T H E U N I V E R S I T Y O F T U L S A

THE GRADUATE SCHOOL

MODIFIED CONTINUATION-NEWTON WITH ADAPTIVE STEPLENGTH SELECTION

by
Ruslan Miftakhov

A thesis submitted in partial fulfillment of
the requirements for the degree of Master of Science
in the Discipline of Petroleum Engineering

The Graduate School
The University of Tulsa

2014

T H E U N I V E R S I T Y O F T U L S A

THE GRADUATE SCHOOL

MODIFIED CONTINUATION-NEWTON

WITH ADAPTIVE STEPLENGTH SELECTION

by

Ruslan Miftakhov

A THESIS

APPROVED FOR THE DISCIPLINE OF

PETROLEUM ENGINEERING

By Thesis Committee

_____, Advisor

Rami M. Younis

_____

Albert C. Reynolds

_____

Mohammad Shahvali

COPYRIGHT STATEMENT

ABSTRACT

Ruslan Miftakhov (Master of Science in Petroleum Engineering)

Modified Continuation-Newton With Adaptive Steplength Selection

Directed by Rami M. Younis

83 pp., Chapter 5: Discussion And Future Work

(278 words)

Nowadays the Reservoir Simulation domain is being constantly challenged by complex emerging physics. The modeling of real world problems require increasing the complexity of partial differential equation, which in addition incorporates physical stiffness onto the formulation, and the main challenge is to assure that the solution for any problem would always be obtained. Currently, the Newton-like methods serve as a good tool for solving nonlinear problems.

The fully implicit scheme is unconditionally stable in terms of discrete approximation, but at each Newton's iteration the solution of the nonlinear residual equation cannot be guaranteed. To facilitate the convergence, many mathematical as well as heuristic safeguarding techniques have been developed. Convergence has been achieved in certain cases, but still a universally efficient algorithm is not claimed. Thus, the simulation of complex processes, for instance, thermo-compositional simulation does not have an efficient safeguarding strategy.

This work devises an algorithm to address the shortcomings of Newton's method for multi-physics problems. The developed nonlinear iterative algorithm is able to acquire the solution at each global iterate, and it represents the solution of a particular timestep size which is smaller than the target step. Moreover, the algorithm can efficiently address complex problems where we do not have reliable safeguarding techniques.

As an example of a complex problem, the three phase simulation on SPE10 geological grid is tested. The results are fascinating; the developed Modified Continuation-Newton requires many fewer linear solves as opposed to the standard Newton method which employs Eclipse timestepping strategy and is safeguarded by Modified Appleyard chop. The implications of this are: convergence is always guaranteed for any well-stated problem and timestep chops do not involve wasted computation.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and thanks to my advisor, Dr.Rami Younis: you have challenged and supported me thought this journey. You also kept reminding me to appreciate intermediate milestones along the path. I could not have imagined having a better advisor and mentor for my study. I would like to thank my committee members, Dr. Albert Reynolds, and Dr. Mohammad Shahvali. You are exceptional lecturers, and your classes were a great help for this work.

I would also like to thank my colleagues: Soham Sheth, Walter Poquioma, Abdulaziz Al-Qasim, Shahriyar Alkhasli, and Rahman Mustafayev for their moral support and constructive feedback.

I would like to thank my wife, Irina Miftakhova. She was always there to cheer me up and stood by me through the good and bad times. I extend my thanks to my parents, Fanis Miftakhov and Gulisa Miftakhova for all of the sacrifices that they have made on my behalf. This work was possible with the financial support from the Republic of Tatarstan Higher Education Fellowship "Algarysh" and FUture Reservoir Simulation Systems and Technology research consortium, The University of Tulsa.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Reservoir simulation is routinely used within reservoir management activities such as the design, forecasting, and optimization of oil and gas production operations. Simulation in these contexts, involves the solution of numerical approximations to the highly nonlinear and coupled systems of Partial Differential Algebraic Equations (PDAEs) that govern flow and transport within the subsurface as well as the connected wells and surface facilities. There are numerous classic and recent introductions to the governing equations, prevalent numerical approximations, and solution procedures that are used in reservoir simulation; see for example [2, 6, 3].

Generally, the reservoir simulation governing equations are systems of coupled, nonlinear, and time-dependent advection-reaction-diffusion equations with spatially varying coefficients. Wells are typically the driving forces behind the temporal evolution, and they are often modeled by nonlinear source terms. Additionally, a number of nonlinear algebraic constraints may be coupled to the system in order to model implicit constraints such as phase behavior.

In terms of numerical approximations, the spatial discretizations that are typically used are either finite-volume or finite-element methods. Owing to the dramatically varying levels in the underlying heterogeneity and local wave speeds, and to the coupled parabolic wave propagation, temporal discretizations invariably involve some degree of implicitness. A typical discretization is the Fully-Implicit Method (FIM) that is a low-order backward Euler discretization that is fully coupled. In practice, less popular alternatives to FIM include an Implicit Explicit (IMEX) splitting that is referred to as the Implicit Pressure Explicit Saturation (IMPES) method, as well as the Adaptive Implicit Method that dynamically and

locally switches between FIM and IMPES. Ultimately, due to the severe levels of nonlinear coupling and stiffness that are present, implicit methods are a staple of reservoir simulation.

While FIM offers unconditional stability in terms of the temporal discretization, the method requires the solution of a large coupled nonlinear system of residual equations in order to advance each timestep. To solve these nonlinear algebraic systems, Newton-like iterative solution processes are the method of choice. Generally, Newton-like methods are not guaranteed to converge, and even their safeguarded variants may converge too slowly. Subsequently, convergence considerations are necessary when timestep sizes are selected. Since, to date, there is no *a priori* characterization of convergence as a function of timestep size, a try-adapt-try-again strategy is on order. That is, *a posteriori* logic must be applied where timestep size is selected iteratively. This strategy is computationally wasteful.

A recent advance proposed the application of a numerical continuation solution process, where the continuation parameter is tied to the timestep size [13]. In the proposed iterative approach, each iterate consists of a state and timestep size pair. The nonlinear solution process produces pairs that are approximate solutions to successively growing timestep sizes. While promising, there remain several opportunities to enhance the method's computational performance. This thesis advances the numerical continuation approach and demonstrates it applicability to general reservoir simulation.

This chapter, reviews the current state-of-the-art in implicit timestepping and outlines the current challenges. This is followed by a review of numerical continuation methods, and their application to implicit timestepping. Finally, the objectives of this work are stated.

1.1 Implicit timestepping in reservoir simulation

The reservoir simulation PDAEs govern the evolution of a set of state variables, $\mathbf{u}(\mathbf{x}, t) : \mathbb{R}^3 \times \mathbb{R}^+ \to \mathbb{R}^{n_{eq}}$, that are defined over the spatial domain $\mathbf{x} \in \Omega \subset \mathbb{R}^3$ and time, $t \geq 0$. The PDAE residual equations are of the form,

$$\mathcal{R}_i(\mathbf{u}) = \frac{\partial \mathcal{A}_i(\mathbf{u})}{\partial t} + \mathcal{F}_i\left(\mathbf{u}, D\mathbf{u}, D^2\mathbf{u}\right), \qquad i = 1, \ldots, n_{eq}.$$

Numerical approximations of the PDAE residual equations introduce a discrete spatial domain that is denoted $\Omega_h \in \mathbb{R}^{n_h}$, and a discrete temporal domain that is defined by the sequence,

$$t^{k+1} = t^k + \delta_t^k, \qquad k = 0, 1, \ldots,$$

where δ_t^k is a *timestep size*. *Timestepping* is the process of generating the sequence of numerical approximations, $U^k \in \mathbb{R}^{n_{eq} \cdot n_h}$, starting from the known initial condition U^0. Over a timestep, the approximation is obtained by solving the discrete system of nonlinear residual equations,

$$R\left(U^{k+1}; U^k, \delta_t^k\right) = A\left(U^{k+1}\right) - A\left(U^k\right) + \delta_t^k F\left(U^{k+1}\right) = 0, \qquad k = 0, 1, \ldots.$$

Next, the current-state-of-the art of three key aspects of implicit timestepping are reviewed. The first aspect concerns the accuracy of the approximation as well as characterizations of the asymtpotic relation between accuracy and timestep size. The second aspect is the nonlinear solution process that is used to solve the residual system at each timestep. Newton-like methods are the universal choice in modern reservoir simulation. Finally, the timestep size selection process is developed.

1.1.1 Accuracy of the temporal discretization

The discrete independent state variable, $U^k \in \mathbb{R}^{n_{eq} \cdot n_h}$ is ultimately an accurate approximation to the exact state variables up to a global approximation error;

$$\epsilon_{global}^k \equiv \| U^k - \mathbf{u}\left(\mathbf{x}, t^k\right) \|, \qquad \mathbf{x} \in \Omega_h, \ and k \geq 0.$$

The global error can be decomposed into a portion that is due to the spatial discretization and one that is due to the temporal discretization. In this work, we will neglect the spatial discretization error, and we assume that the spatial mesh is fully resolved. This allows us to consider only effects due to the time truncation errors. Reliable *a priori* estimates for

the global and local approximation errors are not available for general nonlinear problems. Rather, estimates based on extrapolation are commonly used in practice. In the reservoir simulation context, the local rate of change of the state variables is used as a measure of the anticipated local error that will be accumulated in an upcoming timestep; for example [8, 9].

1.1.2 Newton-like solution methods

Newton's method is the predominant solution procedure that is used to solve the nonlinear residual system over a reservoir simulation timestep. At a particular timestep, $k \geq 0$, the old state, U^k, and the target timestep size, δ_t^k, are determined, and the objective is to solve,

$$R\left(U^{k+1}; U^k, \delta_t^k\right) = 0. \tag{1.1}$$

For notational convenience, we will drop the fixed parameters throughout a timestep solution sequence; for example, $R\left(U^{k+1}\right) \equiv R\left(U^{k+1}; U^k, \delta_t^k\right)$. Newton's method generates a sequence of iterates, $\left[U^{k+1}\right]^n,\ n = 0, 1, \ldots$, starting from the zeroth-order predictor of the solution,

$$\left[U^{k+1}\right]^0 = U^k.$$

The sequence is generated by computing linear updates, δ^n, as,

$$\begin{aligned} \delta^n &= -J^{-1}\left(\left[U^{k+1}\right]^n\right) R\left(\left[U^{k+1}\right]^n\right), \\ \left[U^{k+1}\right]^{n+1} &= \left[U^{k+1}\right]^n + \delta^n, \end{aligned} \tag{1.2}$$

where J is the Jacobian matrix of the residual system. The key computational kernel of this process is the computation of the Newton updates. A secondary, and sometimes overlooked computational expense is in the evaluation of the Jacobian matrix and the residual system itself. *Inexact Newton Methods* apply inexact and typically iterative methods such as preconditioned Krylov acceleration in order to compute the linear updates. Inexact methods are particularily suitable for large-scale simulation where the exact solution of the linear system is prohibitively demanding. *Modified Newton methods* aim to reduce computational expense

by replacing the Jacobian with an approximation to it that is easier to compute and invert. Modified Newton methods such as the rank-one update due to Broyden are known to reduce the convergence rate of the iteration. Because of the levels of nonlinearity involved, and the associated difficulties in producing suitable substitutes for the Jacobian matrix, reservoir simulation timestepping typically uses accurate evaluations of the Jacobian matrix.

Equation 1.2 shows that the solution to the timestep is clearly a fixed point. Moreover, it is easy to show that the fixed point is also attractive. In floating point computation, the iteration is seldom carried through to the attractive fixed point. Rather, a collection of *convergence criteria* are applied in practice in order to terminate the iteration at a point that is deemed close enough. Broadly, three measures of convergence criteria may be employed. Residual measures such as,

$$\|R\left(\left[U^{k+1}\right]^n\right)\| \leq \tau_r,$$

can be computed directly. For ill-conditioned problems, they may be misleading however. Error measures on the other hand are not available since the analytic solution is not known;

$$\|\left[U^{k+1}\right]^n - U^{k+1}\| \leq \tau_e.$$

Numerous estimates have been proposed in practice and these are domain specific (see for example the normalized saturations in [9, pp 388-391]). The third type of estimate, exploits the fact that the solution is an attractive fixed point of the iteration in order to measure proximity to the solution,

$$\|\left[U^{k+1}\right]^{n+1} - \left[U^{k+1}\right]^n\| \leq \tau_d.$$

For a fixed set of convergence tolerances, the number of iterations to convergence is strongly dependent on the initial state and the target timestep size; $N\left(U^k, \delta_t^k\right)$. Moreover, in general, the convergence of Newton's method is not guaranteed.

Numerous safeguarding methods apply damping strategies in order to increase the chances of convergence.

Safegurading strategies: All safeguarded Newton variants are essentially based on specifying the diagonal matrix Δv of the explicit Newton flow formulation (1.3), where such matrix can be viewed as a local stepping,

$$\frac{\partial U}{\partial v} = -\Delta v J^{-1} R([U^{n+1}]^v; U^n, \Delta t) \tag{1.3}$$

Standard Newton makes all the diagonal weights to be unity. However, in the case of complex physics some components experience rapid changes, while others might evolve slowly. In such cases, uniform damping might lead to an unstable behavior of the explicit Newton flow formulation, and produces a divergent iteration.

There are two groups of safeguards: Classical safeguards are mathematically derived approaches from theoretical considerations. Heuristic safeguards on the other hand are derived based on some particular environment and have a narrow applicability. (Figure 1.1)

The classical safeguards are general and could be applied to any problem. The line-search and trust region are typical representatives within this group. The principle they follow is to modify the diagonal matrix uniformly such that when applied to the update, they do not change its direction. Rather, they try to find an appropriate scaling factor for the whole matrix. The choice of scaling factor is dictated by the rate of change in the residual equation along the Newton direction [13].

The heuristic approaches are widely used, but they have limited applicability to those problems for which they have been derived. For instance, the Modified Appleyard Chop (MAC) is designed specifically for two phase flow problems. The heuristic based safeguards employ a principle of cell-by-cell scaling factors. Hence, they result with different entries in the diagonal matrix Δv for distinct physics taking place in the subsurface flow. The fundamental basis is to restrict a rapid change of the unknowns in sensitive physical regions.
The following is a list of common heuristic safeguards:

- Modified Appleyard Chop (MAC): Implementation is applies to saturations only. On

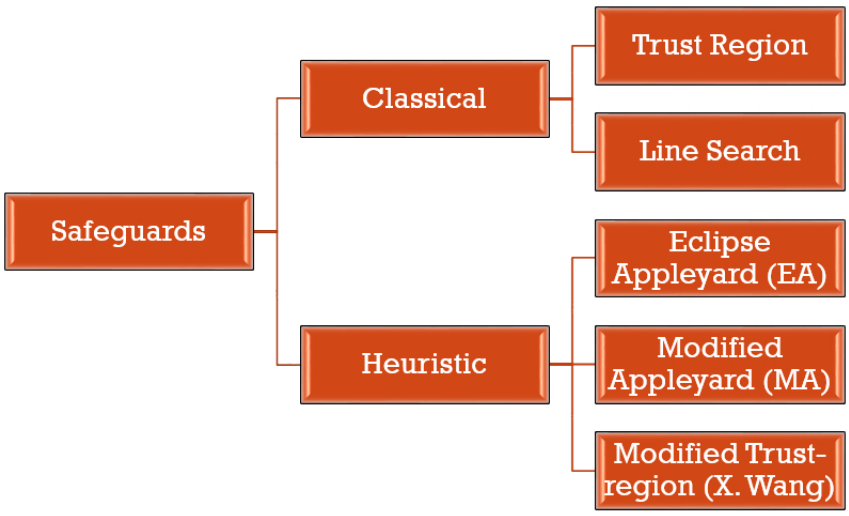

Figure 1.1: Safeguards methods

a cell-by-cell basis, damping factors are selected to ensure that the update is no larger than 0.2, and that changes from immobile to mobile are restricted to barely mobile.

- Trust-regions for two-phase flow flux functions [12]: The flux function is divided into saturation trust-regions by defining the inflection, unit-flux, and end points. The Newton updates are guided in such a way that two successive iterations can not cross any of trust regions.

These cell-by-cell strategies perform well in practice and they secure convergence in the critical regions of a fractional flow curve.

1.1.3 Timestep size selection

At a any given timestep, $k \geq 0$, the old state, U^k is at hand, and the objective is to select a timestep size, $\delta_t^k > 0$ by which to advance time. There are four important considerations to account for:

1. The temporal discretization error and the corresponding accuracy of the approximation are strong functions of timestep size. The FIM is associated with a first-order local error; i.e. $\| e^k \left(\delta_t^k; U^k \right) \| \in \mathcal{O} \left(\delta_t^k \right)$.

2. Discrete events such as the end of the simulation time, or sharp changes in the forcing terms *via* the well controls, may occur. Subsequently, for the solution to be accurate, it is necessary to select timesteps that do not cross the next upcoming discrete event; i.e. $t^k + \delta_t^k \leq t^{event}$.

3. Computational efficiency is directly related to timestep size. Generally, the iterative nonlinear solution process that is applied to solve the timestep may require more iterations to converge for larger timestep sizes. This relation for the number of iterations to convergence, while difficult to characterize, may be expressed as $L\left(\delta_t^k; U^k\right)$. Assuming that all nonlinear iterations require the same amount of computational effort, a measure of computational efficiency is the ratio $\frac{\delta_t^k}{L\left(\delta_t^k; U^k\right)}$.

4. Finally, the condition of finite termination is critical to all computationally iterative methods. We must require that $L\left(\delta_t^k; U^k\right) \leq L_{max} < \infty$.

If the critical error and convergence relations, $\|e^k\left(\delta_t^k; U^k\right)\|$ and $L\left(\delta_t^k; U^k\right)$ respectively, are at hand, then timestep size selection may be performed directly by solving the constrained minimization problem,

$$\begin{aligned} \underset{\delta_t^k}{\text{maximize}} \quad & \frac{\delta_t^k}{L\left(\delta_t^k; U^k\right)} \\ \text{subject to} \quad & \delta_t^k \geq \delta_t^{min}, \\ & \delta_t^k \leq t^{event} - t^k, \\ & \|e^k\left(\delta_t^k; U^k\right)\| \leq \epsilon^{tol}, \\ & L\left(\delta_t^k; U^k\right) \leq L_{max}. \end{aligned}$$

Unfortunately, for general nonlinear problems, neither error $\|e^k\left(\delta_t^k; U^k\right)\|$ nor $L\left(\delta_t^k; U^k\right)$ are available *a priori*. Subsequently, modern timestep size selection algorithms must apply sometimes crude estimates for these relations, and then proceed to adapt them in a trial and error fashion. Inherently, this means that modern timestep selection algorithms have no way of directly optimizing for computational efficiency. Rather, their objective is to simply obtain a feasible timestep size. Listing 2 provides a generic timestep selection algorithm.

Algorithm 1: Try-adapt-try-again timestep selection algorithm

Data: U^k, and tolerance parameters, e.g. L_{max}
Result: solution U^{k+1} and corresponding timestep size, δ_t^k
Select a trial timestep size;
Try to solve corresponding nonlinear residual;
while *nonlinear convergence is too slow or error estimates are too large* **do**
    Adapt timestep size using latest convergence rate information;
    Try to solve corresponding nonlinear residual;

A specific variant of such as strategy is employed in most commercial simulators. For example, the Eclipse simulator [10] implements the following approach. At the beginning of a simulation, Eclipse tries to obtain a solution with an initial timestep size of 1.0 day. If the solution was successfully obtained, then the timestep size will be increased by the factor of 3.0 (the default). The next timestep would be 3.0 days. For all subsequent successful iterations the timestep will be scaled upwards, until it reaches the maximum allowed timestep size of 365.0 days (the default). However, when iterates exhibit some convergence problems, then timestep will be decelerated by the factor of 0.3 (the default). Moreover, if the Newton iteration fails to converge within a maximum numbers of iterations, the factor of 0.1 (by default) will be applied. In such cases the factoring up constant will be decreased to 1.25.

1.2 Motivation for the Continuation-Newton(CN)

The key motivation of devising Continuation-Newton algorithms is to find a more robust nonlinear solver that understands physical changes occurring in a formulation, and that is able to overcome the shortcomings of Newton's method. In the CN algorithm, each iterate is an approximate solution of a particular timestep Δt which is less than the target timestep size Δt_{targ}. The major assets of the CN-like algorithms are based on the following assertion: the convergence is always guaranteed, and no timestep chops are necessary anymore.

1.2.1 High level hypothetical example

Consider a hypothetical example of Buckley Leverett displacement (1.4) in 1 cell

Figure 1.2: One Cell Buckley-Leverett problem.

(Figure 1.2). The governing equation is,

$$\frac{\partial S}{\partial t} + \frac{\partial \mathcal{F}}{\partial x} = 0, \tag{1.4}$$

where S is the saturation and $\mathcal{F}$ is the flux function.

As shown in Figure 1.3, all of the plots are in 3 dimensions. On vertical Z axis is a residual value (affine contravariance), on X is a timestep size Δt, and on Y is a saturation state value S. At the most top plot, the red solid curve represents a known initial saturation state with the timestep size $\Delta t = 0$, and we wish to acquire the solution at the timestep target Δt_{targ} with the corresponding saturation value S^{n+1}, transparent curve. It is well known, the standard Newton takes an old state variable S_{init} as an initial guess to the iterative process. Hence, in the example, after all the iterations it fails to converge (Figure 1.3(a)). The main reason for the failure as was discussed earlier, the standard Newton does not seize the physics which happens in the reservoir and cannot define pathological regions of a flux function. Therefore, a big timestep advancement might bring the initial guess far from the convergence region of the Newton process.

As a result of the failure, a chopping strategy should be applied, which cuts the timestep, to bring the guess closer to the solution. Therefore, the Newton needs to compute an intermediate timestep size Δt_{cut} (Figure 1.3(b)), in order to reach the desired solution at Δt_{targ}. Moreover, even by applying of such harsh restriction on timestep advancement, the convergence sill cannot be guaranteed. According to the hypothetical example, the formu-

lation is finally converged, but it involved a significant waste of computational resources.

Motivated by overcoming the flows of standard Newton's method, Rami Younis in his work [13] proposed to make use of Continuation-Newton, which is applied directly on the timestep size Δt. The high level picture of the derived algorithm is on Figure 1.3(c). The proposed strategy will not use the guess as an old state variable(as it was done in the case of the standard Newton), but it would perform a prediction of a state variable change towards the solution, then it will use a new predicted point as the guess. The CN algorithm understands the variable change direction, and by getting the guess closer to the solution it avoids unnecessary timestep chopping, therefore, secure the convergence.

