A CONTROL PERSPECTIVE TO ADAPTIVE TIME STEPPING IN RESERVOIR SIMULATION

A Thesis

by

DANY ELOM AKAKPO

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Eduardo Gildin
Committee Members,	Maria A. Barrufet
	John Killough
Head of Department,	Daniel A. Hill

August 2016

Major Subject: Petroleum Engineering

ABSTRACT

Reservoir modelling is an important tool in the management of hydrocarbon reservoirs. In fact, reservoir models are often a cost effective and time efficient alternative to a trial-and-error field management approach. Reservoir models allow oil companies to simulate various reservoir conditions and management strategies without having to spend considerable amount of money and time. Consequently, it is crucial to find ways to generate fast and accurate reservoir models to assist in making these crucial decisions. Within a reservoir simulator, the time discretization scheme is one of the most sensitive and computer intensive steps of the entire simulator. As a result, it is vital to find an efficient ways to perform this step in order to optimize the performance of the simulator. During the time discretization process, the choice of the time-step is a crucial decision. In fact, the time-step affects the computation time, the convergence, the accuracy and the amount of memory space used by the computer to run the simulation. We have to pick a time-step that is small enough to allow the solution to converge, but also sufficiently large to avoid high computation times. In order to tackle this problem, there are several adaptive time-stepping methods developed to automatically adjust the time-step and make sure that it remains within an optimal range. In this study, we investigate the effectiveness of using the Proportional-Integral-Derivative controller (PID) to regulate the error and the variations in pressure and saturation during the simulation of a reservoir system. We compare the performance of the PID controller with the basic controller conventionally used in adaptive time-stepping. The results show that PID algorithm used

to control the variations in pressure and saturation can be more efficient than the basic controller as long as the proper PID coefficients are used in the simulation. We were able to reduce the computation cost with the use of the PID controller while maintaining the same level of accuracy as the basic method. The manual tuning of the controller can be time-consuming and future would have to include automatic tuning algorithms specifically tailored for adaptive time-stepping purposes. Otherwise, the benefit associated with using the PID controller would be dwarfed by the time-consuming manual tuning process. We also tested the PID controller to regulate the error within the Newton-Raphson loop. The results showed that the use of the PID controller inside this loop results in instabilities that cause the reservoir simulator to run inefficiently.

DEDICATION

I would like to dedicate this thesis to my parents Solo Akakpo and Christiane Akakpo who both supported me during my studies. I would also like to dedicate this work to my sister Chrystel Akakpo that also helped me get through these two years of graduate school. Finally, I would like to dedicate this thesis to my best friend Ugo Okoli who supported me and helped me get through graduate school.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Gildin, and my committee members, Dr. Barrufet, and Dr. Killough, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my mother and father for their encouragement.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

Time discretization is one the most computer-intensive and error-prone stage of a reservoir simulator. During this process, the time derivatives of the underlying partial differential equation are approximated and the solution of the system is obtained for a small time increment at every iteration. It is important to choose the right time-step during this stage to ensure an optimal performance of the reservoir simulator. The time-step significantly affects the convergence rate, the stability, the accuracy, and the computation time of the reservoir simulator. As the time-step increases, the convergence rate decreases and the solution of the system becomes less accurate. At the same time, an increase in the time-step also leads to a decrease in the computational time. Consequently, the choice of the time-step becomes an optimization problem where the user has to find a happy medium between computation speed and accuracy. In an Implicit Pressure, Explicit Saturation formulation (IMPES), the computation required per iteration is low, but the time-step is limited to a small range due to stability issues [20]. In a relatively more stable fully-implicit reservoir simulator, choosing an optimal time-step becomes even more critical. In fact, the computation cost associated with a fully implicit discretization scheme is substantial and there are less restriction on the time-step due to relatively good stability; it is therefore imperative to maintain the time-step in an optimal range throughout the simulation. To that end, there are numerous algorithms designed to automatically adjust the time-step. Among these methods, the

PID has proven to effectively control the time-step and generate a smooth and robust step response in problems involving ordinary differential equations [11]. Despite these positive results, the application of the PID controller to reservoir simulators has been limited. In our study, we look more in depth at the advantages and drawbacks of using the PID controller on a two-phase flow reservoir simulator. We compare the PID controller's performance to a basic adaptive step-size method used in reservoir simulation. We perform a sensitivity analysis to determine the impact of each PID coefficient on the response of the system. We also look at the effect of permeability and well positions on the performance of the PID controller.

1.1 Literature review

Reservoir modelling has considerably evolved throughout the years from simple analytical models to geologically complex numerical models with intricate fracture networks and grid refinements. In 1942, Bruce [3] published a study describing a mathematical model used to predict the pressure changes during oil recovery. His relatively simple model assumed the oil in the reservoir to be "a homogeneous compressible liquid…contained in an infinite, horizontal, uniformly thick and homogeneous porous medium" [3]. He used a differential equation to represent the distribution of the density across the reservoir and was able to derive the analytical solution. He was also able to relate the change in density at any given point in the reservoir to the pressure change during oil recovery. This mathematical model had a lot of simplifying assumptions and was easy to implement. The simplicity of the model also

led to poor forecasting capabilities in reservoirs with more geological complexities. In order to improve the accuracy of the predictions, engineers had to design more complex models to obtain a realistic picture of the reservoirs' dynamic behavior. With this added complexity, they had to move away from simple analytical solutions and use computational methods to solve intricate reservoir systems [6]. Finites difference was introduced in the 1930s and later improved to solve complex problems as seen in reservoir modelling. In finite difference, the derivatives are approximated using Taylor series expansion and the solution is obtained in small increments [23]. In the 1960s, the black oil model was generated and it initially used finite difference to solve the reservoir system [6]. The black oil model assumed a constant hydrocarbon fluid composition in the reservoir and only distinguished the hydrocarbon as either oil or gas [18]. Black oil reservoirs were effective at predicting the behavior of reservoirs with a mixture of heavy and light hydrocarbon components. However, in reservoirs with a wide range of hydrocarbon compositions, the model broke down and the forecasting became inaccurate [18]. The compositional model was developed in the 1970s to model reservoirs with more diverse hydrocarbon compositions and phase behavior. In this model, the reservoir contains hydrocarbons of different compositions [6]. Each hydrocarbon is treated uniquely and has a distinct phase behavior model in the reservoir simulator. This model is more complex and has a higher computational cost than the black oil model. Today, we are able to model even more complex oil reservoirs with involved well management schemes, fracture networks, intricate grid designs and surface facilities [6]. As the reservoir model gets more complex, so thus the computational cost. The introduction of

parallel computing has significantly helped alleviate this cost [6]. However, the increase in complexity of reservoirs models appears to outpace the upgrades in processors and computing techniques in the oil industry. As a result, it is important to optimally run these new simulators to make sure that the computation time is minimized and the accuracy enhanced. In reservoir models, the time discretization stage is one of the most sensitive stages of the entire simulator. The computing speed and accuracy of the simulator strongly depends on the choice of the time-step during this process. A time-step that is too small can lead to large computing time, while a time-step too large can cause instabilities in the reservoir simulator. With the proper adaptive time-stepping method, we can achieve the best performance and allow practical run time of complex simulators in field environments.

