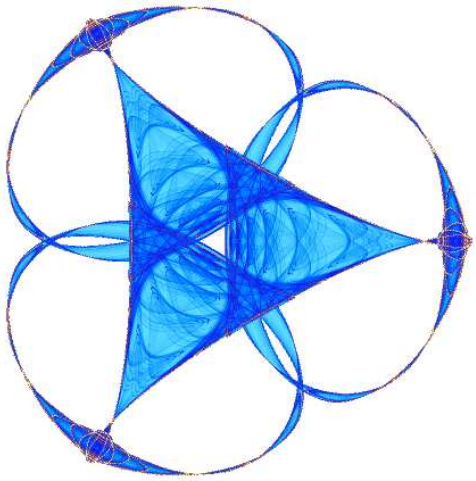
FAST COMPUTATIONAL METHODS FOR RESERVOIR FLOW MODELS

By

Teng Chen, Nicholas Gewecke, Zhen Li,
Andrea Rubiano, Robert Shuttleworth, Bo Yang,
and
Xinghui Zhong

IMA Preprint Series # 2286

(November 2009)



INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA
400 Lind Hall
207 Church Street S.E.
Minneapolis, Minnesota 55455–0436
Phone: 612/624-6066 Fax: 612/626-7370
URL: http://www.ima.umn.edu

Fast Computational Methods for Reservoir Flow Models

Teng Chen* Nicholas Gewecke† Zhen Li‡ Andrea Rubiano§
Robert Shuttleworth¶ Bo Yang ‖ Xinghui Zhong**

August 23, 2009

Abstract

Numerical reservoir simulation models are used in the energy industry to forecast the behavior of hydrocarbon reservoirs and connected surface facilities over long production periods. The simulation of oil and gas reservoirs requires the construction and solution of large, sparse linear systems with millions or tens of millions of unknowns at each step of a Newton iteration. The problem size and complexity of these models is growing faster than advances in computer hardware, therefore more accurate and robust numerical techniques are necessary to reduce simulation turnaround time. In this paper, we compare the computational performance of a Nonlinear Newton-Krylov solver with Inexact Newton-Krylov and Quasi-Newton methods for a compressible, three-phase fluid model with an incompressible rock model. The coupled system of nonlinear partial differential equations (PDEs) that describe fluid flow are discretized on a structured, rectangular grid using finite volumes. An implicit method is used to solve for updates to the pressure and saturations at each time step. Rock properties, like permeability and porosity are generated from the top layer of the SPE10 model problem. We conclude with results comparing different nonlinear and linear solvers and from an optimization strategy for well placement that we experimented with as part of the Institute for Mathematics and its Applications (IMA) summer industrial workshop.

1 Introduction

Numerical reservoir simulation is used in the oil and gas industry to model the flow of fluids (e.g. hydrocarbons, water, vapor) through porous rock space. The ability to accurately

*Department of Mathematics, University of Central Florida tchen@mail.ucf.edu
†Department of Mathematics, University of Tennessee, gewecke@math.utk.edu
‡Department of Mathematics, Iowa State University zhenli @iastate.edu
§Department of Mathematics, Purdue University, arubiano@purdue.edu
¶ExxonMobil Upstream Research Co., robert.r.shuttleworth@exxonmobil.com
‖School of Mathematics, University of Minnesota, yangbo@umn.edu
**Department of Applied Mathematics, Brown University, xinghui_zhong@brown.edu

and quickly simulate the performance of an oil or gas reservoir over time is important for the petroleum industry to make accurate business decisions. Unfortunately, the physical properties of an oil/gas reservoir are not well-known in advance. Therefore, reservoir engineers create multiple reservoir simulation models to examine the sensitivity of the geology, fluid properties, and other physical conditions to evaluate multiple development strategies.

A set of nonlinear, partial differential equations that govern the flow of the various phases of the fluid (oil, gas, and water) is utilized. A derivation of the governing equations motivated by [1, 7] can be found in Section 2. These equations are discretized in space using a finite volume method and in time using a fully implicit strategy (which is discussed further in Section 2.1). At each time step a nonlinear system is solved that requires solving a large, sparse linear system to generate update values for the unknown quantities in question (in our case, pressure and saturations). Solving a linear system can be computationally expensive, and very time consuming. The quality of the nonlinear solver is directly related to the CPU time required to solve the linear system. Ideally, a rapidly converging nonlinear solver and a fast linear solver would be utilized. This paper is concerned with that particular issue. We investigate different nonlinear solvers (Exact Newton and Inexact Newton [3]) along with iterative linear solvers and preconditioning to try to minimize the computational time for a reservoir simulation. In Section 3, we describe the example problem that we used for our numerical experiments. Section 4 describes an optimization strategy used to create a sequence of different problems to test our numerical algorithms. The different choices of nonlinear solvers that we tested is detailed in Section 5 and the linear solvers (along with spectral properties of the coefficient matrices) used to solve for the Newton update is discussed in Section 6. In Section 7 we conclude with some remarks about our findings from this workshop.

2 Reservoir Simulation

Henry Darcy, a French engineer, established a relationship relating the volumetric flow rate, Q, through porous media. The relationship is:

$$Q = -\frac{KA\Delta p}{\mu L}$$

where K is the permeability of the rock, A is the area, Δp is the pressure drop, μ is the viscosity, and L is the length of the media. This relationship is known as Darcy's Law and states that the flow rate is proportional to the permeability, area, and pressure drop; while, the flow rate is inversely proportional to the viscosity and length of the media [1].

In three dimensions the differential form of Darcy's Law becomes:

$$\mathbf{v} = \frac{Q}{A} = -\frac{K}{\mu}\nabla p$$

where K is a symmetric, positive definite tensor. Applying the divergence theorem generates

the equation that represents single phase flow in three dimensions:

$$\nabla \cdot \frac{\rho k}{\mu} \nabla p = \frac{\partial}{\partial t}(\phi \rho)$$