1.2.2 Review of a previous work

The Continuation methods have been used for a long time by scientists and engineers, especially when a good initial guess was not available. The CN implementation in a subsurface flow simulation domain was recently derived, and published by Younis in SPE-119147-MS paper "Adaptively Localized Continuation-Newton Method - Nonlinear Solvers That Converge All the Time". It claims that developed CN solver allows a stable advancement in time without worrying about the timestep chops. [1]

Generally, the solution of PDE is obtained when a pair of an unknown state and corresponding timestep size, forces the residual equation $R(U^{n+1}; U^n, \Delta t)$ go to a zero value. The main task of CN algorithm is to numerically trace the zero residual level curve Equation (1.5), which is a solution path. It can be done by parameterizing the residual equation with respect to some parameter as in Equation (1.6)). For instance, Younis in his paper suggested to use the arc-length λ parameterization.

$$R(U(\lambda), \Delta t(\lambda)) = 0. \tag{1.5}$$

Following from the PDE theory, at every discrete timestep size $\Delta t > 0$, there exists at least one unique solution U^{n+1}, bifurcation points are not considered here to eliminate complicated hypotheses which are not intrinsically important for our discussion. Assuming

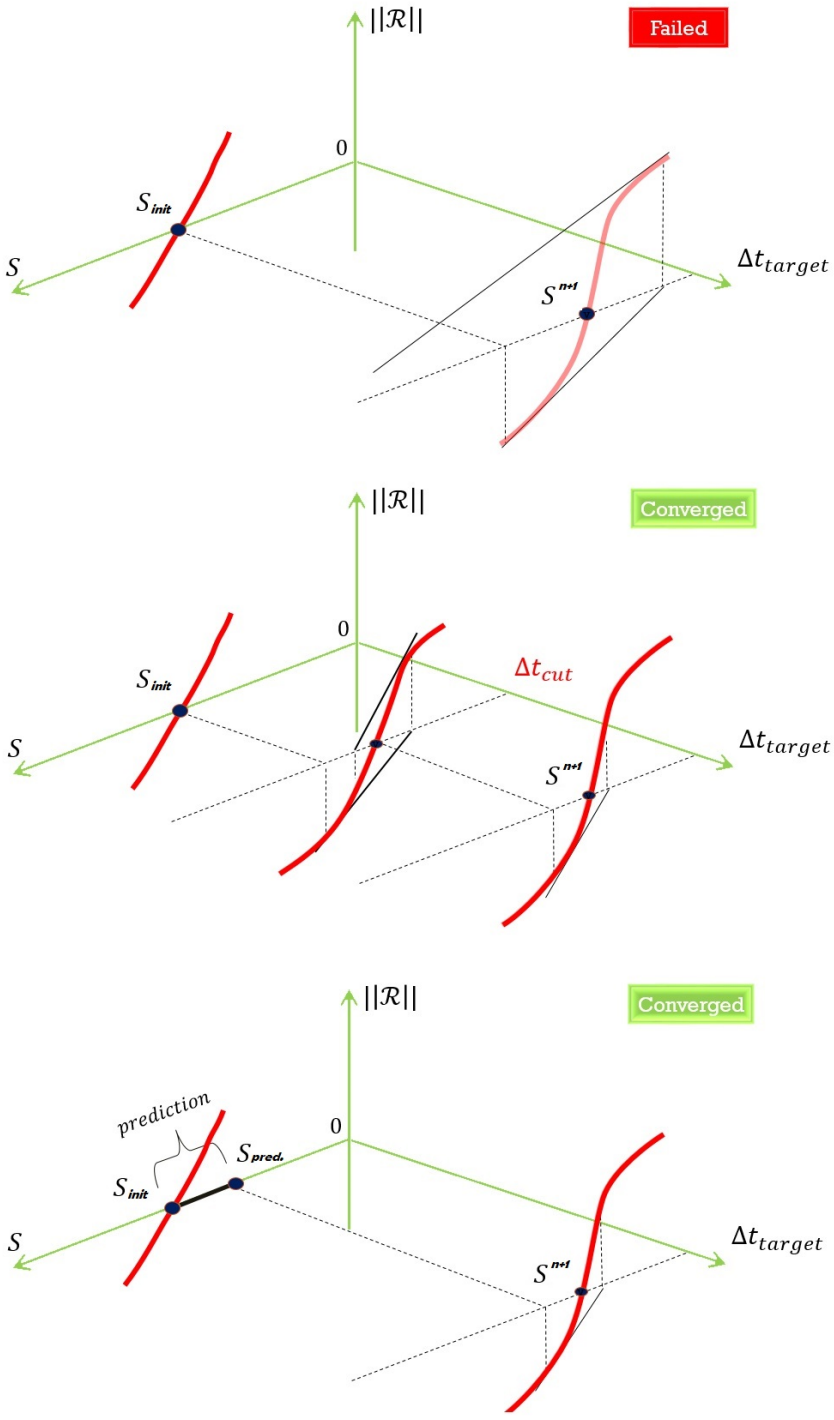

Figure 1.3: Iterative sequence: (a), (b) - Standard Newton, (c) - CN.

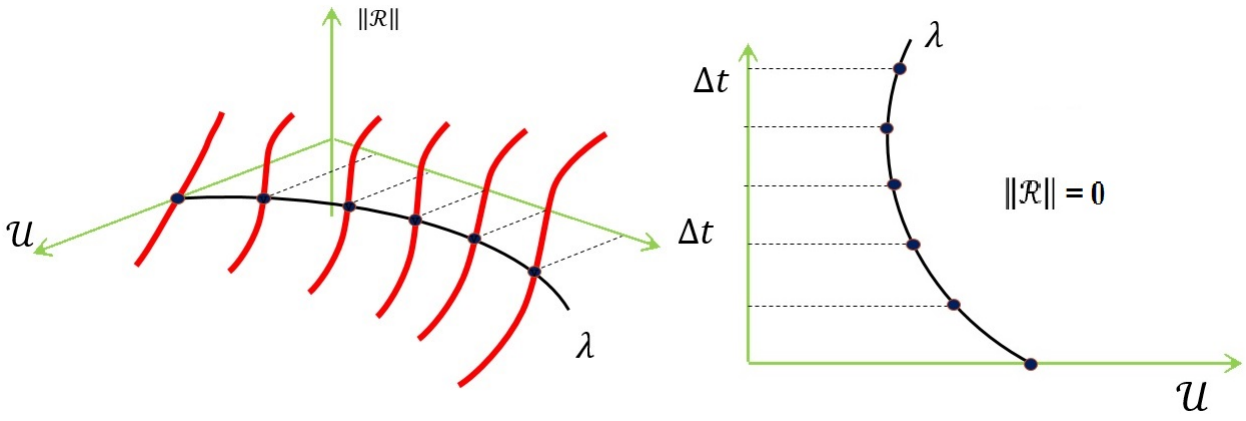

Figure 1.4: Solution path: (a) - 3D, (b) - 2D representation.

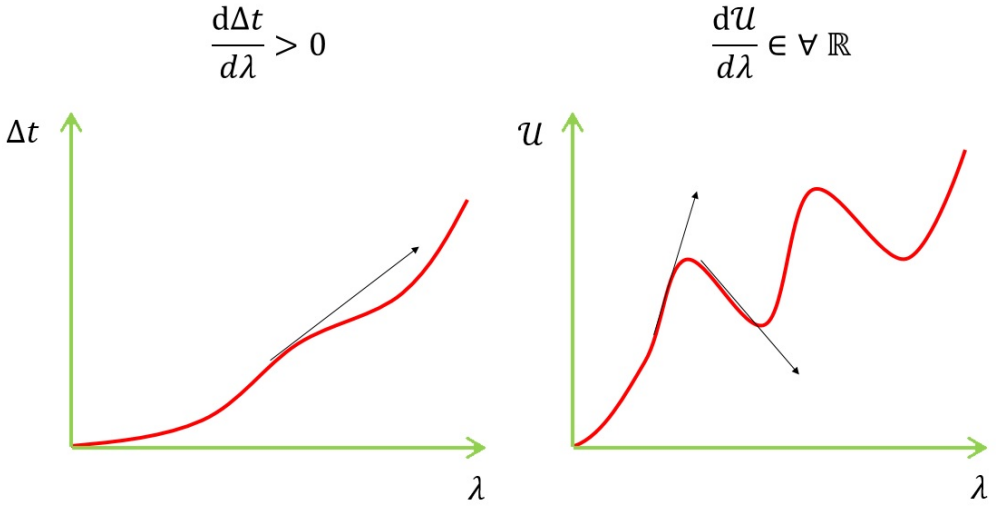

Figure 1.5: The change of independent parameters along the solution path λ: (a) - Δt, (b) - U.

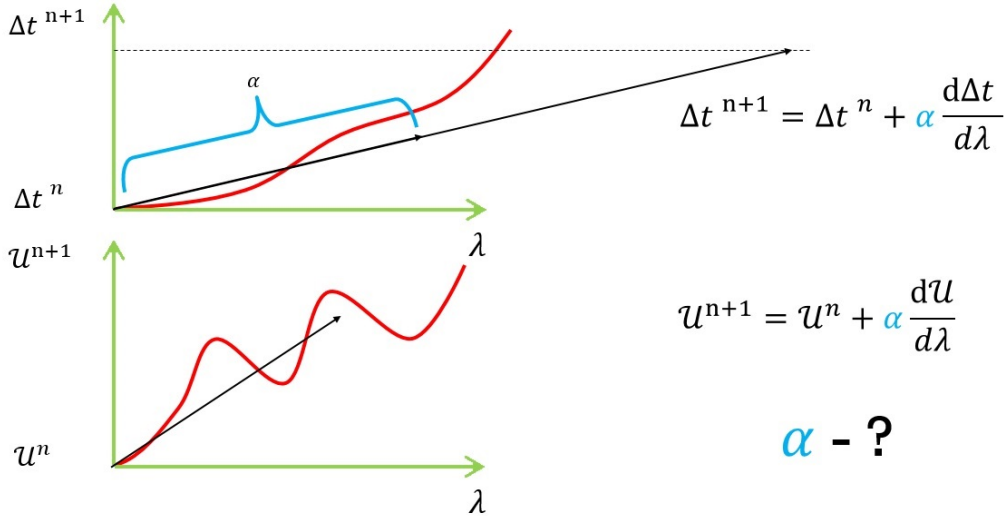

Figure 1.6: Representation of the prediction step in CN algorithm.

that several couples of such unknowns are already computed, those points make the residual equation satisfy the chosen tolerance criterion $\|R\| \leq \varepsilon$. It is possible to connect all these discrete points(supplied with the starting point $R(U_0) = 0$) with a curve which will result in a solution path of underlying formulation(Figure 1.4(a)). A solution path is a continuous curve which lies on zero residual level plane(Figure 1.4(b)), each point on this curve defines a pair of unknown $(U(\lambda), \Delta t(\lambda))$.

The solution path can be computed for any PDE because it is irrelevant to modeled physics. The only difference it would bring is a deformation of a shape of the solution path. As stated in the paper, the residual equation needs to be parameterized with respect to the arc-length λ parameter. Therefore, all the variables involved in the formulation have to be considered to be depended on λ, then the total derivative can be written as follows:

$$\frac{dR}{d\lambda} = \frac{\partial R}{\partial U}\frac{dU}{d\lambda} + \frac{\partial R}{\partial \Delta t}\frac{\Delta t}{d\lambda} = J\frac{dU}{d\lambda} + \frac{\partial R}{\partial \Delta t}\frac{\Delta t}{d\lambda} = 0, \tag{1.6}$$

$$rank(J(\lambda(\alpha))) = N, \quad \forall \alpha \in \Omega, \tag{1.7}$$

where $\frac{dR}{d\lambda}$ is an N by $N+1$ system; $\frac{dU}{d\lambda}$ and $\frac{\Delta t}{d\lambda}$ represent a change in a state variable and a timestep size along the solution path accordingly; α is a steplength along the overall tangent line.

The derived system (1.6) results in N equations and $N+1$ unknowns, which requires either some constitutive relationship or constraint to complete the system of a square matrix. One of the options is to impose a unit tangent condition, $\left\| \frac{dU}{d\lambda}, \frac{d\Delta t}{d\lambda} \right\| = 1$. The other possible way is to force some physical inequality. It is known that there is a unique solution to each timestep size, and we can force the solution path following the uniqueness criterion. For this reason, the change of a timestep size with respect to arc-length has to follow a restricted variation. As a result, the inequality $\frac{d\Delta t}{d\lambda} > C$ has been imposed , where the positive constant C is chosen arbitrary small. Therefore, the change of Δt along the arc-length on the solution path would be bounded above zero, and monotonically increasing at all points (Figure 1.5

(a)). Otherwise, the uniqueness of a solution would be violated. On the other hand, the change in the unknown state variable is not bounded with any constraint (Figure 1.5 (b)) [14]. As a result, the following system can be written:

$$\begin{bmatrix} J & \frac{\partial R}{\partial \Delta t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{dU}{d\lambda} \\ \frac{\Delta t}{d\lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ C \end{bmatrix}, \tag{1.8}$$

where J is a Jacobian $[N\ by\ N]$ matrix, the same as the one used for standard Newton's method, the elements of J are defined as $J_{(i,j)} = \frac{dR_{(i)}}{dU_{(j)}}$, $\frac{\partial R}{\partial \Delta t}$ represents an N dimensional vector of a residual equations change with respect to a timestep size. The system of $N+1$ equation and $N+1$ unknown is completed, and it can be solved simultaneously to obtain the change in the state variable $\frac{dU}{d\Delta t}$ (1.8). Note, this system of equation is always solvable because Jacobian itself is well defined and invertible.

The final step is to define a tangent line equation, which emanates from an origin, as follows:

$$\frac{dU}{d\lambda} = \delta \leftarrow C(-J^{-1}\frac{\partial R}{\partial \Delta t}). \tag{1.9}$$

Based on known initial state variables $(U^n, \Delta t^n)$, which bring a residual equation to the zero level, it is possible to compute tangent lines to all state variables (1.9). Moreover, the first order Euler predictor is employed which takes the tangent values as well as a steplength α along it, and advances the initial state.(1.10)

$$\Delta t_{pred} = \Delta t^n + \alpha\frac{\Delta t}{d\lambda}, \quad U_{pred} = U^n + \alpha\frac{dU}{d\lambda}, \tag{1.10}$$

where Δt_{pred} and U_{pred} is a predicted(expected) change in variables, those predictions will supply an initial guess for the iterative solver.

A big concern for all Continuation-Newton like methods is how to define the steplength α along the tangent line, which is able to directly influence the performance of CN algorithm. The importance of the right choice might be viewed on Figure 1.6. It is a result of the fun-

damental limitation of a linear first order Euler predictor, which accurately predicts the changes in the vicinity of its origin $[U^n, \Delta t^n]$. The prediction is reliable when a sufficiently small steplength α selected to stay within a safe region. In other words, there is some threshold proximity region to the solution path, and those predicted points which are inside the region are most likely to be good predictions. In contrast, the prediction for faraway points might lead to a divergent iterative sequence.

In the SPE paper an algorithm to compute the steplength based on residual evaluations, where so-called the convergence neighborhood is implemented. The algorithm represents quantitative measures of the predicted point proximity to the solution path. The obtained measurements directly define the future steplength. A small tolerance region along the solution path is taken as the convergence neighborhood (Figure 1.7). Those points inside the neighborhood $s(U, \Delta t) \in \mathcal{N}, \quad \|R(s)\| \leq \varepsilon_{tol}$ are assumed to be close enough to the solution path, and they would exhibit a rapid convergence to the solution; whereas, those points outside of the neighborhood are assumed to be bad predictions. The choice of a tolerance proximity is motivated by a local quadratic convergence of the standard Newton's method, which is suggested to take as one or two order in magnitude looser than required to judge if an iterate is an acceptable solution.

Here are presented the flow chart(Figure 1.8) as well as the graphical representation (Figure 1.7) of the proposed algorithm. As follows from Figure 1.7, the tangent line from an initial state is computed, then the algorithm needs to select the steplength along the tangent line which is based on backtracking iteration in order to find such steplength to remain in the convergence neighborhood. At each backtracking iteration the system of residual equations has to be evaluated. If such steplength was found, which falls within the neighborhood region, then the next tangent line would be computed and the steplength selection routine will be invoked again. However, at the time when backtracking iterations results with none steplength α possible in order to remain in the region, some of the Newton corrector iteration needs to be performed to bring the guess closer to the solution path. The algorithm would repeat all those discrete operations until the target timestep size criterion is met.

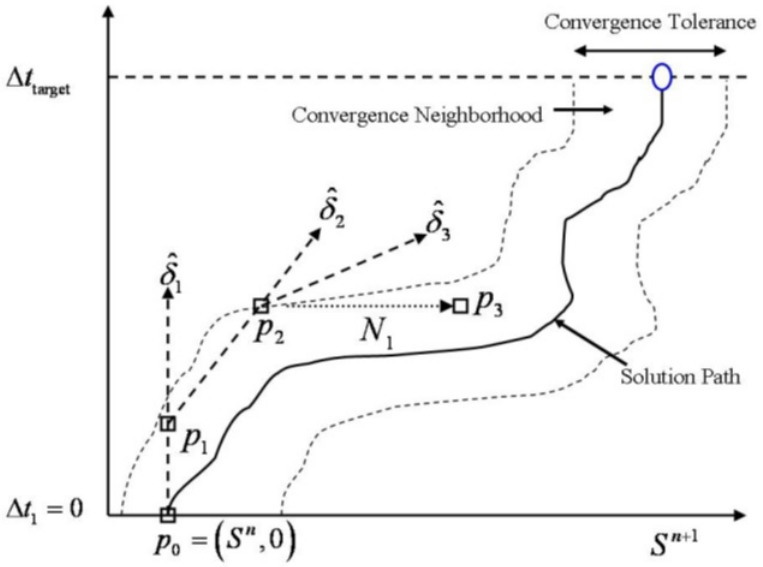

Figure 1.7: Illustration the proposed Continuation-Newton algorithm.

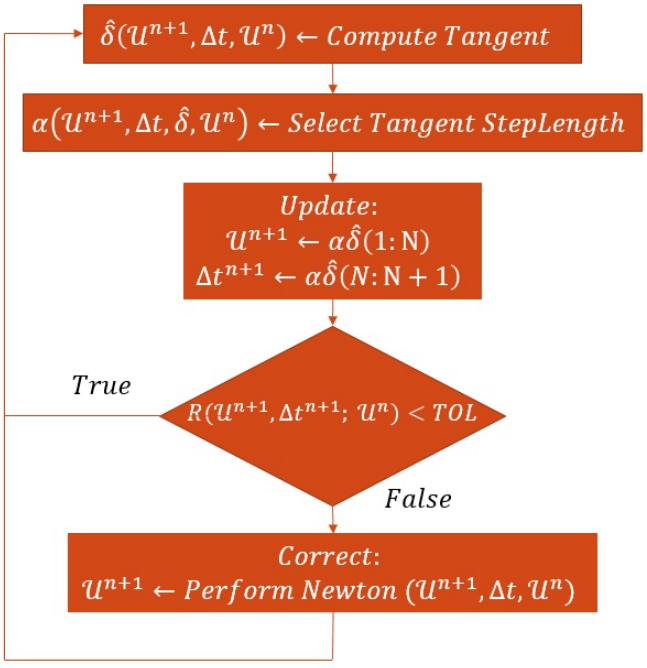

Figure 1.8: Flow chart of the proposed Continuation-Newton algorithm.

1.3 The objectives of this work

The main purpose of the following, is to state the shortcomings of the earlier proposed algorithm and offer some possible solutions. There are several major drawbacks:

1. The convergence neighborhood along the solution path is devised to define a heuristically based local convergence (contraction) region for the corrector process, essentially it represents a zone of small residual proximity (neighborhood tolerance). Those predicted points which fall in such neighborhood are assume to be "close enough" to the solution and will exhibit a rapid convergence. On the other hand, those points which are outside of the region assume to be bad predictions. The detection whenever the predicted point is in the convergence region or not, is implemented by an evaluation of the residual norm. Moreover, a user must supply a convergence neighborhood tolerance criterion around the solution path for various simulation cases. There is not enough theory-based explanation to claim such strategy will be successful at all times.

2. The vector of residual equations has to be evaluated at each backtracking iteration in order to select a suitable steplength along the tangent line. The associated computational deficiency of the algorithm is in such evaluations. For instance, in a compositional flow simulation, it needs to perform not only the routine operations, but also a flash calculation to compose the vector of residual equations.

3. The proposed steplength strategy, which is solely based on the residual evaluation, cannot result neither in an efficient timestep advancement nor an optimal computation, because the norm of the residual equation at most of the time has nothing to do with guaranteeing the convergence. Furthermore, strictly bounding the residual variation with the steplength selection will always result in an inefficient timestepping strategy and wasteful computations.

With this in mind, I will address to the stated drawbacks and will tackle them in the following order:

1. An investigation of possible improvements in the tangent line computations, where the main questions of preferred parameterization parameters, higher order predictors, and stability issues will be covered in the following chapter.

2. The next portion of the work will be focused on a robust steplength selection method, which will not be based on the convergence neighborhood. It will automatically select the steplength based on the convergence history of the previously computed timestep. The proposed steplength selection algorithm can be divided into two stages: At the first stage, needs to acquire the relation between the steplength of the future tangent with an introduced error $\tilde{\varepsilon}_0(\tilde{\alpha})$, derivation based on Taylor series expansion. At the second stage, needs to find an appropriate error model, which could mimic the error decay from the previous timestep. Such model will give leverage in deriving the relation of the future initial error and the numbers of iteration it will take to converge $\tilde{\varepsilon}_0(\tilde{N})$.

   The combination of these two stages, gives us the possibility to adjust the steplength with the specified desired number of iterations. For instance, if a user wants to have only five iteration throughout a simulation run, the algorithm will give the steplength which will reach that.

CHAPTER 2

TANGENT LINE INVESTIGATION

The primary discussion in this chapter will be focused on various tangent line improvements based on theoretical and also practical grounds. A tangent line is an essential kernel of Continuation-Newton like methods, which gives the rate of change for state unknowns with respect to a single parameter, and basically it represents the first order approximation out of an infinite Taylor series expansion of a continuous solution path.

1. As shown earlier, the tangent line was computed with respect to an outside parameter as the arc-length λ of the solution path. However, there are many other parameters with respect to which the tangent line could be computed and it might result in a reduced form of ODE to solve.

2. The Continuation-Newton theory suggests to use as the most efficient, the first order Euler predictor. It is obtained by throwing out all the term higher than linear from an infinite Taylor series expansion of the solution path, which might result with a poor prediction accuracy compare to higher orders. The first order tangent carries an information such as a rate of change of the unknown state with respect to some parameter, but the higher order terms might reveal the information of a curvature. Therefore, it might gives us a considerable gain in terms of accuracy.

   The motivation for testing the higher order predictor came from an observation. The subsurface flow simulators usually encompass two type of equations, such as a parabolic/elliptic-flow equation and a hyperbolic-transport equation. In the flow equation, the changes are occurring globally and to perform prediction is an easy task. On the other hand, the transport equation have a wave propagation-like behavior, where the saturation

changes accruing only in a local region (front). Based on the tests the first order predictor's capability limited only on one cell. In the other words, the first order predictor is able to move the saturation front only on one cell forward. This also motivates us to review higher order terms of the Taylor series expansion, and it is desirable to obtain a prediction which can span the front propagation in multiple cells.

3. Need to address the issue of the tangent line computation stability. The explicit scheme offers a good approximation in the close vicinity to the starting point, but it gives unreliable approximation of a point which is located far from the origin.

   In general, fully-implicit schemes provide any formulation with improved stability. With an intention to develop an implicit-like behavior for the tangent line construction, the associated cost should be evaluated because any fixed point scheme would require the performance of some iterations. Since the ODE, which represents the tangent line equation, needs to invert the system of linear equations in order to be evaluated, the computational cost has to be closely considered.

2.1 Solution path parameterization

It is important to obtain the most simplified formulation of the tangent line equation because it would decrease the complexity of the solution procedure. The zero residual level curve (solution path) can be parameterized with respect to any arbitrary parameter, as well as the independent unknowns involved in the system. As was mentioned, one of the choices is to use the arc-length as the parameterization parameter (2.1), which has already been fully derived in the research work [13]:

$$R(U(\lambda), \Delta t(\lambda)) = 0, \tag{2.1}$$

where each of the unknowns U and Δt are assumed to be dependent on the arc-length λ. The parameterized equation has the following form:

$$\frac{dU}{d\lambda} = -J^{-1}\frac{\partial R}{\partial \Delta t}\frac{d\Delta t}{d\lambda}.$$

Then, in order to advance the state unknown, we need to provide some step length α along the tangent line:

$$U_{pred} = U_{old} + \alpha\frac{dU}{d\lambda}.$$

A disadvantage of the proposed parameterization that the computation of the arc-length based tangent line requires the user to supply an additional closure relation $\frac{d\Delta t}{d\lambda}$ to complete the system of N equations and $N+1$ unknowns. The other option is to choose a parameterization in a way to decrease the number of unknowns in the system. In the following discussion, the use of more natural parameter as the timestep size Δt is derived.

Thus, all the unknowns involved in the formulation are assumed to be dependent on the timestep size Δt:

$$R(U(\Delta t), \Delta t) = 0. \tag{2.2}$$

Then, the zero residual level curve is parameterized with respect to Δt:

$$\frac{dR(U(\Delta t), \Delta t)}{d\Delta t} = 0. \tag{2.3}$$

The total derivative of the residual equation would have the following form:

$$\begin{aligned}
\frac{dR(U(\Delta t), \Delta t)}{d\Delta t} &= \frac{\partial R}{\partial U}\frac{dU}{d\Delta t} + \frac{\partial R}{\partial \Delta t} = 0, \\
\frac{dU}{d\Delta t} &= -(\frac{\partial R}{\partial U})^{-1}\frac{\partial R}{\partial \Delta t}, \\
\frac{dU}{d\Delta t} &= -J^{-1}\frac{\partial R}{\partial \Delta t}.
\end{aligned} \tag{2.4}$$

As provided by Equation (2.4), in order to acquire the tangent line, there is no need to use some other constraint equations. The derived formulation is self-contained and well defined. The solution of (2.4) is always attainable, because the Jacobian matrix is assumed

to be always invertible. Both the timestep size and the arc-length based parameterization require only one inverse of a linear system.

In short, the derived timestep parameterization helps to simplify the calculation of the tangent line, where the system of N equation by N unknowns has to be solved.