There are various time-stepping strategies used in computational methods depending on the type of problem and the user preferences. The use of adaptive time-stepping schemes has proven to be much more efficient than the constant time-step method. In 2010, Bender et al [2] compared an adaptive time-stepping method with a constant time-step scheme on an incompressible fluid flow problem. They were able to show that adaptive time-stepping was much more efficient than the constant time-step scheme. They obtained a lower computation time with the same accuracy in the solution. One of the most popular adaptive time-stepping controllers used to control the error is the basic controller described by Soderlind [22] using the equation below:

$$h_{n+1} = \left(\frac{\epsilon}{r_n}\right)^{1/k} h_n$$

where h_{n+1} is the time-step at iteration $n+1$, h_n is the time-step at iteration n, ϵ is the user-defined tolerance and r_n is the local error estimate at iteration n and k is a constant that depends on the method used to solve the system. If the error r_n is greater than the tolerance ϵ, the ratio $\left(\frac{\epsilon}{r_n}\right)^{1/k}$ becomes less than 1 and so we get a reduction in the time-step for the next iteration ($h_{n+1} < h_n$). Similarly if the local error is less than the tolerance, we get an increase in the time-step at the next iteration. In reservoir engineering, the use of adaptive time-stepping method dates back to the 80's [14]. In 1980, Jensen [14] presented an automatic time-step selection method that he applied to a finite difference steam injection simulator. In his study, he described various schemes that would be applicable to reservoir simulations and compared them to an algorithm that he developed based on a first order predictor-corrector scheme represented on Figure 1. In this algorithm, the goal is to control the maximum change in the solution $max|dc_i|$ by automatically adjusting the time-step throughout the simulation. This maximum change is used as an estimate of the error in the solution. β, ε and α are user-defined and can be adjusted based on the user's preferences. The algorithm uses $\|d_t\|$, the absolute value of the maximum element of the vector dc_i , and compares this value to the tolerance range defined by the user. If the criteria of the user is not met, the current step is rejected and the time-step is divided by two for the next iteration. If the criteria is met, the current step is accepted and the time-step used in the next iteration depends on the value of $\|d_t\|$. In that case, two possible formulas can be used to compute the time-step of the next iteration as described in Figure 1.

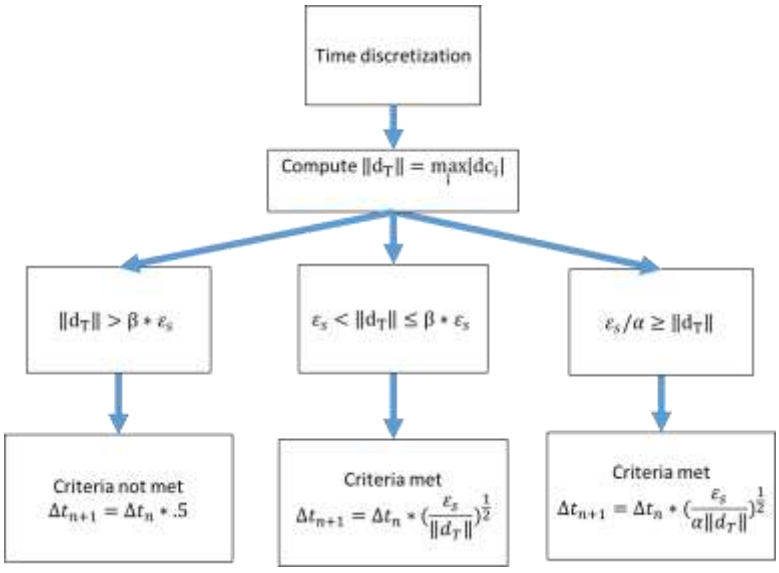

Figure 1: Jensen adaptive time-step algorithm. This time-stepping method is used to control the maximum change in the solution $\|d_t\|$ [14]

Jensen noted that the time-response using this algorithm was easier to implement and resulted in a lower computation time, and a smoother time-step response. In adaptive time-stepping, The PID has also proven to enhance the performance of computer simulations as described by Soderlind [11].

Engineers have relied on PID controllers for almost half a century to regulate a wide variety of industrial processes [25]. Throughout the years, the PID controller has proven to be a stable and reliable control system method. It is a relatively simple and robust control technique that has a broad range of application in engineering. The use of PID controllers to regulate step-sizes in computer simulations is more recent. In 1988, Soderlind [11] used a Proportional-Integral (PI) controller to regulate the time-step of an ordinary differential equation. Soderlind was able to obtain a smooth and robust time-

step response with the PI controller. He tested the PI controller with various tolerances and recorded the local error estimate response, the total number of time-steps used, and the dynamic time-step response of the PI controller. These tests were performed on a variety of differential equations problems that were solved using an explicit Runge-Kutta method. The results showed that the PI controller was faster than the old method while maintaining the same accuracy. Soderlind's study highlighted the potential of the PI controller as a feedback loop in time discretization. It helped pave the way for the use of the Proportional-Integral-Derivative (PID) controllers in other types of problems including fully implicit time discretization simulators. In 2002 Valli et al [24]. used the PID controller as an adaptive time-step method for a 2D viscous flow and heat transfer problem. They used finite element to build the simulator and compare the PID controller to another adaptive time-stepping strategy. The PID controller was used to control both the error and the convergence rate. Valli et al [24]. obtained a smaller number of steps with the PID controller without much loss in accuracy. They achieved a smooth and robust time-step response similar to Soderlind's results.

The use of the PID controller as an adaptive time-step method is not wide spread in reservoir modelling and it effectiveness at controlling reservoir time discretization is not well understood. Consequently, our study aims at correcting this trend and focuses on applying this method to reservoir simulators. This study provides a unique and novel insight into the effectiveness of a PID controller for time-step control. It allows us to better understand the advantages and drawbacks of using such a controller endowed with rigorous error criteria on a fully implicit simulator.

1.2 Objective of the study

In this paper, we assess the efficacy of a PID (proportional-integral-derivative) controller at regulating the time-step used during the time discretization of a reservoir simulator. We formulate the adaptive time-stepping in a control system framework and aim at achieving the following objectives:

1. Assess the efficiency of the PID controller using various metrics (computing time, total number of time steps, number of rejected steps) and compare it to adaptive time-stepping techniques currently used in the industry.
2. Perform a sensitivity analysis to determine the effect of varying PID controller parameters, and how these changes affect the response of the PID feedback loop.
3. Evaluate the robustness of the PID controller by testing it using various types of reservoir simulators.