We begin derivations for a three phase fluid model with incompressible rock, through the form of the phase continuity equation that includes compressibility [1]:

$$\frac{-\nabla \cdot (\rho_\theta v_\theta)}{\rho_\theta} = \phi \left(c_\theta \left(\frac{\partial}{\partial t} p_\theta \right) s_\theta + \left(\frac{\partial}{\partial t} s_\theta \right) \right) - \frac{q_\theta}{\rho_\theta}, \tag{1}$$

where ρ_θ represents density (which is a function of pressure), v_θ represents the velocity, s_θ represents saturation, q_θ represents the source term, c_θ represents the compressibility term and ϕ represents the porosity. Note that θ could either be oil (o), water (w) or gas (g) phases. We require that the saturations for all three phases must sum to one, therefore we have the following relationship:

$$s_g + s_w + s_o = 1. \tag{2}$$

and a similar relationship for capillary pressure:

$$p_{cow} = p_o - p_w \tag{3}$$

$$p_{cgo} = p_g - p_o \tag{4}$$

$$p_{cgw} = p_g - p_w = p_{cgo} + p_{cow} \tag{5}$$

Using these relationships, we can alter the three phase continuity equations to derive a system of three equations with three unknown variables (or vectors): p_w, s_g and s_w. These equations are:

$$\frac{-\nabla \cdot (\rho_w v_w)}{\rho_w} = \phi \left(c_w \left(\frac{\partial}{\partial t} p_w \right) s_w + \left(\frac{\partial}{\partial t} s_w \right) \right) - \frac{q_w}{\rho_w} \tag{6}$$

$$\frac{-\nabla \cdot (\rho_g v_g)}{\rho_g} = \phi \left(c_g \left(\frac{\partial}{\partial t} (p_w + p_{cow}(s_w) + p_{cog}(s_g)) \right) s_g + \left(\frac{\partial}{\partial t} s_g \right) \right) - \frac{q_g}{\rho_g} \tag{7}$$

$$\frac{-\nabla \cdot (\rho_o v_o)}{\rho_o} = \phi \left(c_o \left(\frac{\partial}{\partial t} (p_w + p_{cow}(s_w)) \right) (1 - s_g - s_w) + \left(\frac{\partial}{\partial t} (1 - s_g - s_w) \right) \right) - \frac{q_o}{\rho_o}. \tag{8}$$

After finite volume analysis, Darcy velocity approximation, simplications and discretization, we have:

$$\begin{aligned} F(t, s_w(t), s_g(t), p_w(t), q(t)) =& \Delta \tilde{D}^{-1}(Q_w - A_w p_w) \\ & - [(c_w s_w)_i](p_w^k - p_w^{k-1}) - (s_w^k - s_w^{k-1}) \end{aligned} \tag{9}$$

$$\begin{aligned} G(t, s_w(t), s_g(t), p_w(t), q(t)) =& \Delta \tilde{D}^{-1}(Q_o - A_o(p_w + p_{cow})) - \\ & - [(c_o(1 - s_g - s_w))_i](p_w^k - p_w^{k-1} + p_{cow}^k - p_{cow}^{k-1} \\ & - (s_g^{k-1} - s_g^k + s_w^{k-1} - s_w^k) \end{aligned} \tag{10}$$

$$\begin{aligned} H(t, s_w(t), s_g(t), p_w(t), q(t)) =& \Delta \tilde{D}^{-1}(Q_g - A_g(p_w + p_{cow} + p_{cog})) - \\ & - [(c_g s_g)_i](p_w^k - p_w^{k-1} + p_{cow}^k - p_{cow}^{k-1} + p_{cog}^k - p_{cog}^{k-1}) \\ & - (s_g^k - s_g^{k-1}) \end{aligned} \tag{11}$$

Numerically, we approximate the solution to this set of coupled, nonlinear equations using Newton's method (more information about Newton's method can be found in Section 5). This requires the creation of the Jacobian matrix, so we need to take derivatives of F, G and H with respect to the unknown variables p_w, s_w and s_g [1, 7]. Due to space constraints we omit the partial derivatives required for the computation of the Jacobian used in Newton's method. We state that the Jacobian matrix is of the form:

$$J(p_w, s_w, s_g) = \begin{bmatrix} \frac{\partial F}{\partial p_w} & \frac{\partial F}{\partial s_w} & \frac{\partial F}{\partial s_g} \\ \frac{\partial G}{\partial p_w} & \frac{\partial G}{\partial s_w} & \frac{\partial G}{\partial s_g} \\ \frac{\partial H}{\partial p_w} & \frac{\partial H}{\partial s_w} & \frac{\partial H}{\partial s_g} \end{bmatrix}$$

and that updates for Newton's method are calculating from solving the system of equations

$$J_k(p_w, s_w, s_g)x_k = R$$

where R is the term containing entries F, G, and H, from equation (9).

2.1 IMPSAT

Our workflow for generating updated pressure and saturations is from the IMPSAT (implicit pressure and saturation) procedure described in [6].

1. Load initial phase saturations s_θ^0.
2. For $k = 1, ...K$ do:
   (a) Set well rates q^k
   (b) Solve the discretized state equations for p^k, s_θ^k:

$$\begin{pmatrix} f \\ g \\ h \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

solve $\begin{pmatrix} f \\ g \\ h \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ using Newton's Method where $s_o^k = 1 - s_g^k - s_w^k$.

3 Simulation Model

For the computational experiments in this study, we use a simulation model with porosity and permeability generated from the top layer of the SPE10 model problem (see Figure 1). This problem is 60 units in the $x-$direction and 220 units in the $y-$direction. We use a structured, uniform grid with injecting wells placed at each corner of the simulation domain and producing wells are placed randomly along the interior of the simulation domain. We run sensitivities with this model using a compressible, three phase reservoir simulator with capillary pressure and an incompressible rock model written in Matlab. The permeability field varies by five orders of magnitude, while the porosity varies from 0 to 0.5.

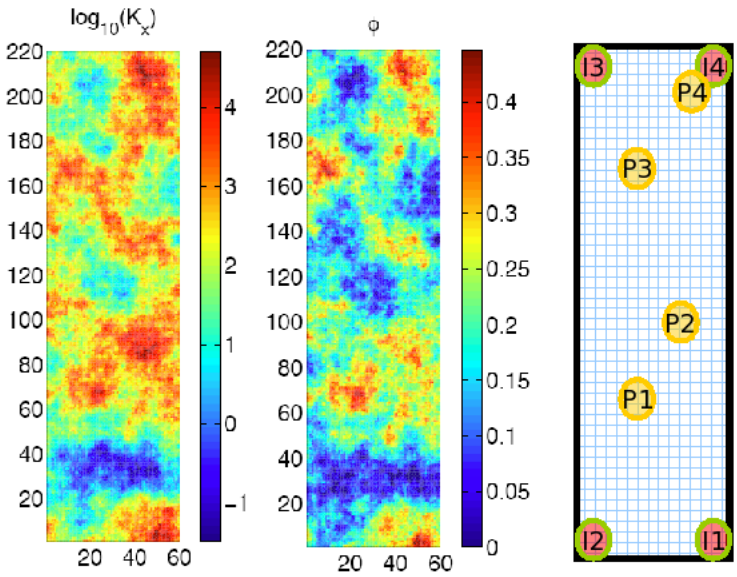


Figure 1: Porosity and permeability fields for the SPE10 Top Layer.

