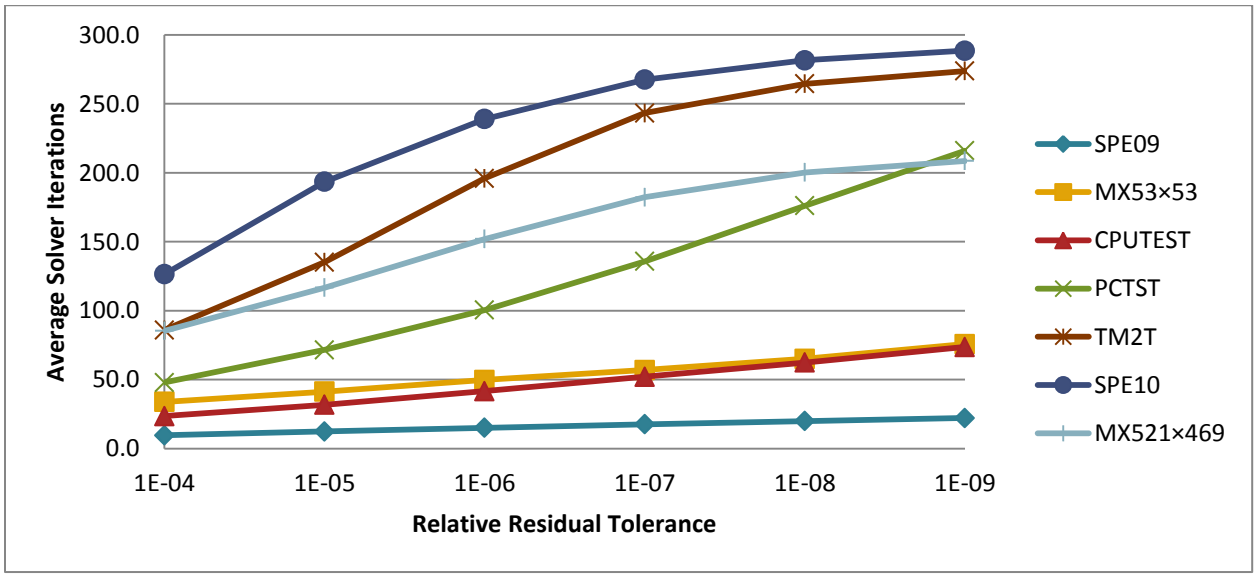


Figure 4.11: The average number of solver iterations (max 300) per Newton cycle required to converge to a given relative residual tolerance for 7 test cases. Results are shown for the ILU only solver.

For each case at each required tolerance the average number of solver iterations required is larger with the ILU only solver than the combinative solver. The fewest average solver iterations required is 9.6 for SPE09 with a relative tolerance of 10^{-4} and the most is 288.4 for SPE10 with a tolerance of 10^{-9}. As the number of average solver iterations increases toward 300 the average is suppressed by the maximum 300 iterations for any Newton cycle. Hence the average appears to approach 300 asymptotically in the SPE10 case. Although there were solver failures for the ILU only solver they only happen at tighter tolerances and for each case and the simulation results matched those found with a relative residual tolerance of 10^{-4}. The fraction of Newton cycles with solver failures (or stalls) is shown in Figure 4.12.

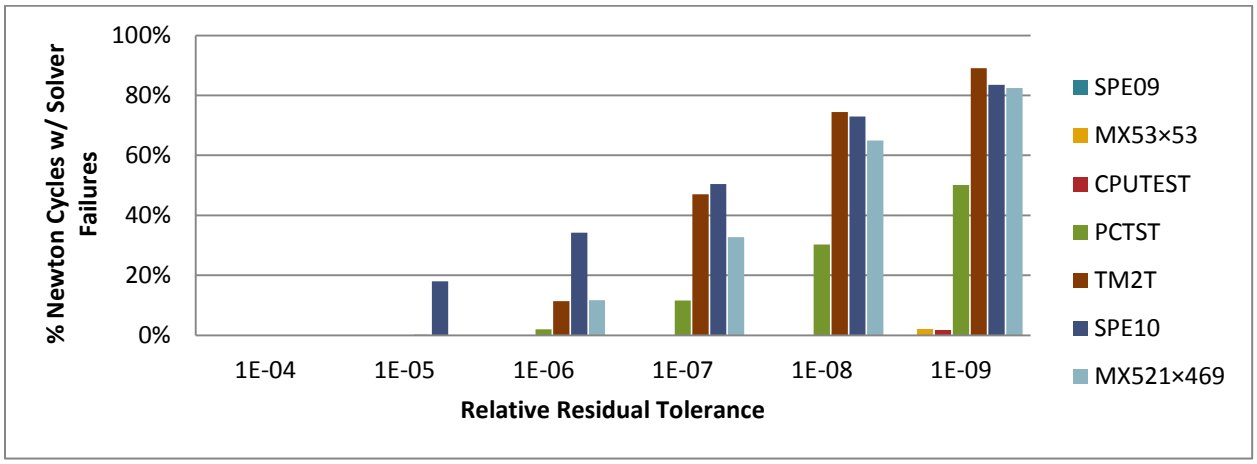


Figure 4.12: The percentage of Newton cycles with solver failures using the ILU solver method for 7 test cases. The relative residual varies from 10^{-4} to 10^{-9}.

Despite the large fraction of solver failures at tight relative residual tolerances the final reservoir simulation solutions were virtually the same as when looser relative residual tolerances were used. Essentially, the tolerance asked for was more precise than the data used in the problem formation allowed. This makes a strong case that the combinative solver not only speeds certain simulations, it can actually be more accurate as well.

4.3.4 Final Results Check

In addition to analyzing the accuracy and performance of the linear solver, the accuracy of the results of full reservoir simulations is also confirmed.

The first check for simulation accuracy is to compare the final average reservoir pressures and cumulative production values of oil, water, gas, and the total fluid at the end of simulation when using both the ILU only preconditioned solver and the combinative preconditioned solver. Results for each of the black-oil test cases are listed in Table 4.1.

Case	Average Reservoir Pressure	Cumulative Oil Production	Cumulative Water Production	Cumulative Gas Production	Total Cumulative Production
SPE09	0.00%	0.00%	0.00%	0.00%	0.00%
MX53	-0.06%	-0.05%	0.00%	-0.03%	-0.04%
CPUTEST	0.00%	0.00%	-0.01%	0.01%	0.01%
PCTST	-0.02%	0.01%	-0.04%	0.18%	0.03%
TM2T	-0.08%	0.02%	-0.24%	-0.07%	-0.02%
SPE10	0.00%	-0.73%	-1.11%	-	-1.00%
MX521	-0.06%	-0.04%	-0.02%	-0.01%	-0.02%
SINC	-0.07%	0.68%	0.73%	1.12%	0.75%
B3MM	-0.03%	0.00%	0.00%	0.00%	0.00%
MAUR	-0.04%	0.00%	0.00%	0.00%	0.00%

Table 4.1: Percent difference between the final average reservoir pressure and cumulative production values between simulations using the combinative solver and the ILU only solver.

The total cumulative production and the average reservoir pressures differ by less than 1% in all cases. This implies that the combinative solver is performing at least as well as the ILU only solver at the level of precision set for the various test cases. There is no divergence significant enough between the two solver methods to produce a different final result.

4.3.5 Reservoir Properties over Time

The second check of accuracy is to examine the change of reservoir properties over the course of reservoir simulation. For this plots of oil production rate, water production rate, average reservoir pressure, and average gas saturation are shown for the TM2T test case in Figure 4.13.

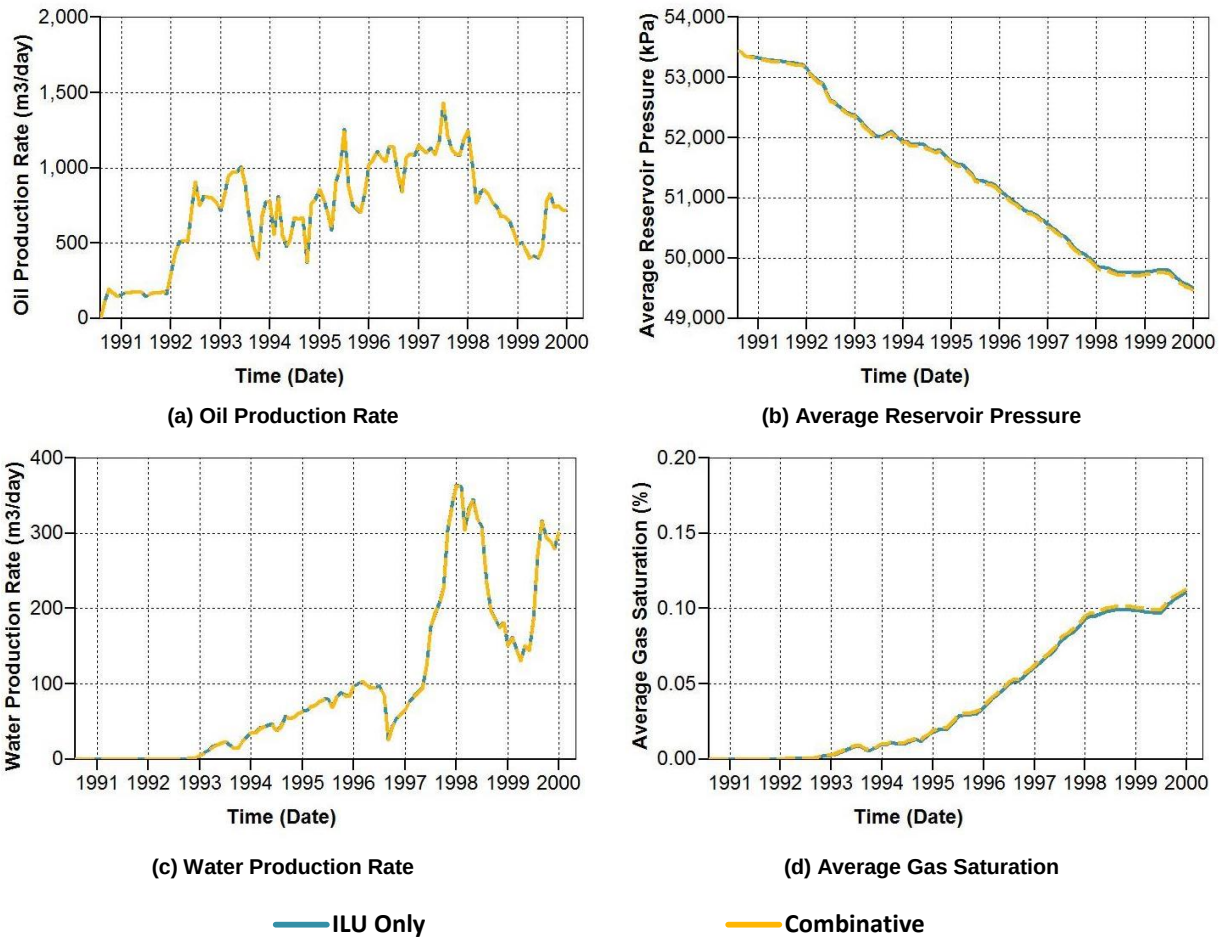


Figure 4.13: Simulation results of the TM2T case using the ILU only preconditioner and the combinative preconditioner after (Gries et al., 2014).

The production rates are perfectly aligned and there is very little difference in average pressure and gas saturation. This is expected because the outer control of the solver in each simulation is identical leaving very little margin for the 'drift' of results.

Producing the same result using the same simulator with different preconditioning methods shows the solver is able to converge well enough for the generated linear systems. However, it is not entirely satisfying. It is possible none of the simulations are producing the *true* result.

To make sure the simulator with the combinative preconditioned solver can produce correct results the SPE09 and SPE10 test cases are examined in more detail.

Results of the SPE10 case using both the ILU only preconditioner and the combinative preconditioner are compared to those from the 10th SPE comparative project (Christie & Blunt, 2001). Comparisons to the accepted solutions presented for the project are made using both preconditioning methods with the IMEX simulator. Results corresponding to Figures 7-10 of (Christie & Blunt, 2001) are shown in Figure 4.14.

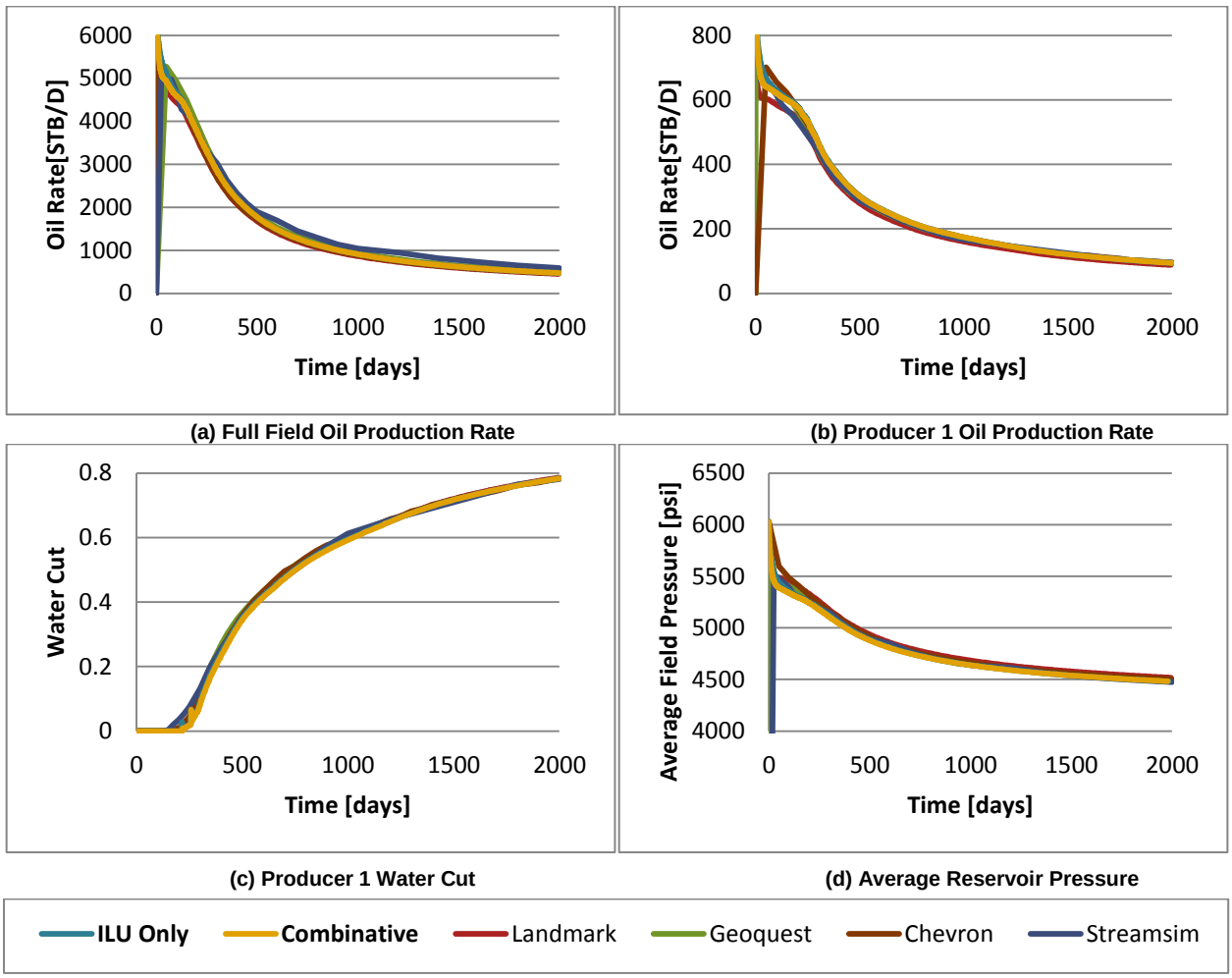


Figure 4.14: Simulation results of the SPE10 case using the ILU only preconditioner and the combinative preconditioner shown with the results of Figures 7-10 of SPE-72469-PA (Christie & Blunt, 2001).

Each of the plots shows a good agreement with both preconditioned methods of the simulator with those of the comparative project. In particular, the results most closely match those of the Chevron data.

Unfortunately, the original data from the 9th SPE comparative project (Killough, 1995) can no longer be obtained. However, results from simulations using the ILU only preconditioner and the combinative preconditioner are shown in Figure 4.15.

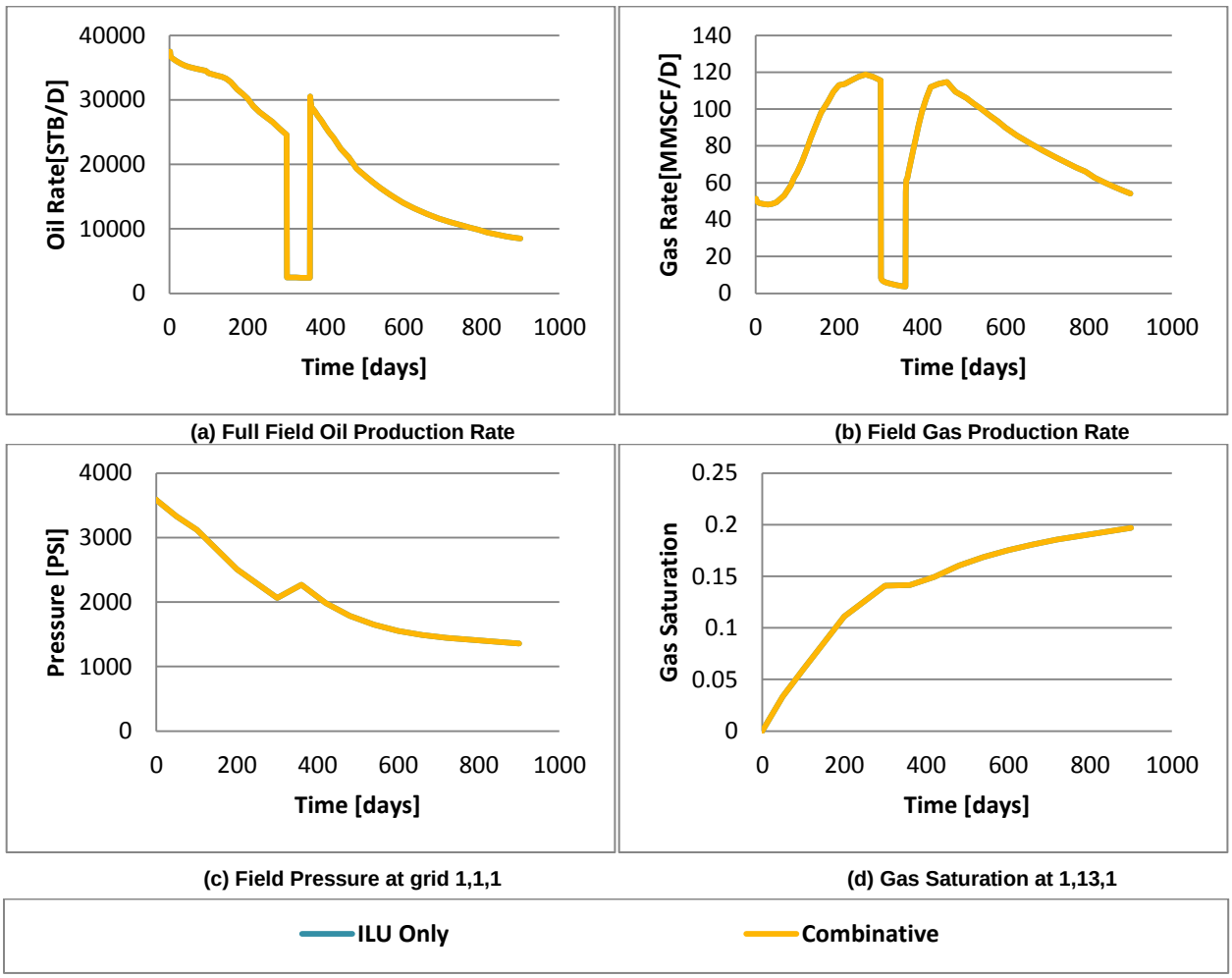


Figure 4.15: Simulation results of the SPE09 case using the ILU only preconditioner and the combinative preconditioner. Plots correspond to Figures 8, 9, 11, and 12 of SPE-29110-MS (Killough, 1995).

ILU only results and the combinative results match exactly to the sixth digit again producing the same result with the different preconditioning methods (the ILU only results lie directly behind the combinative results on the plots). These results were produced requiring a tight relative residual reduction of 10^{-9} and the reservoir grid is small (9000 grid cells) so the nearly exact match is reasonable. The results of these simulations show a qualitative match to Figures 8, 9, 11, and 12 of the 9th SPE comparative project (Killough, 1995) indicating again that the

simulator produces accurate results, up to the standards of the industry, using either preconditioner.

4.4 Eigenvalue Analysis

In this section an analysis of the distribution of eigenvalues from three of the linear systems examined in §4.3.1 is performed. The three smallest problems (SPE09, MX53, and CPUTEST) are chosen due to the high computational cost in determining eigenvalues. For each case, the combinative method is applied using three different decoupling preconditioners; alternate block factorization (ABF), row-scaling (RSCALE), and the dynamic row-sum preconditioner (DRS). More details of each of these preconditioners are given in §3.2 and an analysis of the relative performance of these preconditioners on full reservoir simulations is given in §5.3.4.1. For each system, a relative residual reduction of 10^{-9} is taken as the stopping criterion.

Case	Number of Solver Iterations (and average AMG cycles)						
	ILU Only	Combinative					
		ABF		RSCALE		DRS	
SPE09	27	8	(1)	7	(1)	9	(1)
MX53	65	12	(1.17)	7	(1)	6	(1)
CPUTEST	181*	298*	(4.98)	75	(4.96)	8	(1)

Table 4.2: Number of solver iterations required to reach a residual reduction of 10^{-9} using for solver methods. *Denotes stagnation prior to convergence.

Table 4.2 shows the number of solver iterations for the three cases with each of the solver method and preconditioner combinations. The SPE09 and MX53 matrices show good convergence with all four methods. The CPUTEST matrix was only able to converge to the required residual tolerance with two of the methods and only the DRS preconditioner led to a small number of solver iterations.

Eigenvalues of the extracted pressure aligned sub-matrix, $\widetilde{\boldsymbol{A}}_p$, are compared for the three decoupling preconditioners when used with the combinative solver. The purpose of this work is to improve the convergence properties of the AMG solver for the non-symmetric problems. To

that end, the condition number of the preconditioned matrix plays a rather minor role; in all the three systems studied in detail the RSCALE and DRS preconditioners do not improve the condition number but are better for overall performance. The *distribution* of the eigenvalues is what is most relevant to the performance of the algorithms on the studied problems (Gordon & Gordon, 2010).

4.4.1 SPE09

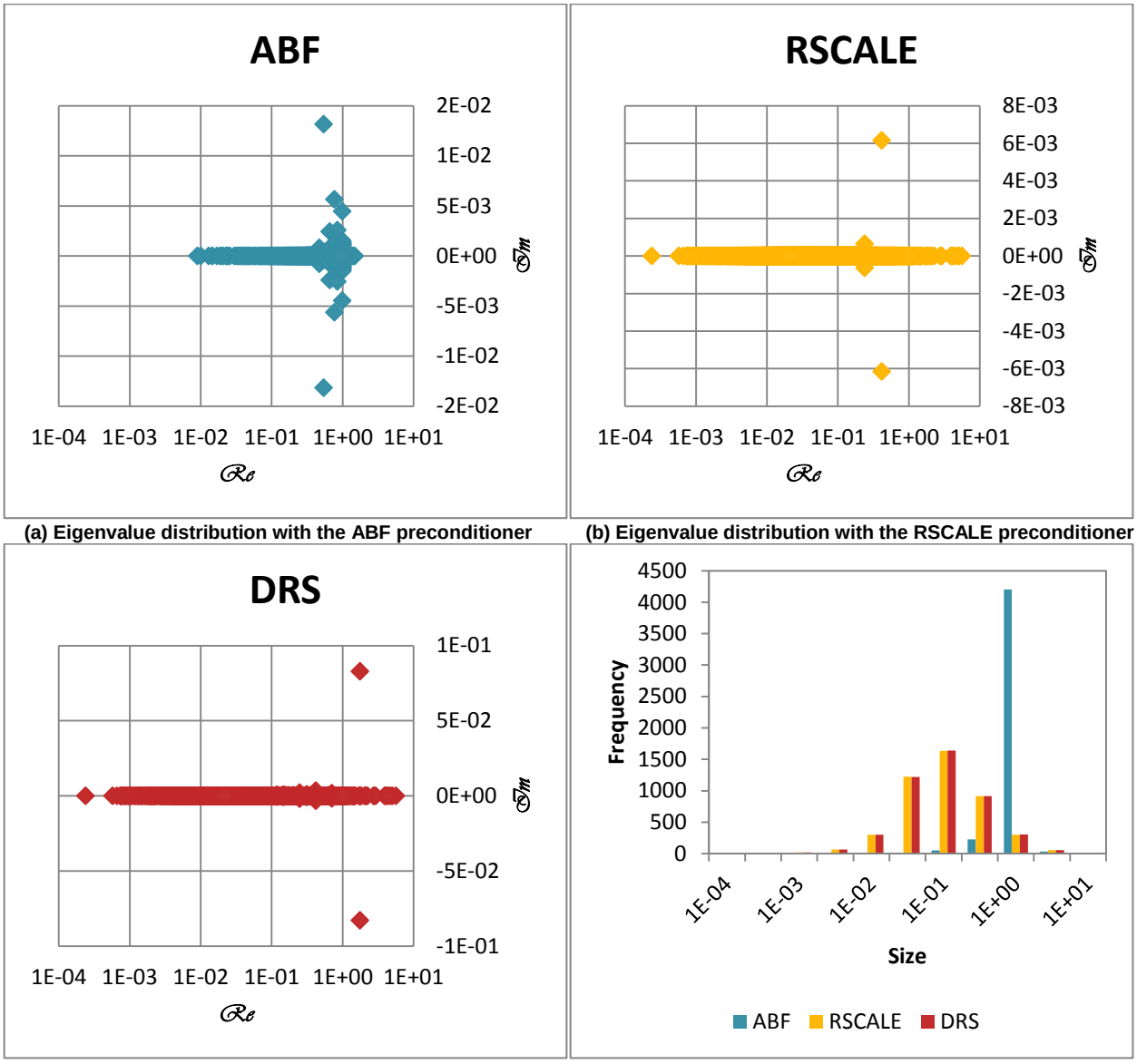
The first linear system examined is from the SPE09 test case. For this linear system each preconditioned combinative method achieves convergence in single digit iterations requiring one AMG cycle per combinative iteration. Table 4.3 shows the values of the minimum and maximum eigenvalues, the condition number, and the number of real and complex eigenvalues for the three decoupled pressure aligned sub-matrices.

Matrix	λ_{max}	λ_{min}	$\lambda_{max}/\lambda_{min}$	No. of Real	No. of Complex
ABF	1.48E+00	8.94E-03	1.66E+02	2168	2358
RSCALE	5.60E+00	2.38E-04	2.35E+04	4426	100
DRS	5.64E+00	2.38E-04	2.37E+04	4420	106

Table 4.3: Basic eigenvalue information for time step 18 from case SPE09

Clearly, the ABF method leads to the smallest condition number by two orders of magnitude. This is due to the normalization of the rows with this preconditioner that is not done for RSCALE and DRS. More than half the eigenvalues are complex using the ABF preconditioner whereas only 2% are complex using RSCALE and DRS.

The distribution of eigenvalues in the imaginary plane with a log axis for the real component is shown in Figure 4.16 a) b) c) with a histogram of the distribution of eigenvalue magnitude in Figure 4.16 d).



(a) Eigenvalue distribution with the ABF preconditioner

(b) Eigenvalue distribution with the RSCALE preconditioner

(c) Eigenvalue distribution with the DRS preconditioner

(d) Histogram of eigenvalues for the pressure matrix

Figure 4.16: Eigenvalue distributions for the pressure matrix determined for time step 18 from case SPE09.

With ABF scaling there is a congestion of eigenvalues with real component near one. The DRS and RSCALE preconditioners give very similar eigenvalue distributions. The eigenspectra are broader than that with ABF resulting in larger condition numbers with those preconditioners. The high concentration of eigenvalues near one is also gone as the distribution is spread out. The

main difference between DRS and RSCALE is the size of the imaginary part of a single pair of eigenvalues.