1.3 Thesis organization

In chapter two, we discuss the fundamentals behind the two phase flow reservoir simulator that we use in this study. We explain the implicit time discretization process and how the Newton-Raphson scheme is used to converge to the true solution. In chapter three, we discuss the fundamental of the PID controller and we go over the typical response of the controller and the effect of each coefficient. We formulate reservoir time-stepping in a control system framework and apply PID control techniques for

adaptive time stepping. We also derive the characteristic equation of the discrete PID used in our study. The PID controller is represented by a system that outputs the time-step of the next iteration using local error estimates, pressure and saturation changes obtained from the previous time discretization. In chapter four, we talk about the results obtained with the PID controller and compare them to the ones obtained using the basic controller. We compare the PID control strategy with other adaptive heuristics methods using the computation time, total number of time steps, number of rejected steps as metrics. We also evaluate the robustness of the PID controller gains, that is, proportional, integral and derivative gains, by varying the initial condition of the reservoir simulator and recording the performance. Chapter five focuses on the conclusion that we reach after analyzing the results and discuss future work.

CHAPTER II

RESERVOIR SIMULATION

Computer simulators are used to represent oil reservoirs for the purpose of predicting flow rate, pressure, saturation among other variables over the life cycle of the reservoir. In our study, we focus on a two-phase flow reservoir simulator containing both oil and water. In this type of reservoir simulator, two sorts of solutions are needed to completely characterize the system: pressure and saturation. In our study, these solutions are obtained by applying finite difference to the fundamental differential equations of the oil reservoir. In finite difference, the derivative of a differential equation is approximated using Taylor's series expansion as described by Li [16]:

$$\frac{\partial u}{\partial t} = \frac{{u_i}^n - {u_i}^{n-1}}{\Delta_t} + R_t \; Backward \; difference \quad [2.1]$$

$$\frac{\partial u}{\partial t} = \frac{{u_i}^{n+1} - {u_i}^n}{\Delta_t} + R_t \; Forward \; difference \quad [2.2]$$

$$\frac{\partial u}{\partial t} = \frac{{u_i}^{n+1} - {u_i}^{n-1}}{2\Delta_t} + R_t \; Central \; difference \quad [2.3]$$

Where u represents the time dependent state variable of the system, Δ_t is the time-step used in the approximation, and the subscripts n represent the time iteration during the reservoir simulation. R_t represents the residual error created with the approximation. There are three types of finite differences methods (FDM): backward, forward and central. The backward and forward differences are first-order approximations while the central difference is a second order estimate. In our simulator, the time and space

derivative approximations are obtained using a combination of first and second order FDM. The error in the approximation depends on the size of the time-step used to generate the estimate. As the time-step of the FDM gets larger, the magnitude of the error increases. Thus, it is important to properly manage the value of the time-step throughout the simulation to keep the error in check. The control of the time-step depends on a lot of factors including the type of reservoir modelled. In our case, we simulate a two-phase flow reservoir with some simplifying assumptions discussed in the next section.

2.1 Two-phase flow simulator

The general governing equation of a two-phase flow described by Aziz et al [1]:

$$\frac{\partial}{\partial t}(\emptyset \rho_l S_l) = \nabla.\left[\frac{\rho_l}{\mu_l} k_{rl} K(\nabla P_l - \rho_l g \nabla Z)\right] - \widetilde{m_l} \quad [2.4]$$

In our study, we assume the reservoir fluids immiscible, the gravitational and capillary forces negligible. We also assume water incompressible and oil only slightly compressible. Permeability varies in the horizontal plane and is assumed to be constant along the vertical axis. We mainly use the forward and central difference to approximate the space and time derivatives.

A system of four equations and four unknowns defines the reservoir system as described in equations 2.5 to 2.8. Equation 2.7 and 2.8 represent the partial differential equation of oil and water respectively. The left-hand side of these equations is composed of space variables while the right-hand side contains the time variables. In equation 2.9

and [2.10], we apply forward difference to the time derivatives to obtain a first-order approximation used in our simulator.

$$\frac{\partial}{\partial x}\left[K_o\frac{h_o}{B_o}\left(\frac{\partial p_o}{\partial x}-p_o\frac{\partial z}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left[K_o\frac{h_o}{B_o}\left(\frac{\partial p_o}{\partial y}-p_o\frac{\partial z}{\partial y}\right)\right]=\frac{\partial}{\partial t}\left(\emptyset\frac{S_o}{B_o}\right)+\widetilde{m_o} \qquad [2.5]$$

$$\frac{\partial}{\partial x}\left[K_w\frac{h_w}{B_w}\left(\frac{\partial p_w}{\partial x}-p_w\frac{\partial z}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left[K_o\frac{h_w}{B_w}\left(\frac{\partial p_w}{\partial y}-p_w\frac{\partial z}{\partial y}\right)\right]=\frac{\partial}{\partial t}\left(\emptyset\frac{S_w}{B_w}\right)+\widetilde{m_w} \qquad [2.6]$$

$$\begin{cases} S_o+S_w=1 & [2.7] \\ p_c(S_w)=p_o-p_w & [2.8] \end{cases}$$

The time derivatives approximations are represented by the equations below

$$\frac{\partial}{\partial t}\left(\emptyset\frac{S_o}{B_o}\right)=\frac{1}{\Delta t}\left[\left(\emptyset\frac{S_o}{B_o}\right)^{n+1}-\left(\emptyset\frac{S_o}{B_o}\right)^{n}\right] \qquad [2.7]$$

$$\frac{\partial}{\partial t}\left(\emptyset\frac{S_w}{B_w}\right)=\frac{1}{\Delta t}\left[\left(\emptyset\frac{S_w}{B_w}\right)^{n+1}-\left(\emptyset\frac{S_w}{B_w}\right)^{n}\right] \qquad [2.8]$$

$$\frac{\partial}{\partial t}\left(\emptyset\frac{S_o}{B_o}\right)\approx-(\emptyset b_o)^{n+1}\frac{Sw^{n+1}-Sw^{n}}{\Delta t}+(1-Sw)^{n}[b_o^{n+1}\emptyset'+\emptyset^{n}b_o']\frac{P_o^{n+1}-P_o^{n}}{\Delta t} \qquad [2.9]$$

$$\frac{\partial}{\partial t}\left(\emptyset\frac{S_w}{B_w}\right)\approx[(\emptyset b_w)^{n+1}-S_w^{n}b^{n}b_w'p_c']\frac{Sw^{n+1}-Sw^{n}}{\Delta t}$$

$$+[S_w'b_w^{n+1}\emptyset'+S_w'\emptyset^{n}b_o']\frac{P_o^{n+1}-P_o^{n}}{\Delta t} \qquad [2.10]$$