4 Optimization

An algorithmic approach is needed for well placement because it is very difficult to tell a priori the best location for producing/injecting wells in the reservoir that maximizes sweep efficiency. One could randomly place wells and see how each set of random well distributions performs, but this would be a very computationally expensive procedure, and it would be possible that a small change in the position of one of the wells could have a significant impact on production over the life of the oil field.

We discuss a framework for performing well placement optimization, and then we present results for a couple of test cases, to demonstrate the value of the optimization procedure.

4.1 Framework

Our goal is to maximize the profit which is based on the production rates of each producer, the injection rates at each injector, the price at which the oil and gas can be sold, the cost to inject water, and the costs of constructing a producer or injector. The general idea is to maximize the revenue from producing oil and gas, while minimizing the costs of injecting water and building wells. Thus, we cannot just build a very large number of wells, which by itself would maximize the production. We also need to know the time-span that we want to optimize over.

Combining these pieces, we created the following objective function,

$$\max F = \sum_{i=1}^{N_P} \int_0^T \left[C_o x_o^i(t) + C_g x_g^i(t)\right]\, dt - \sum_{j=1}^{N_I} \int_0^T C_w x_w^j(t)\, dt - C_{WP} N_P - C_{WI} N_I,$$

where N_P and N_I are the number of producers and injectors, C_o and C_g are the prices at which oil and gas are sold, C_w is the cost of injecting water, C_{WP} and C_{WI} are the costs associated with constructing producers and injectors, x_o^i and x_g^i are the production rates of oil and gas for producer i, and x_w^j is the injection rate of injector j. We will denote the site of producer i by (y_1^i, y_2^i) and injector j by (z_1^j, z_2^j), and the vector of all of these values will be $\mathbf{y}$. F denotes the profit generated by the field.

This must be optimized subject to the constraints

$$N_P, N_I \in \mathbb{Z}_+,\ N_P \geq 1,\ N_I \geq 0,\ x_o^i, x_g^i, x_w^i \geq 0$$
$$(y_1^i, y_2^i) \in D_1, (z_1^j, z_2^j) \in D,$$

where D is the spatial grid of the simulator and D_1 is a restriction of the grid by some number of gridpoints along each side, because we want the producers to be away from the edge of the domain. To make the problem more realistic, we cap the producing and injecting well rates.

This optimization problem is complicated by two important issues. First, analytically calculating the gradient or Hessian matrix of F is impractical, as the x values are generated by the solutions of the partial differential equations which govern the system. Approximating these numerically would also be computationally expensive, especially for the Hessian matrix. Secondly, we cannot simultaneously perform the optimization over the well sites and the number of wells.

The first problem cannot be avoided, although we could pick a method such as gradient ascent to avoid trying to compute the Hessian. Calculating the Hessian might be possible, but given the computational effort our efforts are better utilized in other areas.

The second problem can be alleviated by a careful choice of procedure. We start with a fixed number of producers (N_P) and injectors (N_I), then we maximize the objective function for the well sites with the numbers of producers/injectors fixed. Then we repeat for neighboring points on an (N_P, N_I) grid. For instance, one could solve the optimization problem for the four immediate neighbors, or for the eight surrounding values. Of those values we pick the largest one and then repeat the procedure. To save on overall computational effort, the maximized object values $F^*(N_P, N_I)$ are stored, along with the set of values which defines the well locations. This amount of storage is not very large, especially in comparison to the effort to recompute those values.

The objective function can also be modified to reflect present values or approximate predictions of future prices for the various components. We restrict our focus to the stated case.

4.2 Results

For our test problems, we fix the number of injectors at $N_I = 2$. The injectors are also fixed at the southwest and northeast corners of our computational domain. We solve the optimization problems for $N_P = 1$ with $T = 200$ and $N_P = 2$ for $T = 100$ days, which is a very small time-span for this type of problem. This was done to be able to demonstrate how the method works. We assume that $C_o = 70$ dollars per barrel, which is the approximate price for a barrel of West Texas Intermediate crude oil at the culmination of this workshop in August 2009. We set $C_g = 0$, which essentially means that the gas is flared, and $C_w = 0$, for simplicity. Finally, we let $L_P = L_I =$\$2 million. Due to the short time periods, our objective F is negative. Due to the assumptions, the optimization problem is seen as maximizing oil production.

For both of these problems, we follow a similar procedure. First, we randomly select the initial positions of the producing wells. Then we calculate the value of F for those positions. Next, a numerical approximation of the gradient is computed using the points $(y_1^i + 1, y_2^i)$ and $(y_1^i, y_2^i + 1)$ for each i. If the gradient has an infinity norm greater than one, we normalize the gradient by dividing by the infinity norm. Then, we search in the gradient direction, calculating F at $\mathbf{y} + 2^k \nabla F$ for $k = 3, 2, 1, 0$ (in that order). Such a point will likely not be a grid point, so we round the entries to move the point to a grid point. We stop the search when the value of F at the test point is greater than the value of F at the current point. We repeat this process until the search fails to find an improvement, at which time we stop and accept the current point.

Only one trial with $N_P = 1$ was performed because it is not realistic. The initial point was $(45, 195)$, where the total oil production was calculated as 1863.26 barrels. This corresponds to $F = -5,869,571.80$. The final point was $(49, 190)$ with a total production of 1874.19 barrels, which corresponds to $F = -5,868,806.70$.

Three trials were performed with $N_P = 2$. The data for these trials is listed in Table 1. The first trial has the most dramatic change in well placement and total output. Figure (2)

Initial Well Placement	$(26, 197), (44, 206)$	$(37, 12), (47, 201)$	$(38, 164), (42, 87)$
Initial Output	1838.55	1871.92	1869.33
Initial F	$-7,871,301.50$	$-7,868,965.60$	$-7,869,216.90$
Final Well Placement	$(39, 175), (43, 207)$	$(42, 5), (44, 208)$	$(37, 162), (42, 87)$
Final Output	1880.71	1878.46	1874.54
Final F	$-7,868,350.56$	$-7,868,507.80$	$-7,868,782,20$

Table 1: Initial and final data for three trials of the optimization method with $N_P = 2$

shows the initial and final well placement for this case on top of plots of the permeability and porosity fields.