4.4.2 MX53

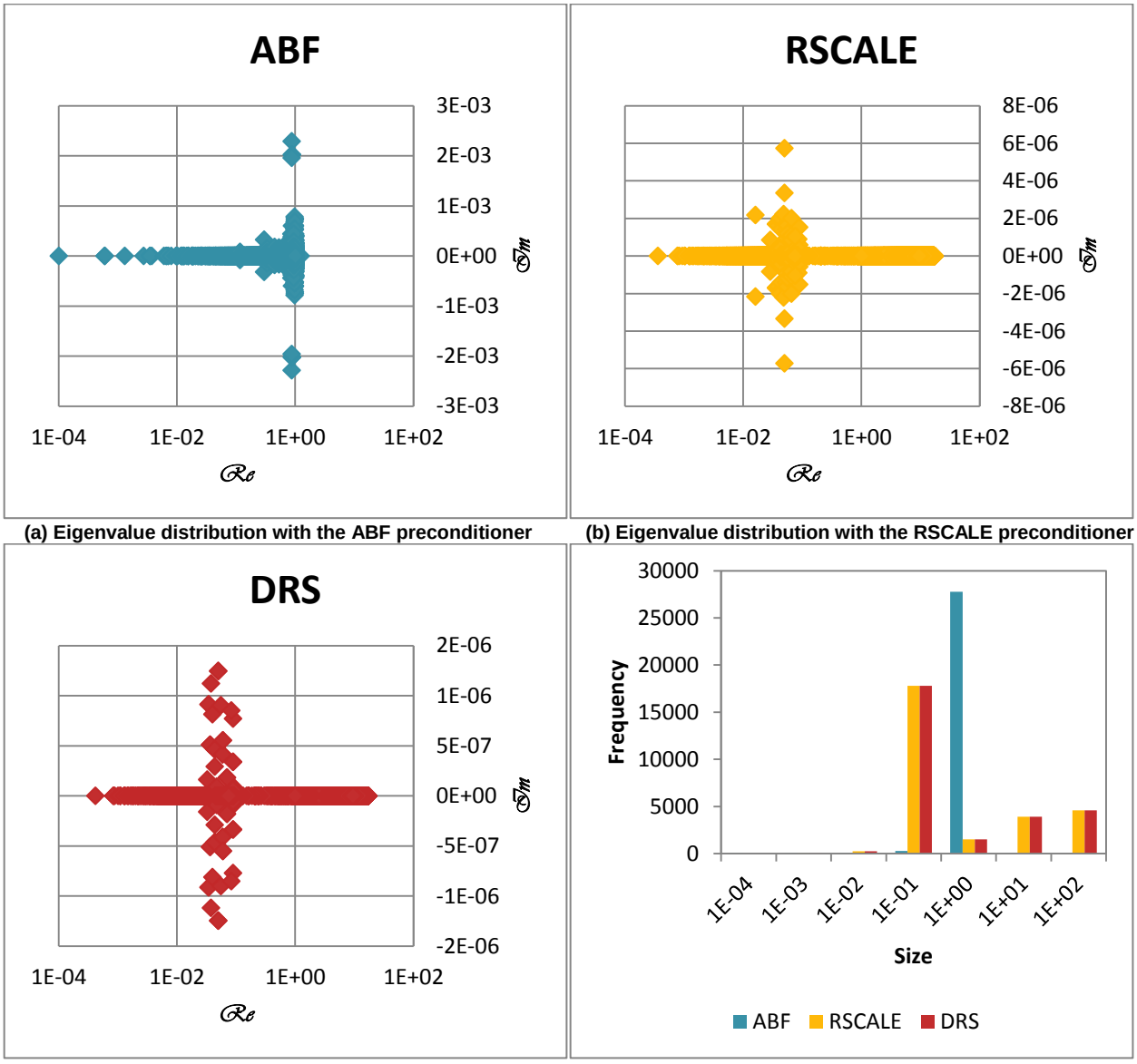
The second linear system examined is from the MX53 test case. For this linear system the DRS and RSCALE preconditioned combinative methods achieve convergence in single digit iterations requiring one AMG cycle per combinative iteration and the ABF preconditioner requires 12 combinative iterations including 14 AMG cycles total. Table 4.4 shows the values of the minimum and maximum eigenvalues, the condition number, and the number of real and complex eigenvalues for the three decoupled pressure aligned sub-matrices.

Matrix	λ_{max}	λ_{min}	$\lambda_{max}/\lambda_{min}$	No. of Real	No. of Complex
ABF	1.23E+00	9.96E-05	1.23E+04	20169	7926
RSCALE	1.73E+01	3.58E-04	4.82E+04	28005	90
DRS	1.73E+01	4.17E-04	4.14E+04	28041	54

Table 4.4: Basic eigenvalue information for time step 211 from case MX53

Again, the ABF method leads to the smallest condition number but for this case all the condition number are of the same order of magnitude. Also, many more eigenvalues are complex with the ABF preconditioner than the others. The DRS preconditioner leads to the fewest number of complex eigenvalues.

The distribution of eigenvalues in the imaginary plane with a log axis for the real component is shown in Figure 4.17a) b) c) with a histogram of the distribution of eigenvalue magnitude in Figure 4.17 d).



(a) Eigenvalue distribution with the ABF preconditioner

(b) Eigenvalue distribution with the RSCALE preconditioner

(c) Eigenvalue distribution with the DRS preconditioner

(d) Histogram of eigenvalues for the pressure matrix

Figure 4.17: Eigenvalue distributions for the pressure matrix determined time step 211 from case MX53

Again, with the ABF preconditioner there is a concentration of eigenvalues near one and the range of the eigenspectra with RSCALE and DRS is larger. The concentration of eigenvalues is not spread as much as in the previous SPE09 case but rather is "pushed" closer to values with a real component near 0.75. The main difference in eigenvalue distributions between RSCALE and

DRS is an increase in pairs of eigenvalues with imaginary values up to three times larger in magnitude.

4.4.3 CPUTEST

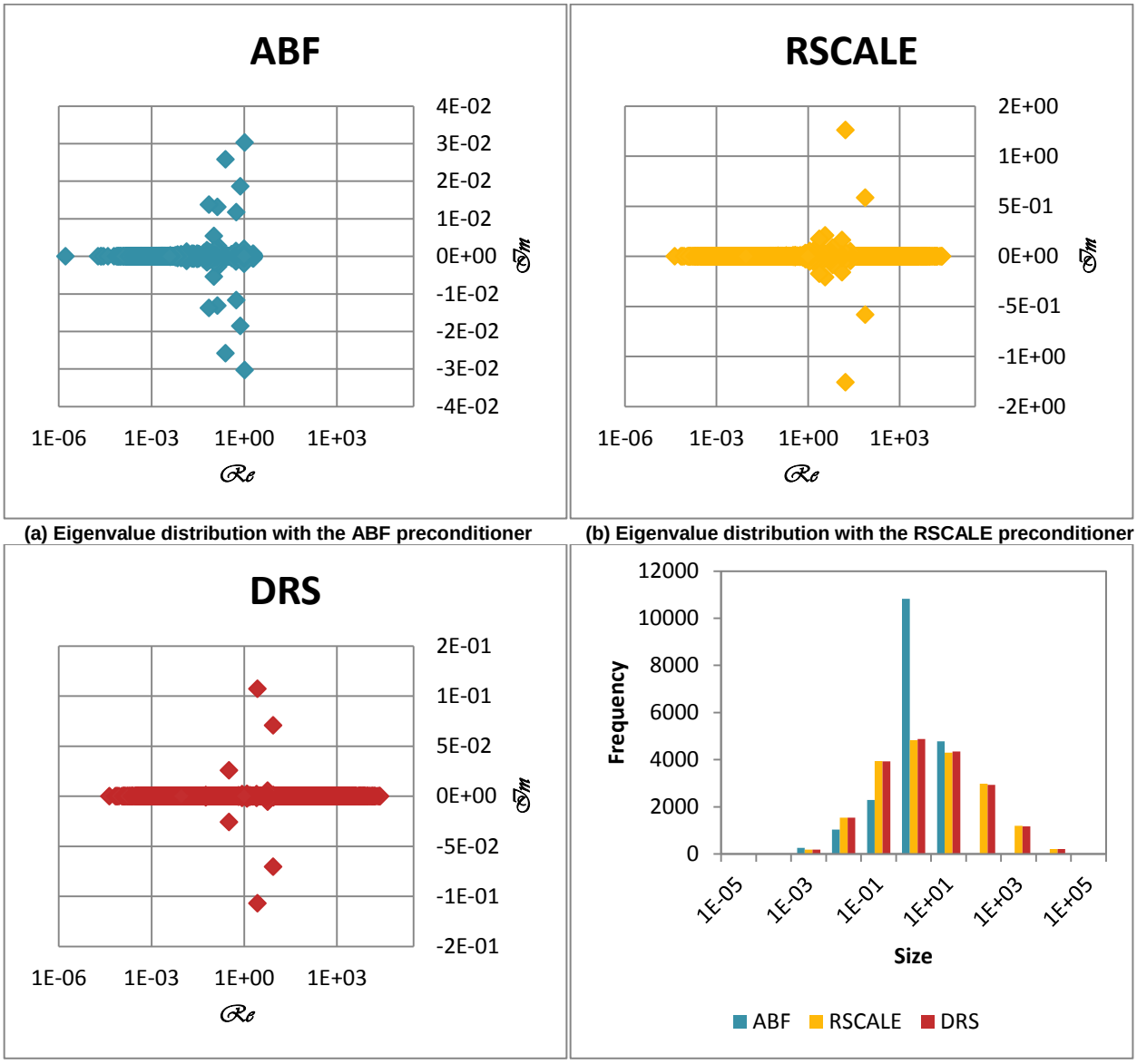
The final linear system examined is from the CPUTEST test case. For this linear system the DRS and RSCALE preconditioned combinative methods achieve convergence with only the DRS method requiring single digit iterations and one AMG cycle per combinative iteration. The ILU only method and the ABF preconditioned combinative method were unable to converge to the required residual tolerance (see Figure 4.8). With the ABF and RSCALE preconditioners the maximum 5 AMG cycles were taken for nearly every solver iteration indicating a lack of convergence of the pressure sub-problem with AMG. The extracted pressure system is indefinite with those two preconditioners, each has eigenvalues in the four quadrants of the imaginary plane. Four of the eigenvalue have negative real parts when using the ABF and RSCALE preconditioners. All eigenvalues have positive real parts with the DRS preconditioner. Table 4.5 shows the values of the minimum and maximum eigenvalues, the condition number, and the number of real and complex eigenvalues for the three decoupled pressure aligned sub-matrices.

Matrix	λ_{max}	λ_{min}	$\lambda_{max}/\lambda_{min}$	No. of Real	No. of Complex
ABF	2.00E+00	1.65E-06	1.21E+06	18639	566
RSCALE	2.47E+04	4.32E-05	5.72E+08	17905	1300
DRS	2.44E+04	4.32E-05	5.65E+08	19129	76

Table 4.5: Basic eigenvalue information for time step 534, Newton cycle 2 from case CPUTEST

Again, the ABF method leads to the smallest condition number by two orders of magnitude. For this case the RSCALE preconditioner gives the largest number of complex eigenvalues and yet again the DRS preconditioner gives the fewest.

The distribution of eigenvalues with positive real component in the imaginary plane with a log axis for the real component are shown in Figure 4.18 a) b) c) with a histogram of the distribution of eigenvalue magnitude in Figure 4.18 d). Note the four eigenvalues with negative real components for ABF and RSCALE are not shown.



(a) Eigenvalue distribution with the ABF preconditioner

(b) Eigenvalue distribution with the RSCALE preconditioner

(c) Eigenvalue distribution with the DRS preconditioner

(d) Histogram of eigenvalues for the pressure matrix

Figure 4.18: Eigenvalue distributions for the pressure matrix determined for time step 534, Newton cycle 2 from case CPUTEST.

Again, the ABF preconditioner leads to a higher concentration of eigenvalues near one. For this case the eigenspectra with ABF is not a small as the two previously examined cases. The range of the eigenspectra with RSCALE and DRS includes many more eigenvalues with sizes greater than one. The concentration of eigenvalues is spread in a similar manner to the previous SPE09 case. The main difference in eigenvalue distributions between RSCALE and DRS is the number of complex eigenvalues and the elimination of eigenvalues with negative real components. The elimination of negative eigenvalues with the DRS preconditioner from a different linear system from the CPUTEST cases was shown previously (Gries et al., 2014). Indeed, the elimination of negative eigenvalues from the pressure for the effective application of the AMG method is the primary motivation behind the design of the DRS preconditioner.

Chapter Five: **Performance**

To measure the potential acceleration of simulation when using the preconditioned combinative method comparisons are made to CMG's IMEX simulator. The IMEX simulator uses a variable degree ILU (ILU(k)) preconditioned GMRES solver referred to in this chapter as the "ILU only" method since it does not use AMG. The two solver methods differ in the application of preconditioners to the GMRES algorithm. The ILU only method uses ABF as a left preconditioner $M_L^{-1} = M_{\mathrm{ABF}}^{-1}$ and uses ILU as a right preconditioner $M_R^{-1} = M_{\mathrm{ILU}}^{-1}$. The developed combinative method uses DRS by default as a left preconditioner $M_L^{-1} = M_{\mathrm{DRS}}^{-1}$. A 2-stage combinative CPR method, with AMG as the inner pressure solver $M_1^{-1} = M_{\mathrm{AMG}}^{-1}$ and variable degree as the outer full system preconditioner $M_2^{-1} = M_{\mathrm{ILU}}^{-1}$, is used as a right preconditioner $M_R^{-1} = M_{\mathrm{CPR}}^{-1}$. The combinative method also uses the *flexible* variant of GMRES meaning the solution vectors are stored per GMRES iteration. The choice of FGMRES allows M_{CPR}^{-1} and specifically M_{AMG}^{-1} to vary for GMRES iterations that broadens the parameter options in SAMG available to study.

Previously, comparisons based on the number of iterations to convergence were made. This is not the best measure of performance since a computationally intensive method may reduce iterations but take much longer. It may be natural to consider the amount of work done using each method by for example counting the number of floating point operations. However, calculating the number of operations is tricky and prone to error. Also, not all operations have the same computational cost. The decision is therefore to consider the motivation of this study (a reduction in the time of simulation) and use running (wall clock) time as the primary metric of interest. Although only the linear solver differs in the methods compared, minimal drift in the system from, for example, convergence rates, time step size selection, and adaptive-implicit

switching can cause other portions of the simulator to take more or less clock time. Results are most often reported as a comparison of the total running time (ILU only / combinative) of simulation. Slower simulations due to variations in the combinative method result in decreased ratios since they produce longer clock times. The ILU only method is used as the numerator in the ratio meaning that values above unity show a speedup in running time when using the combinative solver.

Additionally, the number of GMRES solver iterations of the two methods is compared keeping in mind that combinative preconditioned solver iterations are more computationally expensive than ILU only preconditioned iterations. The value reported is the total number of solver iterations divided by the total number Newton cycles over the entire duration of the simulation. This is equivalent to the average number of solver iterations required for each Jacobian matrix. The accuracy of results of the methods was confirmed by comparing average reservoir pressures and cumulative-production values of oil, water, and gas at the end of the simulation.

5.1 Simulation Parameters

Unless otherwise stated the base set of parameter values for the tests below are as follows. The Jacobian matrices were constructed using the adaptive-implicit technique, generalized red-black ordering was used and the numerical parameters for IMEX were held constant for runs with both preconditioning methods. The combinative method uses the DRS preconditioner with a threshold of 0.3 and row weights based on reservoir volumes. IMPES cells are fully decoupled with ABF for the first stage solution as well as the second with row weights applied as for DRS. Rows associated with wells are included in the first stage solution and are also weighted based on reservoir volumes. A full setup stage of AMG is done for every new time step and a partial setup is done for subsequent Newton cycles. The first and second stage solutions are added with no

weighting and the combinative method is not stopped prior the termination of GMRES. The number of AMG cycles per combinative solver iteration is allowed to vary between 1 and 5 controlled by a relative residual tolerance of 0.07 on the pressure solution. AMG V-cycles are used to precondition the BICGSTAB method for the pressure solution. A single ω-Jacobi relaxation sweep with an ω of ⅔ is used as the smoother with the standard coarsening technique applied to all levels. Intel's Math Kernel Library (MKL) direct parallel solver, PARDISO (Schenk & Gärtner, 2004), was used to find the solution on the coarsest level. All automated checking of the input matrix and residuals by the SAMG software was turned off.

5.2 Parallel Effects

Most of the simulations in this thesis are performed using parallel enabled versions of the software. The number of cores (CPUs) used for each test case is kept consistent throughout and is listed in §4.2. However, in this section the effect of using parallel simulation is examined.

5.2.1 Serial Simulations

Simulations using a single core are done to find a baseline. Results for the seven test cases are shown in Figure 5.1.

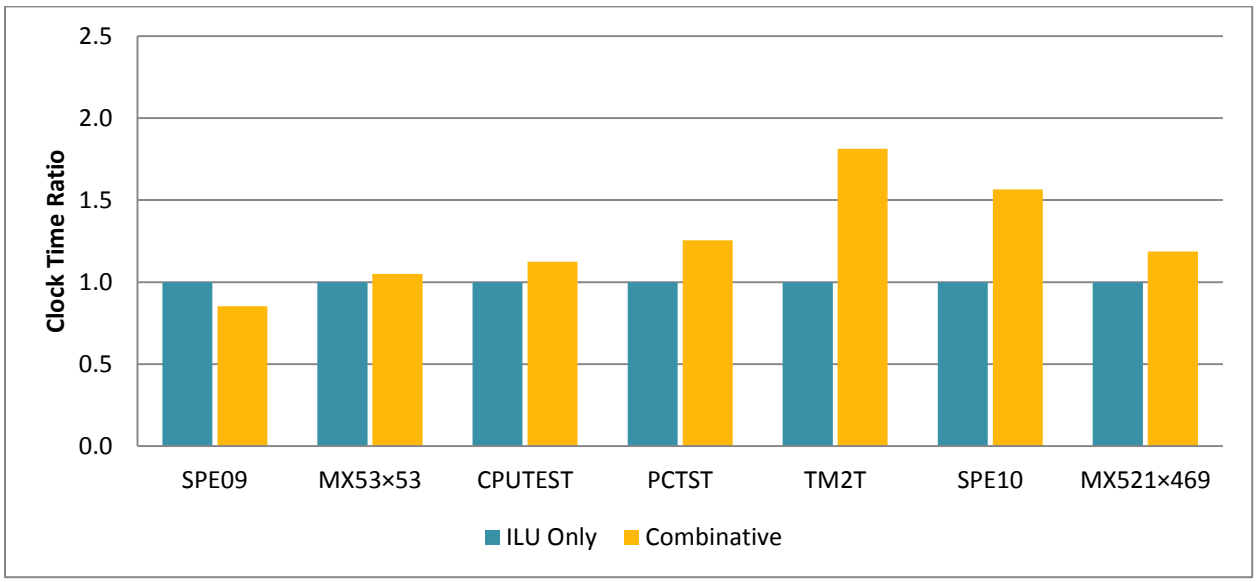


Figure 5.1: Ratio of clock time for ILU only and combinative simulations (ILU only is the numerator). The combinative simulations use the base set of combinative parameter values. All results are from serial simulations.

The results show clock time ratios larger than 1.0 in 6 of the 7 test cases when the combinative solver is used. Only for smallest case (SPE09) does the ILU only solver lead to a faster simulation time than the combinative solver.

The solver portion of the total clock times ranges between 55% and 77% of the total clock time. The solver portion is smaller for runs using the combinative solver in all cases except SPE09. The result is the ratio of solver clock times appears more favourable to the combinative solver in all cases except SPE09. The clock times and ratios are given in Table 5.2.

Case	Total Elapsed Clock-Time [s]		Clock-Time Ratio	Total Solver Clock-Time [s]		Solver Clock-Time Ratio
	ILU Only	Combinative		ILU Only	Combinative	
SPE09	10.47	12.27	0.85	6.70	8.45	0.79
MX53	128.88	122.79	1.05	81.14	74.93	1.08
CPUTEST	372.73	331.34	1.12	229.36	190.60	1.20
PCTST	828.20	660.25	1.25	524.83	365.92	1.43
TM2T	11917.62	6576.61	1.81	9167.92	3838.18	2.39
SPE10	10467.60	6685.55	1.57	7500.38	3743.69	2.00
MX521	16938.00	14285.56	1.19	10909.03	8332.89	1.31

Table 5.1: Simulation clock time comparison when using ILU only solver and the combinative solver. All results are from serial simulations.

The increase in speed when using the combinative solver comes from a reduction in the number of solver iterations required during the simulation. The average number of solver iterations used per Newton cycle is shown in Figure 5.2.

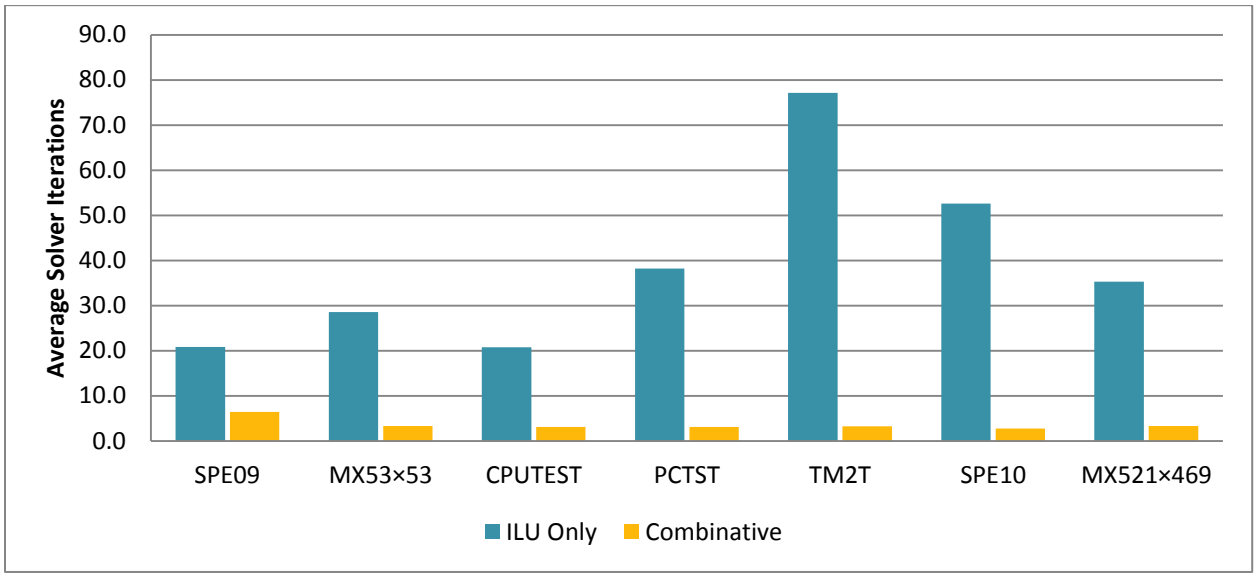


Figure 5.2: Average number of solver iterations per Newton cycle for ILU only and combinative simulations. The combinative simulations use the base set of combinative parameter values. All results are from serial simulations.

Between 20.7 and 77.1 solver iterations per Newton cycle are required when the ILU only solver method is used. This drops to between 2.8 and 6.5 solver iterations per Newton cycle when the combinative solver is used. However, solver iterations with the combinative method are more computationally expensive and there is an additional overhead cost from the AMG solver portion. Thus, the number of solver iterations must be reduced by "enough" to see an increase in speed. It is difficult to quantify how much "enough" is. When using the GMRES (or FGMRES) method the cost of each solver iteration increases with the number of solver iterations per matrix. Also, the computational cost of ILU only solver iterations does not necessarily grow at the same rate as the cost combinative solver iterations as the size of the linear system increases. In the cases above clearly the extra costs are more than compensated for in 6 of the 7 cases. The only

exception, SPE09, is the smallest case and also has the least reduction in the average number of solver iterations (20.8 to 6.5).

5.2.2 Parallel Speedup

The parallel speedup of the ILU only and combinative methods are examined in this section. This work does not consider how the algorithms are parallelized or consider the implementation of parallelism in the software. Rather, only the effect of parallelism on simulation results is examined. For each case simulations were made both using the ILU only solver and the combinative solver utilizing 1 to the full 24 cores available.

Clock time and not CPU time is used because CPU time also counts the time a parallel thread is idle waiting for the slowest thread. The parallel speedup of the three largest of the previous test cases is considered. Parallel speedup is measured by $N_0 \times T_{N_0}/T_N$ where N_0, the base number of cores, is typically 1 and T_N is the clock time when using N cores. In these results $N_0 = 2$ since the number of solver iterations for the MX521 and SPE10 cases using the ILU only method with a single core differs from the rest of the results by as much as 40%. Results are shown in Figure 5.3.

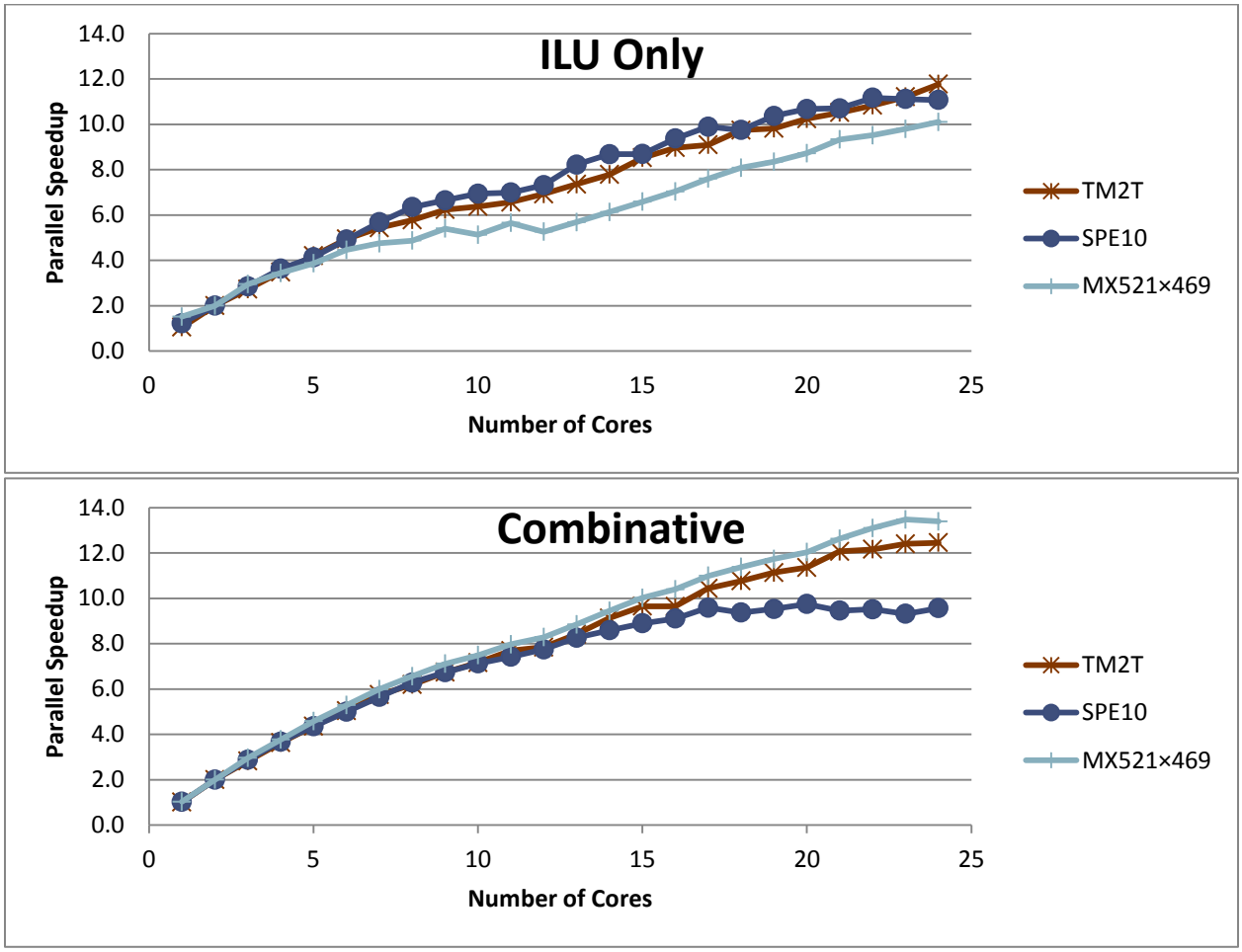


Figure 5.3: Parallel speedup from 1 to 24 cores using the ILU only preconditioner and the combinative preconditioner. Wall clock times are used to determine speedups.

The combinative parallel speedup outperforms the ILU only speedup in each of the test cases for the first 12 cores. For more than 12 cores the speedup of the SPE10 case tails off with the combinative method whereas the other cases remain slightly ahead of the ILU only method. Since the speedup in each case trends the same with both preconditioners the measure of relative clock time will remain relatively consistent. In the case of SPE10 the clock time ratio is less favourable for the combinative method for more than 12 cores when the parallel speedup begins to tail off. For TM2T the clock time ratios are 1.9, 2.1, and 2.0 for 2, 12, and 24 cores used. For SPE10 they are 1.9, 2.0, and 1.6 and for MX521 they are 1.7, 2.7, and 2.3.

5.2.3 Default Parallel Solver Results

The performance results of both the ILU only method and the Combinative method using the default number of threads listed in §4.2 for each test case are shown in Figure 5.4.