The time and space derivative approximations are replaced back into the differential equations. The system of equations generated from all the nodes of the reservoir can be represented in matrix form as described in equation 2.11. The size of the matrices in the reservoir system depend on the total number of nodes that it contains.

$$TX=B\frac{dX}{dt}+Q+G \qquad [2.11]$$

The matrix $X = \begin{bmatrix} Po \\ Sw \end{bmatrix}$ represents the state vector and contains oil pressure and water saturation of each node of the reservoir system. T is the transmissibility matrix and contains coefficients related to the fluid and formation properties. The accumulation matrix D stores the coefficient associated with the derivative of the pressure with respect to time. The matrix G contains the gravitational parameters used to factor in the effect of gravity on the system. Q is the source/sink vector and is used to model the presence of a production or an injection well. These matrices are populated using variables that are in some cases pressure-dependents. In fact, viscosity, formation volume factor, porosity and density are variables that depend on the pore pressure. In our simulator, we use an exponential regression to characterize this dependency. The general form of the exponential function is:

$$F = F_0 e^{a(p - p_{ref})} \qquad [2.12]$$

Where F is the value of the pressure-dependent variable at a pressure p, p_{ref} is the reference pressure and a is a constant. After every iteration, the pressure-dependent variables are updated using equation 2.12. These variables are then used to populate the simulator matrices. During each iteration, once the transmissibility, accumulation, gravity and sink are constructed, the system is solved for the next time increment. This stage of the simulator is one the most sensitive of the entire simulation. In the next section, we discuss this process in more details.

2.2 Time discretization

We use an implicit scheme to obtain the time-dependent solution of the system. The solution at time $n+1$ is guessed and the Newton-Raphson method is used to converge toward the true solution of the system. Equation 2.13 to 2.15 describe the derivation of the residual function used to perform the Newton-Raphson scheme.

$$T^{n+1}X^{n+1} = B^{n+1}\left(\frac{X^{n+1}-X^{n}}{\Delta t}\right) + G + Q \quad [2.13]$$

$$T^{n+1} \approx T^{n} + \vartheta(\nabla t)T^{n}X^{n+1} = B^{n}\left(\frac{X^{n+1}-X^{n}}{\Delta t}\right) + G + Q \quad [2.14]$$

$$\left(T^{n+1} - \frac{1}{\Delta t}B^{n+1}\right)X^{n+1} + \frac{1}{\Delta t}B^{n}X^{n} - G - Q = 0 \quad [2.15]$$

$$F(X^{n+1}) = 0$$

Since our reservoir simulator is in matrix form, we can rewrite the Newton-Raphson scheme in the matrix form:

$$X_{p+1}^{n+1} = X_{p}^{n+1} - J^{-1}\left(X_{p}^{n+1}\right)F\left(X_{p}^{n+1}\right) \quad where\ p \le p_{max} \quad [2.16]$$

Where p represents the Newton-Raphson iteration and n represent the time iteration. J is the Jacobian and F is the residual function. We have also incorporated a maximum number of Newton-Raphson iteration p_{max} in our algorithm to promptly identify divergence and take corrective action. Whenever the maximum number of iterations p_{max} is reached, the time-step is reduced by half and the Newton-Raphson scheme is repeated until the solution converges within the tolerance. We can also observe that the equation 2.16 is dependent on the time-step Δt, which can change from iteration to iteration.

In the Newton-Raphson scheme, the determination of the Jacobian $J(X_p^{n+1})$ can be error-prone and computer-intensive. In our simulator, we approximate the Jacobian using an analytical solution to help minimize the computation cost. In order to solve the Newton-Raphson equation, we use a linear factorization method. This method is more time efficient and accurate than performing a matrix inversion [9].

In our study, we record the total number of Newton-Raphson iterations and the total number of time-steps counted during the simulation. We also use the ratio of the total number of Newton-Raphson iterations to the total number of time-steps:

$$NR_{ratio} = \frac{Total\ number\ of\ NR\ iterations}{Total\ number\ of\ time-steps}$$

This ratio gives an estimate of how efficient the Newton-Raphson scheme is. If the ratio is high, it means that the average convergence rate of the reservoir simulator is low and vice-versa. It is one of the criteria used to evaluate the performance of the PID controller and compare the adaptive time-stepping methods.

The efficiency of the Newton-Raphson method strongly depends on the first guess and the size of the time-step. The stability of the Newton-Raphson scheme increases with decrease in the time-step. As the time-step is reduced, the Newton-Raphson method becomes more likely to converge to a solution. At the same time, reducing the time-step can also lead to an increase in the total number of Newton-Raphson iterations required to obtain the solution. As a result, the choice of the time-step for the N.R scheme becomes a trade-off between stability and total number of iteration.

2.3 Time-stepping

As we see on equation 2.14, the residual function used in the N.R iteration is dependent on the time-step Δt. An increase in the time-step leads to lower convergence rate and higher error [13, 12]. As a result, it is crucial the optimally choose the time-step in order to obtain a solution within the user-defined tolerance. In our study, we adjust the time-step in order to control the change in pressure and change in saturation per iteration. As the time step increase, so thus the change in pressure and saturation. In order to obtain a solution that captures all the critical changes in pressure and saturation, the time-step is reduced whenever there are high pressure and saturation changes. Similarly, the resolution controller increases the time-step whenever there are low changes in pressure and saturation in order to reduce the computational cost of the simulation. In chapter 3, we discuss in more details the resolution algorithm and the PID controller used in our study and we compare it to a basic controller.

2.4 Linear solver

The linear factorization method used in our simulator is the Matlab direct solver function. The algorithm behind this function depends on the nature of the input matrices. The general form of a system of linear equation solved by the backlash function can be written in the form:

$$Ax = b$$

In our simulator, A represents the Jacobian $J\left(X_p^{n+1}\right)$, x represents the solution at the time iteration $n+1$: X_{p+1}^{n+1}, and b is $J\left(X_p^{n+1}\right) X_p^{n+1} - F\left(X_p^{n+1}\right)$. In this thesis, we use the

Matlab direct solver based on the Gaussian elimination with partial pivoting to solve the system of equations [17]. While the backlash function is less error-prone than the matrix inversion approach, there are nonetheless error accrued during this process. These errors are mainly due to the rounding of coefficients. The rounding errors can be estimated using the residual function defined by:

$$R\_LU = J\left(X_p^{n+1}\right) X_p^{n+1} - F\left(X_p^{n+1}\right) - J\left(X_p^{n+1}\right) X_{p+1}^{n+1\prime}$$

Where X_{p+1}^{n+1}'is the solution obtained using the LU factorization. In our simulator, we noted that the residual error of the LU factorization is low compare to the residual error generated from the Newton-Raphson scheme itself. As a result, we did not set a tolerance for the factorization residual and assumed the error to be negligible for the purposes of this simulator.