The well closest to the northeast corner does not move very much, but the other well moves quite a bit. That well moves from a location with low permeability and porosity to a

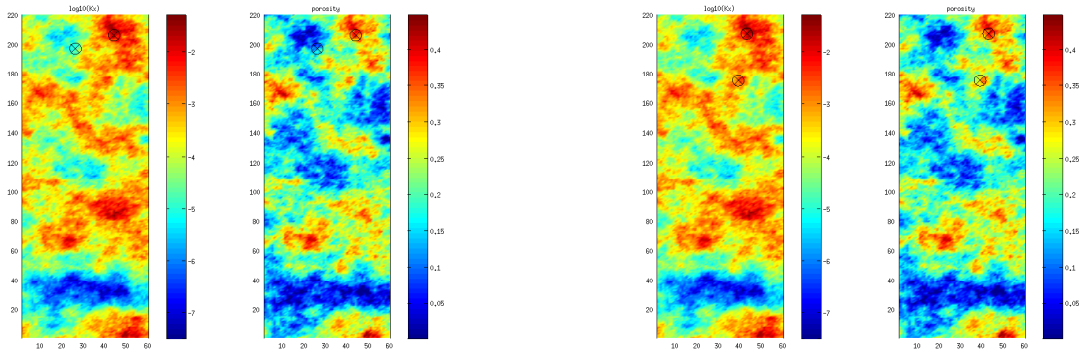


Figure 2: Well positions at the beginning and end of the optimization procedure. The well locations are indicated by black circles with X's inside.

location with higher permeability and porosity.

5 Nonlinear

In a reservoir simulation, the solution of a set of nonlinear equations is required at each time step. For typical grids in the oil and gas industry, this requires a significant amount of computational effort, computer memory, and time. One of the goals of this workshop is to experiment with different types of nonlinear solvers to reduce the computational effort required to solve the governing equations. Traditionally, a classical exact Newton method is used to calculate updates to the nonlinear system. In this paper we explore two different approaches to numerically solve the nonlinear equations that arise after discretization of the governing equations. They are:

- Inexact Newton's method
- Quasi-Newton methods based on Broyden's method

We begin by describing each of these methods, then summarize this section with results we obtained from implementing these numerical techniques in our Matlab simulator.

5.1 Newton's Method

Consider the system of nonlinear equations

$$F(x) = 0 \tag{12}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear mapping such that

i. There exists an $x^* \in \mathbb{R}^n$ such that $F(x^*) = 0$.

ii. F is continuously differentiable in a neighborhood of x^*.

iii. $F'(x^*)$ is nonsingular.

A classical algorithm and perhaps the best known method for finding successively better approximations to the zeroes of a function is Newton's method. The algorithm is:

Given an initial guess x_0, we compute a sequence of steps $\{s_k\}$ and compute a sequence $\{x_k\}$ as follows:

```
FOR  k=0 STEP 1 UNTIL Convergence DO
     Solve F'(x_k)s_k = -F(x_k)
     Set   x_{k+1} = x_k + s_k.
```

Typically, Newton's method converges quadratically for a sufficiently good initial guess. We implemented a Newton-Krylov method for our simulations using GMRES (generalized minimal residual method), preconditioned with an incomplete LU matrix factorization to solve the linear equation for the Newton update. We computed the Jacobian matrix analytically and did not approximate it. Our objective was to compare the number of iterations taken by a Newton-Krylov method with a fixed stopping tolerance with the iteration counts obtained from an inexact Newton method with an adaptive stopping tolerance (see Section 5.3).

5.2 Inexact Newton-Krylov

In last section, we introduced the classical Newton method. The most computationally expensive part of this method is solving the linear equations

$$F'(x_k)s_k = -F(x_k) \tag{13}$$

at each step of the nonlinear iteration for the update, s_k. Solving this equation accurately may not be justified when x_k is far from the solution, x^*, therefore it is reasonable to solve (13) approximately. Such 'Inexact Newton-Krylov' methods offer a trade-off between accuracy with which the Newton equations are solved and the amount of computational work per iteration.

An *Inexact Newton method* is an extension of classical Newton's method for approximating x^* as follows:

$$\|F'(x_k)s_k + F(x_k)\| \leq \eta_k \|F(x_k)\|. \tag{14}$$

Algorithm IN: Inexact Newton Method

Given x_0 (Typically, x_0 is zero or from the previous iterate.
FOR k=0 STEP 1 UNTIL "Convergence" DO
Find $\eta_k \in [0, 1)$ and s_k which satisfies (14)
Set $x_{k+1} = x_k + s_k$.

Note that (14) expresses both a certain reduction in the norm of $F(x_k)+F'(x_k)s_k$, the local linear model of F, and a certain accuracy in solving the *Newton equation*, $F'(x_k)s_k = -F(x_k)$, the exact solution of which is the *Newton step*. In many applications, notably Newton iterative or truncated Newton methods, each η_k is specified first, and then an s_k is determined so that (14) holds. The role of η_k is to force $\|F(x_k) + F'(x_k)s_k\|$ to be small in a particular way; therefore, η_k is often called a *forcing term* [3].

The choices of the forcing terms are to achieve desirably fast local convergence and to avoid oversolving the linear system. All of the proposed choices incorporate information about F but are scale independent so they do not change if F is multiplied by a constant. We examine three choices for η_k from work by [3]. They are:

- *Choice 1.1*:

$$\eta_k = \frac{\|F(x_k) - F(x_{k-1}) - F'(x_{k-1})s_{k-1}\|}{\|F(x_{k-1})\|}, \quad k = 1, 2, \cdots \tag{15}$$

- *Choice 1.2*:

$$\eta_k = \frac{\|F(x_k)\| - \|F(x_{k-1}) + F'(x_{k-1})s_{k-1}\|}{\|F(x_{k-1})\|}, \quad k = 1, 2, \cdots \tag{16}$$