2.2 Higher order predictor

The common way to compute the tangent line is to use only the first order term of the Taylor expansion. The infinite expansion for the arc-length λ based parameterization of the solution path is written in the following form:

$$U = U_0 + \alpha \frac{dU}{d\lambda} + \frac{\alpha^2}{2!}\frac{d^2U}{d\lambda^2} + \frac{\alpha^3}{3!}\frac{d^3U}{d\lambda^3} + \frac{\alpha^4}{4!}\frac{d^4U}{d\lambda^4}\dots \tag{2.5}$$

In contrast, the timestep based parameterization is expressed as:

$$U = U_0 + \alpha \frac{dU}{d\Delta t} + \frac{\alpha^2}{2!}\frac{d^2U}{d\Delta t^2} + \frac{\alpha^3}{3!}\frac{d^3U}{d\Delta t^3} + \frac{\alpha^4}{4!}\frac{d^4U}{d\Delta t^4}\dots \tag{2.6}$$

As was mentioned earlier, the subsurface flow simulators usually incorporate two type of equations one for parabolic/elliptic flow and one for hyperbolic transport. It is been observed that for the transport equation, which has a wave propagation-like behavior, the first order predictor is able to move the saturation front only one cell forward.

The first order term carries information such as a rate of change of an unknown state, which limits the prediction only on the linear extrapolation. The PDEs involved in the simulation are highly nonlinear, and it is desirable to implement a higher order approximation, which might give leverage to the tangent line in spanning along the solution path. Furthermore, it is necessary to compute the second order term, which will provide the predictor with a parabolic behavior. The obtained accuracy improvement might enforce the overall tangent line to predict a propagation of the saturation front on several cells.

The two sets of derivations for the second order term are derived, the first for the arc-length and the second for the timestep size parameterization.

2.2.1 Higher order predictor: the arc-length parameterization

The parameterization begin with the definition of the first order residual parameterization:

$$\frac{dR(U(\lambda),\Delta t(\lambda))}{d\lambda} = J\frac{dU}{d\lambda} + \frac{\partial R}{\partial \Delta t}\frac{d\Delta t}{d\lambda}.$$

Then, it is necessary to differentiate the above equation once again, with respect to the arc-length:

$$\frac{d^2R(U(\lambda),\Delta t(\lambda))}{d\lambda^2} = \frac{d}{d\lambda}[J\frac{dU}{d\lambda}] + \frac{d}{d\lambda}[\frac{\partial R}{\partial \Delta t}\frac{d\Delta t}{d\lambda}],$$

where $J(U(\lambda),\Delta t(\lambda))$ is an N by N Jacobian matrix, which depends on λ. For simplicity of the further derivation, assume $\mathcal{G} = \frac{\partial R}{\partial \Delta t}$.

The second order derivative of the zero residual level curve is presented below:

$$\frac{d^2R}{d\lambda^2} = [\frac{\partial J}{\partial U}\frac{dU}{d\lambda} + \frac{\partial J}{\partial \Delta t}\frac{d\Delta t}{d\lambda}]\frac{dU}{d\lambda} + J\frac{d^2U}{d\lambda^2} + [\frac{\partial \mathcal{G}}{\partial U}\frac{dU}{d\lambda} + \frac{\partial \mathcal{G}}{\partial \Delta t}\frac{d\Delta t}{d\lambda}]\frac{d\Delta t}{d\lambda} + \mathcal{G}\frac{d^2\Delta t}{d\lambda^2} = 0,$$

where the above equation needs to be solved for $\frac{d^2U}{d\lambda^2}$, which represents the second order term of the Taylor series expansion (2.5):

$$\frac{d^2U}{d\lambda^2} = -J^{-1}[(\frac{\partial J}{\partial U}\frac{dU}{d\lambda} + \frac{\partial J}{\partial \Delta t}\frac{d\Delta t}{d\lambda})\frac{dU}{d\lambda} + \mathcal{G}\frac{d^2\Delta t}{d\lambda^2} + (\frac{\partial \mathcal{G}}{\partial U}\frac{dU}{d\lambda} + \frac{\partial \mathcal{G}}{\partial \Delta t}\frac{d\Delta t}{d\lambda})\frac{d\Delta t}{d\lambda}], \tag{2.7}$$

where $\frac{\partial J}{\partial U}$ is a partial derivative of an N by N Jacobian matrix with respect to the vector of N unknowns, which results in a tensor matrix. $\frac{\partial J}{\partial \Delta t}$ is a partial derivative of the Jacobian with respect to the scalar Δt. $\frac{d^2U}{d\lambda^2}$ is an N equations by $N+2$ unknowns system.

As was mentioned in the previous chapter, in order to obtain the first order tangent line, the closure equation is needed, and it was taken as $\frac{d\Delta t}{d\lambda} > 0$. The main challenge of formulation (2.7) is to find an additional restrictive knowledge of a solution path behavior, as $\frac{d^2\Delta t}{d\lambda^2}$, in order to be able to solve the system for the second order term. Such knowledge cannot be obtained because the acceleration of a timestep size with respect to arc-length is defined in a whole real space, so there is no bounds we might enforce. Thus, this leaves us with the incomplete system (2.7) of N equations and $N+1$ unknowns, which might

be solved by applying a more complicated and computationally costlier inverse such as the Moore Penrose pseudoinverse [5].

Finally, we can write a truncated formulation of a predicted point near the solution path:

$$U^{n+1} = U^n + \alpha \frac{dU}{d\lambda} + \frac{\alpha^2}{2!} \frac{d^2U}{d\lambda^2}. \tag{2.8}$$

2.2.2 Higher order predictor: the timestep size parameterization

In the hope that the computation of the second order term system would be possible, we consider the equation of the tangent line based on the Δt parameterization:

$$\frac{dR(U(\Delta t), \Delta t)}{d\Delta t} = J \frac{dU}{d\Delta t} + \frac{\partial R}{\partial \Delta t}.$$

Taking a second derivative of the above equation with respect to Δt,

$$\frac{d^2R}{d\Delta t^2} = \frac{d}{d\Delta t}[J \frac{dU}{d\Delta t}] + \frac{d}{d\Delta t}[\frac{\partial R}{\partial \Delta t}].$$

Assume, the Jacobian dependence on Δt, $J(U(\Delta t), \Delta t)$, and assume $\mathcal{G} = \frac{\partial R}{\partial \Delta t}$. The second derivative of the residual equation with respect to Δt is

$$\frac{d^2R}{d\Delta t^2} = [\frac{\partial J}{\partial U} \frac{dU}{d\Delta t} + \frac{\partial J}{\partial \Delta t}] \frac{dU}{d\Delta t} + J \frac{d^2U}{d\Delta t^2} + \frac{\partial \mathcal{G}}{\partial U} \frac{dU}{d\Delta t} + \frac{\partial \mathcal{G}}{\partial \Delta t} = 0.$$

The following defines the second order term of the infinite Taylor series expansion (2.6):

$$\frac{d^2U}{d\Delta t^2} = -J^{-1}[(\frac{\partial J}{\partial U} \frac{dU}{d\Delta t} + \frac{\partial J}{\partial \Delta t}) \frac{dU}{d\Delta t} + \frac{\partial \mathcal{G}}{\partial U} \frac{dU}{d\Delta t} + \frac{\partial \mathcal{G}}{\partial \Delta t}]. \tag{2.9}$$

The calculation of system (2.9) does not require an additional closure equation; it represents a complete system of N equations and N unknowns. This system can be inverted by using standard linear algebra routines.

Then, the predicted point can be defined as the following:

$$U^{n+1} = U^n + \alpha \frac{dU}{d\Delta t} + \frac{\alpha^2}{2!} \frac{d^2U}{d\Delta t^2}. \tag{2.10}$$

In summary, although it was possible to come up with a constraint equation for the first order tangent line in the arc-length parameterization, we failed to obtain such constraint for the second order term. Therefore, it makes the computation impossible to perform. On the other hand, the timestep size parameterization gives us a more convenient form of the system, and needs no additional constraints However, the computation of the second order term requires the tensor to be composed and many tensor-vector multiplications to be performed.

2.2.3 Higher order predictor: 1-cell simulation case

In order to test the prediction accuracy of different orders of approximation, a simple 1-cell Buckley-Leverett problem is presented in this section, where the governing equation has the following form:

$$\frac{dS}{dt} + \frac{dF(S)}{dx} = 0,$$

where $F(S)$ is a flux function and S is a saturation value in the cell.

The discretized form of the residual equation is

$$R = S^{n+1} - S^n + \frac{\Delta t}{V}[F(S^{n+1}) - F_{inj}],$$

where V is a volume of the cell.

The initial saturation is $S_{init} = S_0 = 0.0$ (Figure 1.2), and we want to obtain the solution at $\Delta t = 0.8$. There are 3 predictor types (Figure 2.1): the first is the predictor of a zero order, which can be viewed as a standard Newton, where the prediction is an old state variable; the second is the first order tangent, which represents a strait line; the third is the second order tangent curve, which encompasses an additional paraboloid behavior.

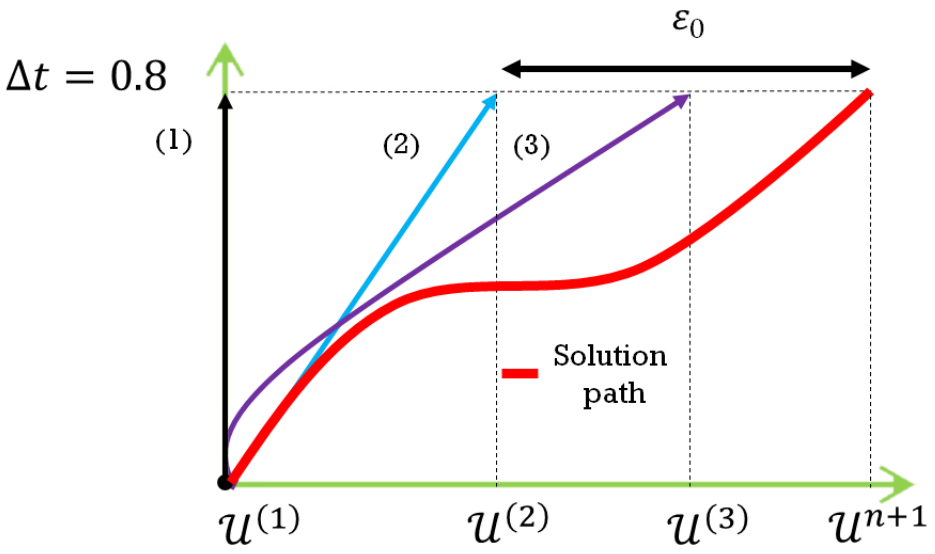

Figure 2.1: Different orders of approximation for the predictor: (1) - zero order, (2) - first order, (3) - second order

Order of approximation	Terms	Predicted value	Error
(1) Zero Order	$S_{pred} = S_0$	0.0	-0.6255
(2) First Order	$S_{pred} = S_0 + \alpha \frac{dS}{d\Delta t}$	0.8	0.1745
(2) Second Order	$S_{pred} = S_0 + \alpha \frac{dS}{d\Delta t} + \frac{\alpha^2}{2!} \frac{d^2S}{d\Delta t^2}$	0.7488	0.1233

Table 2.1: The one cell simulation results

NOTE: In all the cases, the steplength α is chosen to be equal to the full timestep size $\Delta t = \alpha = 0.8$. The converged solution at the prescribed timestep size is $S^{n+1} = 0.6255$.

The results of comparison performance of all three predictor options are in table (2.1) and on Figure 2.1. As was mentioned, the zero order predictor represents the standard Newton's method. It takes the prediction as the old state variable $S_{pred} = 0.0$, and results in a big proximity(Error) to the solution; the first order is closer to the solution $S_{pred} = 0.8$, but it overshoots the solution on $\varepsilon_0 = 0.1755$; The most accurate as we know from the Taylor series expansion and now from the simulation test is the second order predictor, which results in $S_{pred} = 0.7488$ and the error of $\varepsilon_0 = 0.1233$.

The accuracy of the second order predictor is justified in this section, but still a question of computational performance versus gained accuracy needs to be answered since the computation of the second term requires the performance of one additional inverse of a linear system and compute many tensor-vector multiplications. The following test will try to answer that question.

2.2.4 Higher order predictor: 1D simulation case

The test setup is 1D Buckley Leverett problem, the number of gridlocks is $Nb = 1000$, and injection is undergoing from the most left face. The current simulation case has no safeguarding strategies implemented. It will compare the three orders of approximation: zero order which associated with the standard Newton; CN with the first order ; and CN with the second order approximation.

The result of the comparison is shown on Figure 2.2, which depicts the relation of the number of linear iterations versus the timestep size. Each point on the figure represents a full simulation run as the one timestep. At small timestep sizes, the standard Newton converges with less number of iterations compare to other two. However, after some threshold it stars having some trouble to converge. In contrast, CN of the first and the second order, converges reasonably well throughout the all timestep sizes. Moreover, CN of the second order almost all the time lies above the first one. The observed deficiency of the second order is because

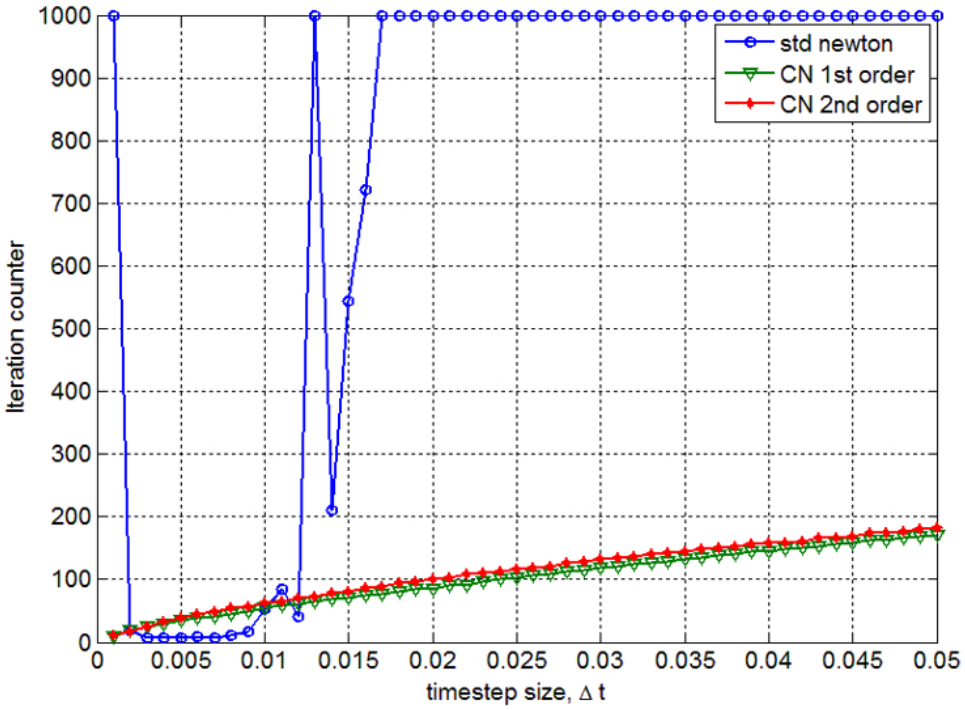

Figure 2.2: One timestep simulation performance comparison

the accuracy improvement don not overweight the cost of the one additional linear inverse. The small variation in predicted value can not be seized by the Newton iterative process, under the circumstances of the contraction region being very close to the solution.

The first order approximation will be used in the following discussions as the most efficient one.

2.3 The level of Implicitness

The stability issue is an important topic for a "good" tangent line computation. The explicit scheme is cheap and offers a fairly good approximation only in the close vicinity. In contrast, Implicit schemes are able to provide an improved stability for faraway points, but it might come at a considerable expense.

The main reason for searching more stable solution of the tangent computation is to have a bounded error along the tangent line. Presumably, the explicit tangent is not able to provide with a good tangent line direction, which would lead to a significant error growth.

However, it assumed that even a slightly introduced implicitness to the formulation would improve the tangent direction, which will have an improved error behavior.

In this section, we will investigate how the choice of a pair of unknowns $[U^{n+1}, \Delta t]$ affects the tangent line as well as the overall performance of the CN algorithm.

First, the standard way of selecting the pair of unknowns is to take them as an old values ($U^{n+1} = U_{old}; \Delta t = \Delta t_{old}$) , then the tangent will be emanating from its origin point $[U_{old}, \Delta t_{old}]$ on the solution path. Moreover, it is might be viewed as an explicit based approach, where the prediction capability is limited in a close area from the origin.

For the current choice of unknowns assume $U^{n+1} = Uold$.

$$\frac{dU}{d\Delta t} = -J^{-1}(U^{n+1}, Uold, \Delta t^n)\frac{\partial R(U^{n+1}, Uold, \Delta t^n)}{\partial \Delta t}. \tag{2.11}$$

As shown on Figure 2.3, even for a simple case of a parabolic function the first order predictor results with a bad extrapolation. The tangent line emanated from the origin $(U_0, 0)$, which brought the predicted point faraway from the solution. Please note here, the standard Newton's initial guess is a lot more closer to the solution at the target timestep size (Figure 2.3). If we were to take the predicted point as a guess for the Newton iterative process, it most probably will spend a lot of iterations to converge or might even diverge. There is no way to take a big enough steplength α and remain in the convergence region with the explicitly defined tangent line.

Second, consider if we would be able to take the tangent line at the point where the solution is being searched $(U^{n+1}, \Delta t_{targ})$. Certainly, it would bring many benefits in bounding the error from infinite growing. However, such stability remedy could be reached only by introducing some sort of implicitness to the formulation. If one of the future unknown state or the timestep is known a prior, then it might be used to obtain an improved tangent line, which would help to get the other unknown by use of some Fixed Point like iteration schemes (Figure 2.4).

Assume, the timestep where we want to get a solution is known. The pair of unknowns

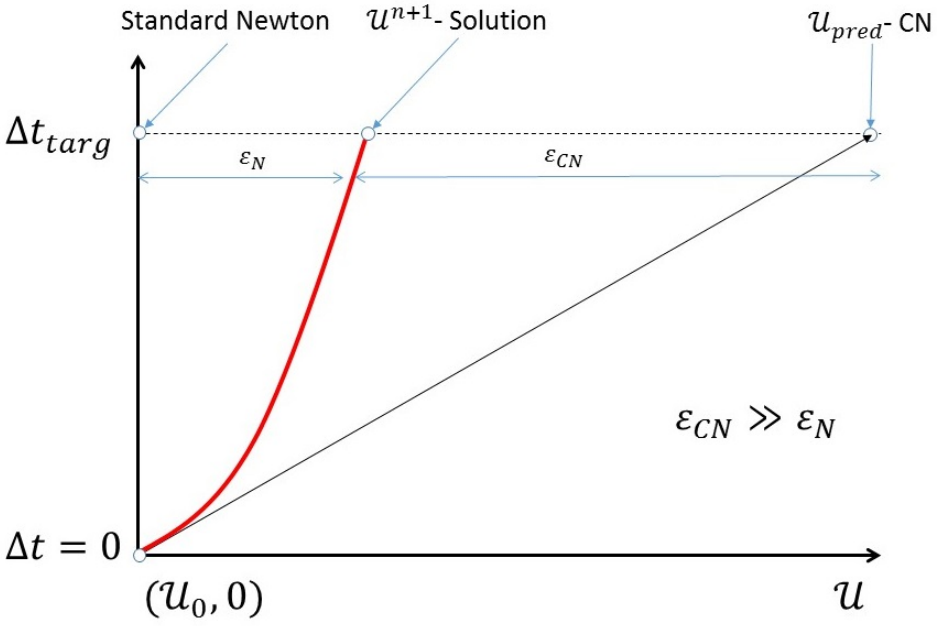

Figure 2.3: Explicit predictor

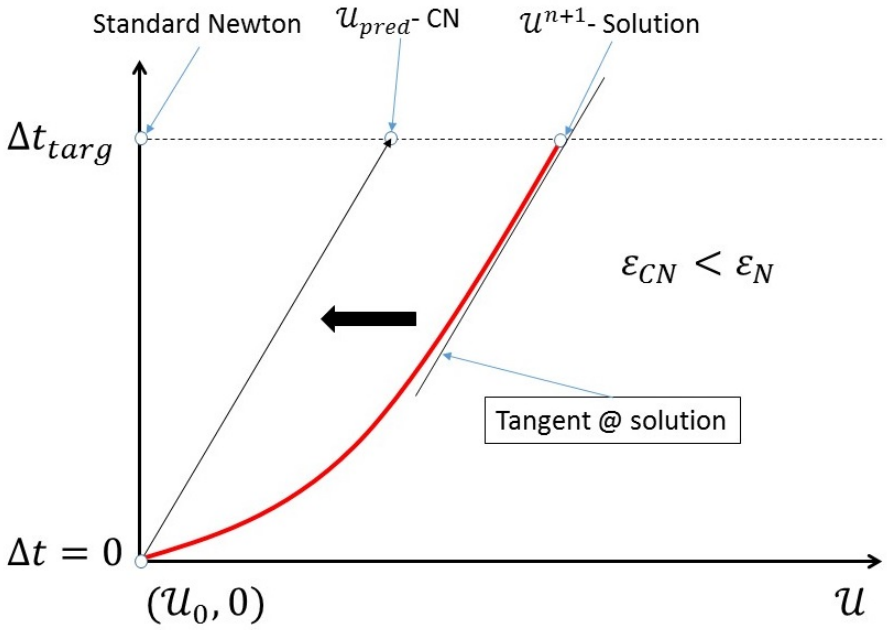

Figure 2.4: Implicit-like predictor

would have the form $[U^{n+1} = U_{old}, \Delta t^{n+1} = \Delta t_{targ}]$. The tangent line equation can be expressed as

$$\frac{dU}{d\Delta t} = -J^{-1}(U^{n+1}, Uold, \Delta t^{n+1}) \frac{\partial R(U^{n+1}, Uold, \Delta t^{n+1})}{\partial \Delta t}, \tag{2.12}$$

where Δt^{n+1} is the future timestep size at the solution, which we want to obtain.

Then, the tangent line could be used to advance the state variables:

$$U^{n+1} = U^n + \alpha \frac{dU}{d\Delta t},$$

where α is the target timestep size divided by the number of fixed point iterations $\alpha = \frac{\Delta t}{N}$.

Algorithm 2: Fixed point iteration algorithm

Data: U_{old}, the target timestep size Δt_{targ}, and the number of iterations N;
Result: predicted point $\tilde{U}^{n+1}$;
Compute the steplength $\alpha = \frac{\Delta t}{N}$
while *the iterations less than prescribed* N **do**
    Obtain the tangent line solution, $\frac{dU}{d\Delta t}$
    Advance the state unknowns, U^{n+1}
The final predicted point would be: $\tilde{U}^{n+1} = U^{n+1}$

Equation (2.12) and the update of the state variables can be repeated until the desired accuracy or maximum iterations bounds are met. Obviously, increasing the number of iterations will eventually lead to a more accurate and reliable tangent line solution.

On Figure 2.4 shown, a hypothetical example of the quadratic function and the tangent line that we wish to obtain. NOTE: The figure does not represent a sequence of fixed point iterations, only one evaluation of the algorithm 2 has been performed with the steplength $\alpha = \Delta t_{targ}$. In this case, the tangent line emanated from the origin is not an exact tangent as at the solution point, but a result of the first approximation (iteration). Even from the figure we can see, the improved stability of the tangent line construction. The error between the predicted point and the solution decreased dramatically. The induced error by the scheme is always less than from explicit tangent line. All of these motivates us to take even bigger steplength along the tangent line and still be withing the error bounds.

In summary, we have devised a new way to compute a more stable tangent line compare to the original (explicit) one. The simulation comparison will be discussed in the next section.

2.3.1 The level of Implicitness: Comparison

In order to understand the ability of each tangent line formulation accurately predict the future occurring changes. The two phase subsurface flow simulator with SPE10 comparative study grid model is proposed to be used as the test case. In order to decrease the computational intensity only the first 2 layers in Z-direction were taken from the full SPE10, which results in an overall grid dimension of $60 \times 220 \times 2$. Six production wells as well as two injection wells were located on 5-spot inverted pattern. All the supplementary physics are included: capillarity, and gravity effects.

Here will be a comparison performance of the explicit and the semi-implicit based tangent line formulations. The explicit based tangent test is done by the following procedure: first, compute the tangent at the origin point ($U^{n+1} = U_{old}, \Delta t = 0.0$), then advance the state variables by using different steplengths. The tested steplengths $(\alpha_1, \alpha_2, \alpha_3)$ were selected in such a way, to allow the Newton iterative process to obtain the solution starting with the predicted point $\tilde{U}^{n+1}$ at a particular timestep size. As it shown on Figure 2.5, the steplength range varies from $0.000005 \leq \alpha \leq 0.005$ and if it increases the upper bound, the formulation does not converge.