CHAPTER III

CONTROL-ORIENTED TIME-STEPPING

In this chapter we present the control algorithms used in our simulation and the derivation of the PID controller characteristic equation. We also present a general introduction to control system theory and how it is applied to adaptive time-stepping. In this chapter, we also derive the basic controller for the change in pressure and saturation used as a performance benchmark for the PID controller.

There have been several adaptive time-stepping methods developed throughout the years in numerical methods [3]. The choice of the method depends on the type of simulator used and user preferences. For error control algorithms, the relationship between the error and the time-step is also important to determine the appropriate adaptive time-step technique to use. For small time-steps and explicit Runge-Kutta methods, the relationship between the time-step and the error can be represented by equation 3.1 [10]

$$r_n = \alpha h_n^k \quad [3.1]$$

where r_n represents the local error estimate, α is the norm of the error function. k represents the order of the method used to solve the differential equation, h represents the time-step and n represents the time discretization iteration. Let's denote ε the error tolerance defined by the user in the reservoir simulator. If we want the local error at the next iteration $n + 1$ to be equal to the tolerance ε, we can replace r_n by ε in equation 3.1 and we get:

$$\varepsilon = \alpha h_{n+1}^{k} \quad [3.2]$$

By writing this equation, we are assuming that the norm of the error function does not change considerably from one iteration to the other. This assumption is generally reasonable, but it is important to note that it is not always the case [10]. We combine equation 3.1 and 3.2 to get:

$$\frac{\varepsilon}{r_n} = \frac{\alpha h_{n+1}^{k}}{\alpha h_n^{k}} \qquad [3.3]$$

Simplifying equation 3.3, we get the basic equation for explicit Runge-Kutta methods described by Gustafson [11]:

$$h_{n+1} = \left(\frac{\epsilon}{r_n}\right)^{1/k} h_n \qquad [3.4]$$

This equation is the most basic control system algorithm and is designed to bring the local error within the tolerance in one iteration.

3.1 Basic controller

We can also derive a similar controller for the change in pressure and saturation of a fully-implicit reservoir simulation. In fact, for small time-steps, based on a first order Taylor approximation [4], we can assume the relation between the change in pressure and time-step as:

$$h_n = a * \Delta P_n$$

where a is norm that depends on the time derivative, h_n is the time-step used at iteration n, and ΔP is the change in pressure that occurs at iteration n. In order to obtain the change in pressure to the desired value $\Delta P_{desired}$ within one iteration, we need:

$$h_{n+1} = a * \Delta P_{desired}$$

Dividing the equation above with the previous one, and assuming that the norm a does not change significantly from one equation to the other, we get:

$$\frac{h_{n+1}}{h_n} = \frac{a * \Delta P_{desired}}{a * \Delta P_n}$$

$$h_{n+1} = \left(\frac{\Delta P_{desired}}{\Delta P_n}\right) * h_n$$

We can obtain a similar equation for the time-step based on the change in saturation:

$$h_{n+1} = \left(\frac{\Delta S_{desired}}{\Delta S_n}\right) * h_n$$

where $\Delta S_{desired}$ is the maximum change in saturation desired per iteration,ΔS_n is the change in saturation at iteration n.This controller is referred throughout this research as the basic algorithm and is used to evaluate the performance of the PID resolution controller. The time-step obtained using the equations above is inputted in the residual function derived in chapter 2 (Equation 2.15).

3.2 Feedback control system

The general block diagram of a control feedback loop can be represented using Figure 2. The reference input represents the target value of the controlled variable defined by the user. The goal of the feedback control system is to make sure that the controlled variable

remains close to the reference input throughout the operation of the plant. The plant can represent any engineering process that needs to be controlled automatically. The controller block symbolizes the algorithm chose to adjust the manipulated variable. This variable is an input to the plant and can influence the controlled variable.

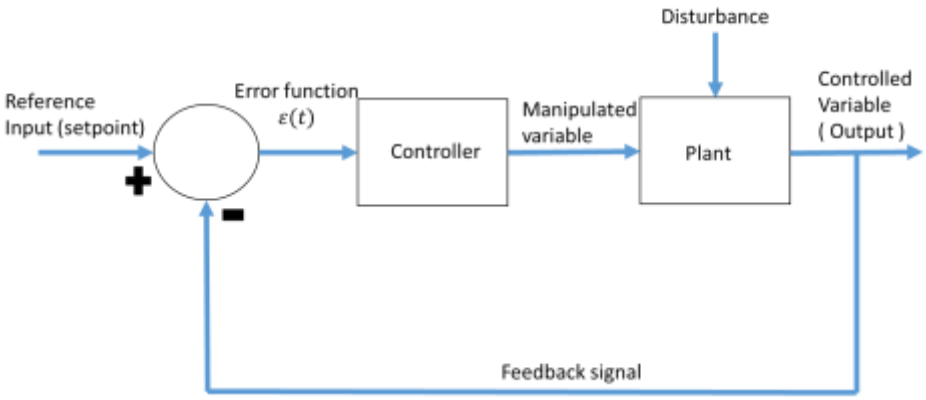

Figure 2: Block diagram of feedback control loop

Ideally, the manipulated variable would be automatically adjusted so that the controlled variable equals to the reference value. However, this never happens in practice. Instead, the controlled variable is maintained within a certain range of the reference value. The controlled variable is also affected by disturbances caused by external forces to the plant system.

Each block of the feedback system block diagram can be represented using a transfer function. The transfer function is a representation of the governing differential equation of the block. We can obtain the output of a block by multiplying the input

signal by the transfer function of the block. For continuous functions, Laplace transforms are used to represent the transfer function of a block in control system. In Figure 3, we illustrate the use of transfer functions in control system:

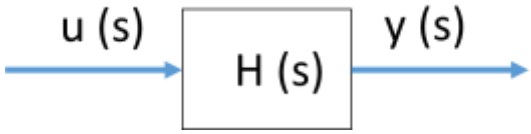

Figure 3: Transfer function of control block

In this example, the output signal $y(s)$ can be obtained using the transfer function of the block $H(s)$ and the input signal $u(s)$ by using the equation:

$$y(s) = H(s) * u(s) \quad [3.5]$$

The transfer function allows us to solve differential equations by performing simple algebraic operations.

3.3 PID controller

The PID controller has three subcomponents in the control system block: the proportional, the integral, and the derivative components. The effect of all three component is added to create the overall response of the PID controller as illustrated in Figure 4. These components operate independently of each other and have a unique effect on the overall response of the PID controller. The proportional term helps the system respond to the latest feedback data coming from the plant.