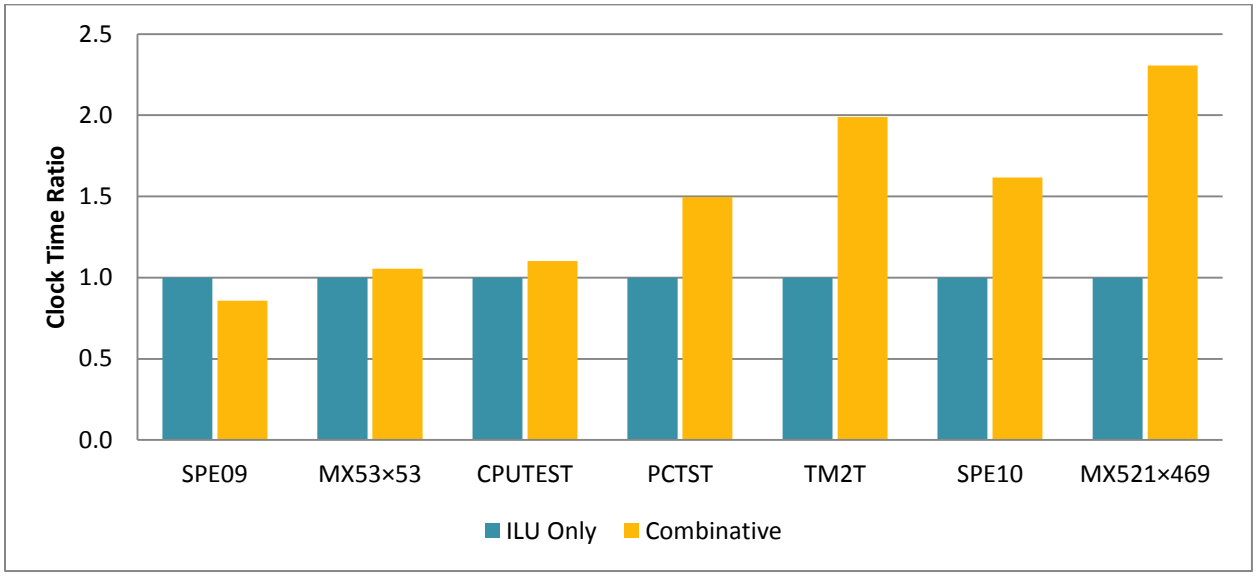


Figure 5.4: Ratio of clock time for ILU only and combinative simulations (ILU only is the numerator). The combinative simulations use the base set of combinative parameter values.

The results show clock time ratios larger than 1.0 in 6 of the 7 test cases when the combinative solver is used. Only for smallest case (SPE09) does the ILU only solver lead to a faster simulation time than the combinative solver. In the two largest cases (TM2T and MX521) the combinative solver leads to simulation times that are at least twice as fast.

The solver portion of the total clock times ranges between 59% and 86% of the total clock time. The solver portion is smaller for runs using the combinative solver in all cases except SPE09. The result is the ratio of solver clock times appears more favourable to the combinative solver in all cases except SPE09. The clock times and ratios are given in Table 5.2.

Case	Total Elapsed Clock-Time [s]		Clock-Time Ratio	Total Solver Clock-Time [s]		Solver Clock-Time Ratio
	ILU Only	Combinative		ILU Only	Combinative	
SPE09	10.0	11.7	0.86	6.5	8.2	0.79
MX53	128.4	121.8	1.05	81.5	74.7	1.09
CPUTEST	82.0	74.4	1.10	51.0	43.7	1.17
PCTST	130.9	87.5	1.50	94.1	50.6	1.86
TM2T	1060.0	533.1	1.99	872.0	344.9	2.53
SPE10	1159.3	717.5	1.62	928.7	490.8	1.89
MX521	2539.1	1101.1	2.31	2171.5	732.1	2.97

Table 5.2: Simulation clock time comparison when using ILU only solver and the combinative solver.

The results of these parallel simulations match those of the serial simulations shown in the previous section. The increase in speed when using the combinative solver comes from a reduction in the number of solver iterations required during the simulation. The average number of solver iterations used per Newton cycle is shown in Figure 5.5.

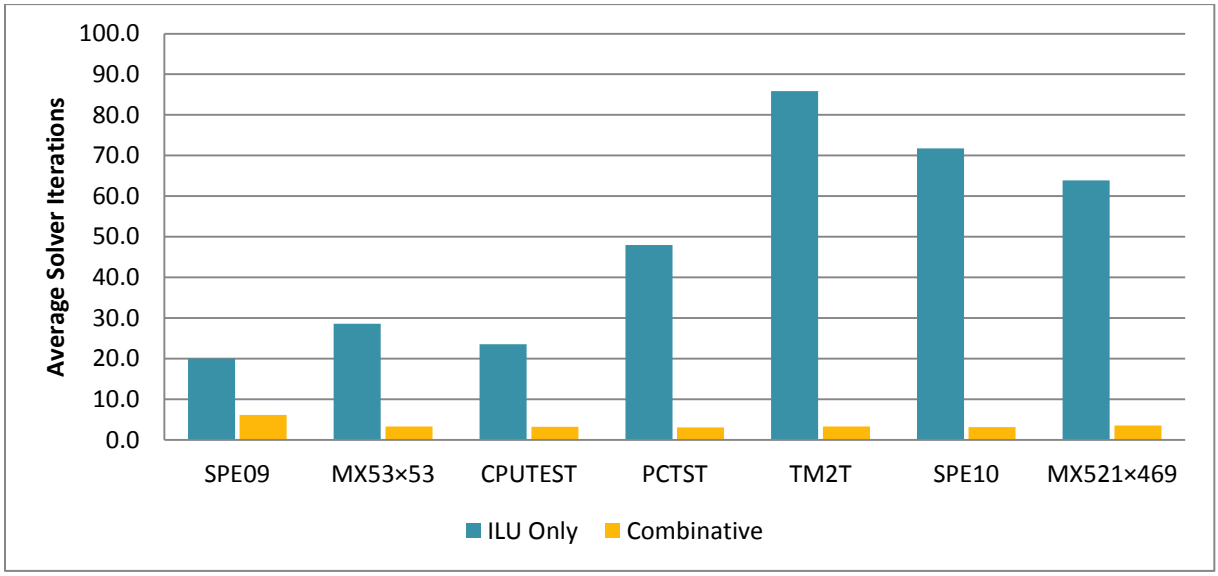


Figure 5.5: Average number of solver iterations per Newton cycle for ILU only and combinative simulations. The combinative simulations use the base set of combinative parameter values.

Between 20.0 and 85.9 solver iterations per Newton cycle are required when the ILU only solver method is used. This drops to between 3.1 and 6.1 solver iterations per Newton cycle when the combinative solver is used. However, solver iterations with the combinative method are more computationally expensive and there is an additional overhead cost from the AMG solver

portion. Thus, the number of solver iterations must be reduced by “enough” to see an increase in speed. It is difficult to quantify how much “enough” is. When using the GMRES (or FGMRES) method the cost of each solver iteration increases with the number of solver iterations per matrix. Also, the computational cost of ILU only solver iterations does not necessarily grow at the same rate as the cost combinative solver iterations as the size of the linear system increases. In the cases above clearly the extra costs are more than compensated for in 6 of the 7 cases. The only exception, SPE09, is the smallest case and also has the least reduction in the average number of solver iterations (20.0 to 6.1).

5.3 Combinative Solver Variations

A parameter study was done with the combinative solver to determine reasonable defaults of the many included parameters. These parameters are for the AMG method, the full system ILU based solver, the newly implemented combinative solver, and the newly developed preconditioner. For each variation of parameter all others are kept consistent unless otherwise stated.

5.3.1 AMG Variations

The SAMG implementation of AMG includes many optional parameters to alter the AMG method. Unless otherwise stated SAMG uses defaults for all parameters. Parameters that control the setup and solution phases of AMG are tested for the effect on the combinative solver. Particular emphasis is placed on the solution phase including the type of smoothing, the number and type of AMG cycle, and the use of Krylov acceleration for AMG.

5.3.1.1 Number of AMG Cycles

The effect of the number of AMG cycles per combinative iteration was tested. A maximum and minimum number of cycles for the solution phase of AMG are set through SAMG. Setting a gap

between the maximum and minimum allows the number of cycles to vary between calls to SAMG. As described in §3.1.1.1 the number of cycles is then determined by a relative residual tolerance passed to SAMG. First, the number of AMG cycles was set to absolute values from 1 to 5 giving the results in Figure 5.6.

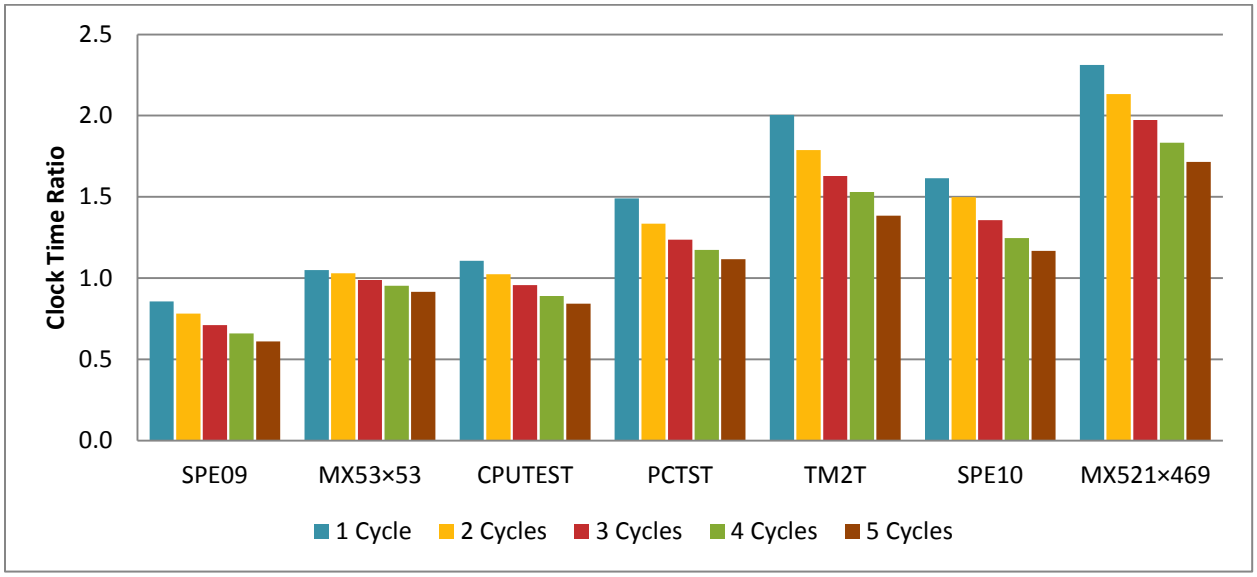


Figure 5.6: Ratio of clock time (ILU only / Combinative). The number of AMG cycles per combinative solver iteration ranges from 1 to 5.

Clearly, the combinative simulations slow down as the number of AMG cycles per combinative iteration is increased.

Next, a range cycles between 1 and 30 was allowed and the relative residual tolerance was used as the primary termination criterion. The relative residual ranged from 0.01 to 0.9. Timing results are in Figure 5.7.

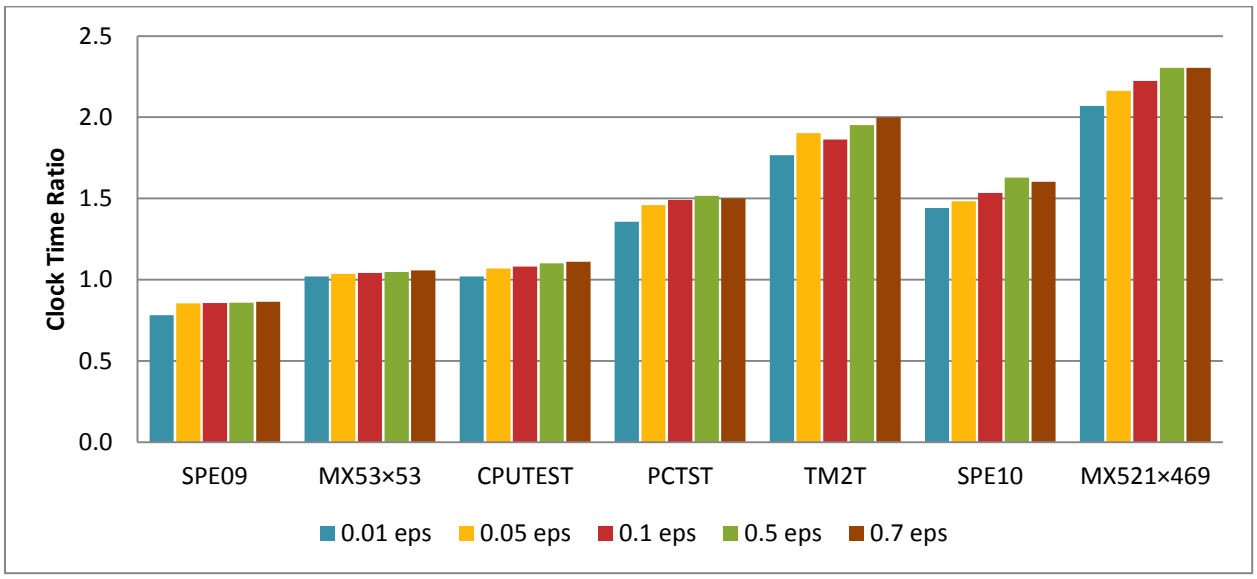


Figure 5.7: Ratio of clock time (ILU only / Combinative). The relative residual tolerance (eps) for the first stage solution ranges from 0.01 to 0.7. A maximum 30 AMG cycles per combinative iteration is enforced.

For each relative residual tolerance and test case the resulting average number of AMG cycles per combinative iteration is given in table Figure 5.8.

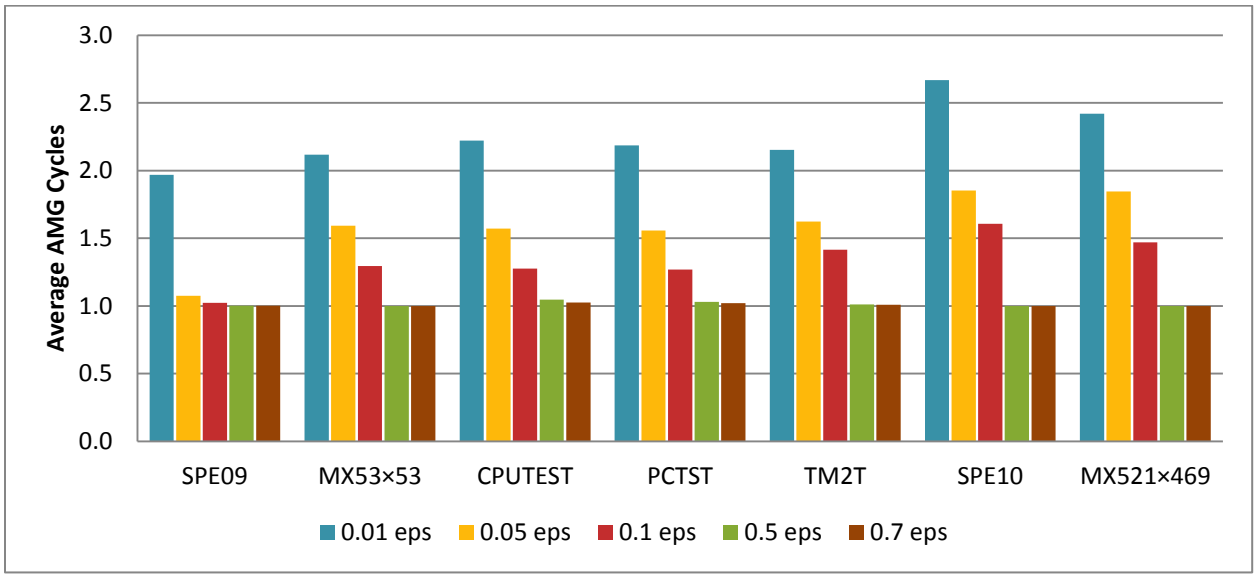


Figure 5.8: Average number of AMG cycles per combinative iteration. The relative residual tolerance (eps) for the first stage solution ranges from 0.01 to 0.7. A maximum 30 AMG cycles per combinative iteration is enforced.

These results show that as the relative residual tolerance is loosened combinative simulations generally run faster. This appears to be a result of using fewer AMG cycles per combinative

solver iteration. For a loose enough relative residual tolerance (0.5 in the cases tested) the average number of AMG cycles per combinative iteration approaches a single AMG cycle. Even looser tolerances do not affect the result because since only the minimum number of AMG cycles is used.

It is important to note that for all of these test cases a relative residual tolerance of 0.5 is reached within the maximum number of AMG cycles. It is possible that for more poorly conditioned systems that AMG may converge more slowly or even diverge. For that reason the choice to use the relative residual tolerance the primary AMG termination criteria with a small maximum number of AMG cycles is made. In particular, given these results the default values for tests of other parameters is to use a relative residual tolerance of 0.7 and allow a range of AMG cycles between 1 and 5. A maximum of 5 AMG cycles per combinative iteration functions primarily as a safety measure to cutoff any pressure stage solutions that are taking too long to converge.

5.3.1.2 AMG Cycle Type

The effect of the type of AMG cycle is examined. Three types of cycles (V-, F- and W-cycles) described in §3.1.1.2 are used. The results are in Figure 5.9.

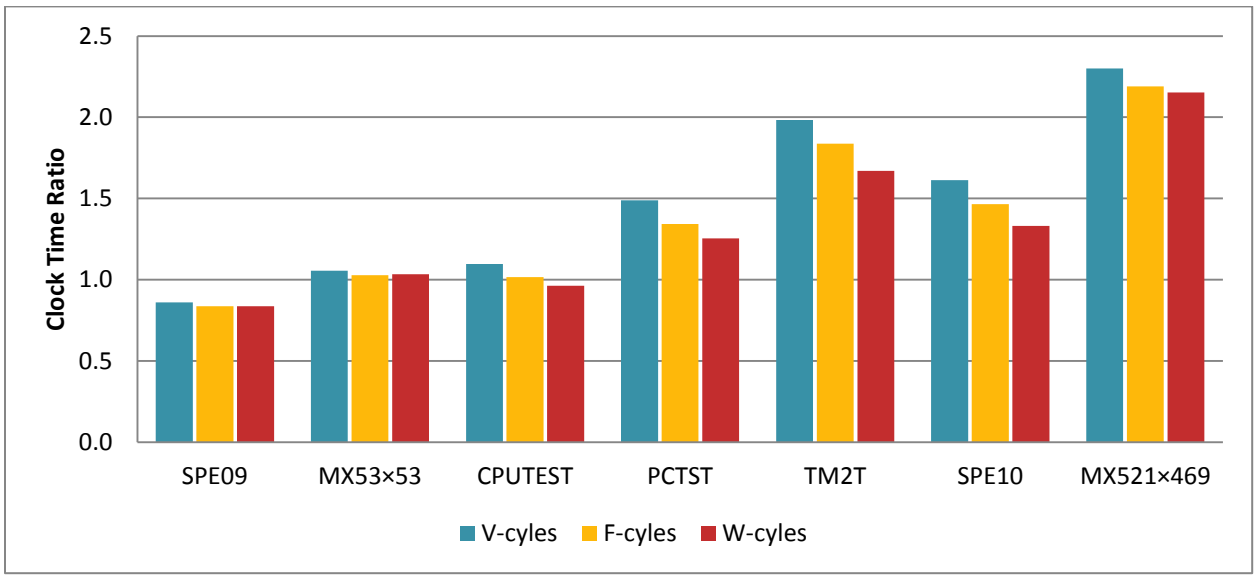


Figure 5.9: Ratio of clock time (ILU only / Combinative) using three types of AMG cycles. V-cycles, F-cycles, and W-cycles are used for the solution phase.

The results show using AMG V-cycles for the solution phase gives the fastest clock times in all 7 cases tested.

5.3.1.3 AMG Cycle Acceleration

In cases where more than one AMG cycle per combinative solver iteration is used AMG can be used as a standalone method or as preconditioner for a Kryolv method. Three options were tested: AMG preconditioned BICGSTAB, AMG preconditioned FGMRES, and standalone AMG iteration. For each of these options the number of AMG cycles per solver iteration was allowed to vary between 1 and 5 controlled by a relative residual tolerance of 0.7. The results are in Figure 5.10.

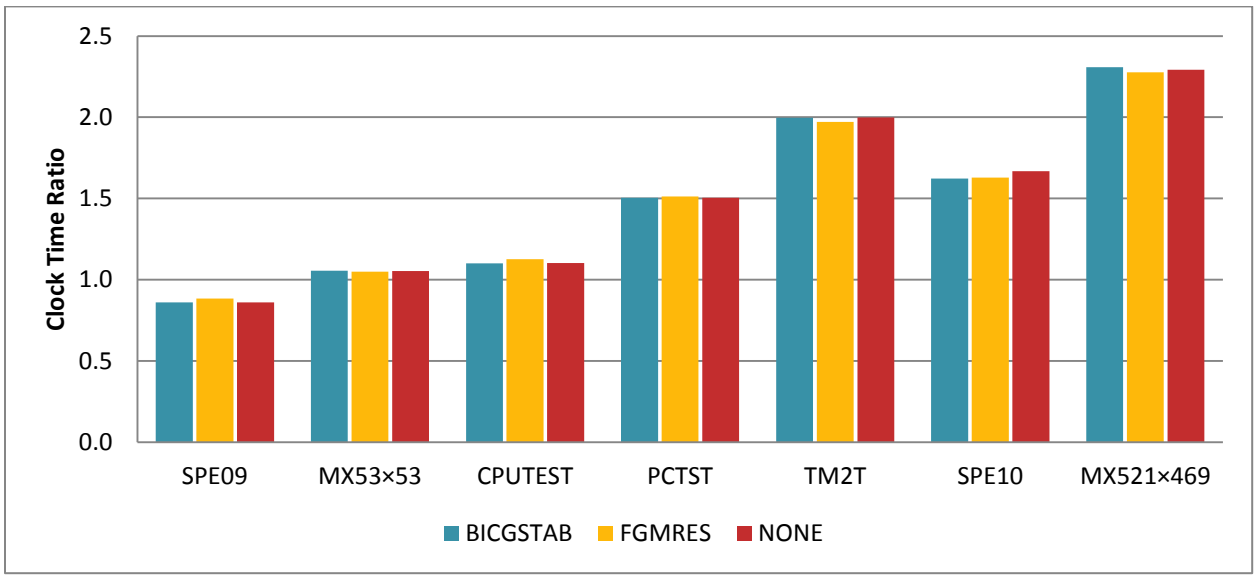


Figure 5.10: Ratio of clock time (ILU only / Combinative) using different Krylov methods for AMG cycles. BICGSTAB, FGMRES, and standalone iteration are used.

There is very little difference in clock times for each of the Krylov methods tested. This is likely the result of requiring only a small number of AMG cycles per combinative solver iteration meaning Krylov acceleration would provide little benefit. To confirm the average number of AMG cycles per combinative solver iteration is shown in Figure 5.11.

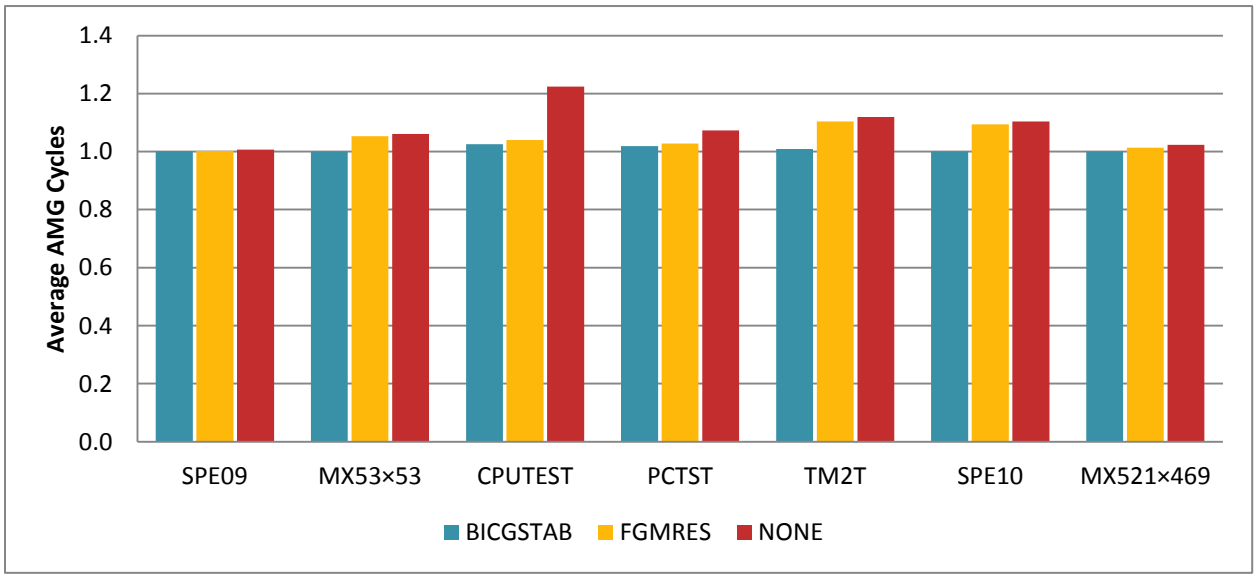


Figure 5.11: Average number of AMG cycles per combinative iteration using different Krylov methods for AMG cycles. BICGSTAB, FGMRES, and standalone iteration are used.

Clearly, for many of the cases tested only a single AMG cycle is necessary to reach a relative reduction of 0.7 in the pressure residual negating much of the benefit of Krylov acceleration. Finally, the average number of solver iterations per Newton cycle is in Figure 5.12.

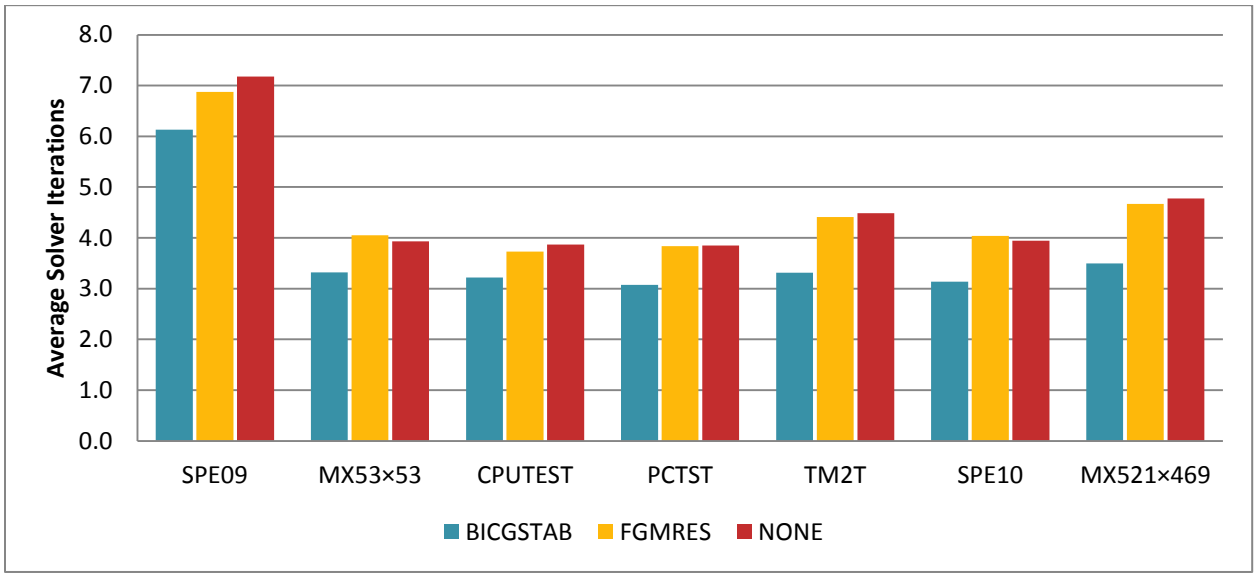


Figure 5.12: Average number of solver iterations per Newton cycle using different Krylov methods for AMG cycles. BICGSTAB, FGMRES, and standalone iteration are used.

The BICGSTAB method for AMG cycles results the fewest number of combinative solver iterations in all test cases. It appears the biorthogonalization coefficients applied on even a single AMG cycle produce the most accurate result. This effect could provide a large benefit in running time for larger systems than those tested here.

Since the average number of AMG cycles per solver iteration is so low in the simulations examined above, additional simulations were made to further explore the effect of cycle acceleration. In these additional tests both BICGSTAB and FGMRES acceleration were used. The difference from previous simulations is that the DRS inclusion parameter was set to 1.0 rather than 0.3. This means that only rows that are strictly diagonally dominant are included in the row sum. In §5.3.4.3 the effect of this parameter is examined in detail, but for this section the

important result is that a DRS inclusion parameter of 1.0 is quite poor in some cases. The results of these additional simulations are in Figure 5.13.