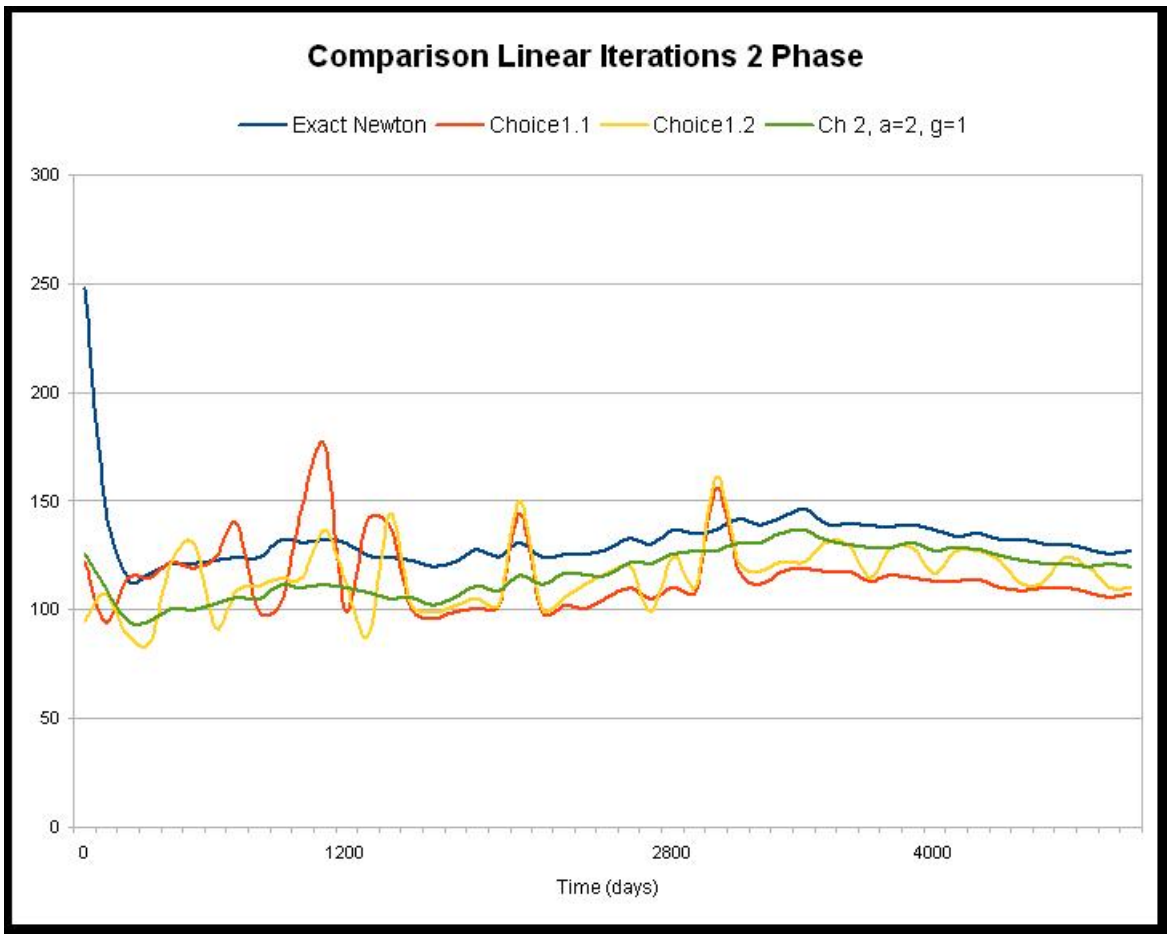
- *Choice 2*: Given $\gamma \in [0, 1],\ \alpha \in (1, 2]$,

$$\eta_k = \gamma \left(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|} \right)^{\alpha}, \quad k = 1, 2, \cdots \tag{17}$$

5.3 Comparison between Exact and Inexact Newton

We apply Alogrithm IN to our two phase problem described in Section 3. The results comparing each forcing term to an exact Newton method for a simulation that was run to 5000 days are summarized in the following table. MNI and MLI are averages of the numbers of inexact Newton steps and linear iterations, respectively. GMNI and GMLI are geometric means of inexact Newton steps and linear iterations, respectively. I is the total number of linear iterations and P is the percentage of improvement with respect to the exact Newton's method.

η_k choice	MNI	GMNI	MLI	GMLI	I	P
0 (exact)	3.06	3.04	131.43	131.03	26285	
Choice 1.1	3.34	3.30	115.15	114.17	23029	12.39%
Choice 1.2	3.18	3.16	115.50	114.49	23100	12.12%
Choice 2, $\alpha = 2$, $\gamma = 1$	3.06	3.05	117.33	116.80	23466	10.72%
Choice 2, $\alpha = 2$, $\gamma = 0.9$	3.06	3.04	117.93	117.41	23585	10.27%
Choice 2, $\alpha = 2$, $\gamma = 0.5$	3.02	3.01	119.78	119.17	23955	8.86%
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}$, $\gamma = 1$	3.23	3.19	117.14	116.49	23427	10.87%
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}$, $\gamma = 0.9$	3.19	3.15	116.35	115.74	23270	11.47%
Choice 2, $\alpha = \frac{1+\sqrt{5}}{2}$, $\gamma = 0.5$	3.07	3.06	116.90	116.36	23379	11.06%



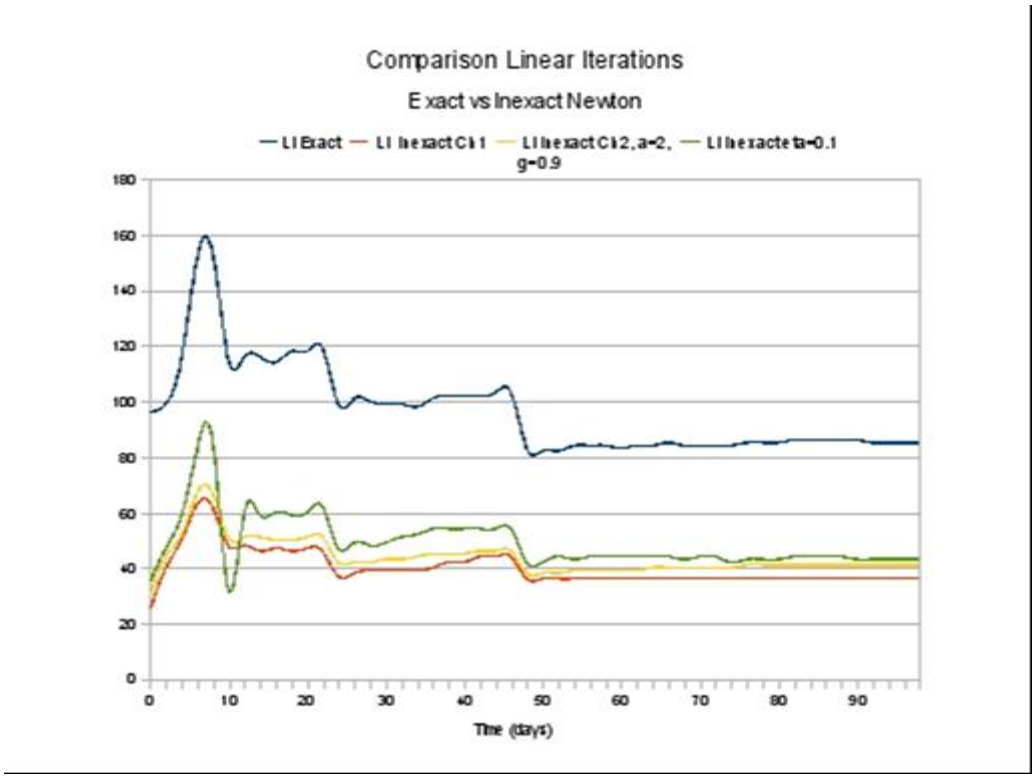
From the table above, the best overall performance was obtained from Choice 1.1 and Choice 1.2. Taking $\gamma = 0.5$ in Choice 2 resulted in significantly less efficiency with $\alpha = 2$.