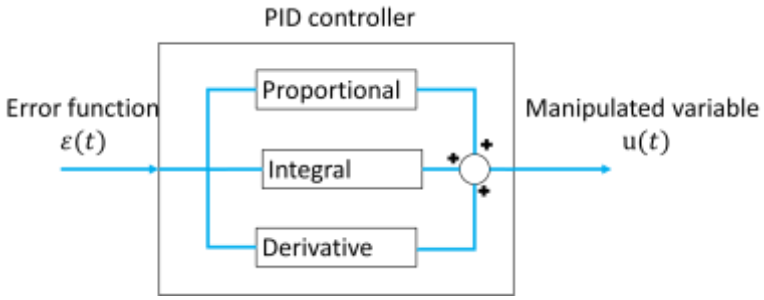

Figure 4: PID controller block

It is the instantaneous response component of the PID controller. In the event of a sudden change in the output of the plant, the response of the control system will first come from the proportional component. The integral component is affected by the cumulative trend of the plant's output. In other words, the integral uses data from consecutive iterations to reduce the gap at steady-state between the controlled variable and the reference input. The derivative component is used to anticipate future plant output and preventively take corrective action. The derivative component is sensitive to the rate of change of the plant's output.

In time-domain representation, the PID controller governing equation can be written in the form [15]:

$$u(t) = K_p * \varepsilon(t) + K_i * \int \varepsilon(t) + K_d * \frac{d\varepsilon}{dt}$$

The continuous transfer function of a PID controller obtained from Crowe et Al [15] is represented using Laplace transform with the equation:

$$G_{PID}(s) = \frac{K_i + K_p s + K_d s^2}{s^2} \qquad [3.6]$$

K_i is the integral coefficient, K_p the proportional coefficient and K_d is the derivative coefficient. The response of the controlled variable depends on these coefficients. In Figure 5, we present a typical PID controller response. The value of the controlled variable is plotted throughout the process. As seen in Figure 5, there is an initial rise of the controlled variable to the reference value, but once the reference value is exceeded, the controller adjusts the manipulated parameter in order to reduce the controlled variable. As the controlled variable drops below the reference point, the manipulated variable is again adjusted leading to an increase in the controlled variable. This process is repeated until the system reaches steady-state. It is important to note that this controlled response does not always happen. In some cases, the controlled variable never overshoots the target but asymptotically approaches the target value until it reaches steady-state. The system response strongly depends on the PID coefficients. During the tuning process, the user can adjust the PID controller parameters to determine the optimal coefficients that generate the desired system response. There are tuning algorithms designed to determine the PID coefficients based on simulator parameters [19]. In our study, we perform the tuning of the PID manually because most of the tuning algorithms are not designed for adaptive time-stepping. As we see from Figure 6, increasing the proportional and the integral coefficients leads to a shorter rise time, but at the same time it leads to higher overshoot. The derivative is not represented in Figure 6, but has a unique effect on the system response. In fact, increasing the derivative coefficient leads to a decrease in the overshoot and the settling time [19].

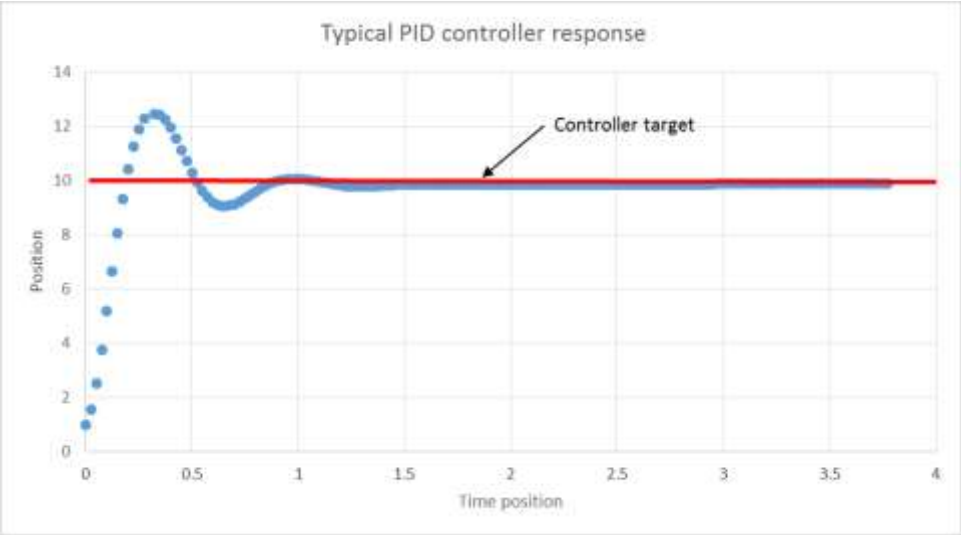

Figure 5: Typical PID controller response

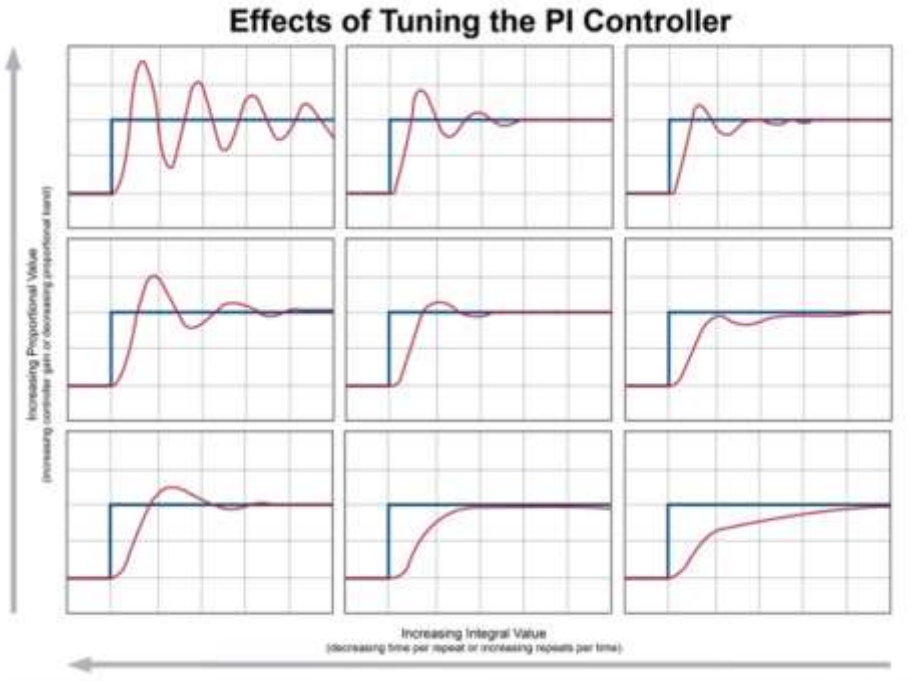

Figure 6: Effect of K_p and K_i on the system response [7]

The general schematic of the PID controller used in our simulator can be represented using Figure 7. In our study, the controller represents the PID algorithm and the plant is the reservoir simulator. The PID controller uses output data from the simulator-plant to generate a new time-step. This new time-step is then sent to the reservoir simulator where it is used to solve the system for the next time increment.