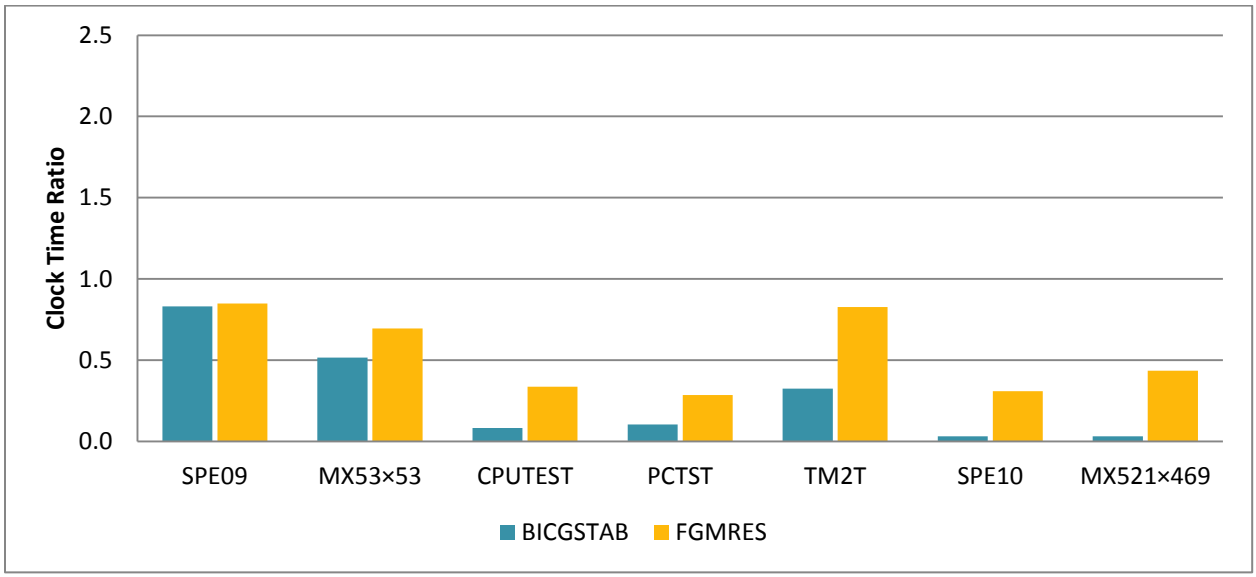


Figure 5.13: Ratio of clock time (ILU only / Combinative) using different Krylov methods for AMG cycles. BICGSTAB and FGMRES are compared in the case the DRS inclusion parameter is 1.0.

In these cases the combinative solver performs worse than ILU only. In the largest two test cases, SPE10 and MX521, the combinative solver simulations took 30 times longer than ILU only when BICGSTAB was used for AMG cycles. In each case using FGMRES for AMG cycles results in much better clock times than BICGSTAB for AMG cycles. The reason is seen by examining the average number of solver iterations per Newton cycle in Figure 5.14.

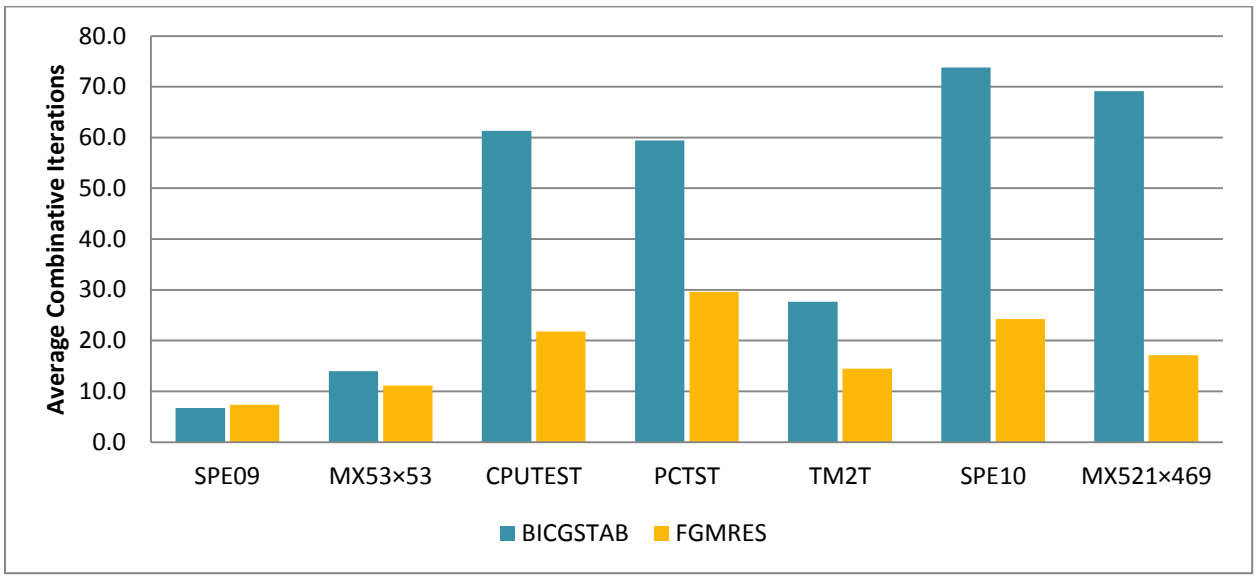


Figure 5.14: Average number of combinative solver iterations per Newton cycle using different Krylov methods for AMG cycles. BICGSTAB and FGMRES are compared in the case the DRS inclusion parameter is 1.0.

In this situation the number of combinative solver iterations per Newton cycle is much higher than when using a DRS inclusion parameter of 0.3. When using the BICGSTAB method for AMG cycles the number of iterations is more than twice as high as when using FGMRES in four of the cases.

Another effect of setting the DRS inclusion parameter to 1.0 is that combinative solver iterations are stopped prematurely in some of the cases because the size of the residual from the pressure only solution is so small. The ILU only method is then used until convergence for the linear system is reached. The *total* number of solver iterations per Newton cycle is graphed in Figure 5.15.

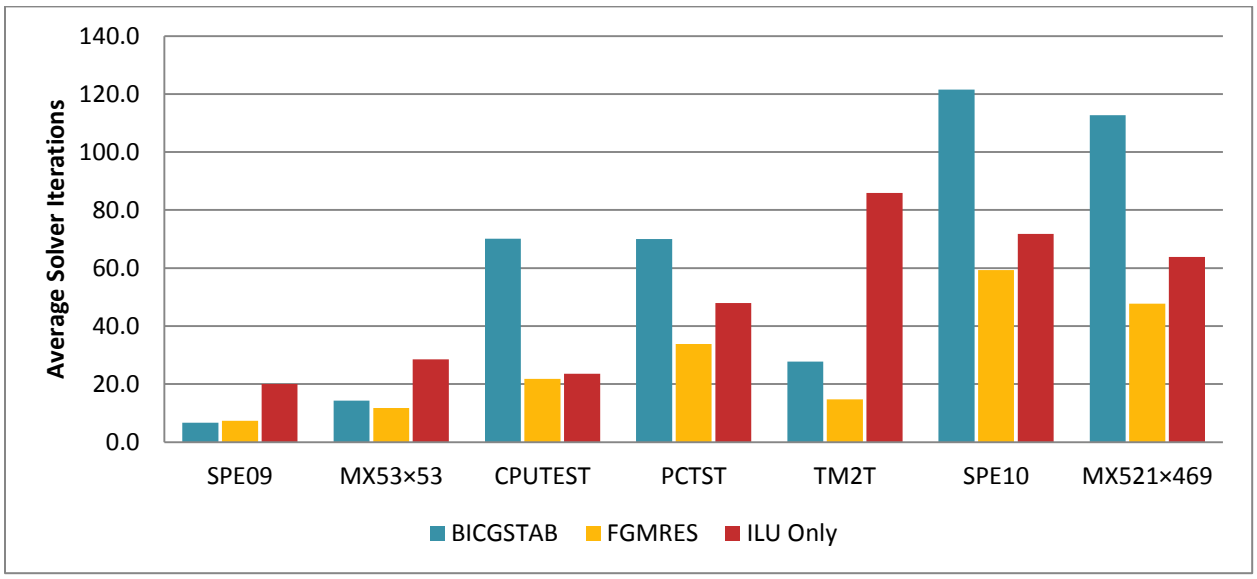


Figure 5.15: Average number of *total* solver iterations per Newton cycle using different Krylov methods for AMG cycles. BICGSTAB and FGMRES are compared in the case the DRS inclusion parameter is 1.0.

The only case where combinative iterations are not stopped prematurely is SPE09. Using FGMRES for AMG cycles leads to fewer total solver iterations per Newton cycle than using BICGSTAB. In the CPUTEST, PCTST, SPE10, and MX521 cases using BICGSTAB actually leads more solver iterations than when using the ILU only method.

The results indicate that for a small average number of combinative solver iterations BICGSTAB leads to slightly better convergence but for simulations where there is poor convergence with the combinative method FGMRES is more efficient.

5.3.1.4 AMG Smoother Type

Four smoothing variations of AMG cycles are tested. Smother types of Jacobi relaxation, Gauss-Seidel relaxation, and ILU(0) are applied at each level of the AMG V-cycle. A fourth variation was also tested in which ILU(0) is the smoother on the finest level and Jacobi relaxation is used for the remainder. The under-relaxed ω-Jacobi variant with $\omega = 2/3$ was the choice of Jacobi relaxation. The results are in Figure 5.16.

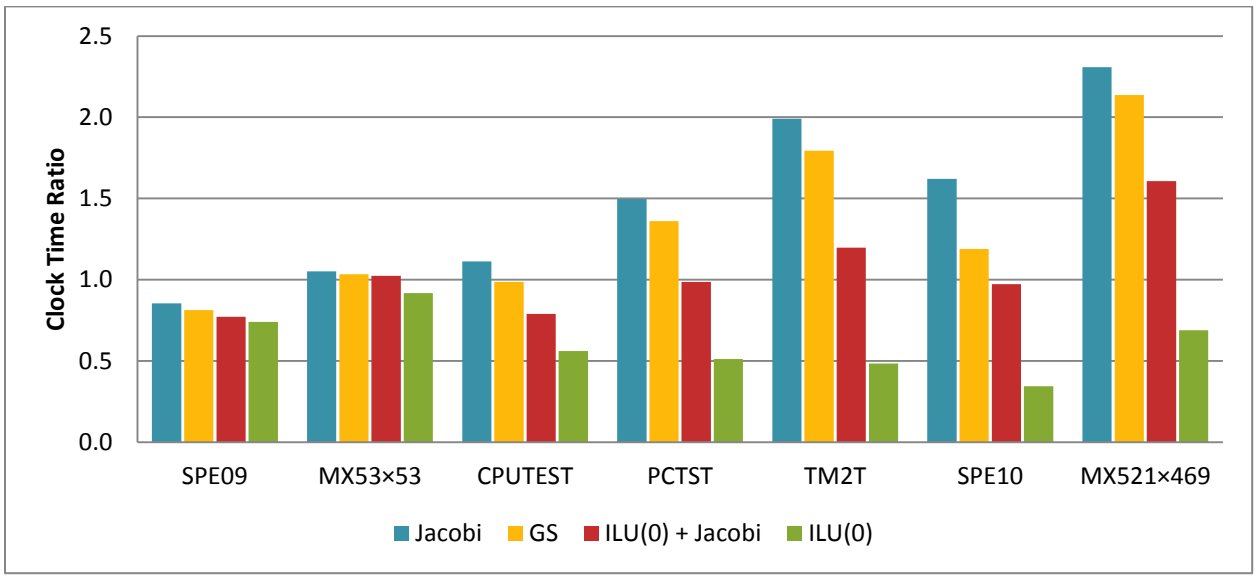


Figure 5.16: Ratio of clock time (ILU only / Combinative) for different smoother types during the AMG solution phase. Jacobi, Gauss-Seidel (GS), ILU(0) on the finest level followed by Jacobi, and ILU(0) on all levels are used as smoothers.

The results show Jacobi smoothing leads to the fastest simulations in all test cases. This is followed by Gauss-Seidel relaxation, then by ILU(0) applied on the finest level, and finally ILU(0) applied on all levels is the slowest. These results indicate the accuracy of the smoother is less important than the speed of the smoother. Jacobi relaxation is likely faster than Gauss-Seidel for parallel runs since it is an inherently parallel algorithm. It is unclear why Jacobi would be faster for serial runs.

5.3.1.5 AMG Coarsening Strategy

SAMG allows a choice in the aggressiveness of the coarsening strategy applied to the setup phase of the AMG method. A more aggressive strategy produces fewer matrix entries on the next level and therefore fewer levels in total. There are four choices of "aggressive coarsening" in addition to standard coarsening. For this study only the most aggressive and standard coarsening are tested. The developers of SAMG suggest aggressive coarsening is most effective when applied to the first level only so that is tested as well. The results are in Figure 5.17.

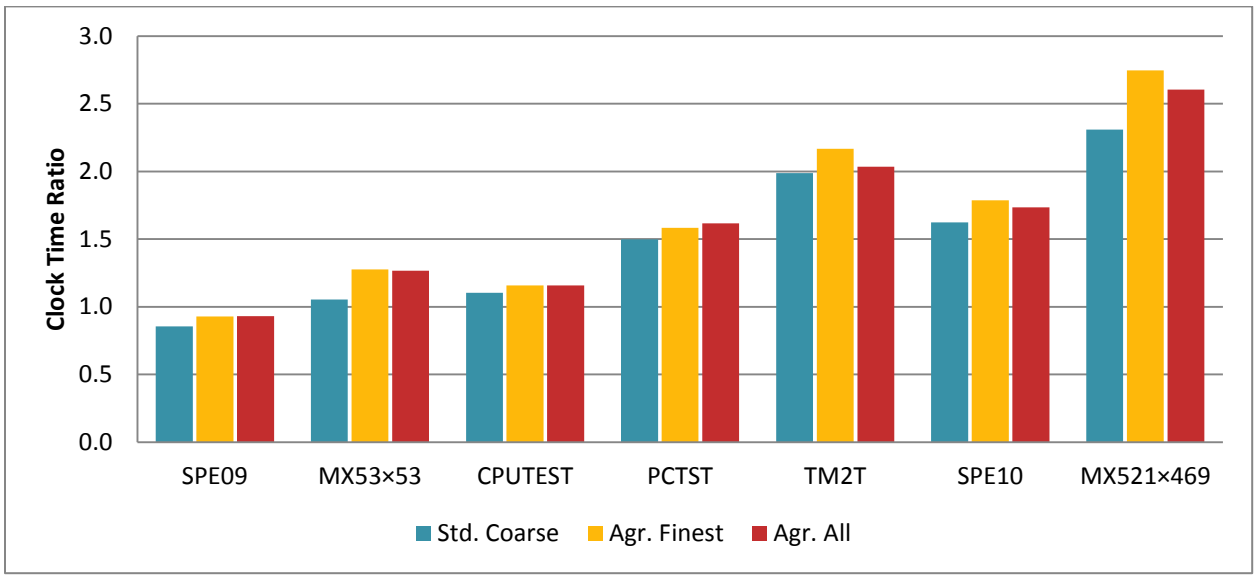


Figure 5.17: Ratio of clock time (ILU only / Combinative) for three different coarsening strategies during the AMG setup phase. The coarsening strategies tested are standard coarsening on all levels, aggressive coarsening on the finest level only, and aggressive coarsening on all levels.

The fastest results of the three coarsening strategies employed come from the suggested strategy of applying aggressive coarsening at the finest level and standard coarsening to all others. The average number of solver iterations per Newton cycle is in Figure 5.18.

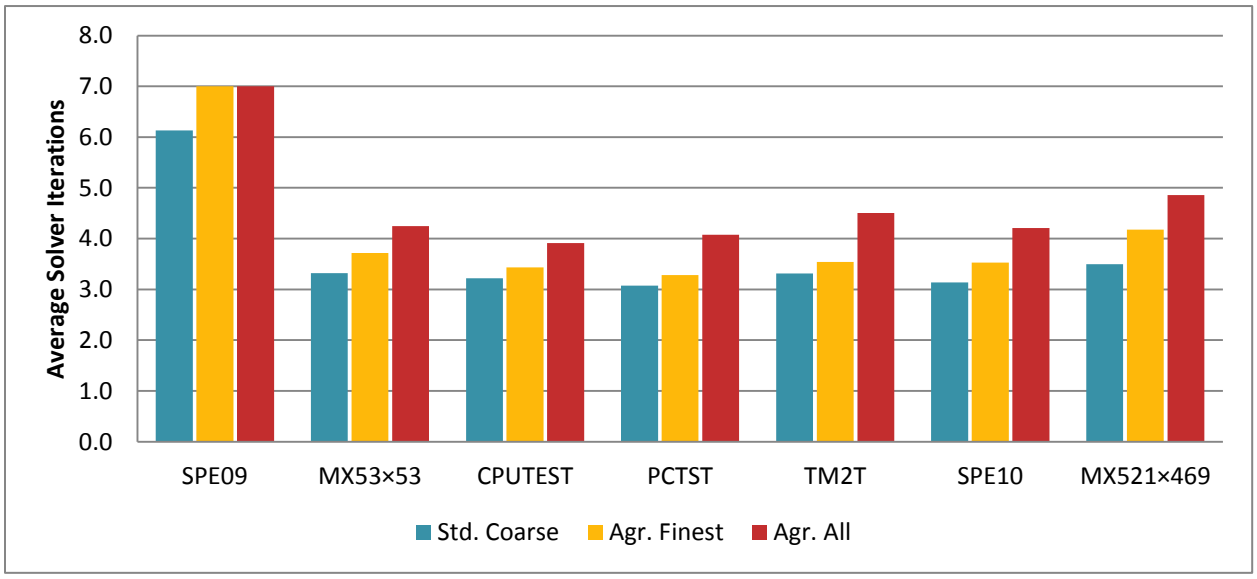


Figure 5.18: Average number of combinative solver iterations per Newton cycle for three different coarsening strategies during the AMG setup phase. The coarsening strategies tested are standard coarsening on all levels, aggressive coarsening on the finest level only, and aggressive coarsening on all levels.

Standard coarsening gives the most accurate solution leading to a reduction in the number of combinative iterations required. However, the speed gained by having fewer levels for the AMG solution phase more than makes up for the reduced number of solver iterations. Finally, the average number of AMG cycles per solver iteration is in Figure 5.19.

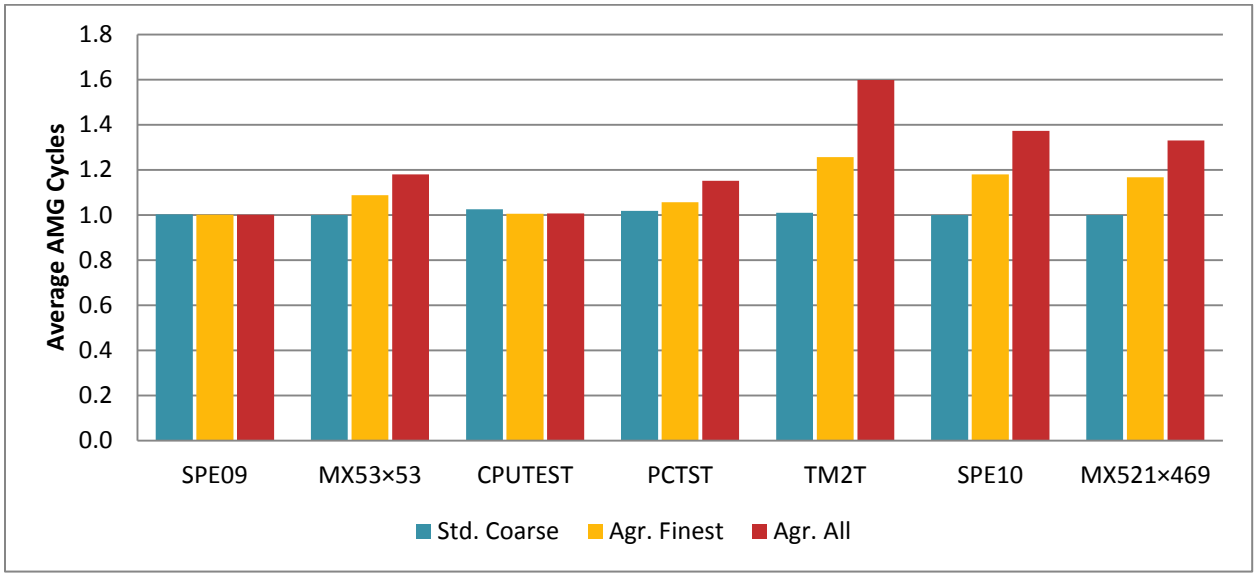


Figure 5.19: Average number of AMG cycles per combinative iteration for three different coarsening strategies during the AMG setup phase. The coarsening strategies tested are standard coarsening on all levels, aggressive coarsening on the finest level only, and aggressive coarsening on all levels.

Again it is clear that standard coarsening gives the most accurate result but the speed gained from aggressive coarsening applied to the finest level more than makes up for it.

5.3.2 Full System ILU and Ordering

The second stage of the combinative solution is a full system solution and the right preconditioner for this stage is exactly analogous to the ILU only method of IMEX used for comparisons. The ILU parameter tested for this study is the level of fill for ILU(k) for inter-class terms. See §3.1.2 for details. The default level of fill for ILU in IMEX depends on the ordering used for the system. Even though ordering of the system is correctly considered as a left preconditioner for the full system the effect of ordering is examined in this section. Ordering and

the level of fill are two of the ways to affect the convergence of the ILU only method. Three combinations for ordering and level of fill were examined. These selections differ from the default used in all other simulations of this study. The default selects red-black ordering with ILU(0) for intra-class terms and ILU(1) for inter-class terms. Obviously this distinction is only important for cases that were tested using parallel simulation.

The first selection examined in this section is nearly the same as the default except it uses ILU(0) for both intra and inter-class terms. The second and third selections use natural ordering of grid cells. In the second selection ILU(0) is used for all terms and in the third selection ILU(1) is used for all terms. In all three scenarios the variation in ordering and level of fill is only applied to simulations run with the combinative method. Although these selections could apply the ILU only method they are not, defaults are used. The advantage of using defaults for the ILU only method is that for the clock time ratio measure the numerator remains consistent and therefore a comparison of ratios shows the effect these parameters have on the combinative method. The results are in Figure 5.20.

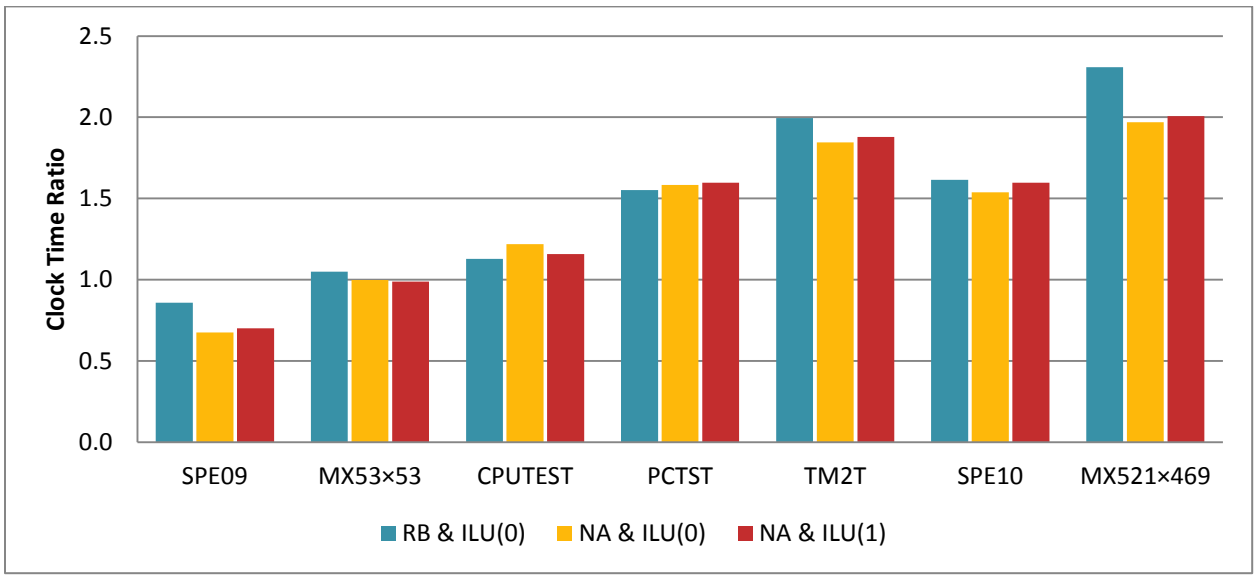


Figure 5.20: Ratio of clock time (ILU only / Combinative) for three variations of solver order and ILU degree. These variations are red-black ordering with ILU(0), natural ordering with ILU(0), and natural ordering with ILU(1). These variations are only applied to the combinative method. For the ILU only method red-black ordering with default ILU degree (1 on domain boundaries, 0 elsewhere) is used.

In five of the seven cases red-black ordering with ILU(0) was the fastest. For CPUTEST natural ordering with ILU(0) was the fastest and for PCTST natural ordering with ILU(1) was the fastest. For SPE09, TM2T, and MX521 red-black with ILU(0) was the best. In the other cases the difference in speed was marginal.

The average number of combinative solver iterations per Newton cycle is shown in Figure 5.21.

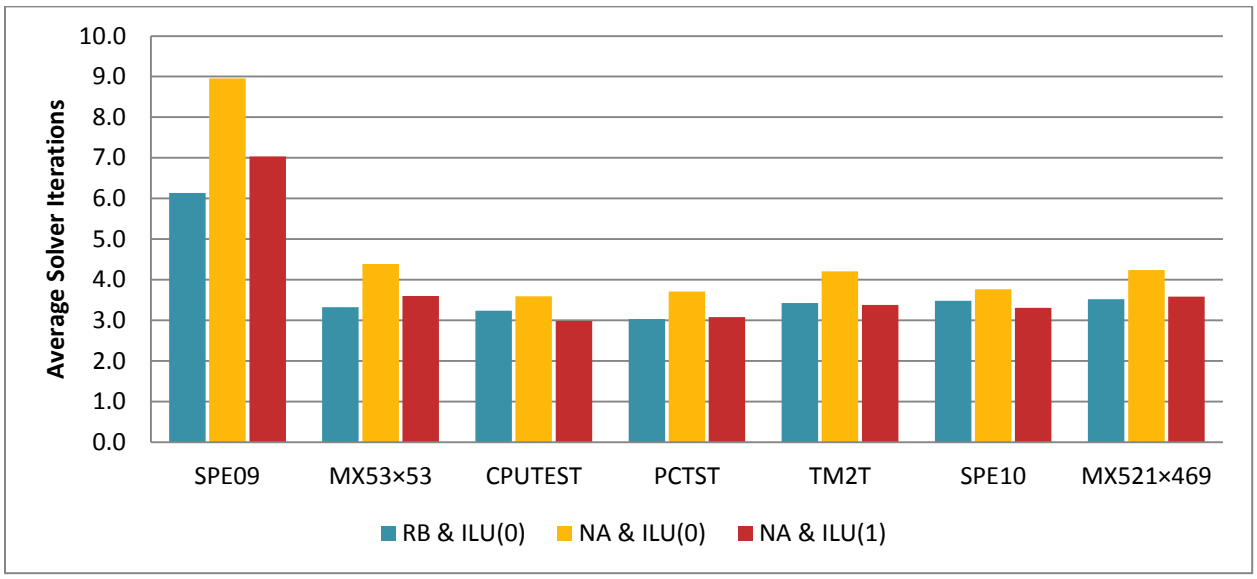


Figure 5.21: Average number of combinative solver iterations per Newton cycle for three variations of solver order and ILU degree. These variations are red-black ordering with ILU(0), natural ordering with ILU(0), and natural ordering with ILU(1). These variations are only applied to the combinative method. For the ILU only method red-black ordering with default ILU degree (1 on domain boundaries, 0 elsewhere) is used.

The results indicate that more iterations are required for natural ordering with ILU(0) as is expected since it leads to the simplest right side preconditioning matrix. The cost of performing iterations is different for each of these choices due to the effect on the sparsity pattern of the ILU preconditioner.

5.3.3 Combinative Parameters

The combinative solver has been developed to include several optional parameters. The results when altering the combinative parameters are studied below.

5.3.3.1 Setup Frequency

SAMG includes the option to perform a full setup, partial setup, on no setup phase for any call after the initial call to SAMG. An option has been added to the combinative preconditioner to change this SAMG parameter at each Newton cycle, time step, well opening / shut-in, or matrix structural change. See §3.1.3.1 for more details. For the test cases considered matrix structural

changes only happen at the initial time step. Nine combinations have been considered. In the following results each combination is given a four-letter abbreviation. The first two letters denote the frequency of performing full setups. The second two letters denote the frequency of partial setups. The two-letter abbreviations are NT for Newton cycle, TS for time step, WS for well opening / shut-in, and ST for matrix structural change. If no partial setups are performed the two letter code NO is used. For example, in the default setup scenario used elsewhere in this study full setups are done every time step with partial setups for subsequent Newton cycles giving the code TSNT. In the first set of results setup options corresponding to NTNO, TSNT, TSNO, WSNT, and STNT are examined and shown in Figure 5.22.