The results for Choice 2 illustrate that more aggressive choices of the forcing terms. i.e., choices that are smaller or result in faster asymptotic convergence, may decrease the number

of inexact Newton steps, but through oversolving the linear system may also lead to more linear iterations. Less aggressive choices may reduce the number of linear iterations, but may also result in increased numbers of inexact nonlinear iterations.

For the three-phase model (simulation run for a total time of 100 days), we found a significant improvement in the number of linear iterations when an inexact Newton method is applied. Our results are summarized in the following table:

η_k choice	MNI	GMNI	MLI	GMLI	I	P
0 (exact)	4.76	4.67	96.76	95.52	4877	0%
Choice 1.2	3.76	3.65	39.82	39.84	1991	58.73%
Choice 2, $\alpha = 2$, $\gamma = 0.9$	3.76	3.65	43.7	43.28	2185	54.59%
0.1	3.76	3.75	48.98	48.15	2487	49.36%



From the data above we can see that the general shape of the graph representing the number of linear iterations for each type of forcing term does not change, but the use of an inexact Newton method reduces the number of linear iterations. Choice 1 is the best option in our simulation, where we were able to obtain an improvement of 58.73% for the Newton method with fixed tolerance for the three phase model. Even though the number of linear

iterations is different between the two methods, the solution, i.e. cumulative gas , water and oil production, is the same. Therefore, the use of these 'inexact' nonlinear strategies have no impact on the accuracy of the solution, but only the computational performance.

5.4 Broyden's Quasi-Newton Method

Broyden's Quasi-Inexact Newton method uses an approximation to the Jacobian to generate a sequence of successive approximations to the nonlinear problem. In general, this method provides superlinear convergence (compared with quadratic convergence for Newton's method), but do not require the calculation of the full Jacobian. In general, a Quasi-Newton method uses some damping parameter (between 0 and 1) to reduce the step size from a full step to a partial step. That is, after obtaining $s_n = -(J_n)^{-1}F_n$ from the equation

$$J_n s_n = -F_n,$$

the next step in the iteration is computed as $x_{n+1} = x_n + \lambda s_n$, where λ lies between 0 and 1. The damping factor, λ, helps prevent oversolving, and thus may increase the rate of convergence.

We tested Broyden's Method, which is a secant-like method. It replaces the Jacobian matrix J_n with an approximate one, B_n, that satisfies

$$B_n s_n = -F_n.$$

While this approximation of the real Jacobian might sacrifice some accuracy, it saves the expensive computation (and storage) of the Jacobian. At each iteration, the Jacobian is updated by

$$B_{n+1} = B_n + \frac{(y - B_n s)s^T}{s^T s},$$

where $y = f(x_{n+1}) - f(x_n)$ and $s_n = x_{n+1} - x_n$. B_n^{-1} can be updated directly using the Sherman-Morrison formula (also known as Woodbury formula) as below,

Sherman-Morrison Assuming C and $I + BVC^{-1}U$ are two nonsingular matrices, then

$$(C + UBV)^{-1} = C^{-1} - C^{-1}U(I + BVC^{-1}U)^{-1}BVC^{-1}.$$

Consequently, we have an update formula for B_n^{-1}'s. In our study, we will use a line search to find the damping term for each step. The drawbacks to Broyden's method are that:

- The line search may fail.
- It is sensitive to the initial data.

When testing this technique inside of our Matlab simulator we found that both the line search failed and that the initial guess we gave to Broyden's method was not close enough to the original. Therefore, Broyden's method did not converge, so we conclude that its not suitable for our problem.

6 Linear Solvers

During a Newton iteration, a large, sparse linear system of equations need to be solved. To speed up the solution process, efficient linear solvers are required. We focus our efforts on iterative solvers because the memory costs associated with sparse direct solvers is prohibitive for current three dimensional reservoir simulation models. We have tried iterative methods, such as GMRES and BICGStab, with preconditioning and looked into using Krylov reuse methods for added efficiency.

6.1 Comparison Iterative Methods with Preconditioners

Each step of a nonlinear solve involves the solution of a linear system. Thus, an efficient linear solver will improve the computational speed of the simulation.

We began by trying linear solvers without any preconditioning. This effort was largely unsuccessful, as convergence was either nonexistent or very slow. The methods tested were GMRES, TFQMR, and BiCGStab. GMRES was restarted every 50 iterations. As we will discuss later, the matrices involved in the linear solve are ill-conditioned and have poor eigenvalue clustering, so creating effective preconditioners for this systems was required.

Preconditioning greatly improves the performance of both GMRES and TFQMR. However, BiCGStab still suffered from poor convergence rates, so it was removed from consideration. The initial preconditioning matrices were incomplete LU decompositions with a drop tolerance of 10^{-4}. The success of this preconditioner led to the investigation of other preconditioning methods, as well as the effects of varying the drop tolerance for the incomplete LU method. One alternative to incomplete LU that we attempted was Jacobi preconditioning. This method uses the diagonal of the matrix as a preconditioner. Unfortunately, this matrix is either singular or very close to singular, so the method failed.

The effects of varying the drop tolerance for the incomplete LU decomposition were reasonably similar for GMRES and TFQMR. Increasing the drop tolerance to 10^{-3} was effective in improving the speed of the overall code, even though both linear solvers required a higher number of iterations to converge. Increasing the drop tolerance further continued to improve the speed, but the number of iterations of the nonlinear solver began increasing. TFQMR began exhibiting stagnation before reaching the convergence tolerance, and GMRES began requiring numerous restarts. It appears that the best choice of drop tolerance for this problem is approximately 10^{-3}. Refer to Table 2 for details for TFQMR, and to Figures 3 and 4 for a comparison of TFQMR and GMRES. The simulation time was 1000 days, with a time-step of 25 days. The norm of the relative residual (or tolerance) used for convergence was 10^{-5}.