The results at each steplength are presented on Figure 2.7. The left part of the figure is dedicated for the pressure and on the other half for the saturation unknown. There are three curves on the figure: the green is initial $U^n = U_{old}$; the blue is predicted $\tilde{U}^{n+1}$; the red is the converge variables distribution U^{n+1}. The horizontal axis represents the grid dimension. As was expected, a good identification of the changes and the scaling with a very tiny steplength $\alpha = 0.000005$. However, as the steplength keeps increasing the profile of changes stays the same, only the magnitude differs. That is the main reason of the prediction deficiency for point faraway from the origin.

The semi-implicit tangent test was implemented in the following order: first, compute the tangent line at the point where the solution is being searched ($U^{n+1} = U_{old}, \Delta t = \Delta t^{n+1} = \alpha$). Keeping in mind the associated cost of the fixed point iteration and in order to have a compatible computational cost with explicit tangent construction, only one iteration of algorithm 2 was computed. Moreover, the chosen steplengths are in the range of $0.000005 \leq \alpha \leq 5.0$, which implies that the semi-implicit tangent line could obtain the solution with a bigger steplength (Figure 2.6).

The simulation based results are shown on Figure 2.8. We should note here, the semi-implicit tangent line has a good prediction accuracy for the small steplength $\alpha = 0.000005$ as the explicit based. Moreover, with the increasing of the steplength the formulation results with a good predictions for the pressure variable, but still straggles to accurately predict the changes in the saturation.

The question of level of implicitness has to be investigated in more details. However, the results are very optimistic and no doubt, it has a good perspective for the future work.

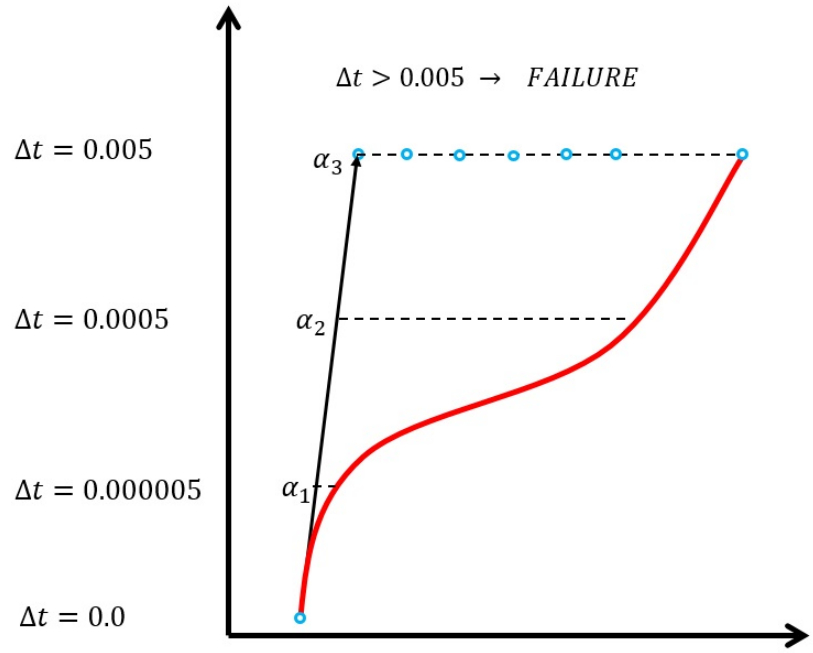

Figure 2.5: Explicit predictor test.

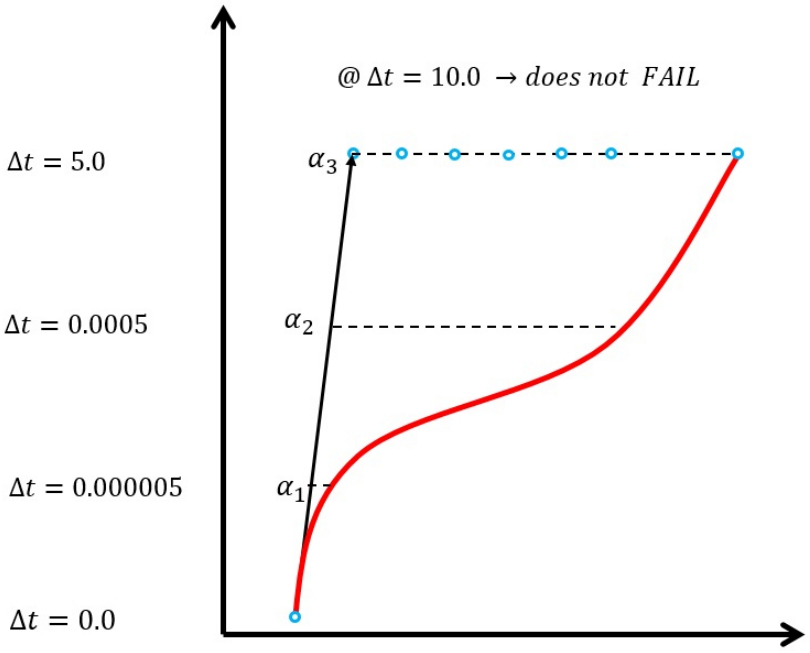

Figure 2.6: Semi-Implicit predictor test.

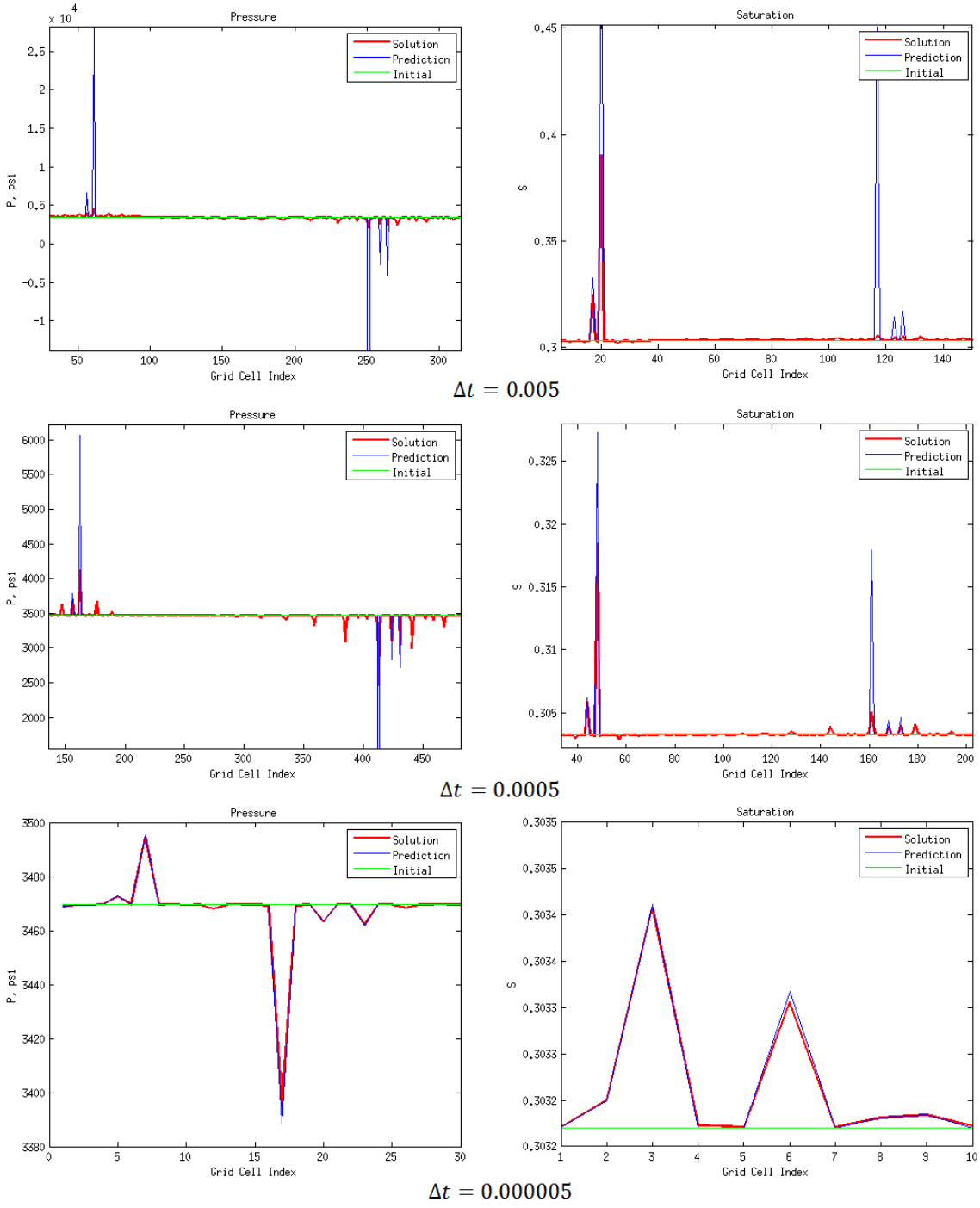

Figure 2.7: Explicit predictor results for the three cases: (a) - $\Delta t = 0.005$ *days*, (b) - $\Delta t = 0.0005$ *days*, (c) - $\Delta t = 0.000005$ *days*

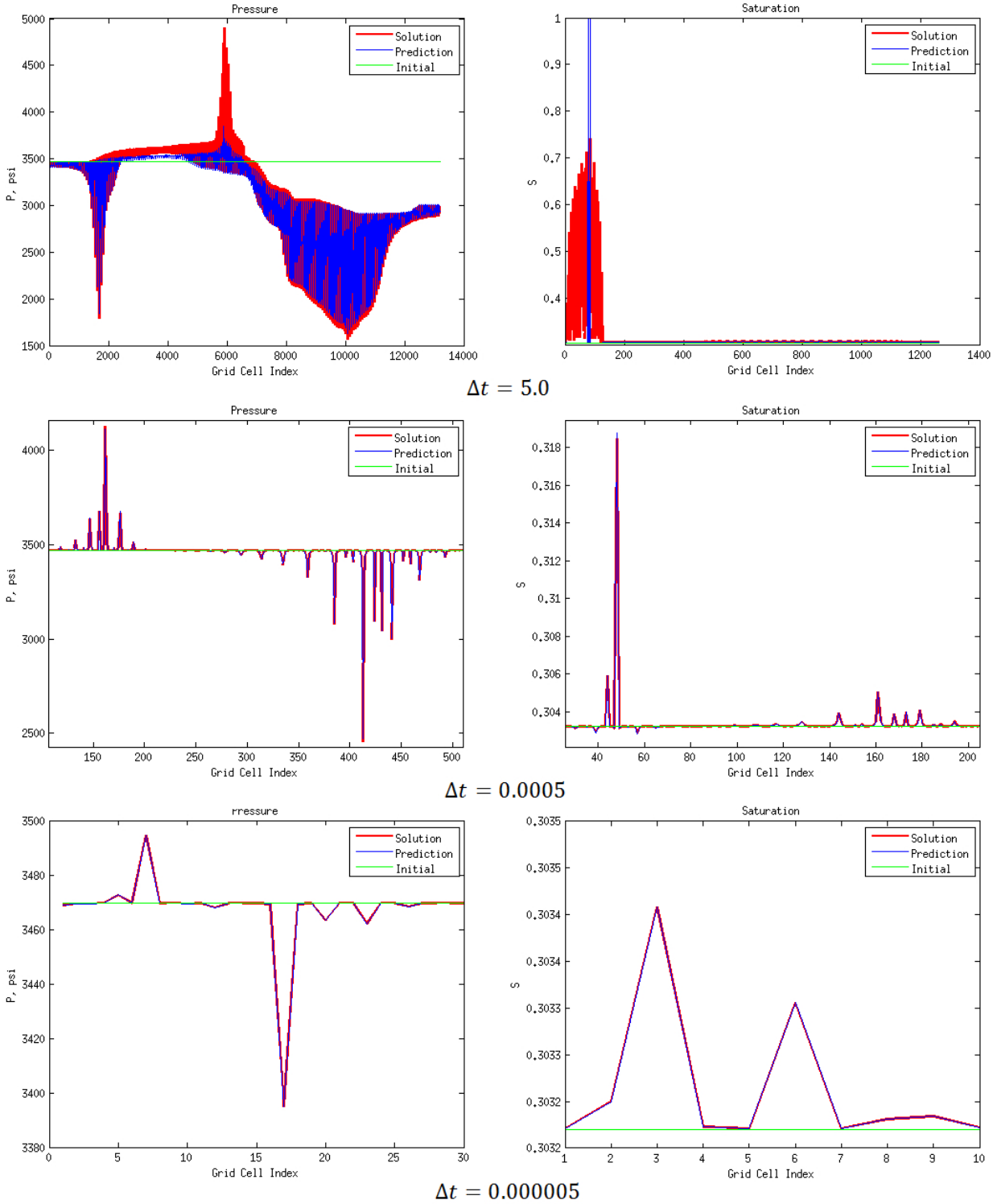

Figure 2.8: Semi-Implicit predictor results for the three cases: (a) - $\Delta t = 5.0$ *days*, (b) - $\Delta t = 0.0005$ *days*, (c) - $\Delta t = 0.000005$ *days*

CHAPTER 3

ROBUST STEPLENGTH SELECTION

The major challenge with Continuation-Newton like methods is the selection of a steplength along the continuation tangent line. For instance, when steplength is taken smaller than subsequent formulation requires, CN will follow the solution path unnecessary closely; on the other hand, if the steplength is chosen bigger when it might take many iterations to converge or even eventually diverge. Poorly chosen steplength will lead to a significant computational waste.

There are many different approaches to define the steplength: adaptation by asymptotic expansion; strategies involving variable order predictors based on a prior estimation of convergence radii; convergence neighborhood which was already used in reservoir simulation problems; etc.

The proposed algorithm here, is able to automatically select the steplength based on an error decay information of the previous timestep. Therefore, having the previous timestep information of the error decay throughout the iterations with the corresponding steplength α, enables to predict a future steplength of the next tangent line. The algorithm can only be used when the corrector step may be viewed as an iterative procedure.

The proposed steplength selection algorithm can be divided into two stages:
The first stage, from the Taylor series representation of the solution path, need to derive the relation between the steplength $\tilde{\alpha}$ on the future tangent line with a growing error $\tilde{\varepsilon}$ along it. The second stage, need to acquire an appropriate error model which could mimic the error decay of the previous timestep. The obtained model will give leverage in deriving the relation of the future initial error $\tilde{\varepsilon}_0$ and the number of iterations it will take to converge N.

Combination of these two stages gives us the ability to adjust the steplength with

a desired numbers of iterations $\tilde{k}$. In other words, the desired numbers of iteration will be a guidance for the steplength selection strategy which is adapted to meet the iteration requirement. For instance, if a user wants to have only five iteration throughout a simulation run, the algorithm will provide the steplength which will guarantee that. This derivation was motivated by the proposed approach of Den Heijer & W. C. Rheinboldt [4].

3.1 Stage 1: the steplength relation with an error

Let U_0 be a predicted point which lies in some neighborhood around the solution path $||R|| = 0$, and T be an iterative corrector process which in our discussion is Newton (Figure 3.1):

$$U_0(\alpha) = U_{init} + \alpha \frac{dU}{d\lambda},$$

where λ is some arbitrary chosen parameterization of the solution path, U_{init} is an initial state unknown.

Then, the corrector may be viewed as an iterative process which brings the predicted point U_0 to the point on the solution path with some prescribed tolerance criterion:

$$U_k = T(U_0(\alpha)).$$

In the further discussion lets assume that for steplength very small $\alpha > 0$, the following limit exist:

$$U_\infty(\alpha) = \lim_{k \to \infty} U_k(\alpha) \in R(0).$$

The corrector process T acts orthogonality to the tangent line $\frac{dU}{d\lambda}$ initiating at U_{init} (Figure 3.1). It is fare to say that the angle between $\|U_0 - U_\infty\|$ and δ is $\pi/2 + O(\alpha)$. Using the Ferent frame representation of the solution path, we can define each point on the curve as:

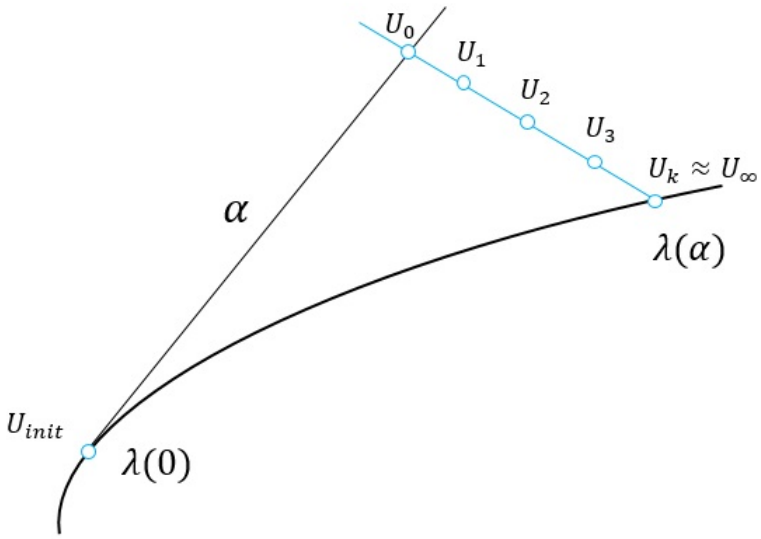

Figure 3.1: Illustration of the local parameterization

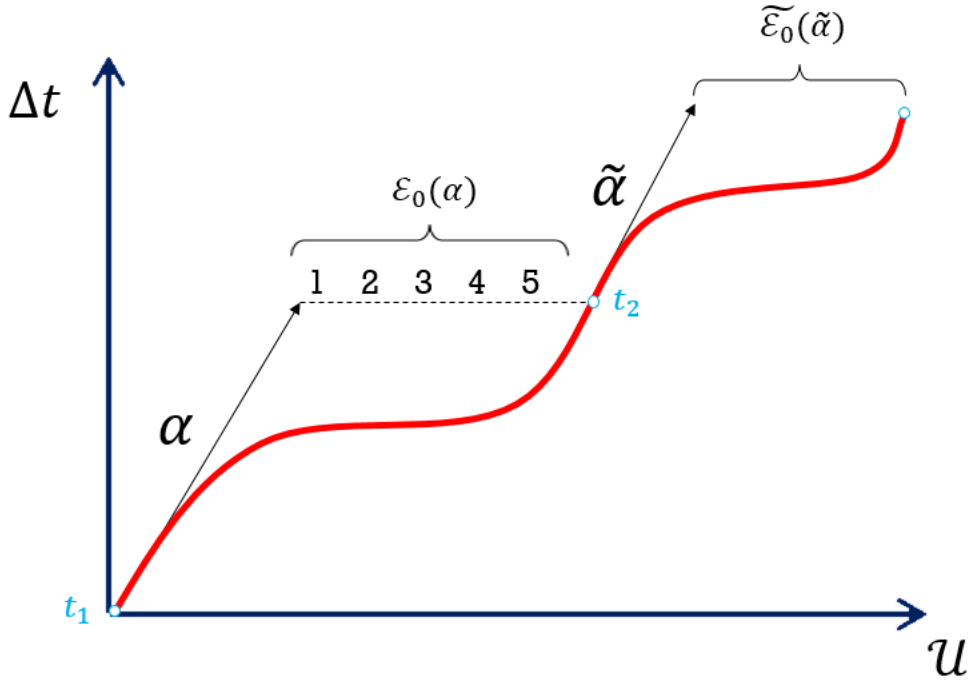

Figure 3.2: The overview of the steplength selection procedure

$$\lambda(\alpha) = \lambda(0) + \alpha\frac{dU}{d\lambda}(0) + \frac{\alpha^2}{2}k(0)N(0) + O(\alpha^3), \tag{3.1}$$

where $k(0)N(0)$ is a curvature and a normal vector to the solution curve accordingly,

$$N(\alpha) = \frac{d^2\lambda(\lambda)}{d\alpha^2} - [\frac{d^2\lambda(\lambda)}{d\alpha^2}, \frac{d\lambda(\alpha)}{d\alpha}]\frac{d\lambda(\alpha)}{d\alpha}, \tag{3.2}$$

$$k(\alpha) = \left\| \frac{d^2\lambda(\lambda)}{d\alpha^2} \right\|. \tag{3.3}$$

Then, the predicted point which lies on the tangent line (Figure 3.1), can be obtained as:

$$U_0(\alpha) = U_{init}(0) + \alpha\frac{dU(0)}{d\lambda}.$$

By throwing out all the terms higher than quadratic, a truncated solution $U_k \approx U_\infty$ could be written:

$$U_\infty \approx U_k = U_{init} + \alpha\frac{dU(0)}{d\lambda} + \frac{\alpha^2}{2!}k(0)N(0). \tag{3.4}$$

The error between the predicted point U_0 and the solution U_∞ is defined as:

$$\varepsilon_0 = \|U_\infty(\alpha) - U_0(\alpha)\| = \frac{\alpha^2}{2!}k(0)N(0) + O(\alpha^3). \tag{3.5}$$

The objective is to obtain an approximation of the future error $\tilde{\varepsilon}_0$, and it's dependence on the future steplength $\tilde{\alpha}$. As one of the ways, it has been suggested by the researchers to take a constant curvature along the solution path. Therefore, the predicted error could be found by using the ratio of two errors (Equation (3.5)) [1]:

$$\frac{\varepsilon_0}{\tilde{\varepsilon}_0} = \frac{\|U_\infty(\alpha) - U_0(\alpha)\|}{\left\|\tilde{U}_\infty(\tilde{\alpha}) - \tilde{U}_0(\tilde{\alpha})\right\|} = \frac{k(0)N(0)\alpha^2}{k(\alpha)N(\alpha)\tilde{\alpha}^2}, \tag{3.6}$$

where $\tilde{U}_i$ is the iterate of the future sequence, $k(\alpha)N(\alpha)$ - curvature and normal vector evaluated at α on the solution path.

By enforcing the assumption $k(0)N(0) = k(\alpha)N(\alpha)$, we can get an equation only with two unknowns, $\tilde{\varepsilon}_0$ and $\tilde{\alpha}$:

$$\frac{\varepsilon_0}{\tilde{\varepsilon}_0} = \frac{\alpha^2}{\tilde{\alpha}^2},$$

$$\tilde{\alpha} = \alpha\sqrt{\frac{\tilde{\varepsilon}_0}{\varepsilon_0}}. \tag{3.7}$$

Equation (3.7), gives the explicit relation of the future error $\tilde{\varepsilon}_0$ with the steplength $\tilde{\alpha}$. Please note here, in order to obtain such relation they imposed rather harsh assumption for the higher order terms of expansion (3.4). They assumed that the curvature of the solution path would have the same value at different points, in other words, the accuracy of the future points will be lagging by one step(previous iteration) behind. Based on many simulation tests this model was proved to be very conservative, and it does not accurately represent the real change in the initial error.

Those critical assumptions in Equation (3.7) were the motivation on deriving a more accurate representation of the error dependency with the steplength $\tilde{\alpha}(\tilde{\varepsilon})$. Begin with the formulation of the error at the future steplength:

$$\tilde{\varepsilon_0} = \left\|\tilde{U}_k(\tilde{\alpha}) - \tilde{U}_0(\tilde{\alpha})\right\| = \frac{\tilde{\alpha}^2}{2!}k(\alpha)N(\alpha).$$

The above equation can be solved explicitly for $\tilde{\alpha}$:

$$\tilde{\alpha} = \sqrt{\frac{2!\ \tilde{\varepsilon}_0}{k(\alpha)N(\alpha)}}. \tag{3.8}$$

The calculation of the second order term might put some additional cost on the algorithm. If there is a need to decrease computational complexity, the second order approximation can be substituted with the norm of the first order. Hence, the higher order terms

of the expansion are always bounded by the norm of lower ones.

$$\frac{\tilde{\alpha}^2}{2!}k(\alpha)N(\alpha) = \tilde{\alpha}\left\|\frac{dU(\alpha)}{d\lambda}\right\|. \tag{3.9}$$

Then, the resultant equation of the first stage will have the following formulation for the future steplength:

$$\tilde{\alpha} = \frac{\tilde{\varepsilon}_0}{\left\|\frac{dU(\alpha)}{d\lambda}\right\|}. \tag{3.10}$$

The derived equation (3.10) will be used as a primary relation of the steplength with the error in the following discussion of the proposed algorithm.