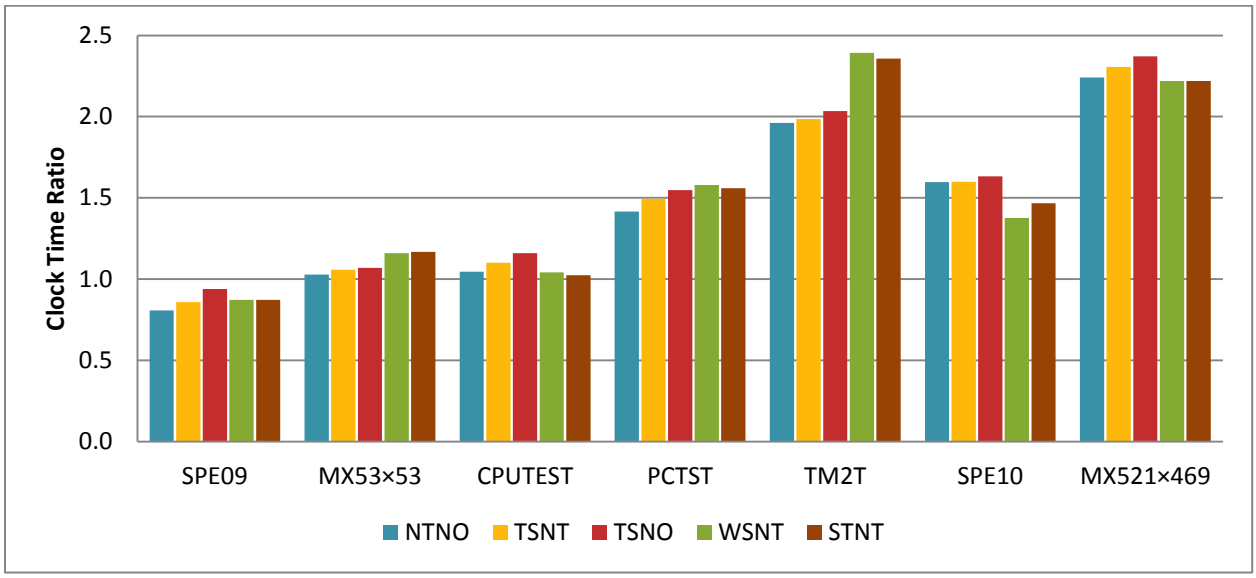


Figure 5.22: Ratio of clock time (ILU only / Combinative) for various AMG setup frequencies. Full AMG setup phases are done every Newton cycle (NT), time step (TS), well shut in / open (WS), or structural change (ST). Partial setups are done every subsequent Newton cycle (NT) or not at all (NO).

For cases SPE09, CPUTEST, SPE10, and MX521 the fastest simulations happen with TSNO, meaning full setups every time step and no partial setups on subsequent Newton cycles. For cases MX53, PCTST, and TM2T the fastest simulations happen with WSNT meaning full setups

every time a well is opened or shut and partial setups for subsequent time steps and Newton cycles.

The average number of solver iterations per Newton cycle for these parameter variations is shown in Figure 5.23.

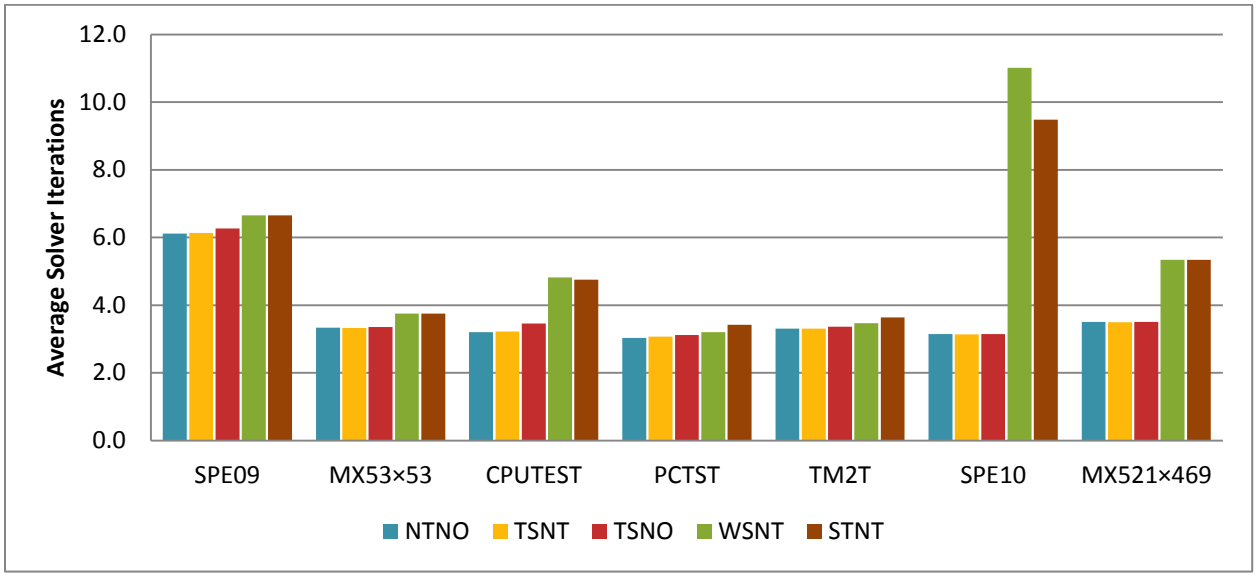


Figure 5.23: Average number of solver iterations per Newton cycle for various AMG setup frequencies. Full AMG setup phases are done every Newton cycle (NT), time step (TS), well shut in / open (WS), or structural change (ST). Partial setups are done every subsequent Newton cycle (NT) or not at all (NO).

The fewest number of solver iterations per Newton cycle corresponds to performing full setups every Newton cycle or full setups every time step and partial setups every Newton cycle. This means TSNT is likely preferable in most cases to NTNO since it is computationally less expensive with no obvious numerical downside. Doing no setups on subsequent Newton cycles (TSNO) leads to nearly the same number of average solver iterations per Newton cycle. This is likely why this parameter variation leads to the best solution times. For cases SPE09, MX53, PCTST, and TM2T the well shut and structure choice for full setup frequency leads to only a few more solver iterations per Newton cycle (6%, 12%, 3%, and 3% more respectively). This marginal increase is likely the reason the WSNT choice performs well in these cases. The cost of

extra iterations is made up for by the gain in the reduced cost of performing full setups. This is not the case for SPE09 though, in which TSNO is the best choice.

In the second set of results setup options corresponding to WSNT, WSTS, WSNO, STNT, STTS, and STNO are examined (Figure 5.24).

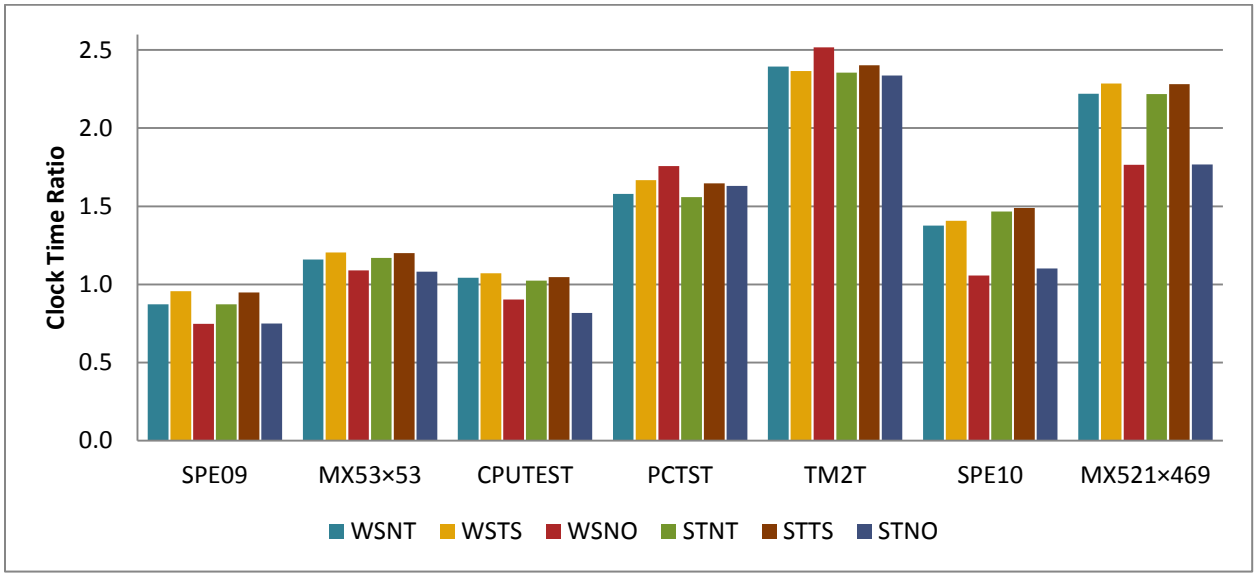


Figure 5.24: Ratio of clock time (ILU only / Combinative) for various AMG setup frequencies. Full AMG setup phases are done every well shut in / open (WS) or structural change (ST). Partial setups are done every subsequent Newton cycle (NT), time step (TS), or not at all (NO).

There is little difference in performance if full setups are done every well shut in and open or ever structural change. In the CPUTEST, PCTST, and TM2T cases the WS option is a little better. In all cases except for PCTST and TM2T simulation times are faster when partial setups are done at least every time step. In the PCTST and TM2T cases doing full setups at every well shut in / open with no additional setups results in the best performance, even better than those in Figure 5.22.

The total number of solver iterations per Newton cycle is shown in Figure 5.25. In some of these cases combinative iteration is prematurely stopped because the residual of the pressure solve is very small.

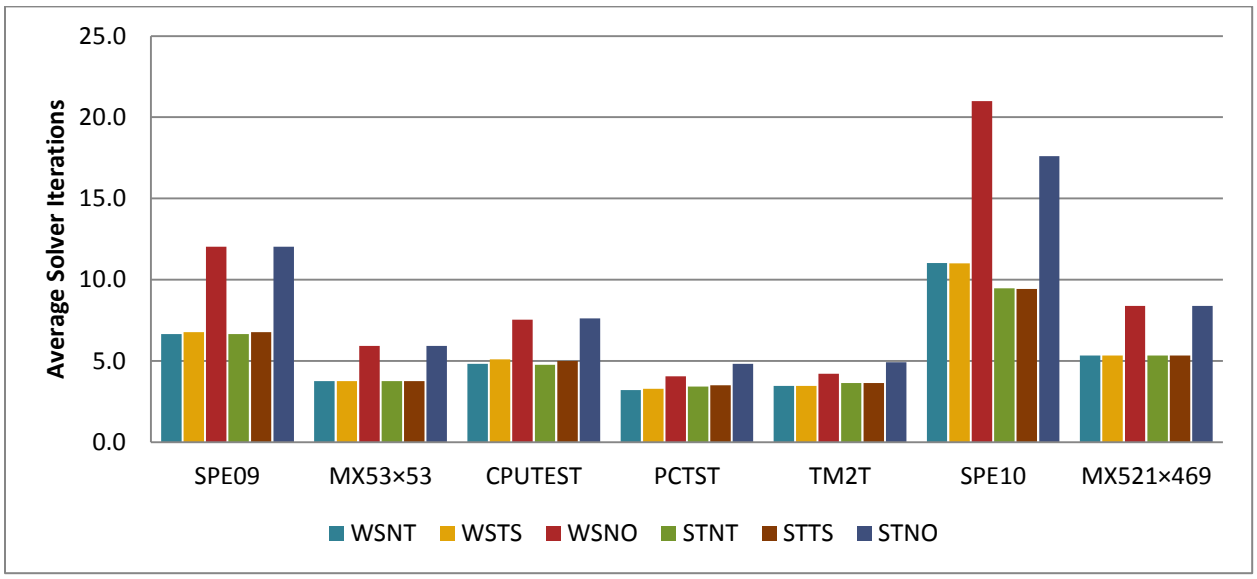


Figure 5.25: Average number of solver iterations per Newton cycle for various AMG setup frequencies. Full AMG setup phases are done every well shut in / open (WS) or structural change (ST). Partial setups are done every subsequent Newton cycle (NT), time step (TS), or not at all (NO).

It is clear that the number of solver iterations increases significantly if there are not at least partial setups done at least every time step. The only cases with a small increase in the average number of solver iterations when not doing at least some setup every time step are PCTST and TM2T. These are also the cases that perform best with the well variation and have the most time steps with wells shutting in or opening.

The average number of AMG cycles per combinative solver iteration is shown in Figure 5.26.

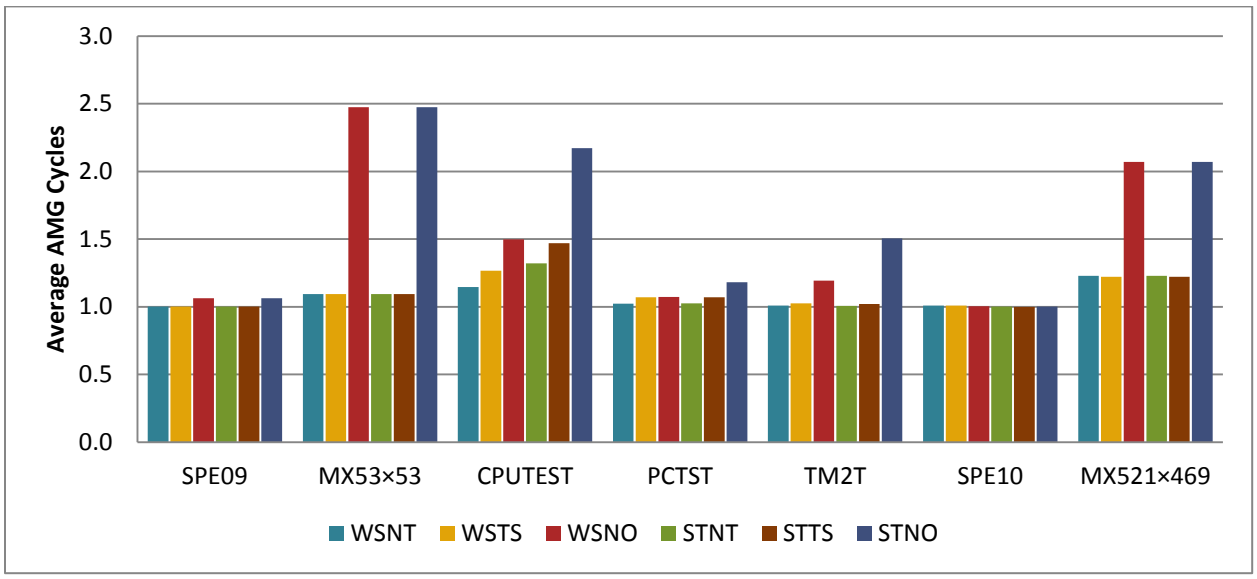


Figure 5.26: Average number of AMG cycles per combinative iteration for various AMG setup frequencies. Full AMG setup phases are done every well shut in / open (WS) or structural change (ST). Partial setups are done every subsequent Newton cycle (NT), time step (TS), or not at all (NO).

The number of AMG cycles required to reach the loose tolerance of the first stage of the combinative method is nearly one as long as at least partial setups are done at least every time step. When no partial setups are done some of the cases take more AMG cycles but the average remains less than 2.5 for the test cases studied.

5.3.3.2 Well Inclusion and Weight

The equations modelling source terms from wells (2.12) are of a different type than those modelling the reservoir. These equations provide a link between the bottom-hole pressure of the wells and the amount of fluid flow into or out of attached grid blocks. The primary variable of well equations is bottom-hole pressure. Since these equations are of a different type it is interesting to see the effect of AMG. An option has been developed to simply exclude well equations and well source terms in reservoir equations from the first stage of the combinative solution. Essentially the effects of wells are only incorporated at the full stage solve just as the effect of saturation is. This means AMG is only used to solve for reservoir pressures resulting

from discretized conservation equations (2.6)-(2.8). The results when excluding wells from the first stage are shown in Figure 5.27.

Another option includes well rows but applies a row weight. Well rows can be weighted based on the total flow of reservoir fluid calculated for each Newton cycle. The magnitude of each well row weight is a summation of the source terms from all reservoir grid blocks connected to the well. In the cases examined reservoir volume weighting was applied to reservoir rows so that the well weight is the total volume of fluid flow at reservoir conditions. If surface volume or mass weighting is applied to the reservoir the well weighting follows. Results of weighting well rows for the matrix are shown in Figure 5.27.

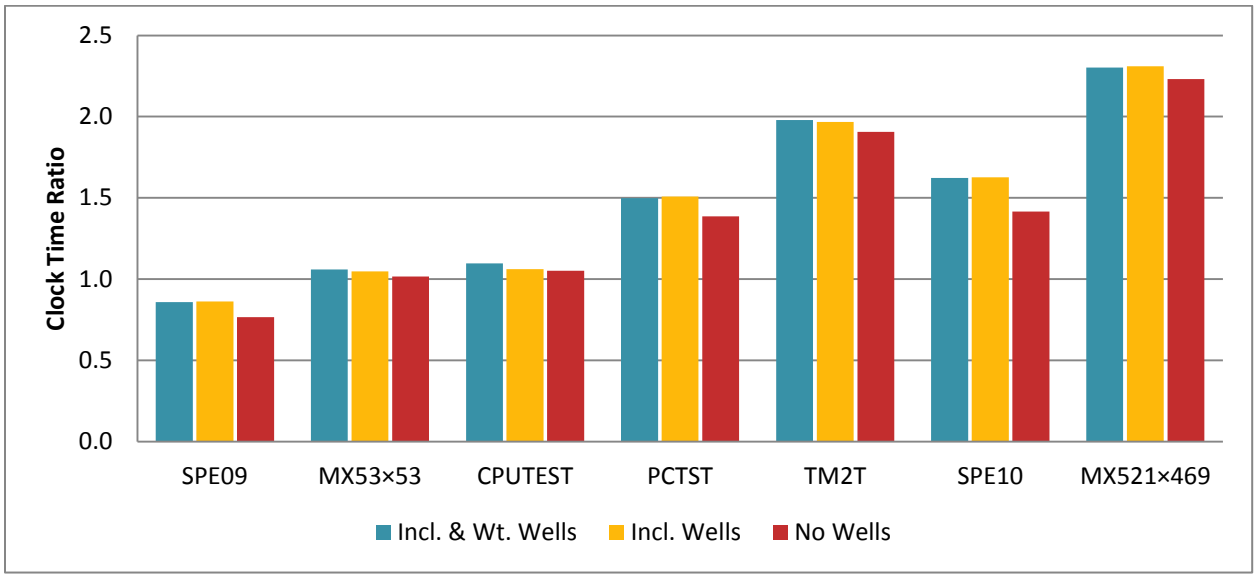


Figure 5.27: Ratio of clock time (ILU only / Combinative) for various well related options in the combinative solver. The options are to include well equations in the first stage solution with and without row weights applied and to exclude well equations from the first stage solution.

The results show the fastest simulations include the wells in the first stage and weight them. Not using weighting of well-rows marginally slows down runtimes and interestingly excluding the wells altogether from the first stage still gives speedups compared to the ILU only simulations.

The average number of solver iterations per Newton cycle is shown in Figure 5.28.

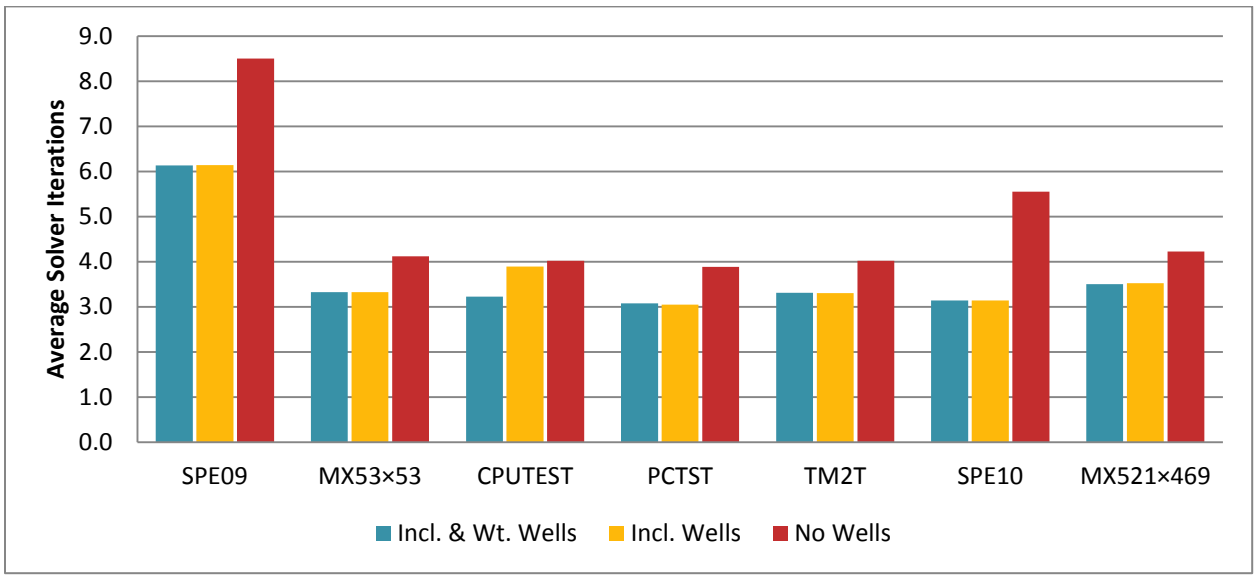


Figure 5.28: Average number of combinative solver iterations per Newton cycle for various well related options in the combinative solver. The options are to include well equations in the first stage solution with and without row weights applied and to exclude well equations from the first stage solution.

The results show that excluding wells from the first stage of the combinative solver leads to a clear increase in the number of solver iterations required. The effect on the running time becomes more pronounced for larger data sets and bigger increases in solver iterations. Weighting well rows produces nearly the same number of combinative solver iterations in all cases. This indicates that well weighting is not very important to the combinative preconditioner.

5.3.3.3 Combinative Weights

The first and second stage of the combinative method are typically added together without modification. Applying scalar weights to the two stages is possible. The right hand side, or update vector, for the second stage is constructed using the solution from the first stage. It is possible to weight the update vector. These weights could be constant or varied per combinative iteration. In this study varied weights are used.

The first combinative weighting option tested applies a weight to both the update vector and the combinative sum of solutions. The chosen weights minimize the residual of the full system with

the pressure only solution. This differs from the first stage of the combinative method, which minimizes the residual of the pressure only system. Essentially, a scalar weight is applied to each pressure only solution after it is determined.

Applying the above weight to the update vector only makes the new update vector $\boldsymbol{v}'$ perpendicular to the product of the full system matrix and the pressure only solution.

The other weight applied to the update vector in this study makes it perpendicular to the original right hand side of the combinative iteration. The idea behind this is to construct a two vector minimization of the combined pressure residual.

Two different weightings of the combined sum of stages are possible. The first is a simple sum with no weight. The second is a weight that minimizes the residual of the combinative solver iteration. More details of the combinative weighting options are in §3.1.3.3.

Performance results of 6 choices of solution and update vector weights along with the base case with no weighting are shown in Figure 5.29.

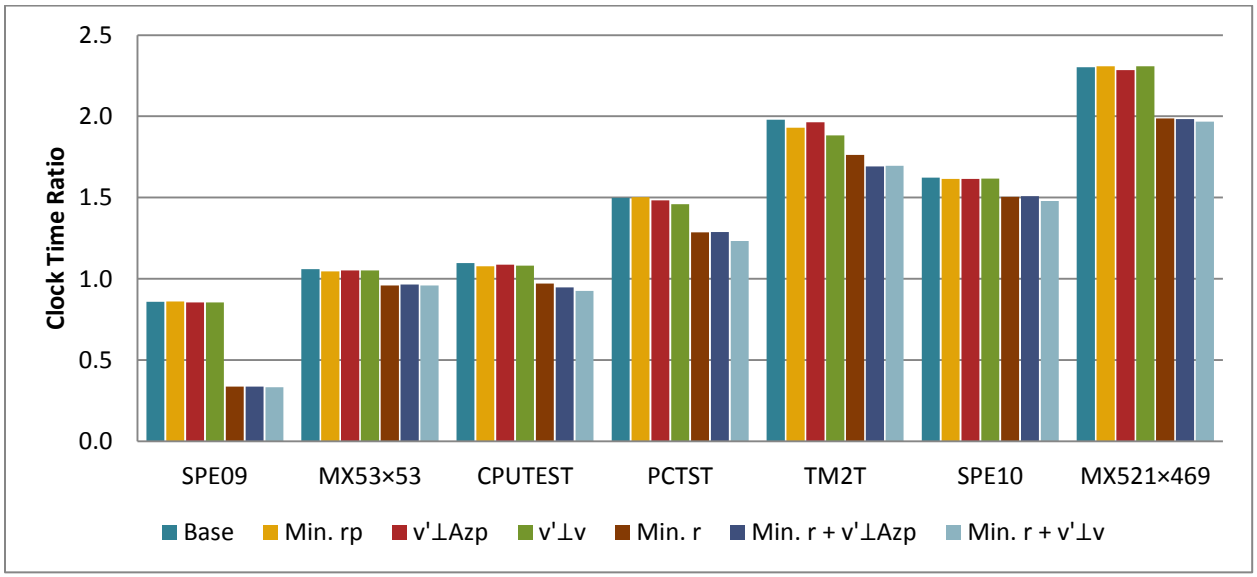


Figure 5.29: Ratio of clock time (ILU only / Combinative) varying the weight of the combinative solution stages. There is either no weight applied or the weights minimize the pressure residual r_p or minimize the combined residual r. The second stage right hand side

v' is either orthogonal to the original right hand side v, orthogonal to the pressure solution product Az_p, or not modified.

The variation in update vector weights shows very little difference in solution times. The base case with no weighting performs the best and minimizing the full system pressure residual is the second best overall. Applying a weight to minimize the total combined residual of each combinative iteration leads to worse performance in all the cases.

The average number of solver iterations per Newton cycle is shown in Figure 5.30.

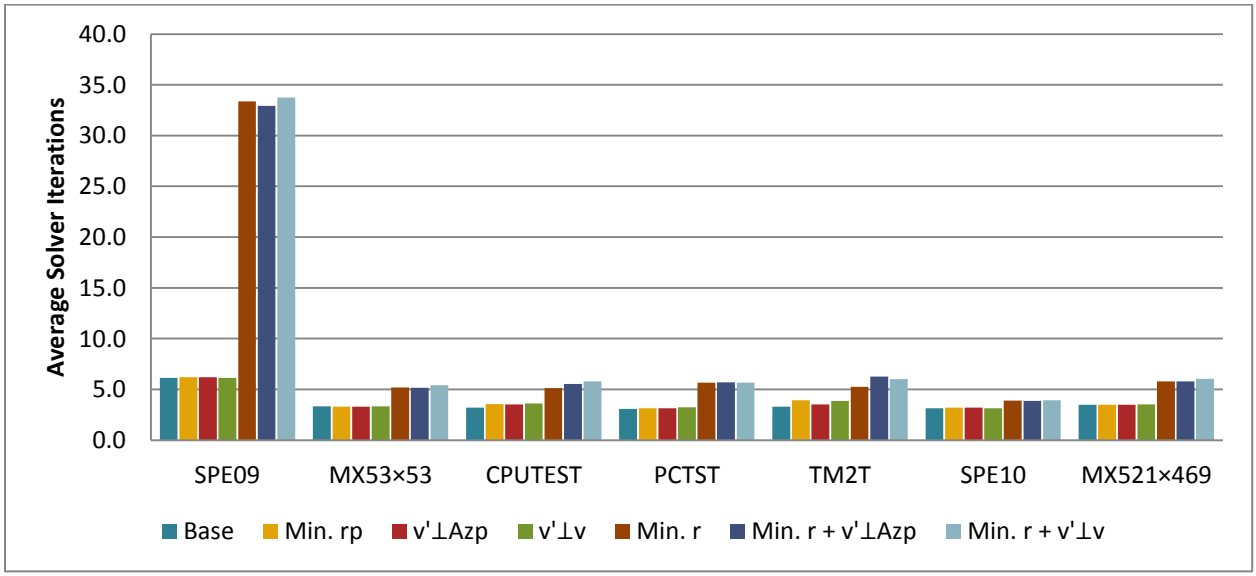


Figure 5.30: Average number of solver iterations per Newton cycle varying the weight of the combinative solution stages. There is either no weight applied or the weights minimize the pressure residual r_p or minimize the combined residual r. The second stage right hand side v' is either orthogonal to the original right hand side v, orthogonal to the pressure solution product Az_p, or not modified.

The number of solver iterations increases when a minimization weights are applied to the combined sum of solutions. These results indicate that using the combinative method without weights is the best choice.

5.3.3.4 Stopping Criteria

Normally the combinative solver uses the 2-stage combinative preconditioner for each FGMRES iteration. However, since the flexible variant of GMRES acceleration is used it is not necessary

to maintain the same preconditioner for each iteration. Therefore the possibility exists to discontinue using the combinative preconditioner and proceed using only the single-stage ILU only preconditioner of IMEX at any point during the solution of a Newton cycle. If the solution for pressure is already quite good and additional solver iterations only serve to improve saturation variables this type of approach may make sense.