Choosing whether to use GMRES or TFQMR is somewhat ambiguous. TFQMR seems to perform slightly better than GMRES, but applying the methods to the full three-dimensional

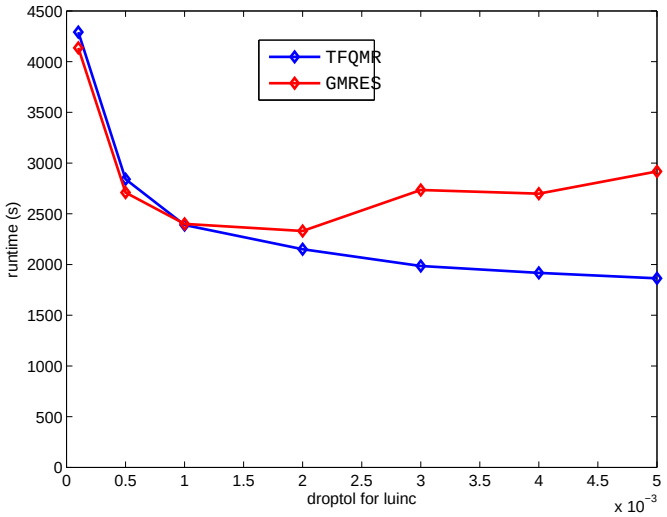


Figure 3: Runtimes (seconds) in Matlab for GMRES and TFQMR plotted against drop tolerance for the incomplete LU factorization

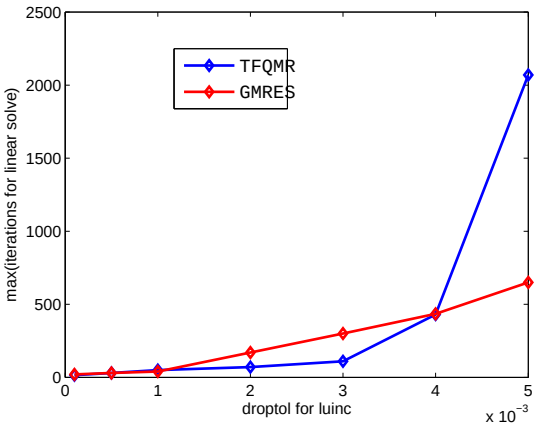


Figure 4: Maximum iterations per solve for GMRES and TFQMR plotted against drop tolerance for the incomplete LU factorization

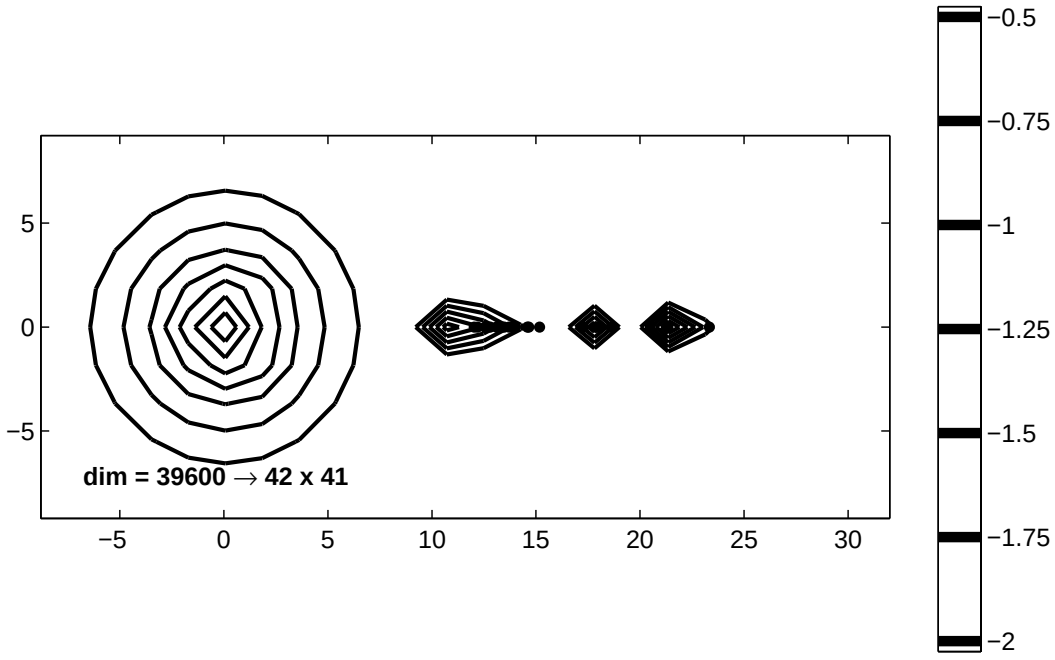
Drop Tolerance	Runtime (s) in Matlab	Maximum iterations per solve
10^{-4}	4291	15
5×10^{-4}	2841	30
10^{-3}	2390	50
2×10^{-3}	2151	70
3×10^{-3}	1985	110
4×10^{-3}	1917	430
5×10^{-3}	1864	2070

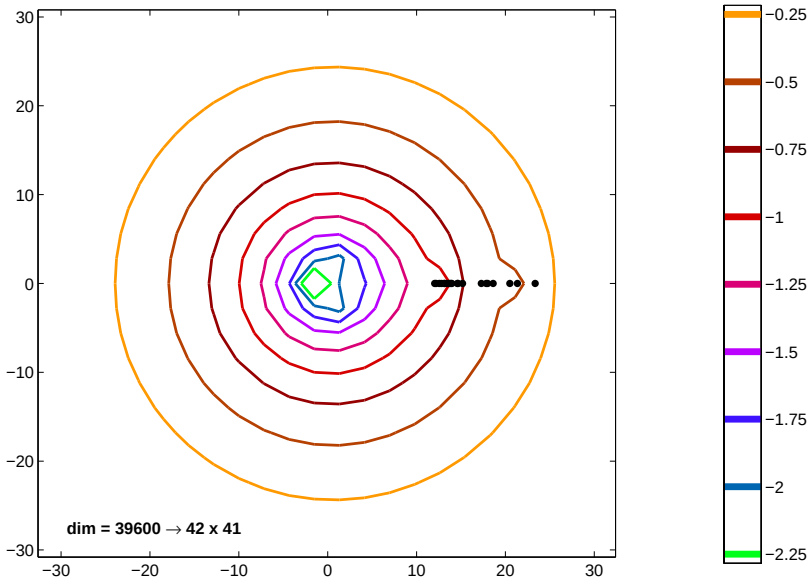
Table 2: Computation details for TFQMR

multi-phase flow problem may change that performance. The major concern with GMRES is that running it without restarts is impractical due to memory concerns, as it requires storing a set of orthonormal vectors. Restarting helps with this, but affects the convergence rate, and can even induce stagnation of the method. TFQMR avoids this issue, as it does not use as much storage as GMRES, but as we have seen, it can also stagnate.

6.2 Spectral properties

Since the coefficient matrix is large and sparse, we investigated the spectral properties of this matrix. In the following figures, we show some graphs of the pseudospectra, where we consider the only 20 largest magnitude eigenvalues.





The pseudospectra is significant because: The eigenvalues of a matrix operator $\mathbb{A}$ are the roots of the characteristic polynomial for $\mathbb{A}$. If $\mathbb{A}$ is highly non-normal, analytic functions of $\mathbb{A}$ such as polynomials are very sensitive to small perturbation of $\mathbb{A}$. The closer the smallest (in magnitude) eigenvalue is to 0, and the more non-normal the operator. From the above two graphs, we see the high singularity of the coefficient matrix. These two graphs show that how the contour lines for different values of ϵ approach a set of disjoint small regions around the eigenvalues of coefficient matrix.