3.2 Stage 2: the relation of a steplength and an iteration number

This section will cover some basic understanding of a good error model necessity, provide the description of the classical model implementation proposed by Rheinboldt, and will discuss the most suitable model for the reservoir simulation cases. As was mentioned earlier the proposed steplength selection algorithm based on the previously obtained iteration sequence data. Therefore, having such history knowledge at hand, an error model which will mimic the error decay throughout the iterations with a good accuracy needs to be found. The accepted error model will be used in order to obtain an estimation for the future error $\tilde{\varepsilon}_0$ at some steplength $\tilde{\alpha}$ (Figure 3.2). Thus, the accuracy of the model is a paramount importance for the reliable prediction and retrieving the convergence quality information. It has been stated by researchers that increasing the accuracy of the error model in describing the error decay at the past timestep would lead to a more efficient steplength selection strategy.

3.2.1 Classical approach

There are several different error behavior regulations on which the error decay models are based. The classical error models employ the following behavior:

$$\varepsilon_{i+1}(\alpha) \leq \varphi(\varepsilon_i(\alpha)), \tag{3.11}$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ is the error model function, such that $\varphi(0) = 0$, the model of this nature has to be applied over and over to observe a decaying tendency throughout iterations. ε_i is an error of the previous iterate of the sequence, regarded as $\|U_i(\alpha) - U_0(\alpha)\|$.

Den Heijer & Rheinboldt [4, 7] proposed to use the following two error models which are derived from Newton-Kantorovich theory:

$$\varphi(\varepsilon) = \frac{\varepsilon^2}{3 - 2\varepsilon}, \tag{3.12}$$

$$\varphi(\varepsilon) = \frac{\varepsilon + \sqrt{10 - \varepsilon^2}}{5 - \varepsilon^2}\varepsilon^2. \tag{3.13}$$

Also one more classical error model of superlinear convergence:

$$\varphi(\varepsilon) = \varepsilon^p, \ p > 1. \tag{3.14}$$

All of these error models provide some estimation of the error decay, based on the assumptions they have derived on. In other words, each of the model will accurately approximate only certain error behavior type(rate of convergence). For example, the models derived based on Newton-Kantorovich theory are employ an assumption of quadratic convergence.

Please note here, the models can be only used with a relative error as an input $0 \leq \varepsilon \leq 1$, because of the error behavior type (3.11).

Suppose for the initially given steplength α (Figure 3.2) computed the predictor step where the tangent line emanates from point t_1, the corrector is stopped after k iteration at a point t_2, fulfilled tolerance requirements, and the corrector process exits with an approximation $U_k(\alpha) \approx U_\infty$. Therefore, it is possible to evaluate the relative error for the last iteration:

$$w(\alpha) = \frac{\|U_k(\alpha) - U_{k-1}(\alpha)\|}{\|U_k(\alpha) - U_0(\alpha)\|} \approx \frac{\|U_\infty(\alpha) - U_{k-1}(\alpha)\|}{\|U_\infty(\alpha) - U_0(\alpha)\|} = \frac{\varepsilon_{k-1}(\alpha)}{\varepsilon_0(\alpha)}. \tag{3.15}$$

Den Heijer & Rheinboldt failed to employ a posterior estimate of a convergence radii around the solution path; They reported such estimates cannot be obtained from the sequence of corrector iterates alone. It will also require having some global information about a function which could be obtained on a balance of a considerable expense. However, the sequence of corrector iterates does allow to compute convergence quality of certain types of correctors.

The following work-flow is used to obtain the estimated error relation with the number of iterations, based on classical error models $\varepsilon(k)$:

1. Need to solve the following ratio which involves one of the proposed monotonic error models φ in order to obtain the convergence quality estimation adjusted to the error model accordingly:

$$\bar{\varepsilon}_0 = \frac{\varphi^{k-1}(\varepsilon_0)}{\varepsilon_0} = w, \quad \varphi(0) = 0, \quad \varphi(1) = 1. \tag{3.16}$$

2. To obtain the future error at the next step which would satisfy the stopping criterion after a given number of desired iterations $\tilde{k}$. The following relation helps to acquire an estimated error $\tilde{\varepsilon}$ for the future timestep as a function of the desired number of iterations (Figure 3.3):

$$\varphi^{\tilde{k}}(\tilde{\varepsilon}_0) = \varphi^k(\bar{\varepsilon}_0), \tag{3.17}$$

where the power indexes $\tilde{k}$, and k are the desired numbers of iterations specified by a user and the number of iterations taken for the current timestep to converge accordingly. Moreover, in the case when the number iterations taken in the current timestep and the prescribed number of desired iteration are equal $k = \tilde{k}$, Equation (3.17) would result with the estimated initial error $\tilde{\varepsilon}_0 = \bar{\varepsilon}_0$, and the future steplength would be of

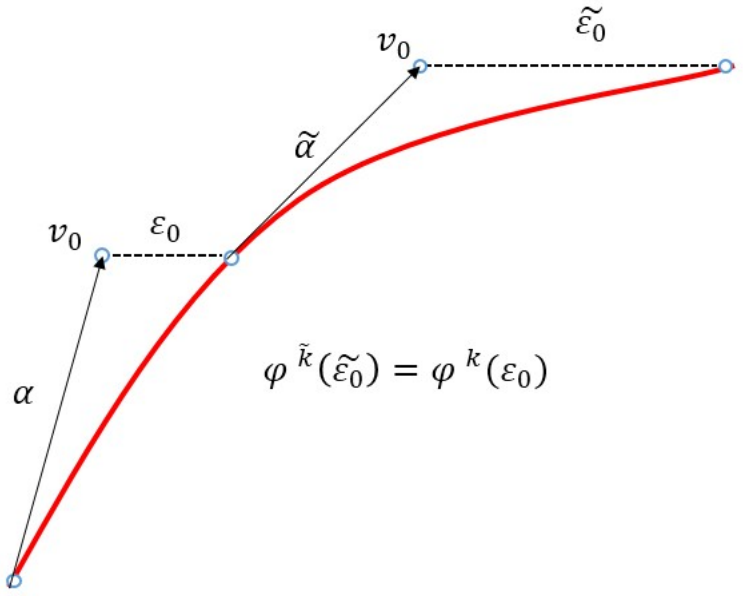

Figure 3.3: The relation of the previous and the future error

the old value $\tilde{\alpha} = \alpha$.

The main objective has been accomplished and the relation of the future error with respect to the number of iteration is obtained $\varepsilon(k)$. Moreover, the resulting error estimation from the proposed algorithm in tandem with steplength Equation (3.7) derived in the first stage, will give such steplength that the corrector would converge only in the desired number of iteration. Therefore, this approach gives control upon the rate of convergence of underlying formulation.

3.2.2 Error model validation

The main objective of this section is to determine whenever the model accurately describe the error decay followed by each iteration in the iteration sequence. Moreover, the order of accuracy is very important for retrieving the reliable estimate of the future error, where the corresponding steplength $\tilde{\alpha}$ will be obtained by solving Equation (3.7) or (3.10). It is highly important to select a reliable error model, taking into account that it will be to be an essential kernel for the whole Modified Continuation Newton algorithm. It was highlighted by many researchers, with increasing the accuracy of the error model in describing the decay

behavior of the previous timestep, would lead to more efficient steplength selection strategy.

In order to validate a suitability of the models for the subsurface flow purposes, each of them has been verified on the two phase flow simulation model with capillary and gravity effects on SPE10 geological grid with 60x220x2 dimensions. 5 production as well as 2 injection wells have been placed on 5-spot inverted pattern on the grid. The capillary gravity equilibrium was computed to initialize distribution of primary reservoir variables. In order to safeguard Newton's updates, the Modified Appleyard damping strategy is employed.

The test simulation is performed for the series of timesteps, $\Delta t = [0.05, 0.5, 1.0, 10.0]$, at each timestep the sequence of iterates is recorded. The motivation of taking variety of timesteps is to bring the predicted point further from the solution at a particular timestep size, which will stress the nonlinear Newton solver. Thus, a sequence will result in an increased number of iterations taken to converge which will give leverage for the proper error model evaluation.

Figures(3.4, 3.5, 3.6) present the comparison, where on vertical axis - the relative error decay measurement; on horizontal axis - the number of iterations; red curve - the recorded sequence of iterations; blue curve - the estimated error computed by one of the models. As was mentioned before, the sequence is the normalized measure of a proximity of each Newton iterate to the solution of a particular timestep, also it can be viewed as the relative error measurement of each iterate. The estimated error is an approximation of the real error computed by the models, based on behavior regulation equation (3.11), where only some of the information of the current iteration sequence was provided. Based on how close the two curves are throughout the error decay, the conclusion of the most suitable model for the discussed purposes could be drawn.

The first classical error model was tested by the formulation given in the equation (3.12), and repeated below:

$$\varphi(\varepsilon) = \frac{\varepsilon^2}{3 - 2\varepsilon}.$$

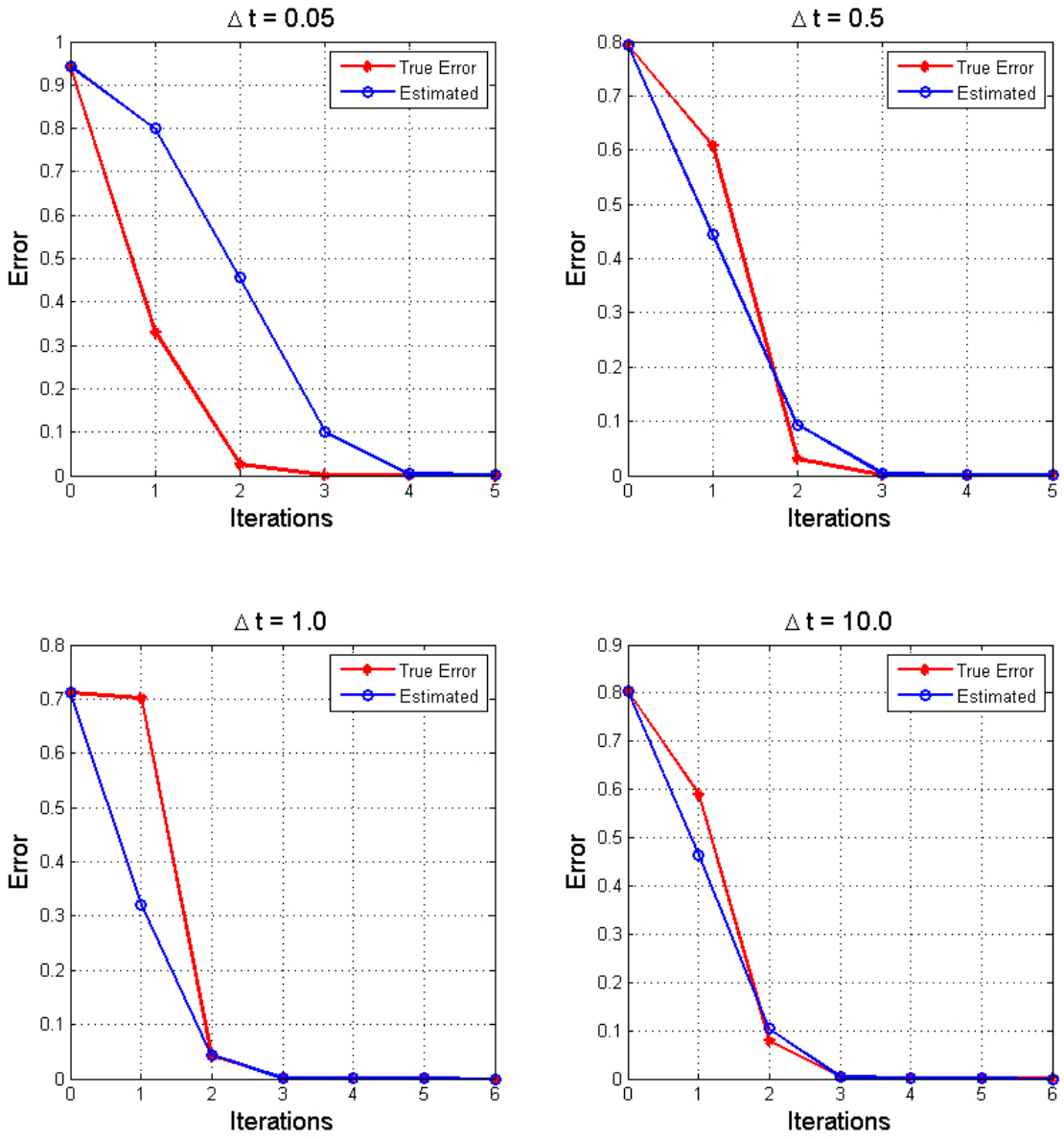

Figure 3.4: Validation of the first error model for the four cases: (a) - $\Delta t = 0.05$ *days*, (b) - $\Delta t = 0.5$ *days*, (c) - $\Delta t = 1.0$ *days*, (d) - $\Delta t = 10.0$ *days*

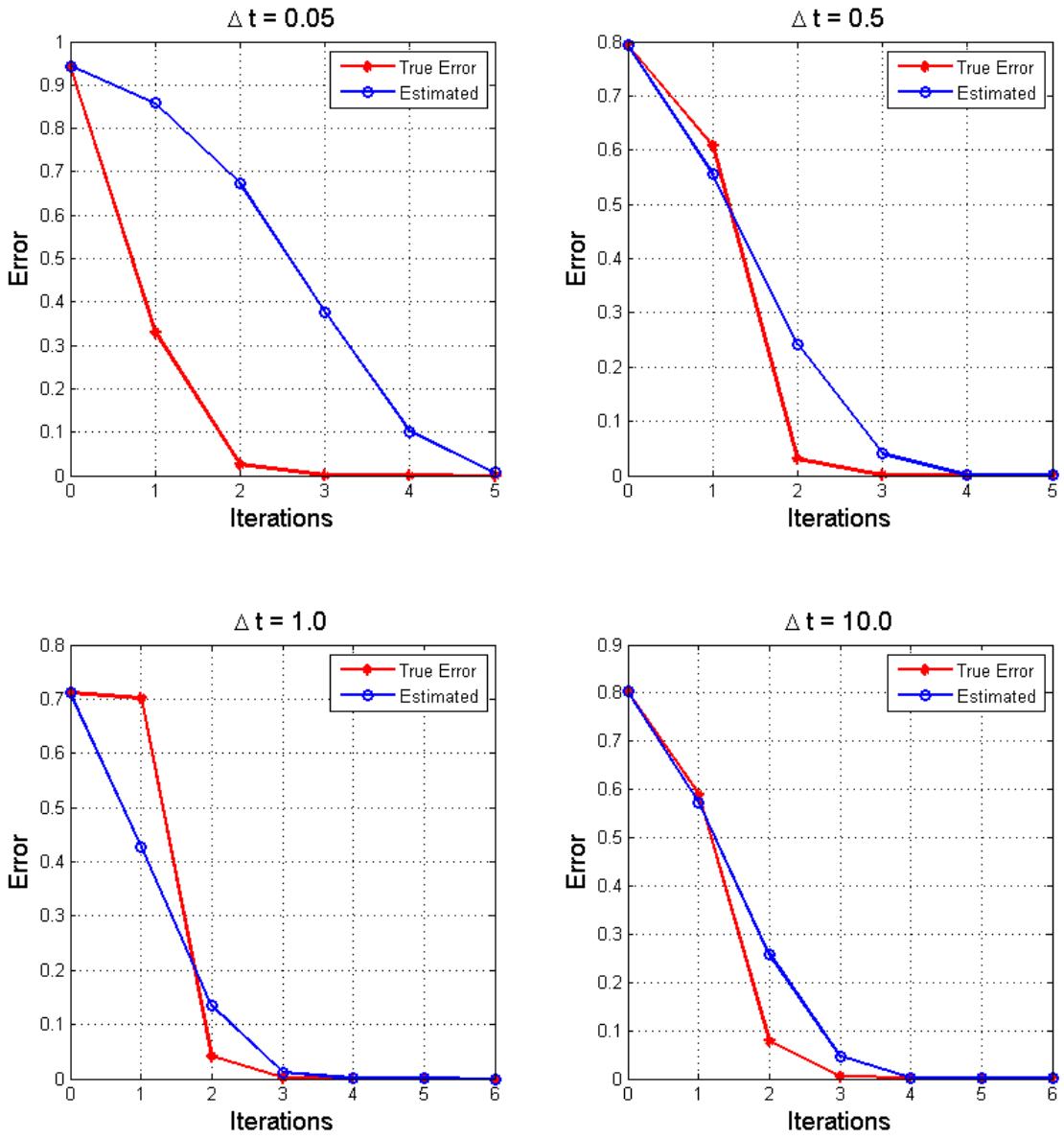

Figure 3.5: Validation of the second error model for the four cases: (a) - $\Delta t = 0.05$ *days*, (b) - $\Delta t = 0.5$ *days*, (c) - $\Delta t = 1.0$ *days*, (d) - $\Delta t = 10.0$ *days*

As presented in Figure 3.4, the estimation obtained from the model is not precisely following the curve of the real decay sequence. Moreover, the estimated curve preserves the uniform shape throughout all the points which is the main reason of the distinction of two curves.

The second classical model stated in the equation (3.13), and repeated below:

$$\varphi(\varepsilon) = \frac{\varepsilon + \sqrt{10 - \varepsilon^2}}{5 - \varepsilon^2}\varepsilon^2.$$

Following from Figure 3.5, we obtained almost the same result as for the first tested model, but at the following timesteps $\Delta t = [0.05, 0.5]$ the estimated error increased the proximity to the real error curve.

Having tested two of the models, an intermediate conclusion could be drawn. The error models which were derived from the Newton-Kantorovich theory have a difficulty in following the error decay of a non-quadratic convergence rate behavior. Hence, tested models have small compatibility in the reservoir simulation, because in rarely cases the reservoir simulation iterative sequence converges with uniform rate, most of the time iterates are experiencing other type of convergence: sublinear or superlinear. Therefore, the estimated error obtained from these models will always be underestimated. Such drawback lessen the applicability of these models in subsurface flow simulation problems.

The last proposed classical model comes from superlinear convergence consideration, presented in the equation (3.14), and also stated below:

$$\varphi(\varepsilon) = \varepsilon^p, \quad p > 1.$$

The error model gives more flexibility to fit the real error curve, because it has a varying exponential parameter p which can be different and be specified at each simulation run. In order to test the sensitivity of the model on the exponential parameter p, the range $1.25 \leq p \leq 4.75$ are tested for various timesteps.

Following from Figure 3.6, the accuracy of the error model varies with different

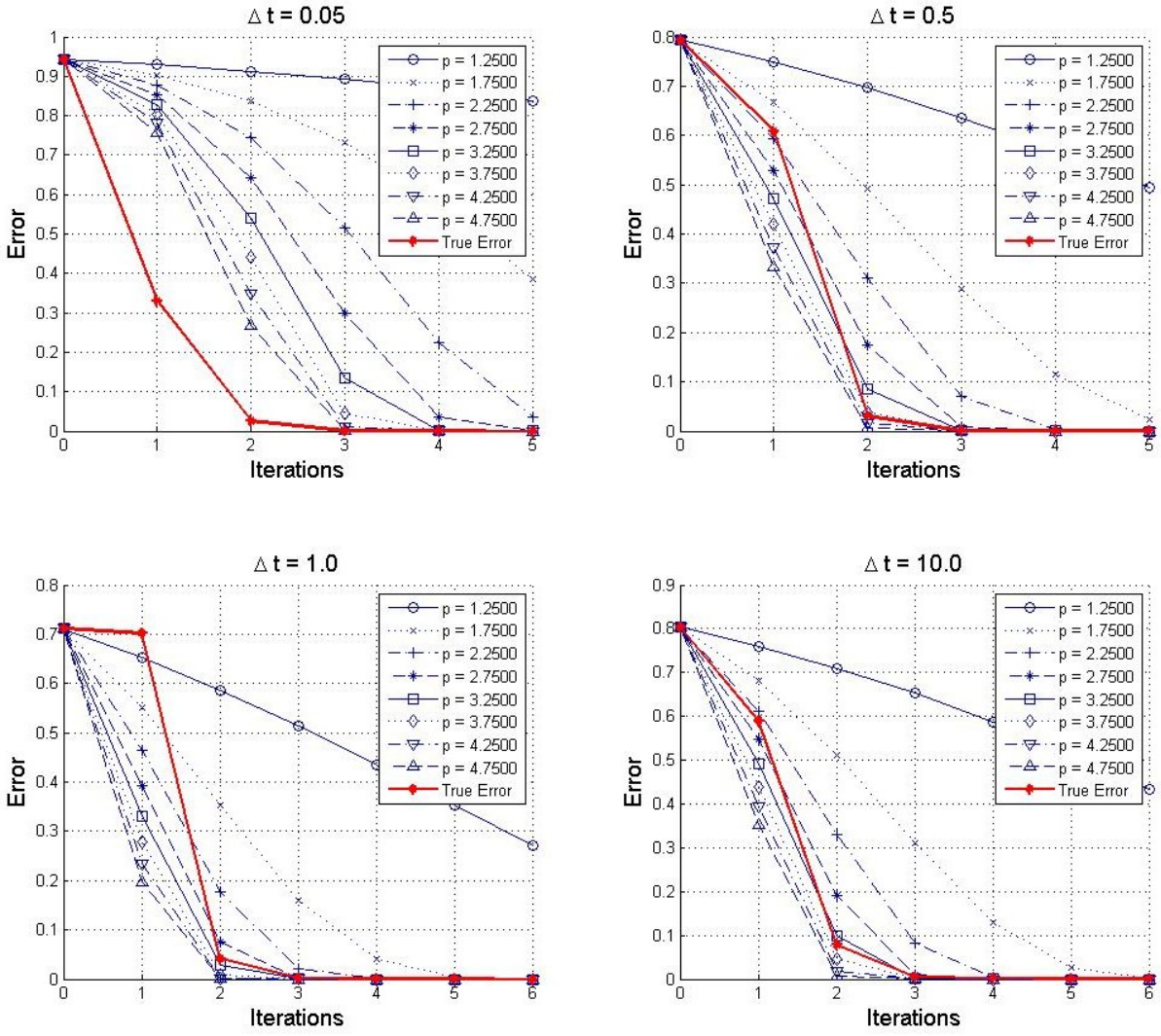

Figure 3.6: The superlinear model validation for the four cases: (a) - $\Delta t = 0.05$ *days*, (b) - $\Delta t = 0.5$ *days*, (c) - $\Delta t = 1.0$ *days*, (d) - $\Delta t = 10.0$ *days*

timestep sizes. Moreover, a good approximation result in cases when the exponential value was known prior the computations. The shape of the estimated error curve highly depend on exponential parameter p and using an arbitrary value will not result in a good approximation because at some timesteps the changes in the reservoir physics can happen faster, which will affect the convergence rate. The shape of the approximated decay in some sense compatible with the previous models, because it also assumes a uniform convergence rate:superlinear.

The previous three error models were based on the the following behavior regulation $\varepsilon_{i+1}(\alpha) \leq \varphi(\varepsilon_i(\alpha))$. Here, will encourage a discussion for a different type:

$$\varepsilon_{K+1}(\alpha) \leq \varepsilon_0(\alpha)\mathcal{F}(K), \tag{3.18}$$

where $\mathcal{F}(K)$ is an arbitrary decaying function, K is an iteration counter which is the forcing term in the formulation. Note, the requirement of having only normalized error $0 \leq \varepsilon \leq 1$ is no longer applicable. The error can be computed as a difference between each iterate of the sequence to the solution $||U_i - U_n||, \quad i = 0 \dots n$.