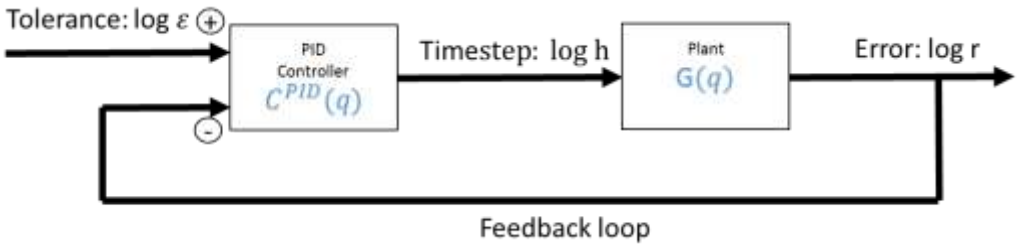

Figure 7: Block diagram of time-step PID controller [10]

The simulator-plant outputs the solution of the differential equation along with error estimates and changes in state variables. This data is fed back to the controller and the process is repeated until the final solution is reached.

In order to obtain the transfer function of the PID controller in the time domain, we introduce the backward shift operator q^{-1} defined by [5] as:

$$q^{-1}u(n) = u(n-1)$$

The difference between two consecutive iterations can be written using the backward shift operator:

$$u(n) - u(n-1) = u(n) - q^{-1}u(n)$$
$$= (1 - q^{-1})u(n)$$

$$= \frac{q-1}{q} u(n)$$

$$= \nabla f$$

$\nabla = \frac{q-1}{q}$ is the backward difference operator. It is the equivalent of the derivative operator for a continuous function. Similarly $\frac{q}{q-1}$ is the summation operator and is the equivalent of the integral operator in the continuous function [2]. Using the difference operator and the summation operation, we can obtain the transfer function of the PID controller in the time domain:

$$\frac{u_n}{e_n} = (K_p + \frac{q-1}{q} * K_d + \frac{q}{q-1} * K_i)$$

Where u_n is the output function of the PID controller and e_n the input error function, both represented in the time domain. From Figure 7, we can replace the input function of our PID controller by the equation:

$$e_n = log\varepsilon - logr_{n-1}$$

Using the backshift operator, we can write:

$$e_n = q^{-1}(log\varepsilon - logr_n)$$

Similarly from Figure 7, we can replace the output function of our PID controller by the equation:

$$u_n = logh_n$$

As a result, we can write:

$$logh_n = q^{-1}\left(K_p + \frac{q-1}{q} * K_d + \frac{q}{q-1} * K_i\right) * (log\varepsilon - logr_n) \quad [3.7]$$

where h represents the time-step, ϵ is the tolerance and r is the local error estimate. We can now identify the transfer function of our adaptive time-stepping PID controller as [21]:

$$C^{PID}(q) = q^{-1}\left(K_i \frac{q}{q-1} + K_p + K_d \frac{q-1}{q}\right)$$

$$\log h = C^{PID}(q)(log\epsilon - logr) \quad [3.8]$$

Substituting the backward shift operator by: $\nabla = \frac{q-1}{q}$ and taking the difference of equation 3.7, we get [21]:

$$\Delta logh = \left(K_I + K_p\nabla + K_D\nabla^2\right).(log\varepsilon - logr)$$

The equation above is equivalent to the recursion [21]:

$$\begin{aligned} &logh_{n+1} - logh_n \\ &\qquad = K_I(log\varepsilon - logr_n) - K_P(logr_n - logr_{n-1}) \\ &\qquad - K_D(logr_n - 2logr_{n-1} + logr_{n-2}) \end{aligned}$$

$$log\frac{h_{n+1}}{h_n} = K_I log\frac{\varepsilon}{r_n} - K_P\left(log\frac{r_n}{r_{n-1}}\right) - K_D\left(log\frac{r_n * r_{n-2}}{{r_{n-1}}^2}\right)$$

Taking the exponential of both sides, we get [21]:

$$\frac{h_{n+1}}{h_n} = \left(\frac{\varepsilon}{r_n}\right)^{K_I} * \left(\frac{r_{n-1}}{r_n}\right)^{K_P} * \left(\frac{{r_{n-1}}^2}{r_n * r_{n-2}}\right)^{K_D} \quad [3.9]$$

From equation 3.9, we can obtain the recurrence relation between the time-step and the local error estimate as described below:

$$h_{n+1} = (\frac{\varepsilon}{r_n})^{K_i} * \left(\frac{r_{n-1}}{r_n}\right)^{K_p} * \left(\frac{r_{n-1}^2}{r_n r_{n-2}}\right)^{K_d} * h_n \qquad [3.10]$$

In this study, we investigate two independent PID control systems algorithms. We test a first PID controller to regulate the local error and a second one to control the resolution of the state variables. The control of the resolution is used to create small time-steps at time intervals with high rate of change in the state variables. Similarly, the algorithm generates large time-step for time intervals with small rate of changes.

3.4 Error control algorithm

For a two-phase flow reservoir simulator, there are two solutions: the pressure and the saturation. They each have a distinct error estimate; the PID uses the highest value of the two error estimates to compute the new time-step [24].

$$e_n = \max(e_p, e_S) \qquad [3.11]$$

$$\text{where } e_p = \frac{\|P^n - P^{n-1}\|}{\|P^n\| tol_P} \; and \; e_S = \frac{\|S^n - S^{n-1}\|}{\|S^n\| tol_S}$$

Substituting equation 3.11 in 3.10, we obtain the characteristic equation of our PID controller described by Geiser [8]:

$$h_{n+1} = (\frac{1}{e_n})^{K_i} * \left(\frac{e_{n-1}}{e_n}\right)^{K_p} * \left(\frac{e_{n-1}^2}{e_n e_{n-2}}\right)^{K_d} * h_n$$

In Figure 8, we describe the algorithm that we use to control the error during the time discretization process. In this algorithm, the controller is located inside the Newton-Raphson loop. In the event that Newton-Raphson does not converge to the solution, the PID controller uses the error estimate to adjust the time-step and improve the chances of convergence. If the error criteria are met, the algorithm accepts the solution of the current iteration and uses a basic controller to adjust the time-step of the next iteration.