6.3 Krylov Subspace Reuse

When solving a nonlinear problem using a Newton-Krylov method, a sequence of linear systems must be solved to calculate an update to the Newton step. Therefore, it seems promising to try using a Krylov subspace reuse method to speed up the convergence of the iterative method. During this workship, we compared a Krylov reuse method, [9], with ordinary GMRES preconditioned with an incomplete LU decomposition.

The method developed in [9] could be applied to the solution of sequences of general matrices without assuming these matrices are pairwise close or convergent, hence this method should be very adaptive especially for our problem which has pairwise close matrices. In addition, the methods discussed in this report should be adopted to get different subspaces, based on the computed information from the matrix and the context of the problem.

Before describing the Krylov reuse method, recall the Krylov Subspace (for a linear system $Ax = b$) is

$$K^r(b) = \text{span}\{b, Ab, Ab^2, ...Ab^r\}.$$

The Arnoldi recurrence in GMRES leads to the following relation, which we denote as the Arnoldi relation

$$AV_m = V_{m+1}\underline{H}_m,$$

where $V_m \in \mathbb{C}^{n\times m}$, and $\underline{H}_m \in \mathbb{C}^{(m+1)\times m}$ is upper Hessenberg. Let $H_m \in \mathbb{C}^{m\times m}$ denote the first m rows of $\underline{H}_m$.

In the previous simulations, the Krylov spaces generated during each solve of the linear system are discarded after each linear solve. In this section, we describe a strategy to save a subspace of the Krylov vectors to reduce the number of iterations for solving the next system. The basic idea about recycling the Krylov subspace is that we pick the recycled subspace, and during each cycle, we minimize the residual over this subspace, keeping the orthogonality with the image of this space in the Arnoldi recurrence.

We know there are the other ways of solving the sequence of linear problem, but most of the other techniques have more strict solution requirements. (i.e. require that the systems are available simultaneously).

A good mechanism is needed to find the best vectors to keep. In general, we truncate when we lose orthogonality, and will result in a suboptimal correction. Basically,the worst-case bound on the convergence is reduced by improving the spectrum. The method used in this workshop, GCRO-DR, is combination of GMRES-DR and GCRO to solve the sequence of the linear systems. After computing the i^{th} system of the equation, GCRO-DR retains k approximate eigenvectors, as follows:

$$\tilde{Y_k} = [\tilde{y_1}, \tilde{y_2}, ...\tilde{y_k}]$$

In our results, we found that the GCRO-DR improves the computation, even for a small sequence of linear systems by significantly reducing the number of matrix-vector products when compared with GMRES.

Also, when the dimension of the recycle subspace increases, the elapsed time decreases, so the dimension of the subspace does effect our result. Since we only tried the three phase case, which means that the size of the systems may not be big enough. The difference between the GRCO-DR and GMRES is small , but as the size goes up, the advantage of the invariant subspace we selected grows [9].

Tolerance	GMRES	Krylov Reuse
10^{-3}	45	34
	2140.27	2013.8
10^{-4}	45	52
	2073.93	2031.67

This algorithm did not work well when the tolerance was set at $1e^{-}4$. We have two possible reasons for this unexpected result, one is the size of the sequence of the system. The GRCO-DR method may perform better with a larger number of systems to solve. Another idea is that by including more information about the problem itself, we could make small changes to accommodate the algorithm. Both of these possibilities are worth considering in future work.

7 Conclusion

In this paper, we began by describing (from first principles) a compressible, three phase fluid model with incompressible rock described by a coupled system of nonlinear PDEs. The system of coupled nonlinear PDEs that describe fluid flow are discretized spatially using a finite volume technique on a structured, rectangular grid. The state variables (pressures and saturations) are defined at node centers and the fluxes represent connections from one node to another. We tried different linear and nonlinear solvers to speed up the calculations. For linear solvers, we tried GMRES, BICGStab and TFQMR methods coupled with an ILU preconditioner. We investigated the spectral properties and compared a Krylov reuse method with GMRES. The preliminary results for this technique look promising. For nonlinear solvers, we tried a Newton method, Inexact Newton method and Broyden line search techniques. The inexact Newton method saved 50% percent of the linear iterations when compared to the exact Newton method for a three phase model. Finally, we consider a optimization problem with respect to the numbers of injectors, producers and placements to maximize profit for a given oil field.

8 Acknowledgment

We would like to express our deepest appreciation to the organizers of this workshop and the IMA for hosting this wonderful and successful workshop, without their help this meeting would not have been possible. We thank the generosity of NSF and the University of Minnesota for all of their thoughtful arrangements and help. We would like to thank our mentor, Robert Shuttleworth, for being such an interesting, resourceful and helpful person, whom we have learned a lot from. We are grateful to ExxonMobil and the other industrial sponsors which gave us this opportunity to learn from all the mentors. Finally, we want to acknowledge our teammates and students from other groups for their support and fellowship throughout the workshop.

References

[1] Aziz, K. and Sattari, A. *Petroleum Reservoir Simulation*. Chapman and Hall, 1979.

[2] Dembo, Ron S.; Eisenstat, Stanley C.; Steihaug, Trond *Inexact Newton methods*. SIAM J. Numer. Anal. 19 (1982), no. 2, 400–408.

[3] Eisenstat, Stanley C.; Walker, Homer F. *Choosing the forcing terms in an inexact Newton method*. Special issue on iterative methods in numerical linear algebra (Breckenridge, CO, 1994). SIAM J. Sci. Comput. 17 (1996), no. 1, 16–32.

[4] Kelley, C.T. *Solving Nonlinear Equations with Newton's Method* North Carolina State University. Raleigh, North Carolina. Society for Industrial and Applied Mathematics Philadelphia. 2003. Society for Industrial and Applied Mathematics.

[5] Kelley, C. T. *Iterative methods for linear and nonlinear equations*. Frontiers in applied mathematics, 16. 1995. Philadelphia: Society for Industrial and Applied Mathematics.

[6] Quandalle, P.; Savary, D. An Implicit in Pressure and Saturations Approach to Fully Compositional Simulation Society for Petroleum Engineers, 1989.

[7] Peaceman, D. W. *Fundamentals of Numerical Reservoir Simulation*. Elsevier Scientific Publishing Company, 1977.

[8] K. D. Wiegand. *GMRES experiments with a matrix derived from the inhomogeneous lattice Dirac equation Parts I and II*.

[9] Parks, M.; de Sturler, E.; Mackey, G.; Johnson D.; Maiti, S.. *Recylcing Krylov Subspaces for Sequences of Linear Systems*, SIAM J. Sci. Comput. 28 (2006), no. 5, 1651–1674.