The rest of this section will be dedicated to the famous exponentially decaying error model. The model is used in a variety of engineering applications for modeling naturally accruing processes. Below, presented a differential form of the model:

$$\frac{d\varepsilon}{dK} = -\eta\varepsilon. \tag{3.19}$$

The solution of the ordinary differential equation (3.19):

$$\varepsilon(K) = \varepsilon_0 exp(-\eta K), \tag{3.20}$$

where η is an exponential decay constant, and ε_0 is the initial deviation from the solution point. The model takes into account a changes in the error along the iteration sequence. In order to increase the accuracy and the ability of fitting the exponential model, an additional degree of freedom introduced to the model by a coefficient β:

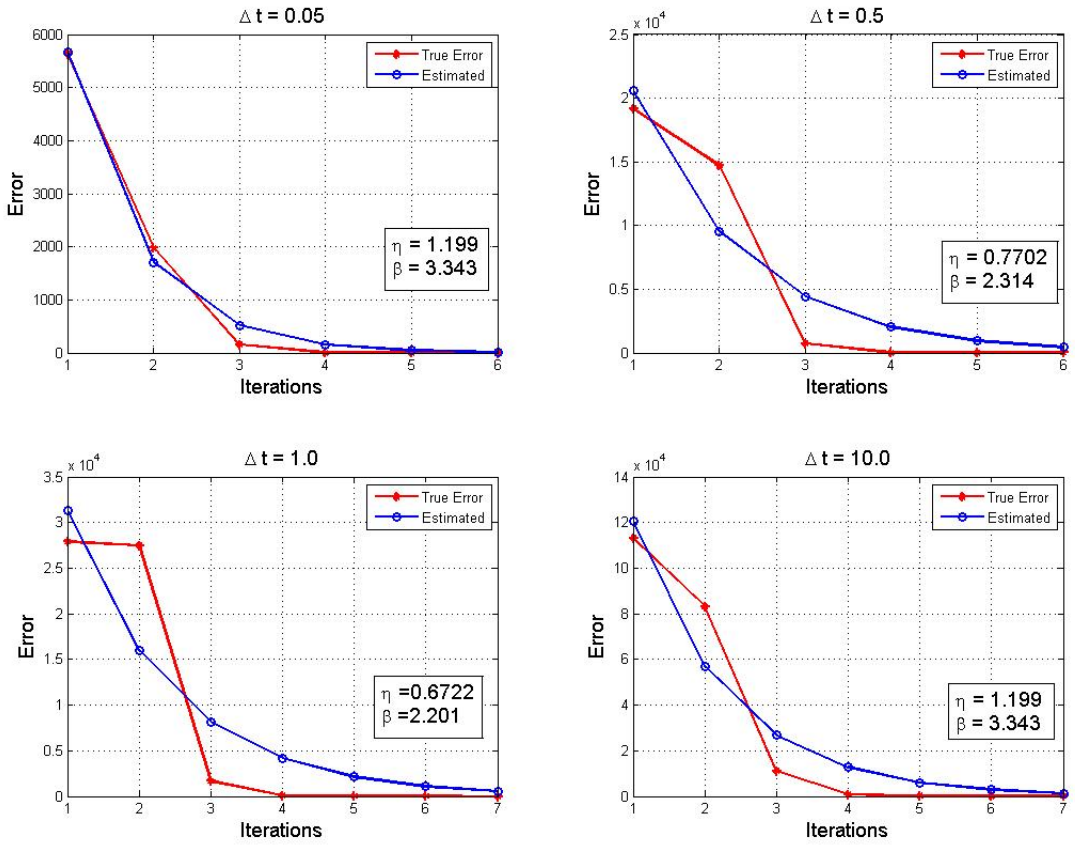

Figure 3.7: Exponential model validation for the four cases: (a) - $\Delta t = 0.05$ *days*, (b) - $\Delta t = 0.5$ *days*, (c) - $\Delta t = 1.0$ *days*, (d) - $\Delta t = 10.0$ *days*

$$\varepsilon(K) = \beta\varepsilon_0 exp(-\eta K). \tag{3.21}$$

The results are shown on Figure 3.7, the model adjusted by the coefficients η and β for each of the tested timesteps. The model shows a good accuracy in the first timestep $\Delta t = 0.05$ test, but it still lacks flexibility for other ones. The main reason again is in the convergence rate of the formulation. Moreover, even with perfectly fitted exponential model the end points are asymptotically approaching the zero, but with a slower rate. Hence, the difference at the end points could be big.

In summary, each of the tested error models has some ability in accurate description of certain parts of the real error curve. Where, the first two model of Kantorovich theory as well

as exponential have a good degree of accuracy in the regions of the quadratic convergence; the second model of superlinear convergence results with a good approximation where primarily the superlinear behavior occurs.

Hence, we need to develop an error model, which will be able to account for for different convergence rate in the same iteration sequence.

3.2.3 Reservoir Simulation Error Behavior

The reasons for the failures of the previous models were in the conservative estimation of the convergence rate throughout the iterations. The uniform convergence rate consideration cannot be applied for the wide range of simulation cases. Mainly, the shortfall of a uniform convergence is due to the guess being outside the quadratic convergence region. The reasons for stated in the previous sentence problem: first, the complex modeled physics which is able to change dramatically even with a small value of the timestep size; second, taking the timestep size bigger than some threshold for moderate physics.

The typical example of the reservoir simulation error decay is shown on Figure 3.8. At the early stage when a few iterations performed, it might be observed that the error stalls, which can be described as the sublinear convergence rate region. After some iteration it enters a new stage of superlinear convergence, where the changes are occurring faster than in the previous. Finally, when the performed iterations brought the guess in the quadratic convergence region, the convergence happens with quadratic speed.

The main objective of this section is to find an error model which could encompass various convergence rates inside one formulation.

It seems that a logistic function has all the features to be an appropriate fit for the peculiar error decay in the reservoir simulation cases. Moreover, it has very long history in various scientific applications: ecology, modeling population growth, statistic, logistic regression; physics, Fermi distribution etc. It essentially represents an "S" shape curve which defined for all real values and has a non-zero first derivative at each point. The end points are asymptotically approaching zero of the function (3.22) as $x \to \pm\infty$. The general

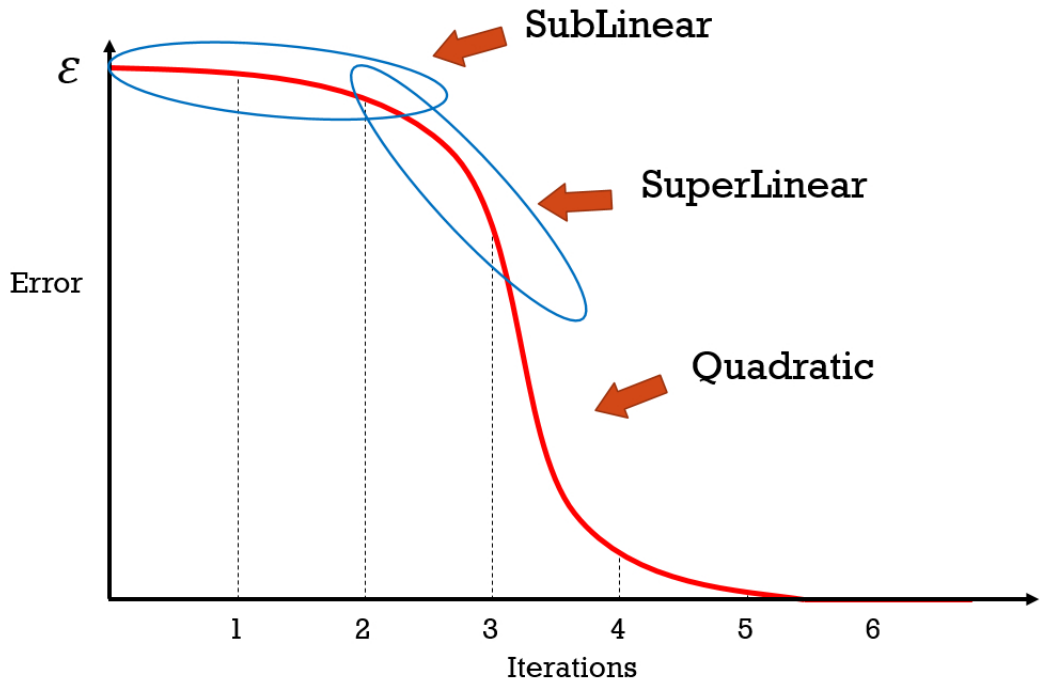

Figure 3.8: Reservoir Simulation error decay

form of the logistic function stated as:

$$\mathcal{F}(x) = \frac{1}{1 + \exp^{-x}}. \tag{3.22}$$

A successful logistic function for reservoir simulation obtained by introducing two additional degree of freedom B and M to the equation (3.22). In the following, the error model formulation could be written:

$$\varepsilon(N) = \varepsilon_0 \frac{1 + \exp(B - M)}{1 + \exp(-M + B(1 - N))}, \tag{3.23}$$

where $\varepsilon(N)$ - the error decay value throughout the iterations N; ε_0 - initial error of the iteration sequence and B and M are fitting coefficients.

Carefully chosen coefficient place in formulation (3.23), enables to change some of the important regions on the logistic function in order to increase the accuracy of the approximation. B coefficient accounts for prolongation of the quadratic region or in other words, the sharpness of the error decay curve (Figure 3.9). For instance, increasing B will result in

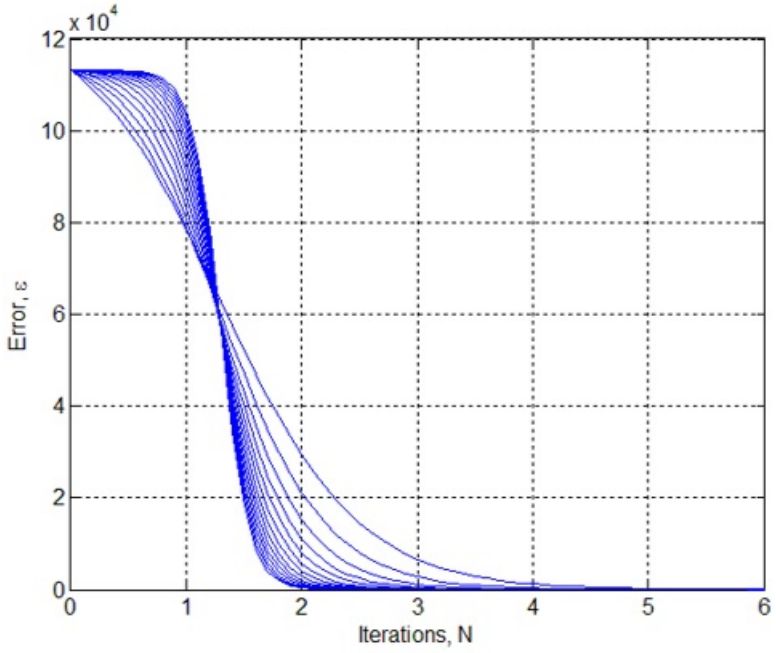

Figure 3.9: Logistic Function: B dependency

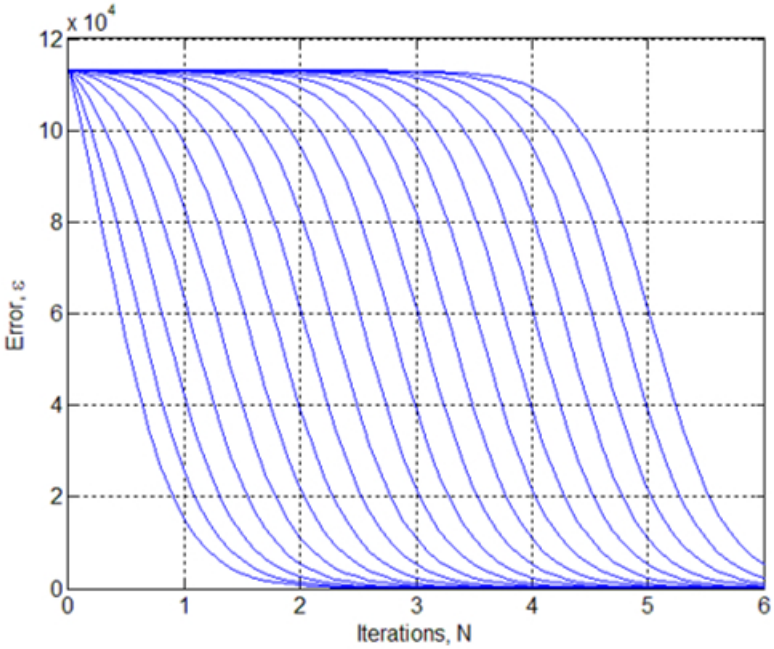

Figure 3.10: Logistic Function: M dependency

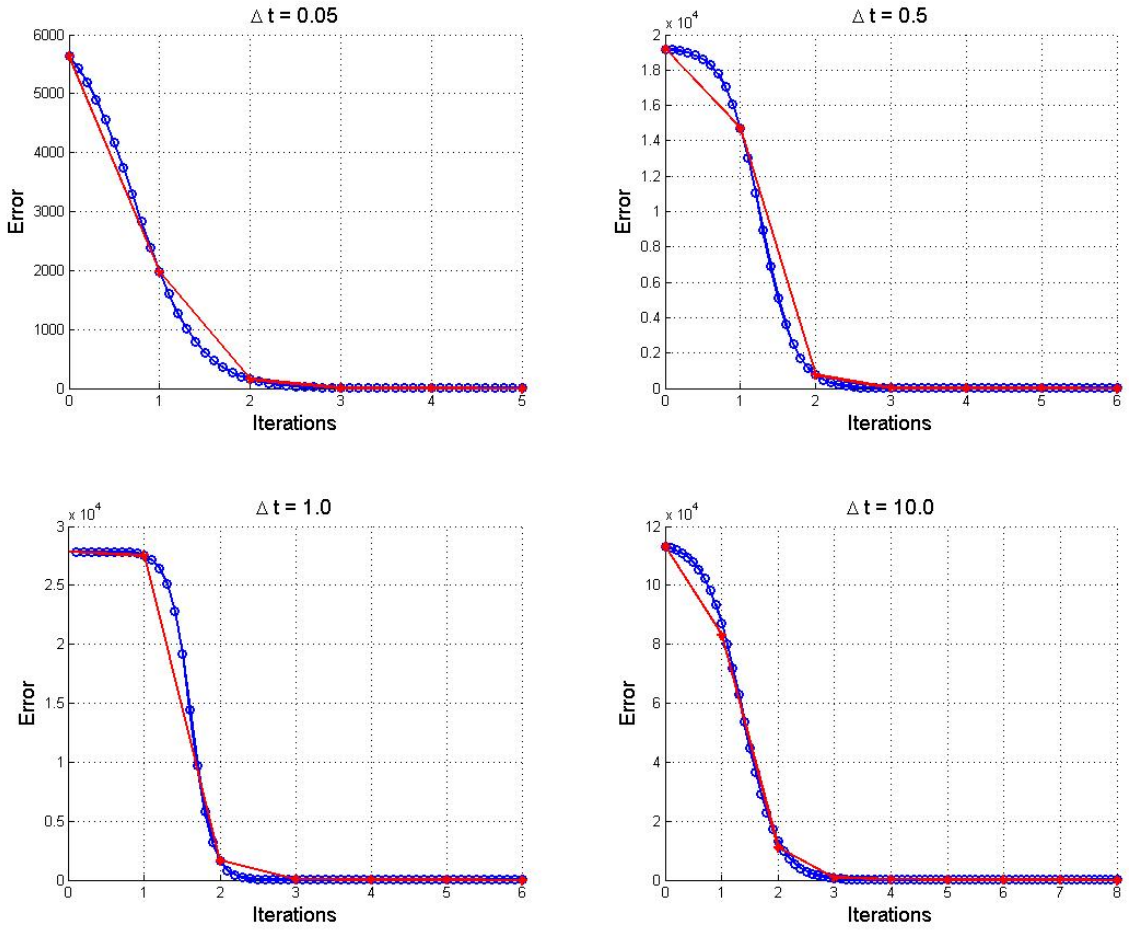

Figure 3.11: The logistic function model validation for the four cases: (a) - $\Delta t = 0.05$ *days*, (b) - $\Delta t = 0.5$ *days*, (c) - $\Delta t = 1.0$ *days*, (d) - $\Delta t = 10.0$ *days*

a greater sublinear and smaller superlinear region. M coefficient responsible for the length of the sublinear region (Figure 3.10).

In order to validate the model for the subsurface flow simulation purposes. The same test case used as for previous error models: 3-dimensional, the two phase flow simulation model with capillary and gravity effects on SPE10 geological grid with 60x220x2 dimensions, and 5 production as well as 2 injection wells. The following series of timesteps $\Delta t = [0.05, 0.5, 1.0, 10.0]$ *days* are tested.

The results are shown on Figure 3.11. The B and M coefficients were computed for each case, based on the two given points on real error decay curve and the initial error ε_0.

It can be observes that the model has a remarkable accuracy to the real error decay, even a stall in the case $\Delta t = 1.0$ was not a problem.

3.3 Conclusion

This section concludes the proposed steplength selection algorithm which is based only on theoretical considerations. In contrast, the previously used steplength algorithm (Convergence Neighborhood) in reservoir simulation was based on heuristic which had a shortage of applicability for general cases. The proposed algorithm is able to select the most appropriate steplength along the tangent line according to a user given iteration constraints. The strategy is broken down into two stages:

1. Stage 1: A more accurate and reliable relation between a steplength and an initial error which does not have major restrictions is derived. The relation does not employ the assumption of having the same curvature value at neighboring points. The following represent the relation:

$$\tilde{\alpha} = \frac{\tilde{\varepsilon}_0}{\left\| \frac{dU(\alpha)}{d\lambda} \right\|}.$$

2. Stage 2: The derivation of the steplength on the iteration number dependency required a deep investigation. The relation could be obtained by enforcing Equation (3.17) or the following formulation:

$$\varphi^{\tilde{k}}(\tilde{\varepsilon}_0) = TOL, \tag{3.24}$$

where φ is the error model; $\tilde{k}$ is the desired number of iterations; TOL is an error value that a user wants to observe after $\tilde{k}$ iterations and $\tilde{\varepsilon}_0$ is an initial error for the future timestep which in this equation is an unknown.

The main challenge of the above formulation is to define the error model φ which will fit for reservoir simulation cases. A different speed of convergence is the major

restricting obstacle for many error models. The best fitted error model was derived based on the logistic function:

$$\varepsilon(N) = \varepsilon_0 \frac{1 + \exp(B - M)}{1 + \exp(-M + B(1 - N))},$$

where B and M are fitting parameters of the model.

The overall algorithm can be written as follows:

Algorithm 3: The steplength selection algorithm

Data: U_i - iteration sequence, $\tilde{k}$ - desired number of iterations
Result: Future steplength $\tilde{\alpha}$
Obtain the coefficients B and M from the sequence U_i
Solve for $\tilde{\varepsilon}_0$ from the logistic function error model (3.23) and relation (3.24)
Finally, compute the future steplength along the tangent line using formulation (3.10)

CHAPTER 4

SIMULATION EXAMPLES

The Modified Continuation-Newton (MCN) is a result of a combination of two essential components: the devised tangent line formulation and the steplength selection routine. A simulation case of three phase flow with gravity is used to investigate the performance of the proposed MCN algorithm.

4.1 The three phase compressible flow

The back oil simulation involves a compressible flow of three phases. Oil and gas are non-wetting phases, and water is wetting. Mass of each component is conserved, but mass of a phase is not. Assumptions, no mass transfer between water and hydrocarbon phases, and no hydrocarbon component exist within the water phase. For instance, if a pressure is greater than a bubble point pressure P_b at a particular part of a reservoir, then the gas component is dissolved into the oil phase and not into water. Moreover, the oil phase consists of the gas and the oil components; vapor phase only gas component.

PVT behavior of the system is expressed by formation volume factors as follows:

$$B_o = \frac{[V_o + V_{dg}]_{RC}}{[V_o]_{STC}}$$

$$B_w = \frac{[V_w]_{RC}}{[V_w]_{STC}}$$

$$B_g = \frac{[V_g]_{RC}}{[V_g]_{STC}}.$$

The mass transfer between oil and gas phases is described by the solution gas-oil ratio:

$$R_s = \left[\frac{V_{dg}}{V_o}\right]_{STC}.$$

The governing equation for the oil phase: [2]

$$-\nabla\left[\frac{1}{B_o}u_o\right] = \frac{\partial}{\partial t}\left[\frac{1}{B_o}\phi S_o\right] + q_o, \tag{4.1}$$

where B_o is the oil formation volume factor (PVT behavior); u_o - Darcy velocity of the oil phase and ϕ is the porosity of the reservoir and q_o is a sink/source term.

The water equation:

$$-\nabla\left[\frac{1}{B_w}u_w\right] = \frac{\partial}{\partial t}\left[\frac{1}{B_w}\phi S_w\right] + q_w. \tag{4.2}$$

The vapor equation:

$$-\nabla\left[\frac{R_s}{B_o}u_o + \frac{1}{B_g}u_g\right] = \frac{\partial}{\partial t}\left[\phi\left(\frac{R_s}{B_o}S_o + \frac{1}{B_g}S_g\right)\right] + q_g + R_s q_o. \tag{4.3}$$

The Darcy's relation between the rate and a pressure gradient can be written in the following form:

$$u = -\frac{k}{\mu}\left(\nabla p + \rho g \Delta z\right), \tag{4.4}$$

where k is a permeability tensor of the reservoir; μ and ρ is the viscosity and the density of a fluid accordingly, g is gravitational acceleration and Δz the difference in the vertical direction.

The Black Oil model is computed on SPE10 (Comparative Solution Project model 2) geological grid. The model dimensions are 1200×2200×170 (ft.), and fine scale cell size is 20(ft)×10(ft)×2(ft). Due to computational time, only several layers of Tarbert formation were taken.

The two wells(production and injection) were located on the opposite grid corners. The injection well operates based on the flow rate control mechanism, whereas the production

on the pressure control P_{bh}

The primary unknowns are depended on the number of phases at a particular condition. When the pressure less than bubble point, and the gas exist as a separate phase, the choice of primary unknowns comes to the pressure of oil P_o, the water Sw and the gas saturation Sg. In the case of the pressure higher than bubble point, the primary unknowns are: the pressure of oil, the water saturation, and the dissolved gas Rs factor.

4.2 Algorithm

The overview of the proposed algorithm and a general implementation for solving any nonlinear problem are the main topic of a discussion here. As was mentioned in the earlier chapters, the Modified Continuation-Newton has to incorporate at least two separate routines in order to be able to solve a stated nonlinear problem:

1. The routine for computing a tangent line has to be developed. In the chapter 2 dedicated to tangent lines, some of the improvements for the tangent line computation and accuracy were reviled. The final algorithm up-to some degree uses those improvements. First, compute simulationationally-wise the timestep size parameterization of the zero residual level curve $\frac{dR(U(\Delta t),\Delta t)}{d\Delta t} = 0$ is chosen. Second, it was proven in the chapter 2, that the higher order predictor results with a slightly better approximation, but it comes on a balance of a considerable expense. Hence, it has not found its implementation in the final algorithm. Third, in order to improve the stability of the predictions, the Heun's method is implemented [11].

   The Heun's method implementation in predictor-corrector scheme represents a morphism of an Euler first order predictor and a trapezoidal rule. Consider, the tangent line differential equation for the reservoir simulation cases:

$$\frac{dU}{d\Delta t} = -J^{-1}\frac{dR}{d\Delta t} = f(U, \Delta t), \quad U(\Delta t_0) = U_0,$$

   where U_0 and Δt_0 are the known state and associated timestep size accordingly

First, from the current state U_0, calculate a predicted value by the Euler method:

$$\tilde{U}^{n+1} = U_0 + \alpha f(U_0, \Delta t_0).$$

Then, improve the prediction by using the trapezoidal rule:

$$U^{n+1} = U_0 + \frac{1}{2}\alpha(f(U_0, \Delta t_0) + f(\tilde{U}^{n+1}, \Delta t^{n+1})). \tag{4.5}$$

The resulted guess U^{n+1}, is more stable withing a considerable range of a steplength α.

2. The routine of the steplength selection. Although, the newly derived error model which is based on the logistic function, has shown a good approximation to the real error behavior in reservoir simulation cases, it can not be implemented for a big scale problems at the current state of the research. The main reason, the computation of the error decay requires to record each Newton's iterate, and only when Newton process converges it is possible to compute the proximity of iterates to the obtained solution. The discrete Newton iterate represents a vector of unknowns with the size of $N_b \times N_{eq}$ number of grid blocks times the quantity of primary equations at each cell. Hence, without knowing a priory how many such iterations will Newton take to converge for big scale problems, it increases the risk of trashing the memory(RAM) or getting out-of-bounds allocation(bad alloc). The possible remedy is to use less informative but more memory efficient quantities. For example, a norm of the residual equation. The residual based error model needs to take only a single sample of the residual vector, which is a scalar. However, at this point the investigation of error model based on other measurements, has to be conducted in the future work.

   In the finial steplength selection algorithm for multi-scale problems, some of the alternation had to implemented:

   First, at the beginning of the simulation the arbitrary small initial error ε_0 for each

variable needs to be stated and a target number of iterations N_{targ} should be specified. The target number of iterations is used as an adjustment factor for the future error.

Second, the steplength needs to be computed based on the provided error variation ε_0. In order to obtain the error relation with the steplength, the Taylor series expansion of the solution path has to be written:

$$\varepsilon_0 = ||U_{pred} - U_\infty|| = \frac{\alpha^2}{2}\frac{d^2U}{d\Delta t^2}, \tag{4.6}$$

where the terms higher than the second were omitted.