Five premature stopping criteria for the combinative solver have been developed (see §3.1.3.4). The first stopping criterion is a check on the size of the residual of the pressure solution. If the magnitude of residual is small, i.e. $\|\boldsymbol{r}_p\| < 10^{-16}$, then combinative preconditioning is stopped. The second combinative stopping criterion is a test of the number of combinative iterations per Newton cycle. Once the number of combinative iterations reaches a maximum for a given Newton cycle combinative preconditioning is stopped. Various maximums were tested producing the results shown in Figure 5.31.

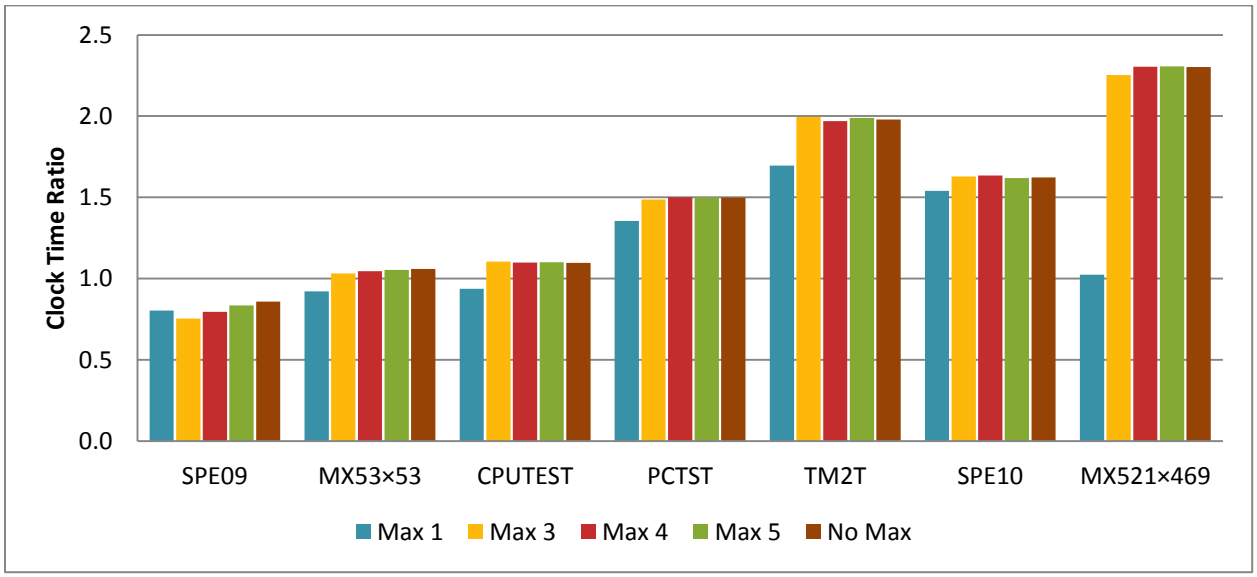


Figure 5.31: Ratio of clock time (ILU only / Combinative) for simulations using a maximum number of combinative solver iterations per Newton cycle.

The results show setting a maximum of just a single combinative iteration per Newton cycle leads to the worst performance in all cases except SPE09. The best performance for all cases

occurs when no maximum is set. This is likely because so few solver iterations per Newton cycle are needed in these cases when no maximum is set. The number of *total* solver iterations per Newton cycle is shown in Figure 5.32.

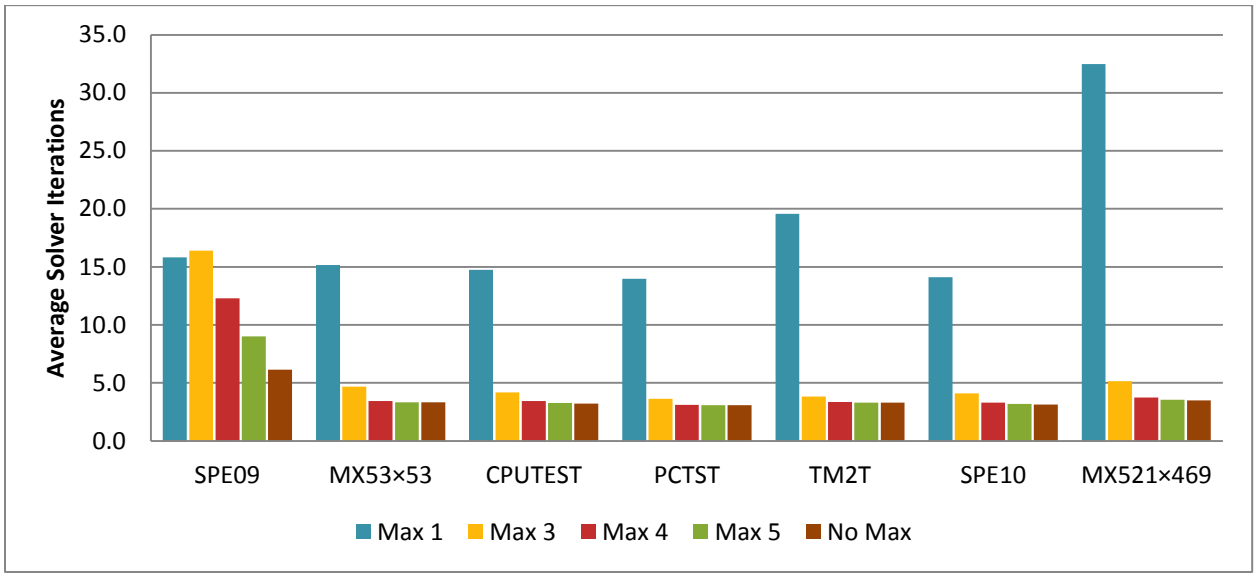


Figure 5.32: Average number of *total* solver iterations per Newton cycle where a maximum number of combinative solver iterations per Newton cycle is set.

In all cases, except for SPE09, the number of solver iterations per Newton cycle is nearly the same whether there is a maximum of three combinative iterations and no maximum at all. In the SPE09 case the number of solver iterations per Newton cycle decreases as the maximum number of combinative solver iterations increases after a maximum of three combinative solver iterations.

The reason so little difference is seen in these cases for a varying maximum number of combinative solver iterations per Newton cycle is that the maximum is not reached very often. The average number of *combinative* solver iterations per Newton cycle is plotted in Figure 5.33.

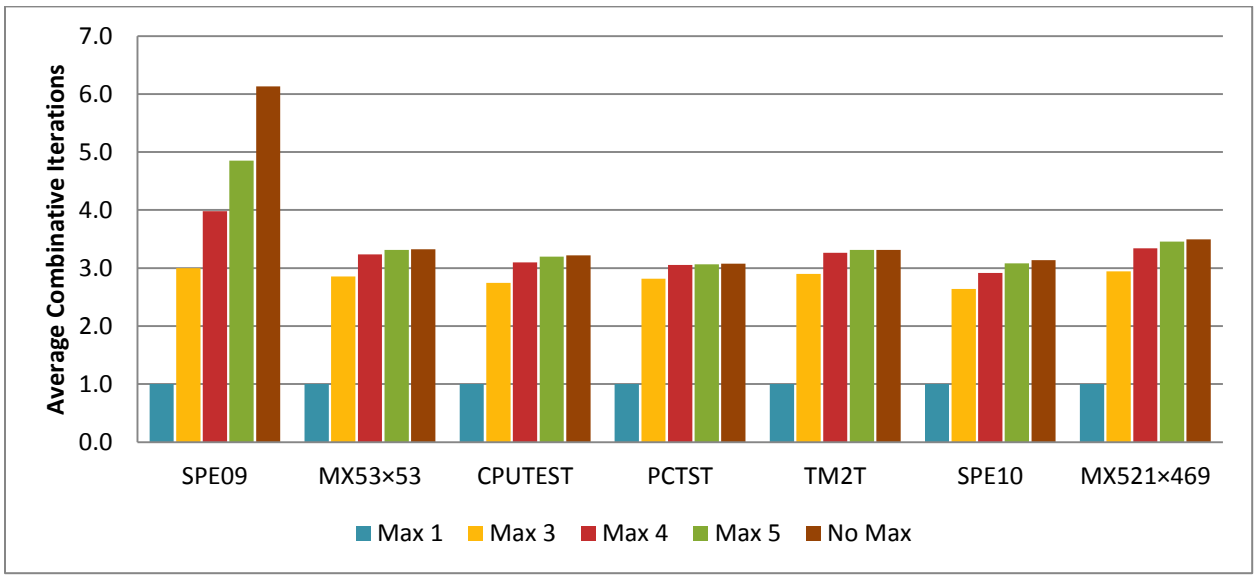


Figure 5.33: Average number of *combinative* solver iterations per Newton cycle where a maximum number of combinative solver iterations per Newton cycle is set.

There is very little increase in the average number of combinative solver iterations when a maximum of three or larger is set. The SPE09 case uses an average of 6.1 combinative solver iterations per Newton cycle when no maximum is set. It is the only case requiring more than 3.5. For SPE09 the simulation time improved as the maximum increased.

The third combinative stopping criterion is the growth of the first stage pressure residual from iteration to iteration. Simulations were run where combinative iteration was stopped when the pressure residual grew 10 times and 1000 times larger than the previous iteration.

The fourth stopping criterion calculates the approximate contribution of the current pressure solution to the total solution. Simulations were run where combinative iteration was stopped if the magnitude of the contribution of the current pressure solution was less than 0.1 and 0.01 of the initial residual.

The last stopping criterion tested is an option to stop combinative iteration if SAMG reports a negative on the diagonal of a coarse level during the setup phase of AMG. The application of this criterion differs from the previous ones since combinative iteration is not stopped during a

Newton cycle. Rather combinative iteration is done for some of the Newton cycles but not others. Results of simulations using the second, third, fourth, and fifth stopping criteria are in Figure 5.34.

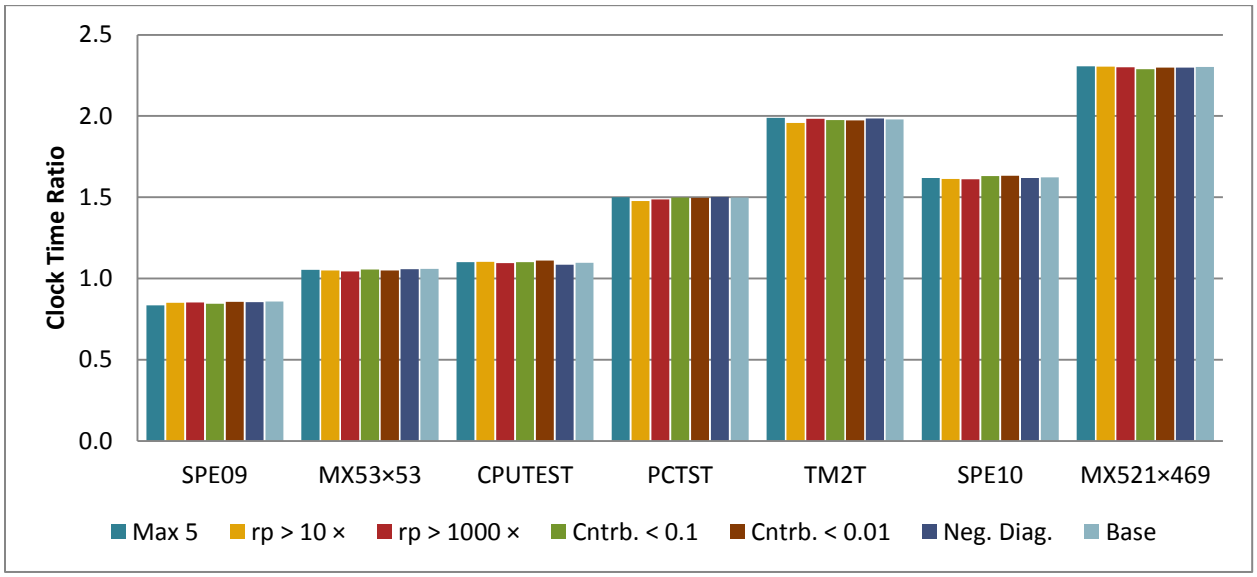


Figure 5.34: Ratio of clock time (ILU only / Combinative) for various combinative stopping criteria. Criteria used are: a maximum of 5 combinative iterations, a pressure residual greater than 10 or 1000 times the previous pressure residual, a pressure contributions less than 0.1 or 0.01 of the initial residual, and if a negative diagonal is found by SAMG. The base case where combinative iteration is not stopped is also shown.

Little variation of simulation clock times with the applied stopping criteria is evident in these cases. This is because these cases perform well with the combinative solver and are rarely, if ever, stopped.

Additional simulations in which the performance of the combinative solver is poor were made to further explore the effect the combinative stopping criteria. The difference from previous simulations is that the DRS inclusion parameter was set to 1.0 rather than 0.3 and FGMRES was used for AMG cycle acceleration rather than BICGSTAB. The results of these additional simulations are in Figure 5.35.

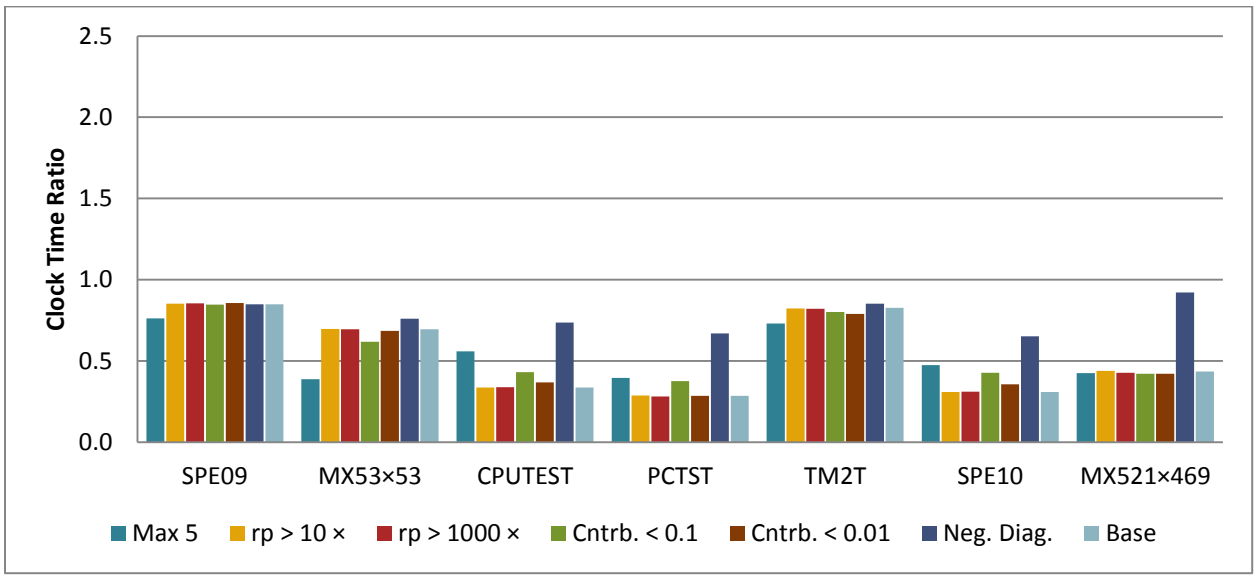


Figure 5.35: Ratio of clock time (ILU only / Combinative) for various combinative stopping criteria. Criteria used are: a maximum of 5 combinative iterations, a pressure residual greater than 10 or 1000 times the previous pressure residual, a pressure contributions less than 0.1 or 0.01 of the initial residual, and if a negative diagonal is found by SAMG. The base case where combinative iteration is not stopped is also shown. The DRS inclusion parameter is 1.0 and FGMRES is used for AMG cycle acceleration.

The best performing simulations from these cases are those using the negative diagonal stopping criterion. Forcing a maximum of 5 combinative iterations per Newton cycle gives the next best simulation times for CPUTEST, PCTST, and SPE10. However, it is the worst performer for SPE09, MX53, and TM2T. The other two criteria at both chosen parameter values perform similarly to the base case with no stopping criteria applied. The pressure residual check gave results within 2% of the base case and the pressure contribution check was within 40% of the base case.

The average number of *total* solver iterations per Newton cycle for the various stopping criteria is shown in Figure 5.36.

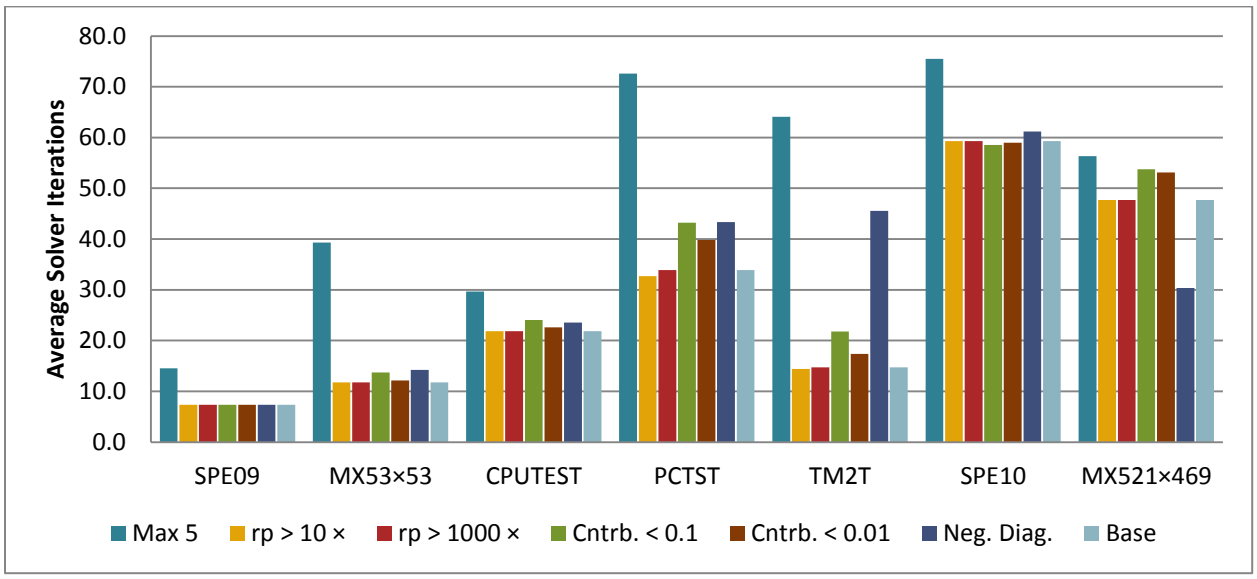


Figure 5.36: Average number of *total* solver iterations per Newton cycle for various combinative stopping criteria. Criteria used are: a maximum of 5 combinative iterations, a pressure residual greater than 10 or 1000 times the previous pressure residual, a pressure contributions less than 0.1 or 0.01 of the initial residual, and if a negative diagonal is found by SAMG. The base case where combinative iteration is not stopped is also shown. The DRS inclusion parameter is 1.0 and FGMRES is used for AMG cycle acceleration.

Using a maximum number of combinative iterations leads to the most total solver iterations in all cases. As shown previously this sometimes leads to faster simulations due to the relative inexpensive cost of ILU only iterations however, not always. The negative diagonal check gave the best simulation times and used the next most iterations in 5 of the cases and the fewest iterations in the MX521 case. The pressure residual check was almost never reached and results in nearly an identical number of iterations as the base case. The pressure contribution check resulted to within 10 solver iterations per Newton cycle in each case. None of the stopping criteria except the maximum combinative iterations were reached in the SPE09 case.

The average number of *combinative* solver iterations per Newton cycle for the various stopping criteria is shown in Figure 5.37.

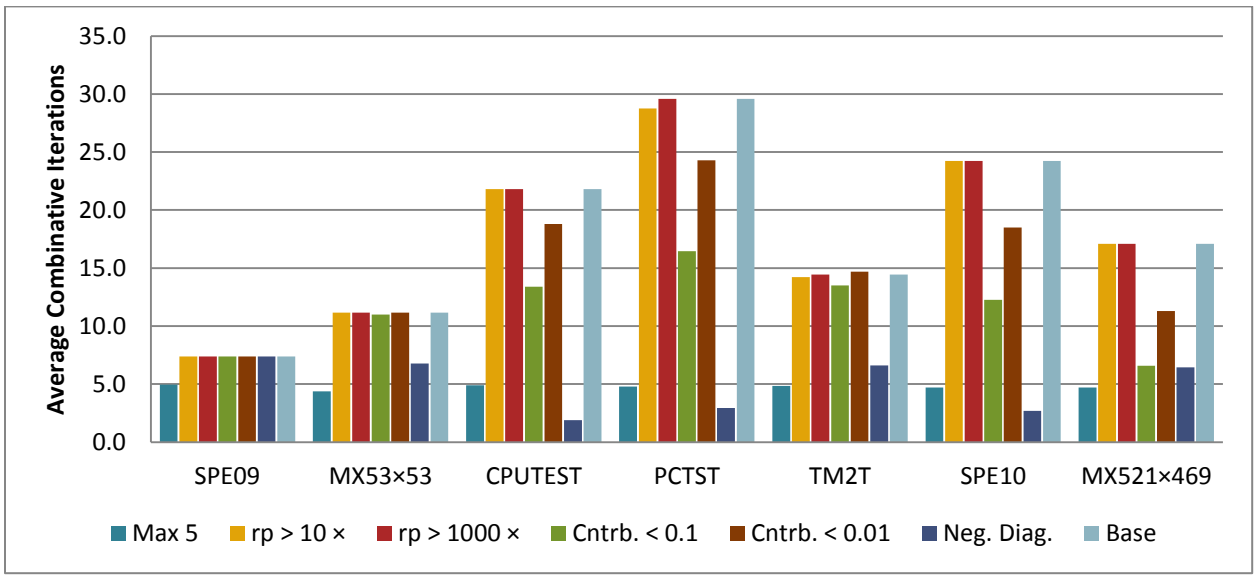


Figure 5.37: Average number of *combinative* solver iterations per Newton cycle for various combinative stopping criteria. Criteria used are: a maximum of 5 combinative iterations, a pressure residual greater than 10 or 1000 times the previous pressure residual, a pressure contributions less than 0.1 or 0.01 of the initial residual, and if a negative diagonal is found by SAMG. The base case where combinative iteration is not stopped is also shown. The DRS inclusion parameter is 1.0 and FGMRES is used for AMG cycle acceleration.

These results clearly show in which cases a combinative iteration has been stopped prematurely. Forcing a maximum of 5 combinative iterations occurs in all cases. The negative diagonal check stops the next most combinative iterations. It is the only other criterion applied to the MX53 case. The residual pressure check only occurs for the stronger value and only for the PCTST and TM2T cases. Even in those cases the check has little effect. The pressure contribution check occurs relatively frequently in the CPUTEST, PCTST, SPE10 and MX521 cases. These results appear to indicate that the negative diagonal check works the best in cases where the combinative preconditioner performs poorly.

5.3.4 Decoupling Preconditioner Variations

The decoupling preconditioner is a left preconditioner of the linear system. For the split preconditioned FGMRES algorithm (Figure 2.1) the decoupling preconditioner is M_L^{-1} and is

applied to the matrix and initial right hand side once prior to performing iterations. The development of preconditioner options for the combinative solver is discussed in §3.2.

5.3.4.1 Preconditioner Type

Three left preconditioner types are available for the combinative solver. The first is alternate block factorization (ABF) which is the preconditioner used for all ILU only simulations in this study. ABF defining characteristic is a local decoupling and normalization of matrix rows. The normalization of matrix rows is expected to perform poorly for the AMG method since it relies on the size of matrix entries during the setup phase.

The second preconditioner type is simple row scaling. After applying ABF each matrix row is scaled in order to provide AMG with more information.

The third preconditioner type is the weighted dynamic row-sum preconditioner (DRS). This preconditioner has been developed for this study. Each matrix row associated with a particular grid cell is assigned a weight and a weighted sum is constructed. The new summed row is used for pressure extraction in the combinative method. There is an inclusion parameter ε_{dd} associated with DRS. If the ratio of the magnitude of the diagonal row entry to the sum of the magnitude of all off-diagonal row entries is less than ε_{dd} the row is not included in the sum. If $\varepsilon_{dd} = 0.0$ every row is included and if $\varepsilon_{dd} = 1.0$ only diagonally dominant rows are included in the sum.

More details of each of these preconditioners are given in §3.2.

In Figure 5.38 the clock time ratio for each of these three preconditioners used with the combinative solver is given. For the row-scaling (RSCALE) and DRS preconditioners reservoir volume weighting is used and for DRS the inclusion parameter is 0.0.

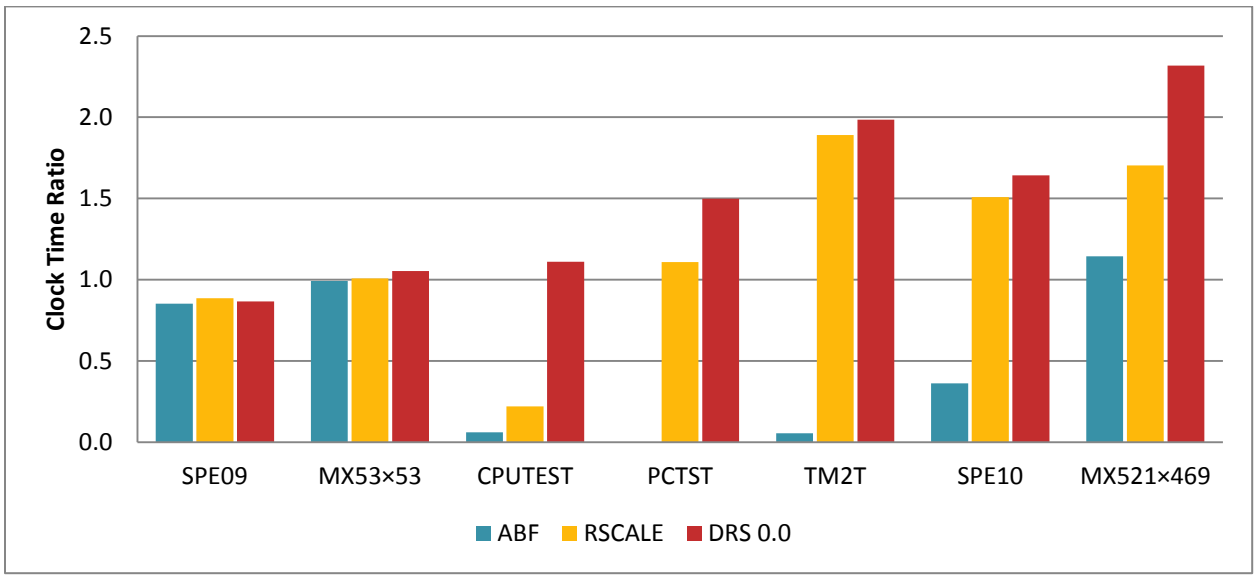


Figure 5.38: Ratio of clock time (ILU only / Combinative) for three different left preconditioners used with the combinative preconditioner. In all cases ABF is the left preconditioner for ILU only simulations. ABF decouples grid cells locally. Row-scaling (RSCALE) decouples and applies a scalar weight to each row. Dynamic row-sum (DRS) applies a weighted sum of matrix rows associated with each grid cell.

Clearly, the ABF preconditioner is the poorest in terms of performance of the three. The CPUTEST and TM2T cases took more than 16× and 18× longer respectively with the combinative preconditioner and the PCTST case failed to run to completion. The RSCALE preconditioner ran better than ABF as expected but did not outperform DRS. The CPUTEST case took 4.5× longer than ILU only when using the RSCALE preconditioner and PCTST was only 1.1× faster. The DRS preconditioner performed the best. Only the SPE09 case was slower with the combinative preconditioner than ILU only. In each other case the DRS preconditioner led to the fastest runtimes with the base set of combinative parameters. The best case was MX521, which was 2.3× faster with DRS and the combinative preconditioner.

The average number of total solver iterations per Newton cycle is shown in Figure 5.39.

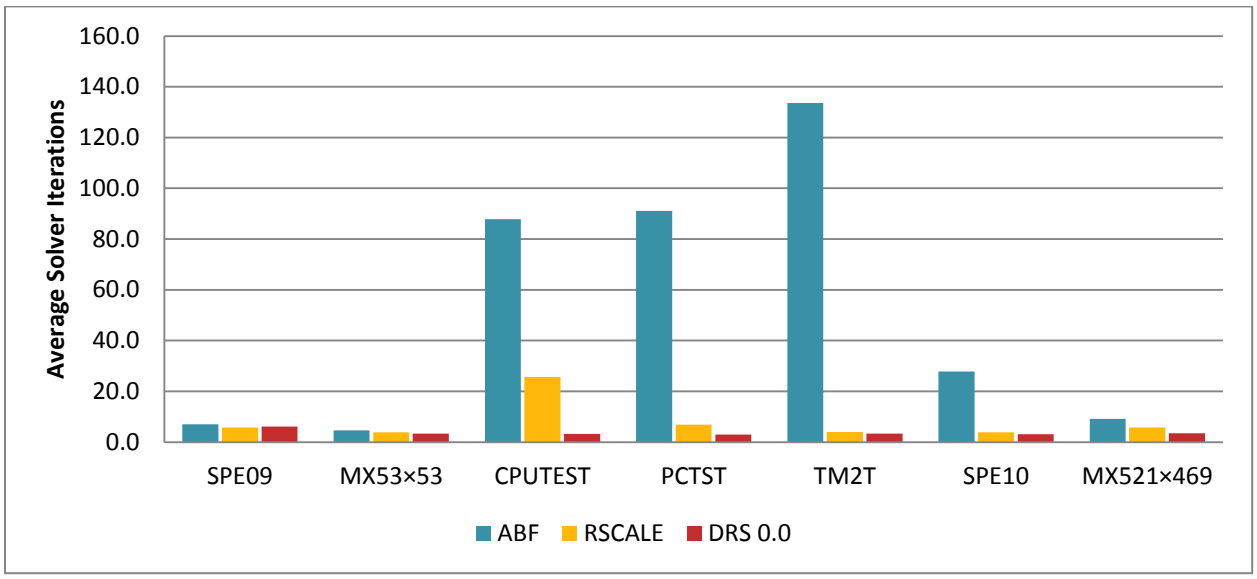


Figure 5.39: Average number of combinative solver iterations per Newton cycle for three different left preconditioners used with the combinative preconditioner.