To eliminate the tedious computation of the second order derivative $\frac{d^2U}{d\Delta t^2}$, which results in a tensor, a norm of the first order is computed instead. Knowing that higher order term are always bounded by the norm of a lower orders, the following relation can be written:

$$\frac{d^2U}{d\Delta t^2} = \alpha \left|\left| \frac{dU}{d\Delta t} \right|\right|. \tag{4.7}$$

Then, by substituting the relation (4.7) to the error formulation (4.6), the steplength along the tangent line can be computed as:

$$\alpha = \frac{\varepsilon_0}{\left|\left| \frac{dU}{d\Delta t} \right|\right|}. \tag{4.8}$$

Because the initial error is define arbitrary, it needs to make adjustments throughout the simulation run in order to adapt the most reliable steplength. The proposed way is to compute the ratio of the target N_{targ} and taken N number of iterations as follows:

$$\varepsilon_0^{n+1} = \varepsilon_0^n \frac{N_{targ}}{N^b}, \quad b = \frac{\left|\left| \frac{dU}{d\Delta t} \right|\right|_{old}}{\left|\left| \frac{dU}{d\Delta t} \right|\right|}, \tag{4.9}$$

where ε_0^{n+1} is the adjusted error value, b is the ratio of the old tangent and the current one. The b ratio gives more stable timestep advancement because it changes according

to the undergoing physics.

The algorithm is presented on flow chart 4.1 and on the schematic plot 4.2. First, the small error for each of the independent variables needs to be chosen in such a way that it would result in sufficiently small first steplength α. As shown on Figure 4.2, the first tangent line $\hat{t}_1$ is emanating from the origin point $S_0 = (U^n, \Delta t)$, which is a known state. The Newton's method is used to bring the predicted value U_{pred} to the solution point U_∞. After obtaining the solution, the error adaptation has to be computed in order to get the steplength consistent with the target number of iterations. These discrete procedures will be repeated until the timestep size requirement is met.

4.3 Results

As was stated before, due to computational time the first 5 layers of SPE10 were chosen for the simulation (Figure 4.3). The two cases were drawn based on SPE10, where both of them have the same grid dimension of $[NX = 60, NY = 220, NZ = 5]$ and the gravity forces acting in the formulation. The first case is dedicated to test the possible limitations of MCN algorithm in solving the problem with moderate physics. It also includes two wells: one is production, which operates based on the pressure control at 2000 psi and the other one is injection well, which is the rate control with an injection rate of 0.05 $\frac{rb}{day}$. The second case is meant to test the ability of the MCN to handle stiff physic problems, where one additional couple of injection and production wells was placed on the grid. Those wells have the increased injection rate to 0.1 $\frac{rb}{day}$ and decreased bottom hole pressure for production wells to 1000 psi. The correctness of the developed simulator is verified based on the comparison with Eclipse commercial simulator in the Appendix A of this work.

For each simulation case, the parameters as the target timestep size, the number of iterations, and the initial error are to be selected. The initial errors for both of the cases are chosen to be the same. Moreover, it assumed that the future pressure change will be bounded by the value of the error, which $\varepsilon_0(P)$ is 25 psi. The error for the saturation of water and gas is chosen as 0.2.

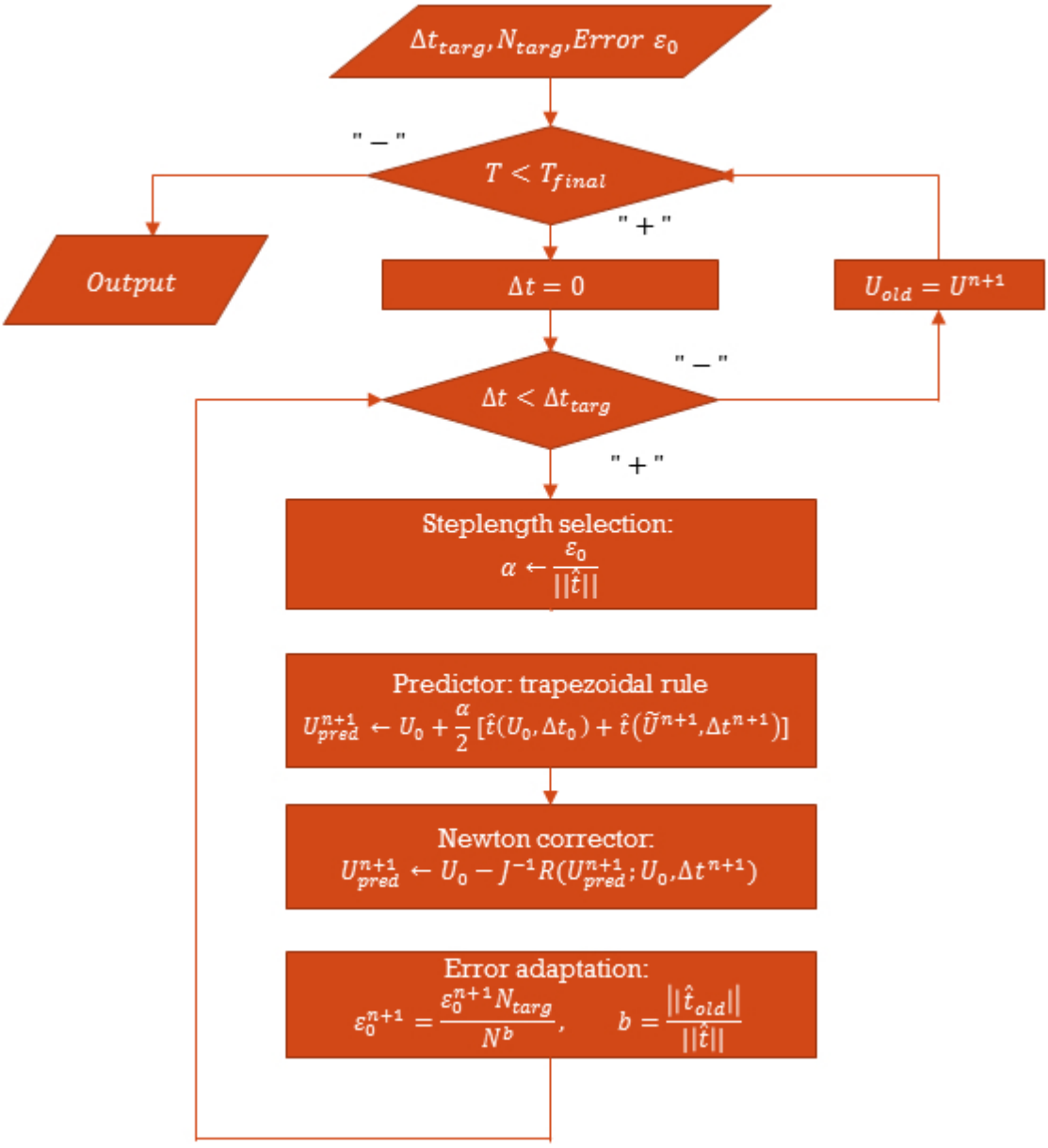

Figure 4.1: Flow chart of the proposed MCN.

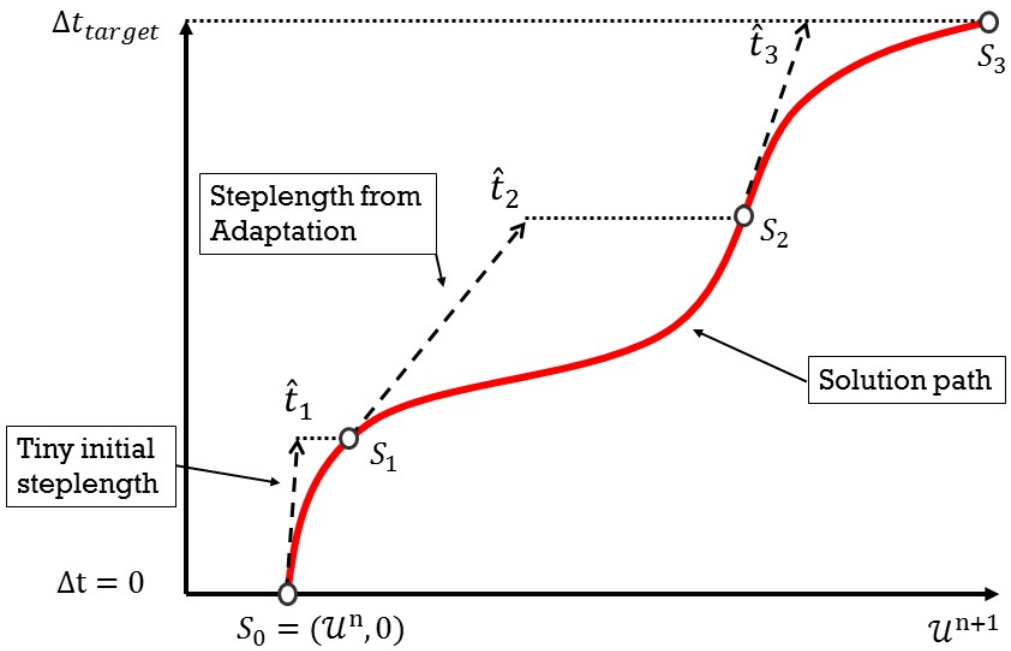

Figure 4.2: Illustration of the MCN algorithm

The selection of the target number of iterations is purely based on the performance considerations. One might choose the target to be 3 iterations, and follow the solution path closely which will guarantee the convergence all the time. Whereas others might choose it to be 10 or more iterations, it increases the efficiency but acquires the possibility of cutting the steplength at some point. Such possibility is dramatically decreased in the proposed algorithm, by the use of some physics change information. For both cases the target number of iterations N_{targ} is selected to be 6.

The target timestep size as well plays an important role in the performance of the algorithm because along the solution path if something major happens it would significantly change the shape. for instance, a new well comes on, phase appearance/disappearance. Therefore, following such solution path would result in many iterations. Hence, the old state variables need to be updated with a certain frequency. For both of the cases, the target timestep size is chosen to be 50 days.

The algorithm, however, needs to have a prepossessing routine, which will be able to test the different convergence criteria. It would propose the most applicable parameters

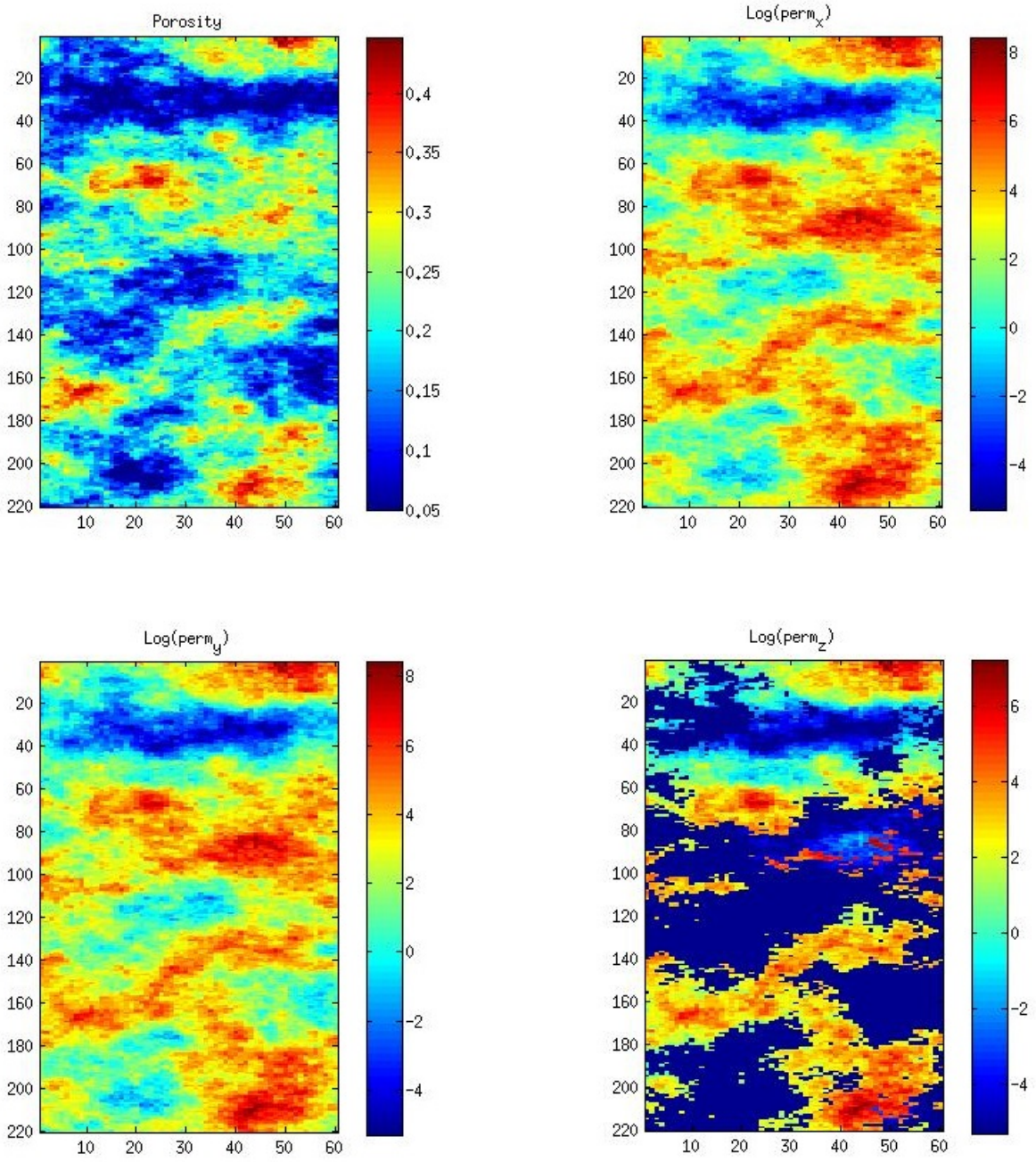

Figure 4.3: The parameters distribution for the first layer

of the target timestep size and number of iterations for the specific simulation case. Such routine is in a developing stage and will be presented in the following works.

The following represents a one timestep simulation test for the moderate physic case.

4.3.1 One timestep simulation

In this section, the full simulation is performed as one timestep in order to be able to measure the maximum capability of timestep convergence and performance of the proposed algorithm and the standard Newton's method. One timestep simulation was performed based on moderate physic case because it would not limit the convergence of standard Newton, which would give leverage in appropriate comparison. The range of timestep sizes initiates from 0 to 57 days, which would be enough to compare the two methods.

The results are presented on Figure 4.4, where horizontal axis is the range of timestep sizes and vertical axis is dedicated for number of iterations taken at particular timestep size. It can be seen that the standard Newton works good in the earlier timestep sizes, but after $\Delta t = 10 days$ it stops converging. On the other hand, the proposed Modified Continuation-Newton performs reasonably well throughout all the timestep sizes.

The obtained results could be interpreted as: the standard Newton converges with higher rate where it can, but when the timestep size takes the corresponding solution further from the initial guess, the standard newton does not converge within the prescribed number of iterations $MAXITER = 500$. In contrast, the MCN algorithm is not very sensitive on the initial guess and converges every timestep.

4.3.2 Full simulation

In both cases, the full simulation was computed up-to 1000 days. The comparison were made between the proposed MCN algorithm and the-state-of-art Newton's solver, which is safeguarded by the modified Appleyard chop, and employs Eclipse timestepping strategy.

The results for the first case of moderate physic are shown on Figure 4.5. The horizontal axis is the simulation time and the vertical axis is the number of linear solves the simulator performed during the calculations. It can be noted that the Eclipse based

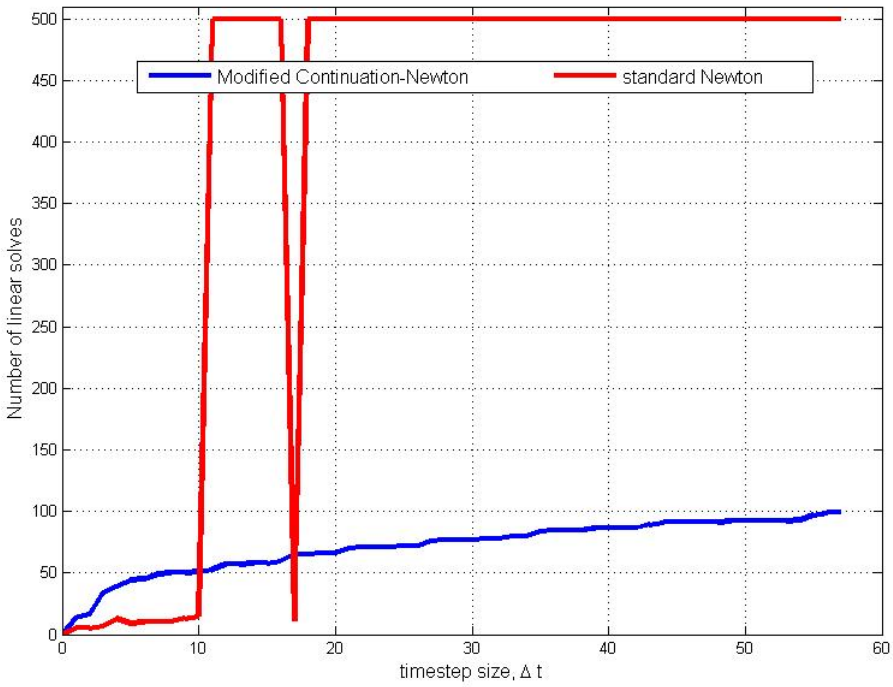

Figure 4.4: The full simulation as one timestep size

standard Newton (red curve) is lying always below the MCN (blue curve). Moreover, the standard Newton after 1000 days resulted with 281 linear solves, whereas MCN has 465 linear solves. The main reason for the deficiency of the MCN is that the physics undergoing in the reservoir is not stiff enough, which prevents the use of the knowledge of the solution path. The MCN lacks the sense of a good convergence rate at a particular timestep size. For instance, the MCN at some point in the simulation takes steplength to be 5 days, which converges after 6 iterations. On the other hand, the standard Newton takes a bigger timestep size of 10 days and would have the same number of iterations to converge. In the light of the obtained results, the error adaptation routine has to be modified, which should not be based on the number of iterations counter. Figure 4.7 shows, the timestep size when the standard Newton could not obtain the solution and where the chop is applied. The first layer variables distribution after 1000 days of simulation is presented on Figure 4.9.

The comparison results of the stiff physic case are presented on Figure 4.6. It can be seen that the standard Newton lies above the MCN curve all the time because of the

complex physic and its inability to identify the changes in the state unknowns, which makes the standard Newton to cut the timestep size along the simulation run. As shown on Figure 4.7, the simulation based on the standard Newton had to be terminated after 431 days because the timestep size was prohibitively small $\Delta t < 0.0001\ day$. Therefore, the MCN took 913 linear solves for 1000 days and the standard Newton took 7505 linear solves for 431 days. The problem with convergence might be related to a variable switching. On the other hand, the MCN algorithm has no problem with this case. The first layer variables distribution after 1000 days of simulation is presented on Figure 4.10.

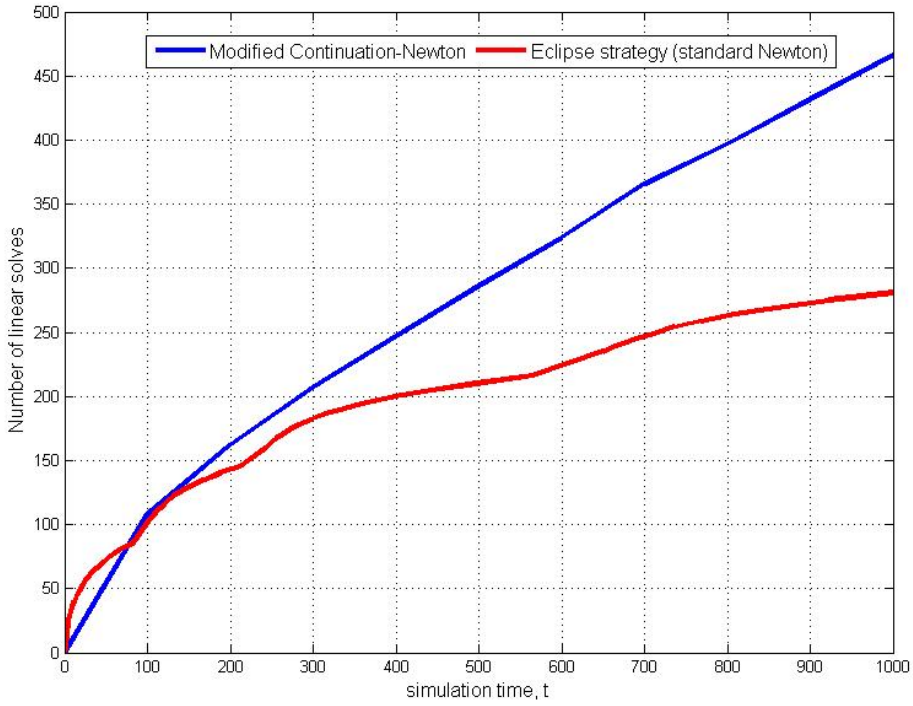

Figure 4.5: First case: the full simulation for 1000 days.

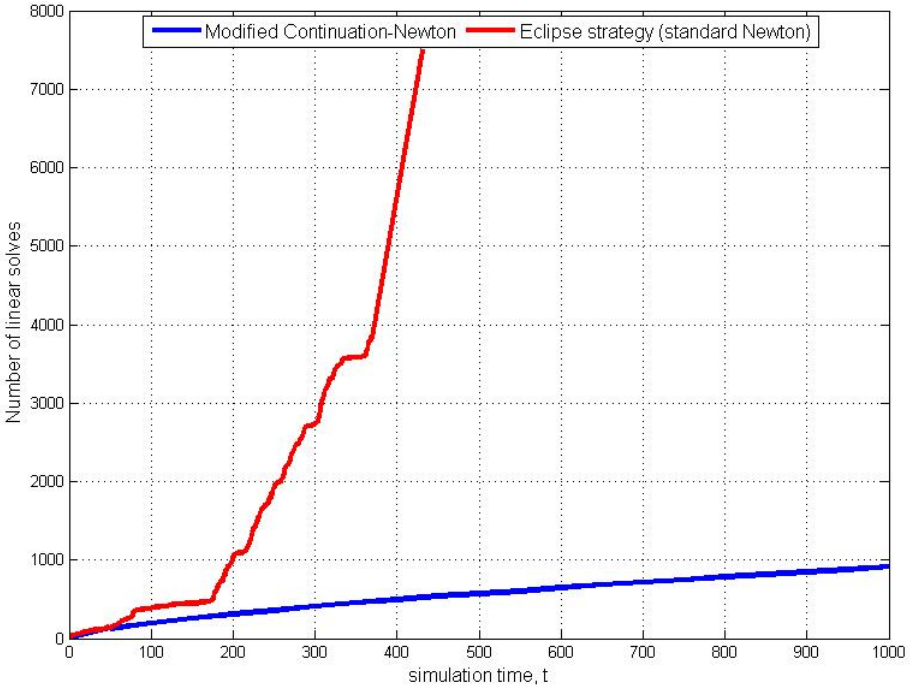

Figure 4.6: Second case: the full simulation for 1000 days.

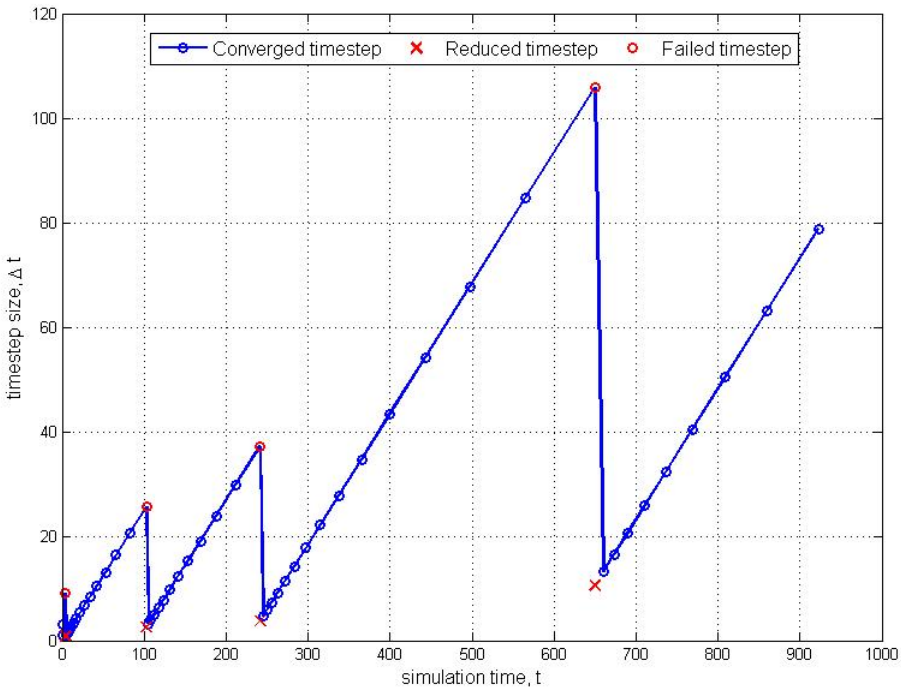

Figure 4.7: First case: standard Newton timestep reduction.

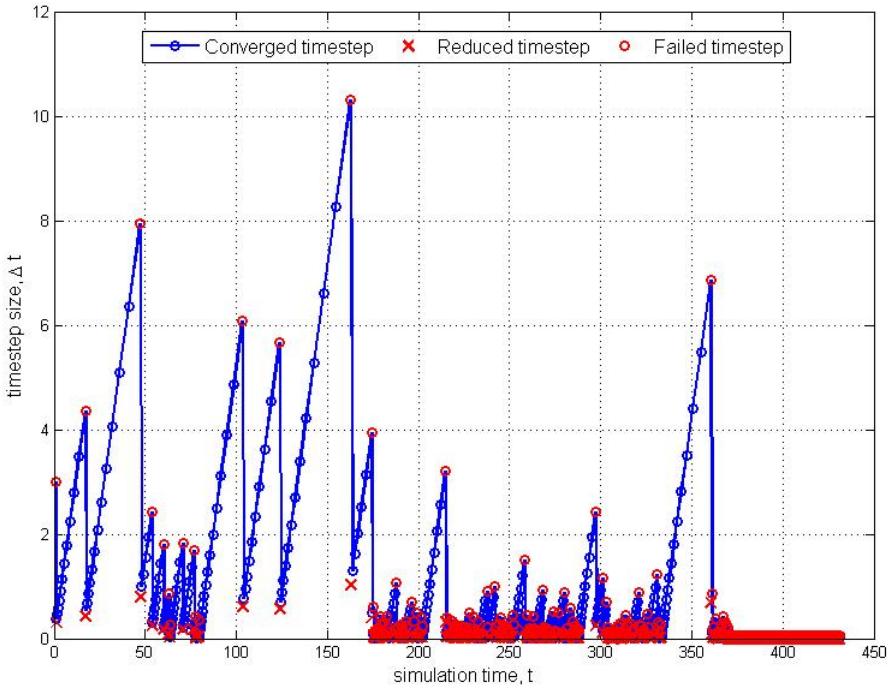

Figure 4.8: Second case: standard Newton timestep reduction.

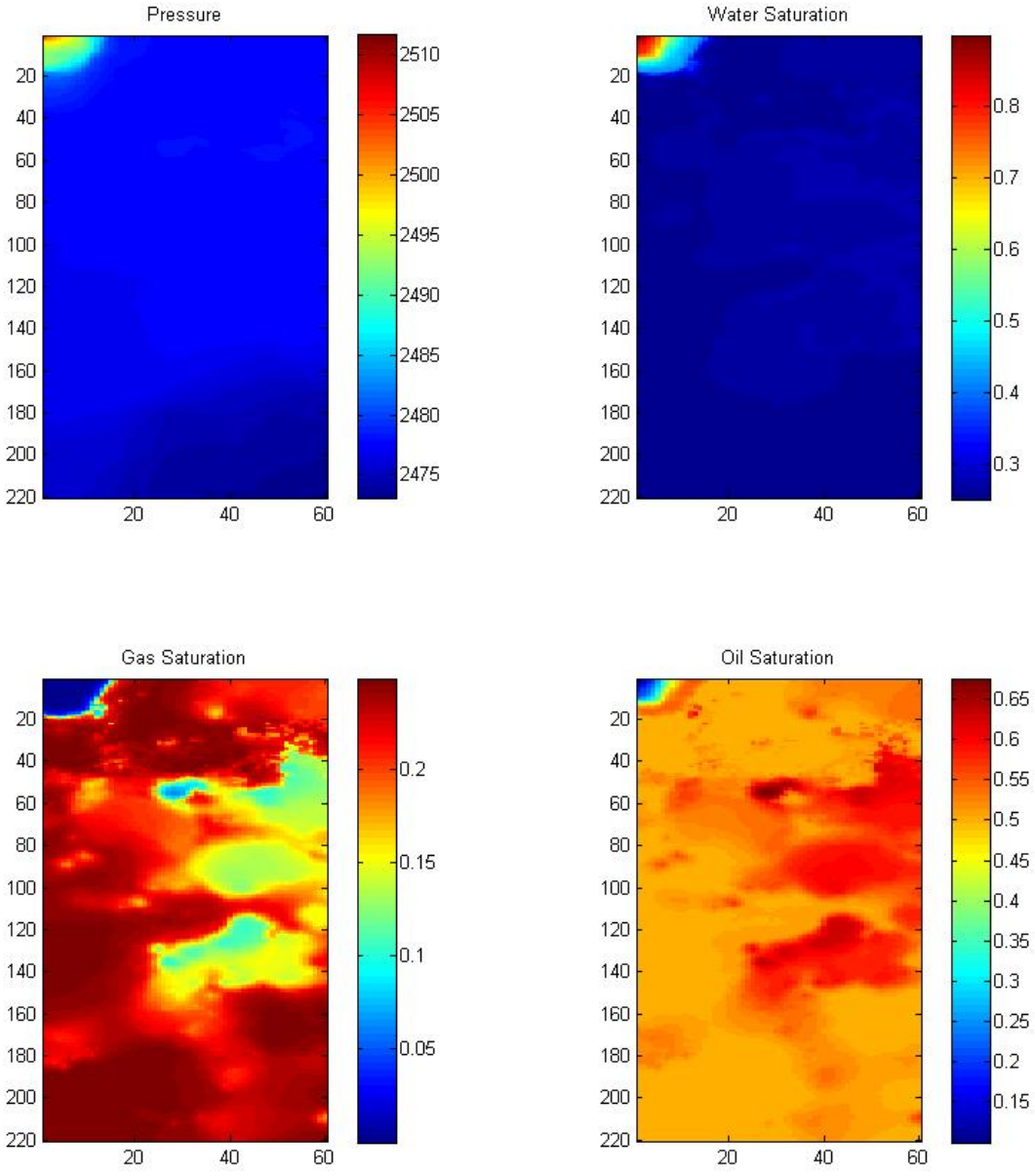

Figure 4.9: First case: the variables distribution after 1000 days.

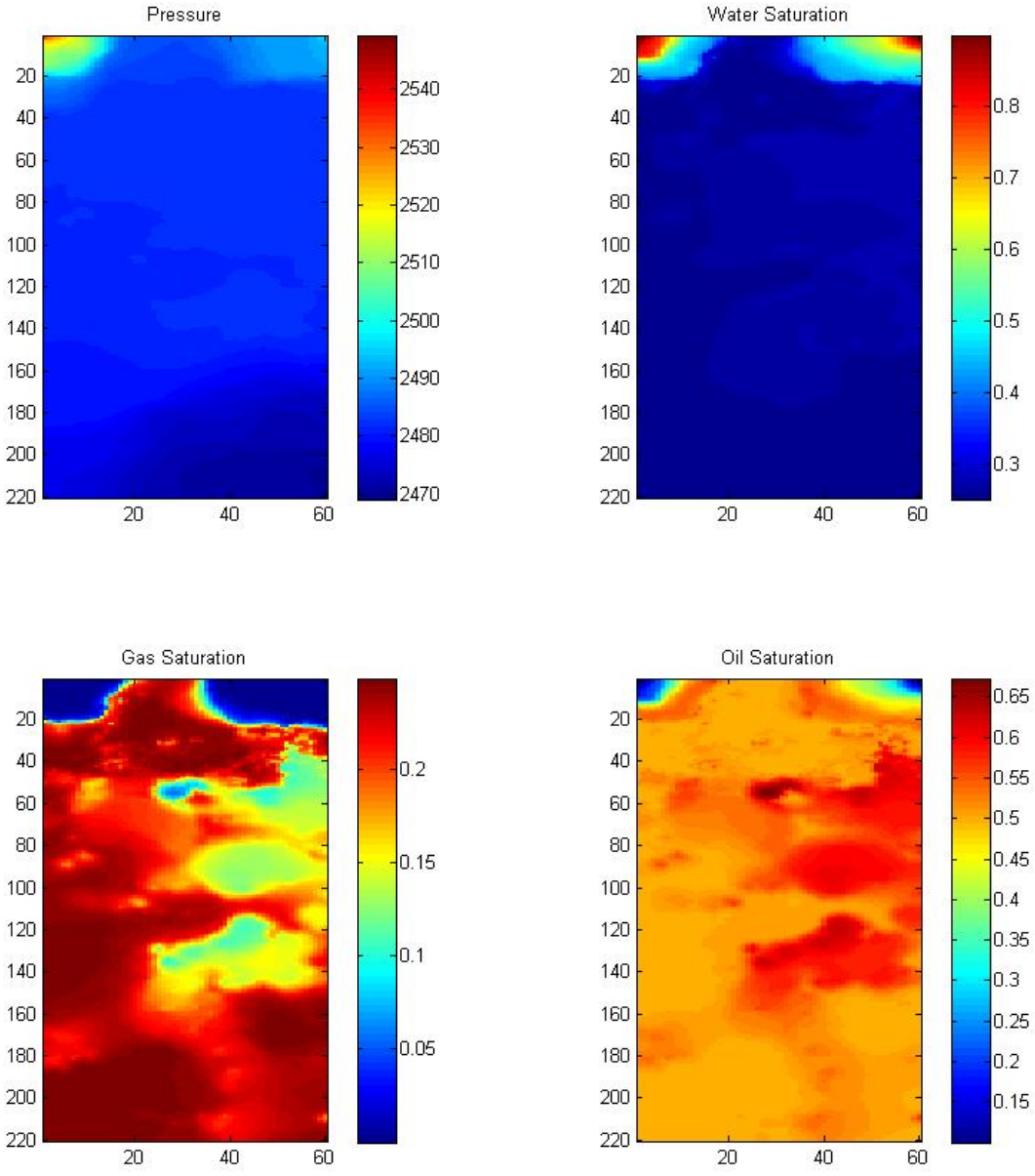

Figure 4.10: Second case: the variables distribution after 1000 days.

CHAPTER 5

DISCUSSION AND FUTURE WORK

We developed and illustrated the Modified Continuation-Newton algorithm, which is based on the predictor-corrector nature, for solving implicitly defined residual systems. The algorithm follows the solution path of a given nonlinear problem, which employs the knowledge of a uniqueness of a solution at a particular timestep. Even if the sequence of iterations is halted at some point, the algorithm will result with a solution for a smaller than a target timestep size. It is possible because each iterate of MCN is a solution for a specific timestep. The performance of the algorithm was illustrated using challenging Black Oil simulation on SPE10 geological grid, where the MCN showed a superior performance over the state-of-the-art Newton's method guided by Eclipse timestepping strategy and safeguarded by Modified Appleyard chop. Yet the algorithm can be further improved.

The MCN employs a trapezoidal rule for improving the stability of the tangent line computation. Thus, the steplength could be taken bigger as opposed to the first order explicit scheme(Euler). But, we envision a possible improvement in a stability based on conducted trial tests. Those tests motivate to focus our attention toward the local fixed point iteration of ordinary differential equation which rises at the tangent line formulation.

The implemented steplength selection algorithm is based on the allowed error variation thought the simulation run. The performance of the MCN is essentially based on the initially given parameters to the algorithm as the target timestep size and the number of iterations. Therefore, it needs to have a prepossessing routine, which will be able to test the different convergence criterion, and will propose the most applicable parameters for the specific simulation case. Such routine is in the developing stage, and it will be presented in the following works.

The other way to select the steplength is to use an error decay model(chapter 3). The derived logistic error model shows its ability in accurate describing the error decay of sub-surface flow formulations. It also proved its efficiency for 1D Buckley-Leverett, and small 3D two phase flow problems. However, in order to efficiently use any of the models for multi-scale problems, we need to find a replacement for the input variables. The possible option is to use less informative but more memory friendly measurements, such as the norm of the residual equation. Moreover, the residual based error is easier to obtain, because it needs to take only a single sample of the residual vector at each Newton iteration, which results in a scalar. Thus, the whole mechanism and the theory has to be revised to fit new measurements.

BIBLIOGRAPHY

[1] E.L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*. Springer-Verlag, Berlin Heidelberg, 1990.

[2] K. Aziz, K. Aziz, and A. Settari. *Petroleum reservoir simulation*. Applied Science Publishers, 1979.

[3] Z. Chen, G. Huan, and Y. Ma. *Computational Methods for Multiphase Flows in Porous Media*. Computational Science and Engineering. Society for Industrial and Applied Mathematics, 2006.

[4] C Den Heijer and WC Rheinboldt. On steplength algorithms for a class of continuation methods. *SIAM Journal on Numerical Analysis*, 18(5):925–948, 1981.

[5] M. S. Hussein J. C. A. Barata1. The moore-penrose pseudoinverse. a tutorial review of the theory. *Brazilian Journal of Physics*, 42(1-2):146–165, 2012.

[6] D.W. Peaceman. *Fundamentals of Numerical Reservoir Simulation*. Developments in Petroleum Science. Elsevier Science, 2000.

[7] Werner C Rheinboldt and John V Burkardt. A locally parameterized continuation process. *ACM Transactions on Mathematical Software (TOMS)*, 9(2):215–235, 1983.

[8] PH Sammon and B Rubin. Practical control of timestep selection in thermal simulation. *SPE Reservoir Engineering*, 1(02):163–170, 1986.

[9] Schlumberger. *Eclipse Version 2013.1: Technical Description*, 2013.

[10] Schlumberger Information Solution, San Felipe, Suite 100, Houston, TX 77056-2722. *Traning and Exercise Guide*, 2008.

[11] Endre Süli and D. F. (David Francis) Mayers. *An introduction to numerical analysis*. 2003.

[12] Xiaochen Wang and Hamdi A Tchelepi. Trust-region based solver for nonlinear transport in heterogeneous porous media. *Journal of Computational Physics*, 253:114–137, 2013.

[13] Rami Younis, Hamdi A Tchelepi, and Khalid Aziz. Adaptively localized continuation-newton method–nonlinear solvers that converge all the time. *SPE Journal*, 15(02):526–544, 2010.

[14] Rami Mustafa Younis. *Modern advances in software and solution algorithms for reservoir simulation*. PhD thesis, Stanford University, 2011.

APPENDIX A

RESERVOIR SIMULATOR VALIDATION

The motivation for writing this appendix is to provide the researchers and interested people with the prove that the developed reservoir simulator is compatible with commercial versions. Three phase sub-surface flow problem on 2D SPE10 geological grid is the main focus of a comparison. The Schlumberger-Eclipse is the world leading standard for sub-surface flow simulation was taken as a commercial simulator.

In order to be able to accurately verify the developed piece of software several cases were tested:

First, we need to validate the correctness of the residual and Jacobian equation for certain simulation. As a primary test the full simulation run up-to 100 days was taken. Then, the plots of the relative error for each cell and each variable (P, S_o) is presented on the couture maps (Figures A.1 - A.3). The relative error was computed by using the following formulation:

$$\varepsilon_i = \frac{|U_{ecl} - U_{c++}|_i}{|U_{ecl}|},$$

where U_{ecl} and U_{c++} are the value of variables after 100 days based on Eclipse simulation and developed code in C++ accordingly. Based on the presented Figures A.1 - A.3), the maximum relative error is not exceeding 8%.

Second, need to validate Newton's method. The simulation starts from $t = 0$, where the timestep size grows $\Delta t = 1, 2, ...$ until the target timestep size is reached. At each Δt, the number of Newton's iterations from Eclipse and C++ were recorded. From Figure A.4(b) can be observed that the difference between Eclipse and the test simulator (C++) is only +/- 1 to 2 iterations per timestep size. The total number of iteration taken by C++ code is

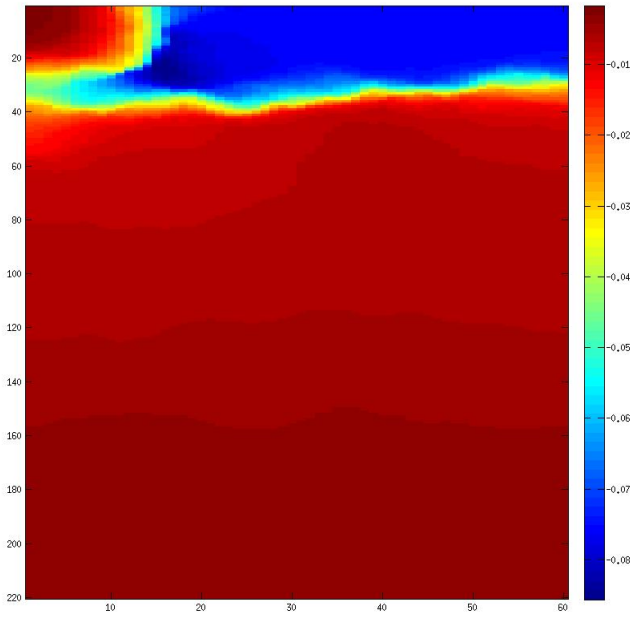

Figure A.1: The relative error for the pressure variable.

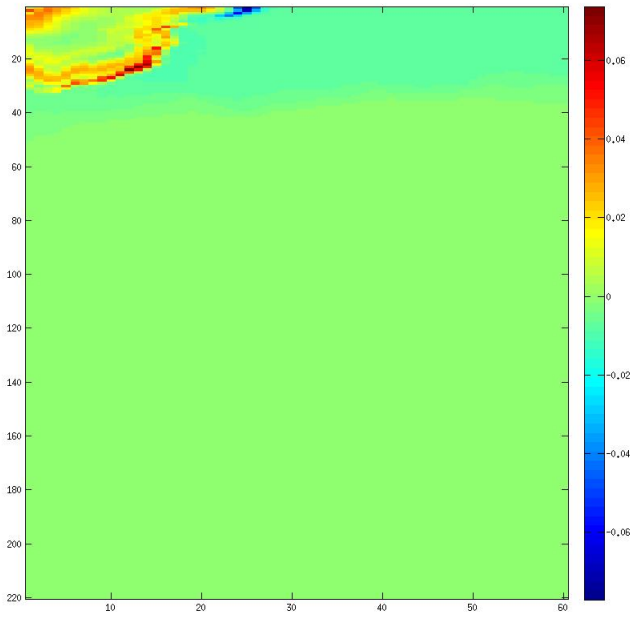

Figure A.2: The relative error for the oil saturation variable.

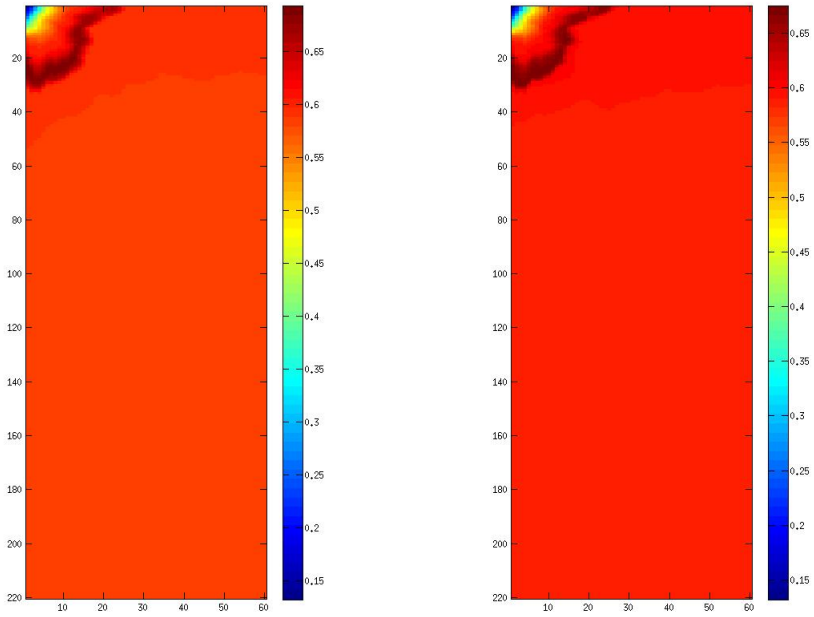

Figure A.3: An example of the oil saturation variable distribution: (a) - Eclipse, (b) - C++.

also compatible with Eclipse (Figure A.4(a)).

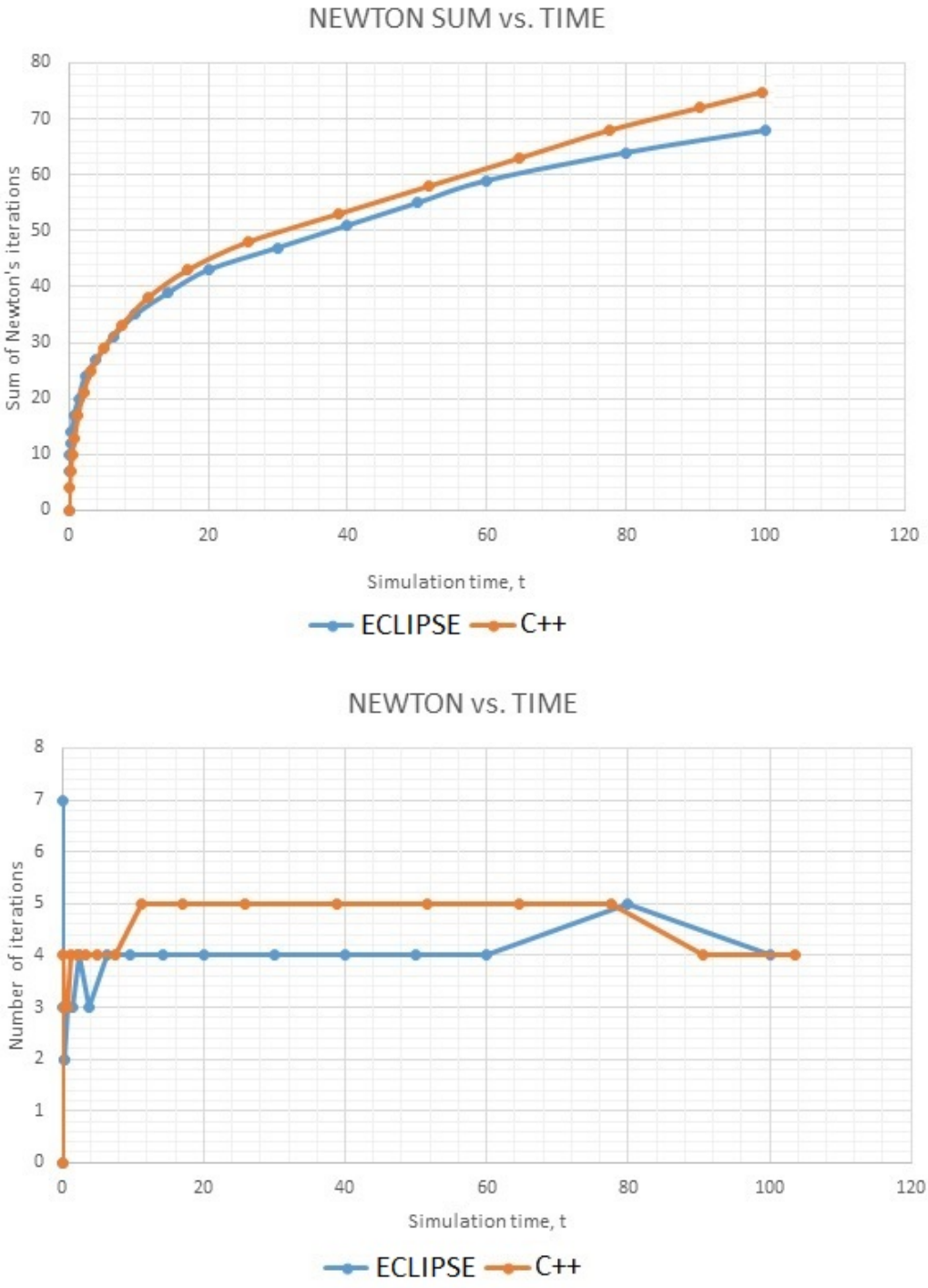

Figure A.4: Performance comparison: (a) - Total number of Newton's iteration versus simulation time, (b) - number of Newton's iterations at each timestep.