The CPUTEST and TM2T cases with the ABF and RSCALE preconditioners stopped combinative iteration prior to the completion of FGMRES iteration for some matrices since the pressure solution was considered converged. In each simulation less than 0.1% of iterations were "ILU only". The ABF preconditioner led to the most solver iterations. For the PCTST case the average number of solver iterations with ABF is shown for completed Newton cycles before the simulation aborted due to lack of convergence. The RSCALE preconditioner led to 18-64% more average solver iterations than DRS in 4 of the cases. Cases PCTST and CPUTEST took 2.3× and 8× more average iterations respectively.

The average number of AMG cycles per combinative solver iteration for each of the three preconditioners is shown in Figure 5.40.

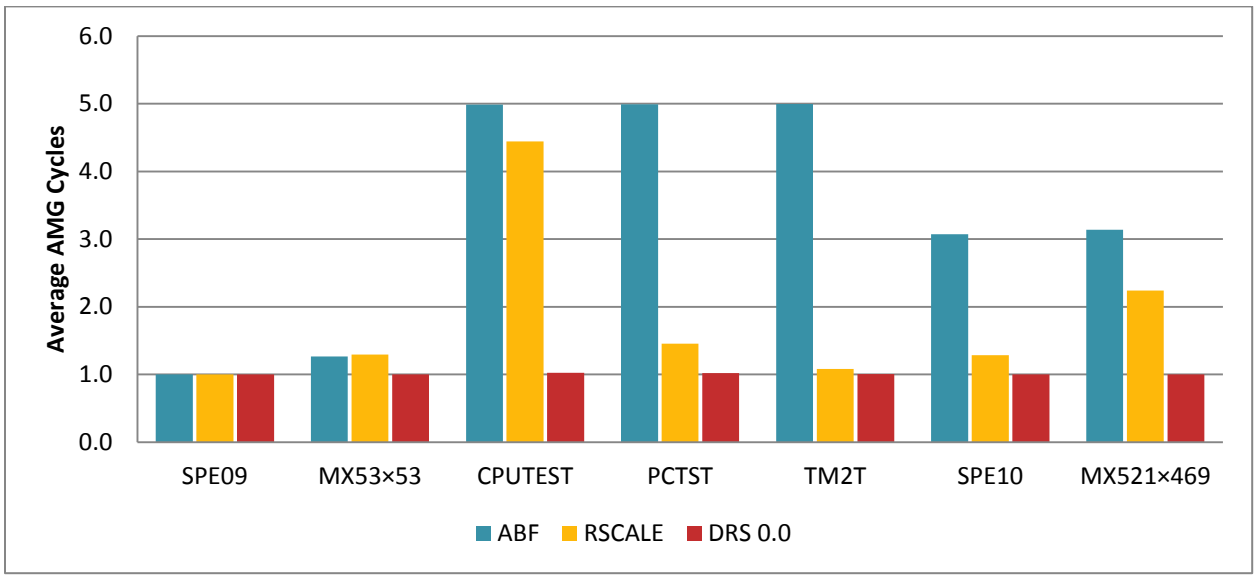


Figure 5.40: Average number of AMG cycles per combinative iteration for three different left preconditioners used with the combinative preconditioner.

The maximum number of AMG cycles per pressure matrix solution is capped at 5. With the ABF preconditioner the CPUTEST, PCTST, and TM2T cases each used 5 AMG cycles per combinative iteration indicating the failure of the AMG method to converge to the extremely loose relative convergence tolerance of 0.7. ABF preconditioned simulations used the highest average AMG cycles, RSCALE the next most and DRS the fewest. The average for the RSCALE preconditioner in the CPUTEST case was 4.4 indicating a failure of AMG convergence for some of the pressure solutions. The DRS preconditioned simulations required only a single AMG cycle for each solver iteration in all cases to reach the convergence tolerance.

5.3.4.2 Row Weighting Options

The matrix formulation of IMEX produces matrix rows associated with the reservoir weighted by surface volume. Three choices of scalar row weights are available to the RSCALE and DRS preconditioners. The first is to not modify the weight and use surface volume weighting. The second uses the state of the reservoir at the current Newton cycle to determine the reservoir

volume of each component equation. The third is to use the surface density to determine the mass of each component equation. More details are found in §3.2.2.

The RSCALE preconditioner applies these weights to the matrix entry aligned with pressure for each component equation and sums them. The resulting scalar quantity is a numerical derivative of surface volume, reservoir volume, or mass of a grid cell with respect to the change of pressure in the grid cell. The scalar weight is then applied to the entire row of the matrix and the right hand side vector for pressure extraction and solution by AMG.

In Figure 5.41 the clock time ratio for each of these three weights for the RSCALE preconditioner used with the combinative solver is given.

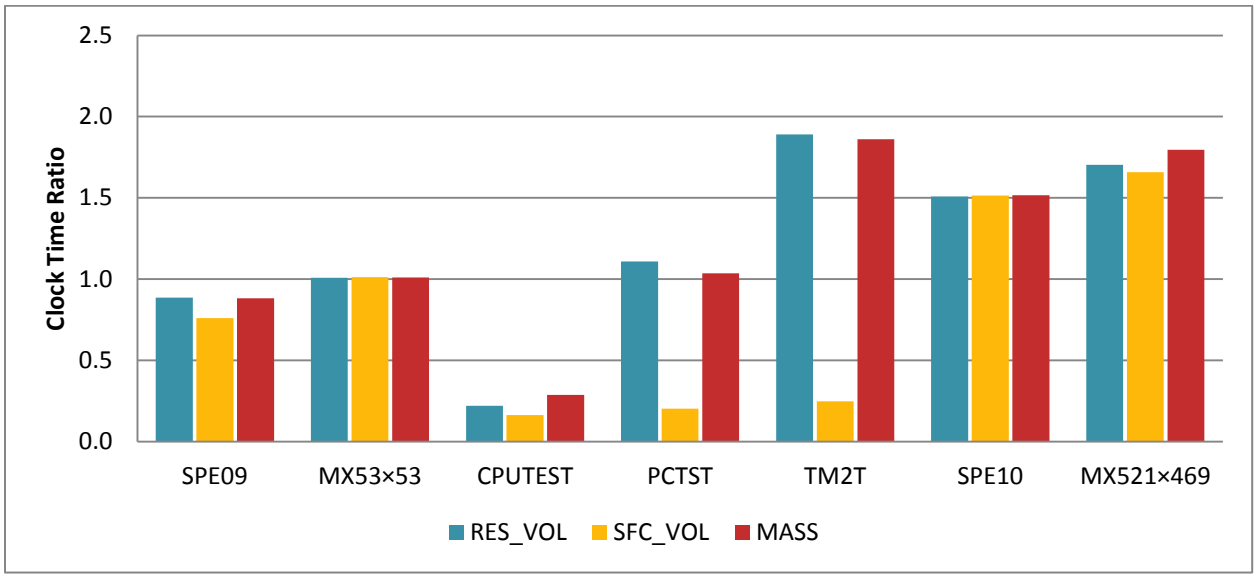


Figure 5.41: Ratio of clock time (ILU only / Combinative) for three scalar weights of the row-scaled preconditioner used with the combinative solver.

Of the three weights surface volume weighting led to the poorest results, especially in the CPUTEST, PCTST, and TM2T cases. Reservoir volume and mass weighting led to similar simulation times though reservoir volume was better in the PCTST and TM2T cases.

The DRS preconditioner applies the determined scalar weights to each row of the matrix and right hand side vector and sums them for each grid cell. The resulting component equation

replaces the original equation in the full system and is the one used for pressure extraction and solution by AMG. The matrix entry aligned with grid cell pressure in the summed equation is exactly the same as the one produced by the RSCALE preconditioner. It is the off-diagonal entries that differ. The DRS can only be applied to the full system for fully implicit grid cells. Otherwise row scaling is used for IMPES grid cells. In Figure 5.42 the clock time ratio for each of these three weights for the DRS preconditioner used with the combinative solver is given.

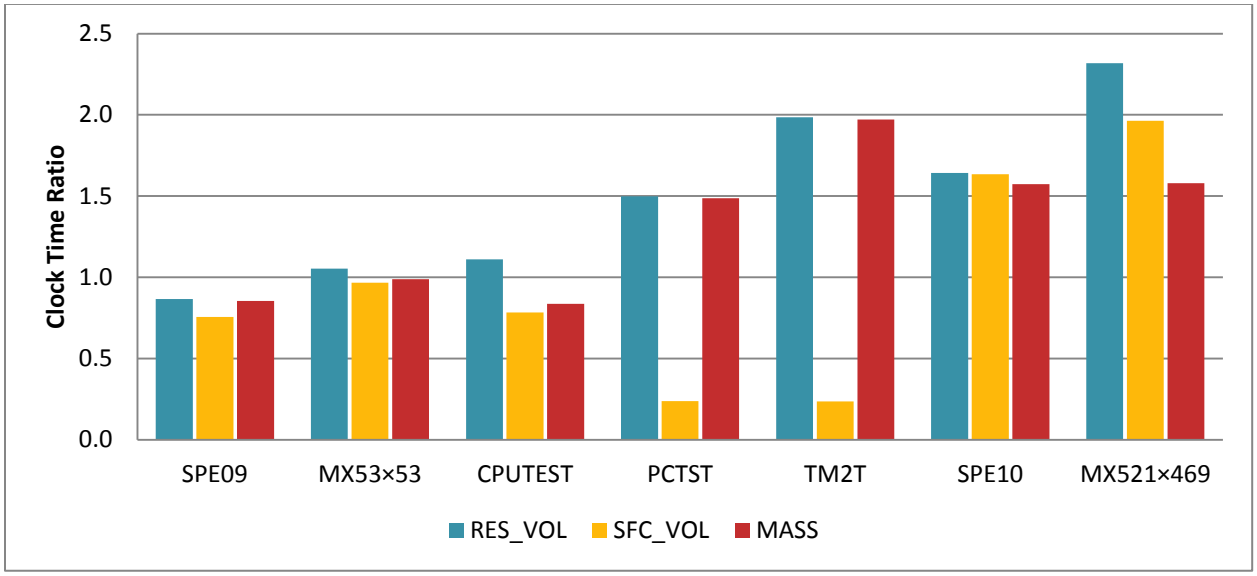


Figure 5.42: Ratio of clock time (ILU only / Combinative) for three scalar weights of the DRS preconditioner used with the combinative solver.

The results with the DRS preconditioner mirror those with the RSCALE preconditioner. The CPUTEST case clearly performs better with DRS than RSCALE for all scalar weight options while the other cases have a similar trend. The reservoir volume weighting gives the best performance in all test cases. For the MX521 case reservoir volume weighting shows a marked improvement from the RSCALE preconditioner to the DRS preconditioner. Surface volume weighting is again the poorest overall with mass weighting falling somewhere between the others. For the SPE10 and MX521 cases the trend is the opposite with surface volume weighting outperforming mass weighting.

The average number of *combinative* solver iterations per Newton cycle is shown in Figure 5.43.

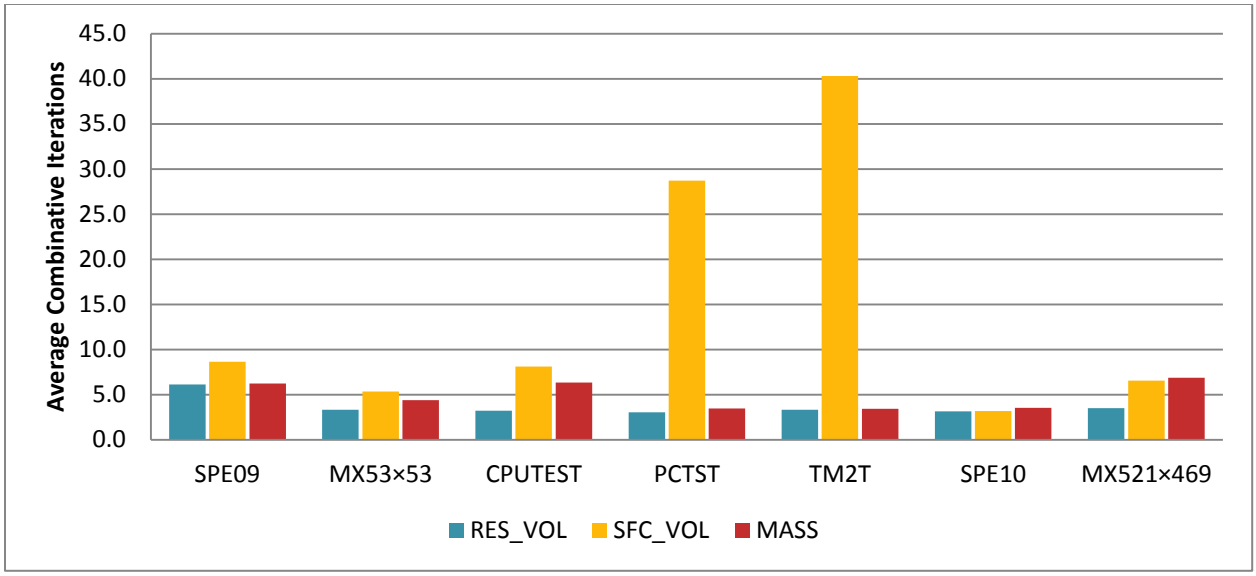


Figure 5.43: Average number of *combinative* solver iterations per Newton cycle for three scalar weights of the DRS preconditioner used with the combinative solver.

The reason for the above performance results is clear. Surface volume weighting leads to the most combinative solver iterations, mass weighting the next most, and reservoir volume weighting the fewest in most cases. For the SPE10 and MX521 cases mass weighting leads to the most solver iterations. The poorest performing cases, surface volume weighting for PCTST and TM2T, required significantly more solver iterations.

5.3.4.3 Dynamic Row Sum Inclusion Parameter

The DRS inclusion parameter can be any non-negative value. Practically, values larger than 1.0 make little sense since in that case the matrix row would already be diagonally dominant. Four values; 0.0, 0.3, 0.9, and 1.0, of the parameter were selected for simulation runs. The performance results are in Figure 5.44.

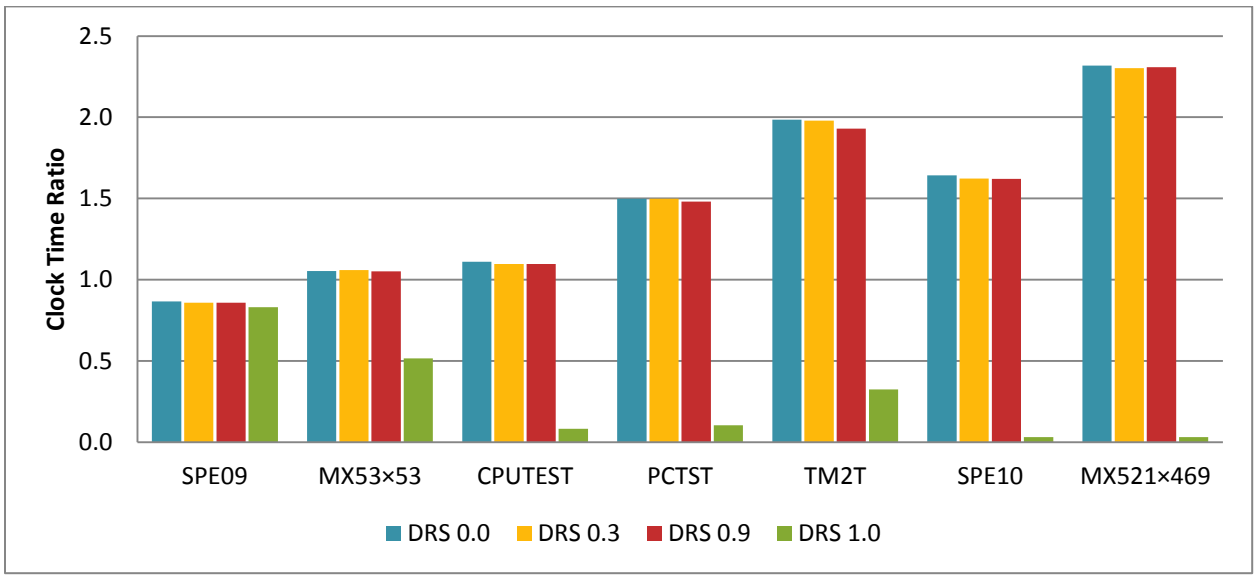


Figure 5.44: Ratio of clock time (ILU only / Combinative) for four DRS inclusion parameters which determine the nearness to diagonal dominance necessary for a matrix row to be included in the sum.

The results show little difference in the performance of each case when the parameter varies between 0.0 and 0.9. The best performance is for 0.3 in the SPE09, MX53, and PCTST cases and 0.0 in the CPUTEST, TM2T, SPE10, and MX531 cases. The difference in these two choices is less than 1.3% for these cases. A parameter value of 1.0 clearly performs the worst in all cases with both SPE10 and MX521 taking more than 30× longer than the ILU only method.

The average number of *combinative* solver iterations per Newton cycle is shown in Figure 5.45.

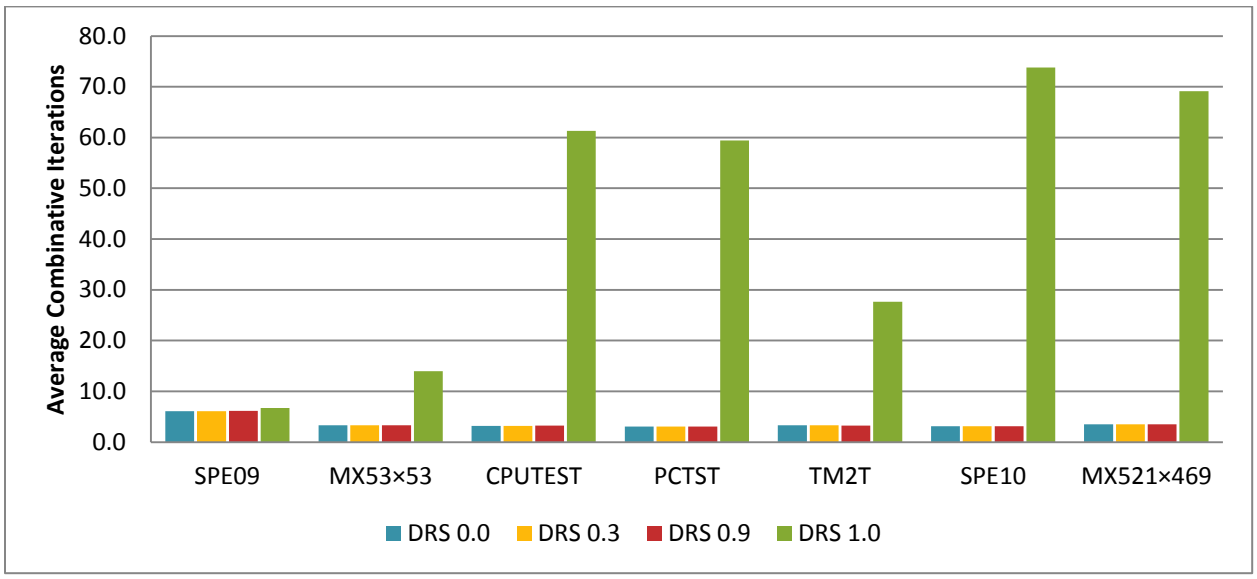


Figure 5.45: Average number of *combinative* solver iterations per Newton cycle for four DRS inclusion values which determine the nearness to diagonal dominance necessary for a matrix row to be included in the sum.

An inclusion parameter of 1.0 clearly leads to the most combinative solver iterations. Lower values result in nearly identical average number of combinative solver iterations. The differences are less than 1.3% in all the cases.

The average number of AMG cycles per combinative iteration is shown in Figure 5.46.

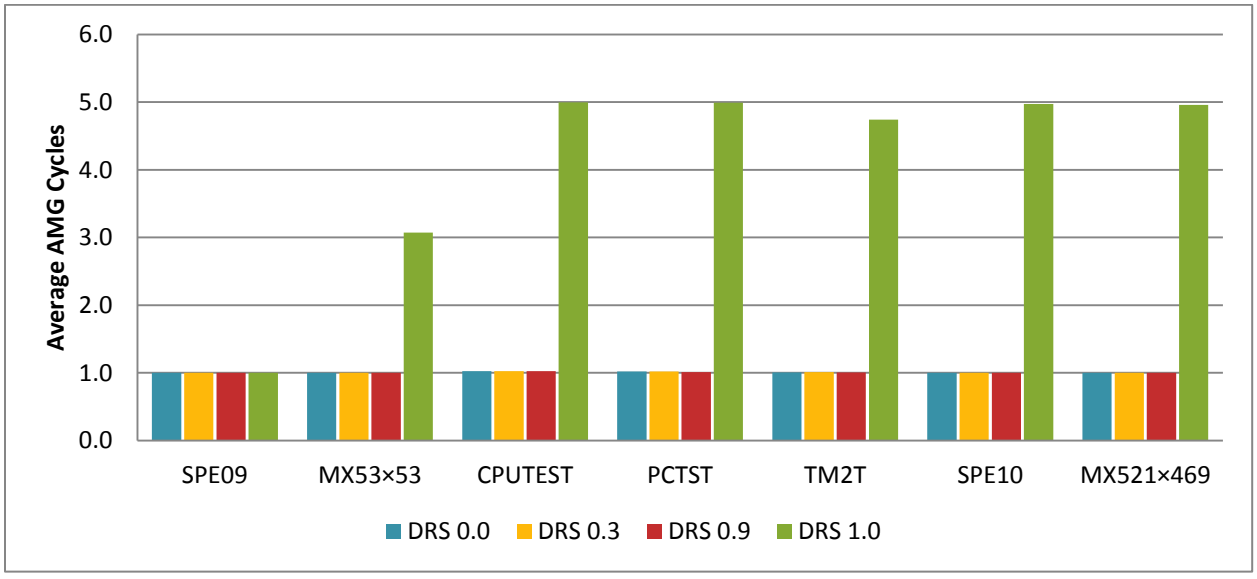


Figure 5.46: Average number of AMG cycles per combinative iteration. Four DRS inclusion parameter values are shown which determine the nearness to diagonal dominance necessary for a matrix row to be included in the sum.

In five of the cases a parameter value of 1.0 leads to the maximum or nearly the maximum 5 AMG cycles per pressure solution. This means that the pressure solution is not converging in those cases, which is what leads to so many extra solver iterations and longer simulation times. For DRS inclusion parameters less than 1.0 no case requires more than 1.03 AMG cycles per solver iteration to reach convergence for the pressure solution.

5.3.4.4 DRS for IMPES Cells

The DRS preconditioner is not applied to IMPES grid cells for the full system. The default choice for combinative preconditioned simulations in this study is to not apply the DRS preconditioner at all for IMPES grid cells. Instead the RSCALE preconditioner is applied to IMPES grid cells. There is an option to apply the DRS preconditioner to the first stage only of the combinative solver. Details are given in §3.2.3. The resulting pressure matrix differs from the default DRS in the magnitude of the off-diagonal entries associated with IMPES grid cells. This option would have a large impact if there were additional non-mass conservation equations modelled. In such a case the elimination of the non-mass conservation equation via the RSCALE preconditioner could modify the "pressure" matrix to a great extent. For example, if enthalpy were being modelled the elimination of an enthalpy equation would lead to temperature and energy effects included in the pressure equation. The assumptions of the combinative solver that allow the extraction and solution of the pressure system would not be valid. For the models used in this study all equations are mass conservation equations so the effect of varying the treatment of IMPES cells is expected to be minimal.

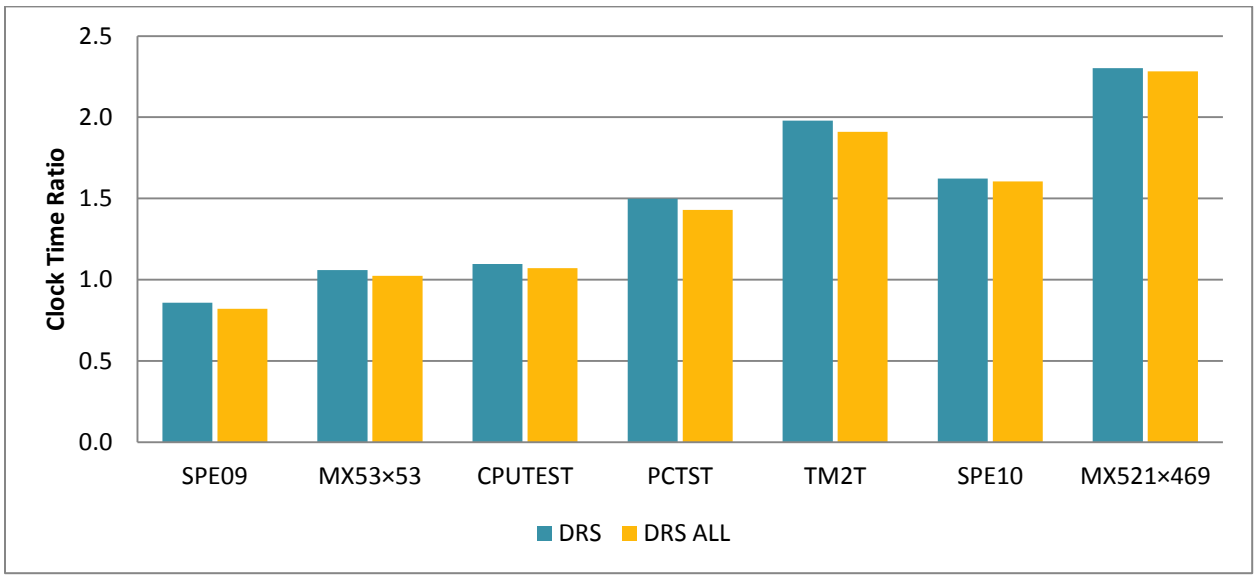


Figure 5.47: Ratio of clock time (ILU only / Combinative) varying the application of the DRS preconditioner to IMPES grid cells.

The default application of DRS where IMPES grid cells are treated with the RSCALE preconditioner is faster for all the test cases. The differences are slight implying little difference in any of the convergence properties in both DRS options. A likely possibility of the difference in speed is the extra calculation in the DRS for all cells option, which requires a conversion between DRS for the first stage and RSCALE for the second stage of the combinative solver. To confirm the convergence hypothesis the average number of solver iterations per Newton cycle is shown in Figure 5.48.

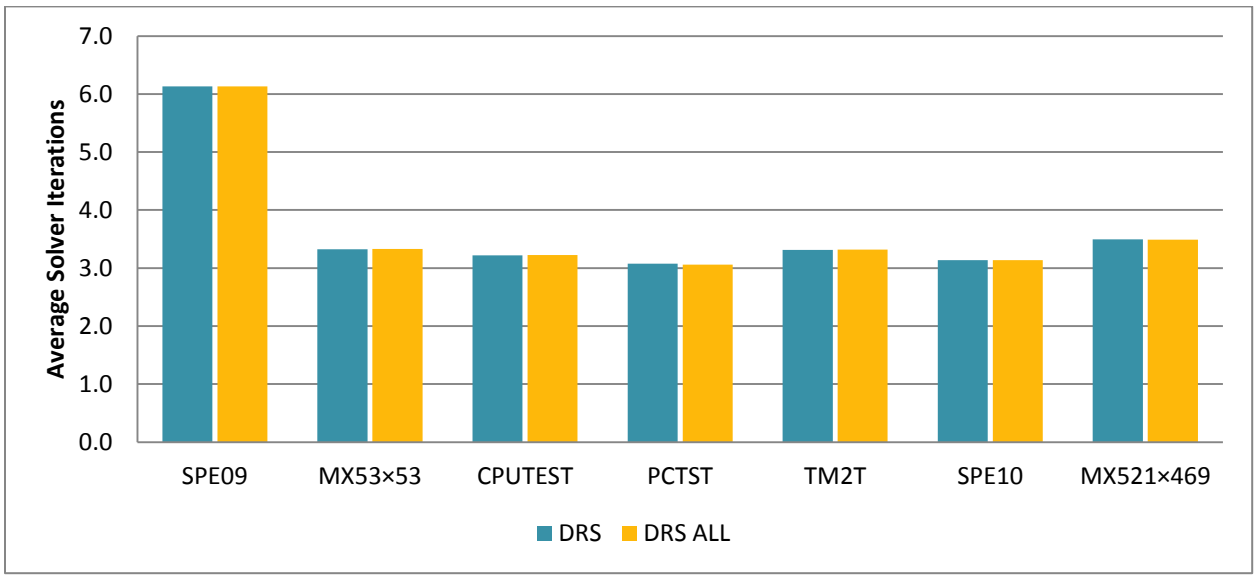


Figure 5.48: Average number of combinative solver iterations per Newton cycle. DRS is either applied to only fully implicit grid cells or to all grid cells.

There is less than a 0.5% difference in the average number of solver iterations per Newton cycle for both applications of DRS to IMPES grid cells. This confirms that the convergence of these two methods is the same for these test cases.

5.4 Fine Grid Effect

An AMG solver should be more efficient for a finer grid solution of a given problem than a single level method due to its design. Increasing the number of grid cells to cover the same area requires more iterations of a single level solver to reach the same solution. This assumes the correct solution is achieved for both grids and there is a significant enough elliptic component of the equations to affect convergence.

This effect can be seen in the results of Table 5.1 comparing the PCTST case with the TM2T case in which each grid cell has been divided into 3×3 cells areally totalling the same area. The ratio of clock time improves from 1.25 to 1.81 with a finer grid.

To confirm this effect the MX53 case was modified to use finer grids in the horizontal direction. The same case was solved using grid cells with the original horizontal extent of 15m as well as

finer grids with horizontal extents of 5m, 3m, 1.5m, 1m, 0.75m, and 0.6m. Each version of the data was run in the standard simulation environment using 9 CPUs allowing a maximum of 150 solver iterations per Newton cycle. The average number of solver iterations used per Newton cycle vs. the number of horizontal grid cells between wells is shown in Figure 5.49.

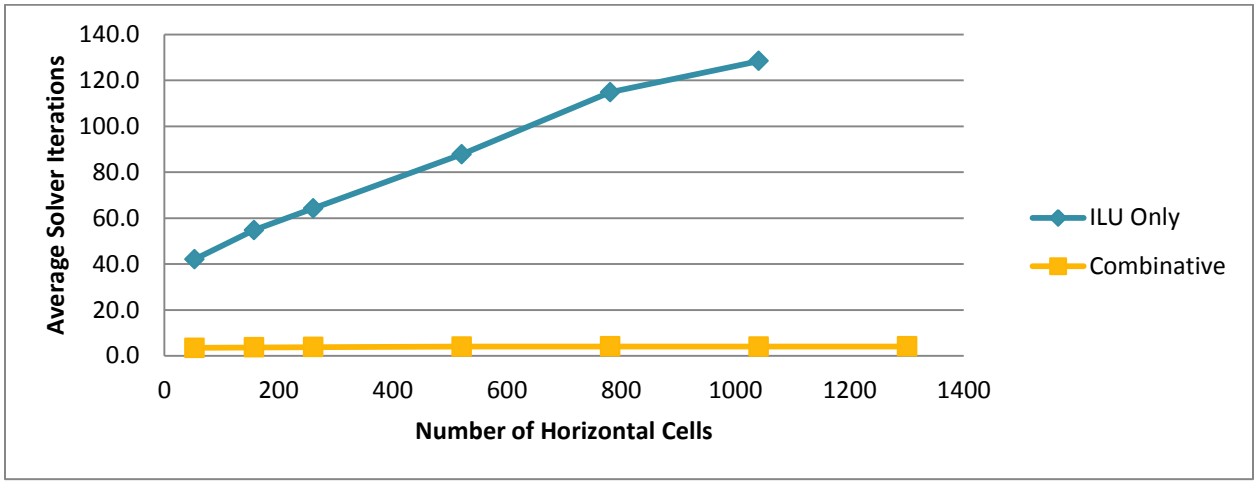


Figure 5.49: Average number of solver iterations per Newton cycle for ILU only and combinative simulations of a single 5 spot pattern. The horizontal extent of a grid cell varied between 15m and 0.6m.

These results clearly show an increase in the number of solver iterations required for the ILU only method with a finer grid and no corresponding increase with the combinative method. The performance results using the clock time ratio of the ILU only method to the combinative method is plotted against the number of horizontal grid cells between wells in Figure 5.50.

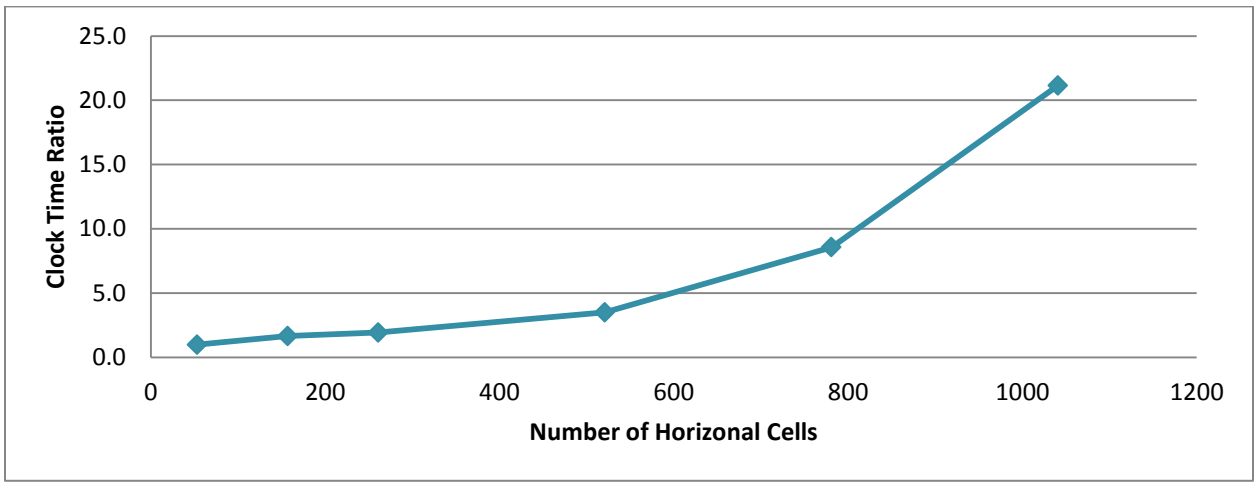


Figure 5.50: Ratio of clock time (ILU only / Combinative) for simulations of a single 5 spot pattern. The horizontal extent of a grid cell varies between 15 m and 0.6 m.

The results show an improvement from a ratio of 1.0 for 53 horizontal grid cells to 21.1 for 1041 horizontal grid cells. The improvement is due to the increase in solver iterations for the ILU only method and the increased length of the solution vectors used by the outer FGMRES method. Clearly, the combinative method improves the running time of cases which use a fine grid resolution.

5.5 Large Field Black Oil Results

The indented benefit of the combinative solver is primarily for large field simulations that take significant computation time. In addition to the cases studied in §5.3 results of three large field cases, SINC, B3MM, and MAUR, are examined. For these cases parameter values giving the best results with the combinative solver were used. The results of simulation are summarized in Table 5.3.

Case	Number of Active Grid Cells	Total Elapsed Clock-Time [s]		Clock Time Ratio	Solver Iterations per Newton Cycle		AMG Cycles per Combinative Iteration
		ILU Only	Combinative		ILU Only	Combinative	
TM2T	1,640,916	1070	422	2.54	85.9	3.7	1.20
SPE10	1,093,078	1171	593	1.97	71.8	3.5	1.13
MX521	4,886,980	2510	898	2.79	63.9	4.3	1.12
SINC	2,638,003	11648	13797	0.84	25.8	6.6	1.08
B3MM	2,193,565	15917	15783	1.01	19.1	3.5	1.11
MAUR	3,529,192	21485	3175	6.77	152.2	3.4	1.06

Table 5.3: Numerical performance comparison of large field scale simulations. Performance using the ILU only solver is compared with the combinative solver.

In 5 of the 6 large test cases the total elapsed clock time is faster using the combinative solver. This is due to a large drop in the average number of required solver iterations per Newton cycle. In 4 of these 5 cases the average number of solver iterations per Newton cycle is less than 4 and the other, MX521, is less than 4.5. Clearly the preconditioned linear solver converges well. In 3 of the cases using the combinative solver is nearly twice as fast and in the MAUR case the combinative solver is nearly 7× faster. The MAUR case is ideal for AMG. It is a water injection simulation with a high water cut at production wells. At least 95% of the production is water. Essentially the mobile reservoir fluid behaves very similarly to a single-phase nearly incompressible fluid. Because of this the pressure in the system can be expected to behave in a nearly elliptical manner.

In the SINC case using the combinative solver slows the simulation. The running time is still within 80% of the time using ILU only preconditioning. This case is one of the less difficult for the ILU only preconditioned solver in terms of the total number of solver iterations required. It has the lowest decrease (by percentage) in the number of average solver iterations when switching to the combinative solver. It appears the time improvement gained from the reduction of solver iterations does not compensate for the extra numerical costs. The SINC case is also the most difficult for the combinative solver. It requires nearly twice as many solver iterations per

Newton cycle as some of the other cases. Because the FGMRES method is used for solver iterations the increase in numerical cost for each additional solver iteration is nonlinear. The time saving from the reduction in iterations does not compensate the overhead and extra work per iteration of the combinative solver.

5.6 Other Fluid Models (and verification)

The combinative solver has been tested on all of the IMEX templates and quality assurance (QA) data sets. All of the templates and QA data can run with the combinative solver option and in the vast majority of the cases the number of linear solver iterations is greatly reduced. In the aggregate over 592 data sets tested the number of linear solver iterations drops from 1,346,878 to 339,280. The average number of solver iterations per Newton cycle drops from 11.0 to 2.8. It appears from both the above test cases and the QA data sets that at least 25,000 grid blocks and an average of 30 solver iterations per Newton cycle normally would run in less time using the combinative solver.

For the QA data sets using the volatile oil and pseudo-miscible fluid models (2.15) and (2.14), the total elapsed clock time is not a good comparison measure because many of them are very small. The overhead using SAMG would overwhelm any gain in speed. The only comparison performed is a measure of average solver iterations required for convergence.

The average number of solver iterations per Newton cycle using the volatile-oil fluid model is shown in Figure 5.51.

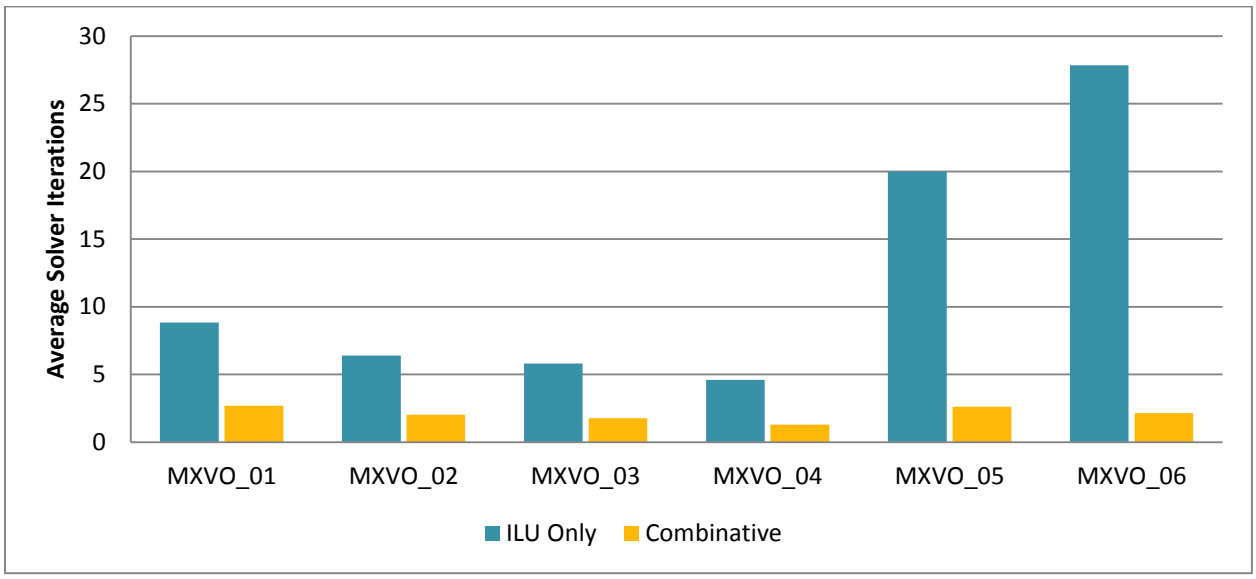


Figure 5.51: Average number of solver iterations per Newton cycle for 6 data sets using the volatile-oil fluid model. Results of the ILU only solver and the combinative solver are compared.

The average number of solver iterations per Newton cycle using the pseudo-miscible fluid model is shown in Figure 5.52.

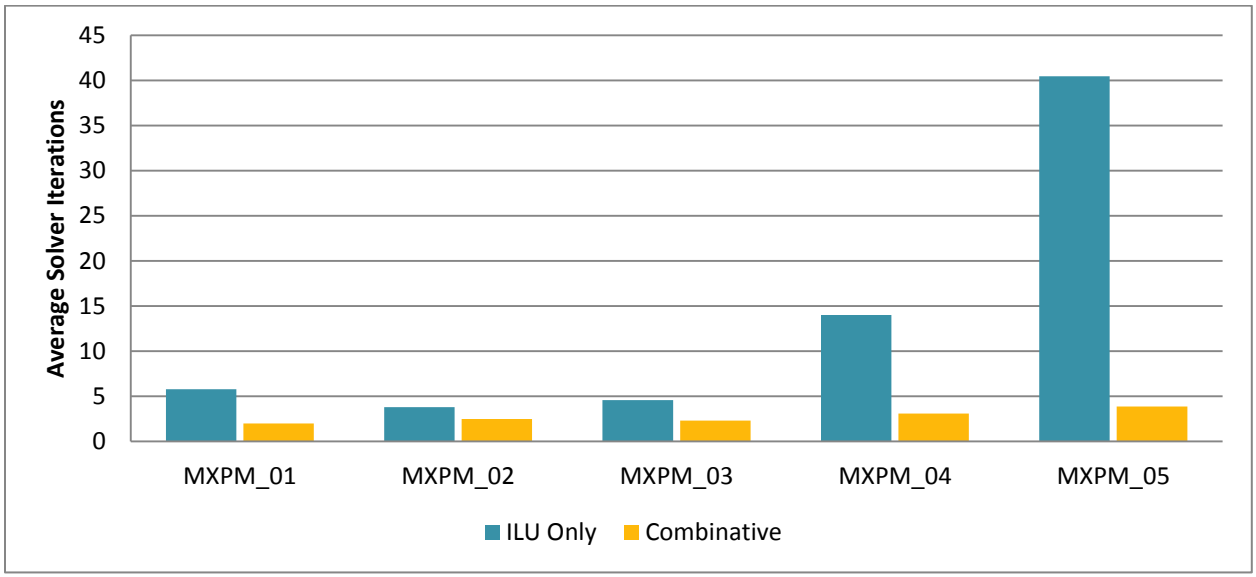


Figure 5.52: Average number of solver iterations per Newton cycle for 5 data sets using the pseudo-miscible gas fluid model. Results of the ILU only solver and the combinative solver are compared.

The results of simulations of these additional fluid models show a reduction in the average number of solver iterations required in every case tested. This provides further evidence of the

viability of DRS preconditioned combinative iterations and shows the method leads to faster convergence. Reservoir weighting for the DRS preconditioner is effective for multiple fluid models.

Chapter Six: **Conclusions and Recommendations**

A two-stage CPR preconditioner with decoupling has been developed and incorporated into an existing adaptive-implicit black-oil simulator. The preconditioner combines Fraunhofer SCAI's SAMG solver for the first stage solution and CMG's ILU preconditioned PARASOL solver for the second stage solution. A physically weighted dynamic row-sum decoupling preconditioner has been developed with the goal of constructing a pressure matrix that is well suited to AMG while at the same time accelerating the CPR process. The decoupled combinative solver functions for a variety of fluid models and processes including solvent injection and volatile-oil models. Simulations of several models, both synthetic and based on real field data, have been completed using the developed solver methods.

6.1 Accuracy and Verification of Results

Results of simulations using the DRS decoupled two-stage combinative solver are compared to the results of the same models using the existing ILU preconditioned solver.

The combinative method was more accurate in general than the ILU only preconditioned method. An average relative residual reduction of 10^{-9} was achievable with the combinative solver for each model tested. Each reduction of the size of the residual by an order of magnitude required an average increase in the number of solver iterations by a constant factor between 0.8 and 1.2 for the cases tested. The residual reduction was reached in fewer solver iterations using the combinative solver for each individual matrix studied. When the same relative residual reduction was prescribed for both methods there was an insignificant difference in the final results for the models tested. Therefore, there was no significant divergence in the solutions of the two solvers. Results using the combinative method matched those of two SPE comparative solution projects.

6.2 Eigenvalue Analysis

The eigenvalues were found for three pressure matrices from different reservoir simulations using three different decoupling preconditioners. The dynamic-row sum (DRS) preconditioner resulted in larger condition numbers and a wider ranger of eigenvalues than the alternate block factorization (ABF) preconditioner. The DRS preconditioner also reduced the number of complex eigenvalues compared to ABF and eliminated eigenvalues with negative real component in one case leading to much better AMG convergence and therefore combinative convergence. A simple row-scaling (RSCALE) of the ABF preconditioner using the diagonal values produced by DRS gave similar results to the DRS preconditioner. However, RSCALE was unable to eliminate the negative eigenvalues found in one case.

6.3 Combinative Solver Performance

The combinative solver led to overall simulation clock time speedups of many simulations with a clock time ratio in the 1.5 to 3 range. In one case using the combinative solver was 7 times faster. Reduction in simulation time was due to substantial reduction in the number of linear solver iterations. The speedup occurred despite the fact solver iterations under the combinative method are more computationally expensive than under the ILU only method and there is an additional overhead cost from the AMG solver portion. Tests of the software indicated that models with at least 25,000 grid blocks requiring an average of 30 solver iterations per Newton cycle with an ILU only method would normally run in less time using the combinative solver.

6.3.1 Parameter Variations

A study of parameters used in the combinative method was performed. This included parameters for the first stage AMG solver, the second stage full system solver, the combinative method itself, and the decoupling method.

Variation of AMG parameters indicate performing a small number of AMG cycles to reach a relatively loose solution is best. Requiring a high accuracy in the AMG solution was detrimental to performance since the first stage pressure solution in the combinative method is only a rough approximation. In general, parameter choices which increased the speed of the AMG solution at the cost of reduced the accuracy led to the best performance. In particular, using V-cycles, Jacobi relaxation, and aggressive coarsening gave the fastest overall speeds.

Variation of full system solver parameters was inconclusive. For three of seven models using red-black ordering with an ILU(0) preconditioner performed the best. In the other cases the difference in speed when varying parameters was marginal.

The development of the combinative solver included several optional parameters. The best choice of frequency in the setup phase of AMG appeared to be problem dependent. For many models performing a full setup phase every time step and partial setups for subsequent Newton cycles generally gave the best performance. Other models benefitted from less frequent setup phases. The weighting and inclusion of well equations in the AMG solution led to better performance of the combinative solver. Weighting the first and second stage results of the combinative method did not improve simulation times and led to very slow simulations in some cases. There was no benefit found when stopping the combinative method prematurely and replacing it with an ILU only method.

Several variations of decoupling preconditioners were studied. The use of physical weights during decoupling had the largest effect. The best results were found using a reservoir volume weight. The weighted DRS preconditioner outperformed a simple row-scaled preconditioner in many cases. The DRS preconditioner performed best when the majority of rows were included in the row sum. In particular, an inclusion parameter between zero and one showed the best results.

Excluding all rows that were not strictly diagonally dominant from the row sum led to poor performance.

6.4 Recommendations for Future Work

There are several possibilities of future research which would follow up this study and compliment the results. Three such possibilities of particular interest are described below.

On the theoretical side, the eigenvalue results of a few select cases are insufficient to draw general conclusions regarding the DRS preconditioner. An analytic study of the DRS preconditioner and the combinative method would be valuable. Further numerical analysis is needed to discover which types of systems benefit from a combinative approach and why.

On the practical side, the combinative method could be applied to linear systems selectively. Simple metrics such as row and column sums could be calculated during the construction of a decoupling preconditioner or even the Jacobian matrix itself. These metrics could then be compared against threshold values to predict the likelihood of poor (or slow) combinative solver performance for a given linear system. If such values could be determined then the simulation would adaptively switch between the combinative preconditioning method and the ILU only method. The difficulty is choosing the correct measures of the linear system and determining reasonable threshold values. It could be that such threshold values are so problem dependent that they are not practically useful.

Another area of further study would be the extension of the combinative method and the DRS preconditioner to more complicated fluid models. In particular, thermal models and compositional models warrant study. Weighted DRS decoupling would require modification for each of these models. In the case of a thermal model the energy of the model may exhibit elliptic or nearly elliptic character which could be a good candidate for solution with the AMG method.

Determining which equations and how to construct them to for solution with AMG is one avenue of future study. How to combine the results of a pressure and temperature decoupled system is another.

References

Appleyard, J. R., & Cheshire, I. M. (1983). Nested Factorization. In *SPE Reservoir Simulation Symposium, 15-18 November, San Francisco, California* (pp. 315–324. SPE–12264–MS). http://doi.org/10.2118/12264-MS

Aziz, K., & Settari, A. (1979). *Petroleum reservoir simulation*. Applied Science Publishers.

Bank, R. E., Chan, T. F., Coughran, W. M., & Smith, R. K. (1989). The alternate-block-factorization procedure for systems of partial differential equations. *BIT*, *29*(4), 938–954. http://doi.org/10.1007/BF01932753

Behie, A., & Forsyth, P. A. (1984). Incomplete Factorization Methods for Fully Implicit Simulation of Enhanced Oil Recovery. *SIAM Journal on Scientific and Statistical Computing*, *5*(3), 543–561. http://doi.org/10.1137/0905040

Behie, A., & Vinsome, P. K. W. (1982). Block iterative methods for fully implicit reservoir simulation. *SOC. PET. ENGRS. J.*, *22*(5), 658–668.

Brown, G. L., Collins, D. A., & Chen, Z. (2015). Efficient Preconditioning for Algebraic Multigrid and Red-Black Ordering in Adaptive-Implicit Black-Oil Simulations. In *SPE Reservoir Simulation Symposium*. Society of Petroleum Engineers. http://doi.org/10.2118/173231-MS

Cao, H., Tchelepi, H. A., Wallis, J. R., & Yardumian, H. E. (2005). Parallel Scalable Unstructured CPR-Type Linear Solver for Reservoir Simulation. In *SPE Annual Technical Conference and Exhibition, 9-12 October, Dallas, Texas* (p. SPE–96809–MS). http://doi.org/10.2118/96809-MS

Chen, Z., Huan, G., & Ma, Y. (2006). *Computational Methods for Multiphase Flows in Porous Media*. Society for Industrial and Applied Mathematics.

Christie, M. A., & Blunt, M. J. (2001). Tenth SPE comparative solution project: A comparison of upscaling techniques. *SPE Reservoir Evaluation & Engineering*, *4*(04), 308–317. SPE–72469–PA. http://doi.org/10.2118/72469-PA

Cleary, A. J., Falgout, R. D., Henson, V. E., Jones, J. E., Manteuffel, T. A., Mccormick, S. F., … Ruge, J. W. (2000). Robustness and scalability of algebraic multigrid. *SIAM Journal on Scientific Computing*, *21*(5), 1886–1908.

Clees, T., & Ganzer, L. (2010). An Efficient Algebraic Multigrid Solver Strategy for Adaptive Implicit Methods in Oil-Reservoir Simulation. *SPE Journal*, *15*(3), 670 – 681. SPE–105789–PA. http://doi.org/10.2118/105789-PA

Coats, K. H., Thomas, L. K., & Pierson, R. G. (1998). Compositional and Black Oil Reservoir Simulation. *SPE Reservoir Evaluation & Engineering*, *1*(04), 372–379. http://doi.org/10.2118/50990-PA

Collins, D. A., Grabenstetter, J. E., & Sammon, P. H. (2003). A Shared-Memory Parallel Black-Oil Simulator with a Parallel ILU Linear Solver. In *SPE Reservoir Simulation Symposium, 3-5 February, Houston, Texas* (p. SPE–79713–MS). http://doi.org/10.2118/79713-MS

Darcy, H. (1856). *Les fontaines publiques de la ville de Dijon: exposition et application...*

Falgout, R. D. (2006). An Introduction to Algebraic Multigrid Computing. *Computing in Science & Engineering*, *8*(6). http://doi.org/10.1109/MCSE.2006.105

Forsyth, P. A., & Sammon, P. H. (1986). Practical considerations for adaptive implicit methods in reservoir simulation. *Journal of Computational Physics*, *62*(2), 265–281. http://doi.org/10.1016/0021-9991(86)90127-0

Gordon, D., & Gordon, R. (2010). Row scaling as a preconditioner for some nonsymmetric linear systems with discontinuous coefficients. *Journal of Computational and Applied Mathematics*. http://doi.org/10.1016/j.cam.2010.05.021

Gries, S., Stüben, K., Brown, G. L., Chen, D., & Collins, D. A. (2014). Preconditioning for Efficiently Applying Algebraic Multigrid in Fully Implicit Reservoir Simulations. *SPE Journal*, *19*(04), 726–736. SPE–163608–PA. http://doi.org/10.2118/163608-PA

Henson, V. E. (2002). Multigrid Methods for Nonlinear Problems: An Overview.

Killough, J. E. (1995). Ninth SPE Comparative Solution Project: A Reexamination of BIack-Oil Simulation of the Reservoir. *13th SPE Symposium on Reservoir Simulation*, 135–147. http://doi.org/10.2118/29110-MS

Lacroix, S., Vassilevski, Y. V., & Wheeler, M. F. (2001). Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS). *Numerical Linear Algebra with Applications*, *8*(8), 537–549. http://doi.org/10.1002/nla.264

LeVeque, R. J. (2002). *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press.

Molenaar, J. (1995). *Multigrid methods for fully implicit oil reservoir simulation*. Citeseer.

Peaceman, D. W. (1977). *Fundamentals of Numerical Reservoir Simulation*. Elsevier.

Ruge, J. W., & Stüben, K. (1987). Algebraic Multigrid. In S. F. McCormick (Ed.), *Multigrid Methods* (Vol. 3, pp. 73–130). Society for Industrial and Applied Mathematics. http://doi.org/doi:10.1137/1.9781611971057.ch4

Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics. http://doi.org/10.2113/gsjfr.6.1.30

Scheichl, R., Masson, R., & Wendebourg, J. (2003). Decoupling and Block Preconditioning for Sedimentary Basin Simulations. *Computational Geosciences*, *7*(4), 295–318. http://doi.org/10.1023/B:COMG.0000005244.61636.4e

Schenk, O., & Gärtner, K. (2004). Solving unsymmetric sparse systems of linear equations with PARDISO. In *Future Generation Computer Systems* (Vol. 20, pp. 475–487).

Stüben, K. (2001a). A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, *128*(1-2), 281–309.

Stüben, K. (2001b). An introduction to algebraic multigrid. In *Multigrid* (pp. 413–532). A. Schüller, Academic Press.

Stüben, K. (2015). SAMG. Fraunhofer SCAI.

Stüben, K., Clees, T., Klie, H., Lu, B., & Wheeler, M. F. (2007). Algebraic Multigrid Methods (AMG) for the efficient solution of fully implicit formulations in reservoir simulation. In *SPE Reservoir Simulation Symposium, 26-28 February, Houston, Texas, U.S.A.* (p. SPE–105832–MS). http://doi.org/10.2118/105832-MS

Trangenstein, J. A., & Bell, J. B. (1989). Mathematical Structure of the Black-Oil Model for Petroleum Reservoir Simulation. *SIAM Journal on Applied Mathematics*, *49*(3), 749–783. http://doi.org/10.1137/0149044

Trottenberg, U., Oosterlee, C. W., & Schüller, A. (2001). *Multigrid*. Academic Press.

Wallis, J. R. (1983). Incomplete Gaussian Elimination as a Preconditioning for Generalized Conjugate Gradient Acceleration. In *SPE Reservoir Simulation Symposium, 15-18 November, San Francisco, California* (p. SPE–12265–MS). http://doi.org/10.2118/12265-MS

Wallis, J. R., Kendall, R. P., & Little, T. E. (1985). Constrained Residual Acceleration of Conjugate Residual Methods. In *SPE Reservoir Simulation Symposium, 10-13 February, Dallas, Texas* (p. SPE–13536–MS). Society of Petroleum Engineers. http://doi.org/10.2118/13536-MS

Weinstein, H. G., Chappelear, J. E., & Nolen, J. S. (1986). Second Comparative Solution Project: A Three-Phase Coning Study. *Journal of Petroleum Technology*, *38*(03), 345–353.