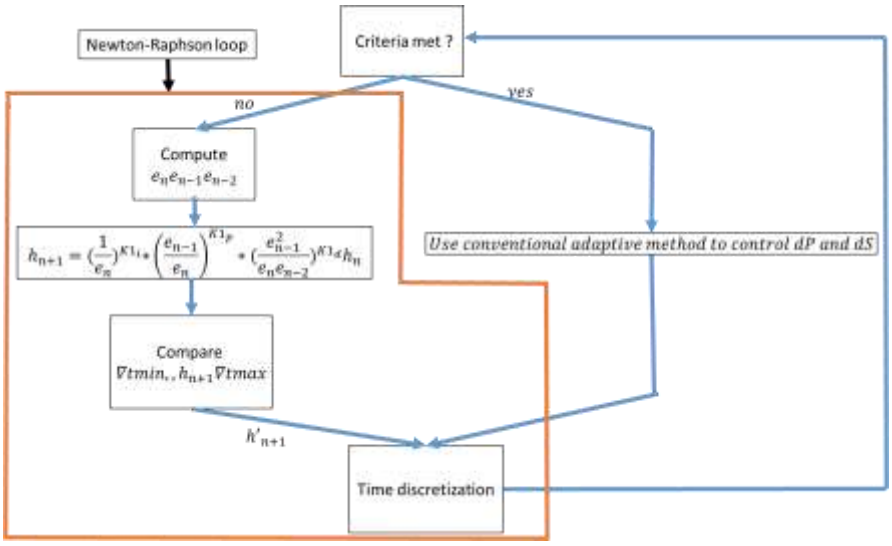

Figure 8: Algorithm to control error

3.5 Resolution control algorithm in Reservoir simulation

We also test a second PID controller to regulate the resolution of the solutions: In regions of fast pressure change and saturation, the control system would reduce the time steps to better capture the variations of the solution. Similarly, in areas of slow pressure

change and saturation, the system would allow larger time steps to reduce the computation time. A PID recurrence equation to control the resolution is described by the equation below:

$$h_{n+1} = (\frac{1}{C_n})^{K_i} * \left(\frac{C_{n-1}}{C_n}\right)^{K_p} * \left(\frac{C_{n-1}^2}{C_n C_{n-2}}\right)^{K_d} * h_n$$

$$C_n = \max(C_p, C_S) \text{ where } C_p = \frac{\|dP^n\|}{\|dP\_desired\|} \; C_S = \frac{\|dS^n\|}{\|dS\_desired\|}$$

The controller adjusts the time-step to make sure that the pressure changes and saturation changes are lower or equal to the desired value set by the user. It is also important to note that the basic controller is a special case of the PID controller for coefficients K_p=0, K_i=1 and K_d= 0. The workflow of the PID algorithm used to control the pressure change and saturation is summarized in Figure 9.

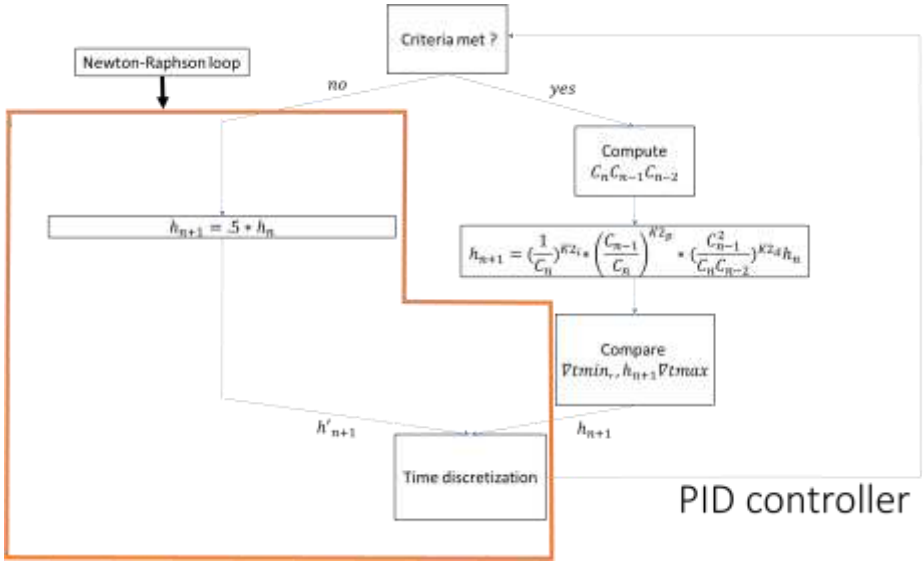

Figure 9: Algorithm to control resolution

After the time discretization process, if the solution doesn't meet the criteria, then it is rejected and the next time-step is obtained by reducing the current time-step by 50%. If the solution meets the user-defined criteria, it is stored and used as input to the PID controller. The time-step obtained from the PID controller is compared to the minimum and maximum time-step value set by the user. These limits are placed in order to prevent the system from becoming unstable. If the PID time-step is lower than the minimum time-step, then the system uses the user-defined minimum time-step. Similarly, if the PID time-step is higher than the maximum time-step, then the system uses the user-defined maximum time-step for the next iteration.

In the next chapter, we apply the algorithm discussed in this chapter to a two-phase reservoir simulator. We compare the performance of the PID controller with the basic controller discussed in this chapter. We investigate various initial conditions scenarios to assess the robustness and the effectiveness of the PID controller at adjusting the time-step throughout the simulation.

CHAPTER IV

SIMULATION RESULTS

In this chapter, we present the results of the PID controller's performance as a tool for adaptive time-stepping. We first perform a tuning of the PID parameters and a sensitivity analysis of the adaptive time-step controller to determine the optimal PID coefficients and how their change affect the time-step response. We then use these coefficients to run our simulator using a set of base-case parameters. We introduce disturbances in the form of change in bottom hole pressures and record the performance of the controllers. We also vary the controller parameters, the permeability and the well location and observe how it affects the response of the controllers.

4.1 Base case scenario

In our base case scenario, we use the permeability and the well location presented on Figure 10.

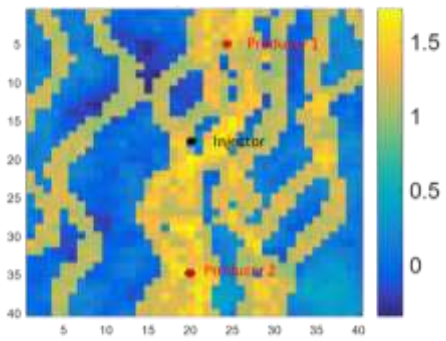

Figure 10: Permeability and well positions of base case. The image represents log (k) where k is in millidarcies

We use a two-phase flow simulator with properties presented in Table 1. As shown on the Figure 10, the base case simulator has two producer wells and one injector well. The injector is located at the coordinates (23, 20) and the two producers are located at the coordinates (24, 5) and (20, 35). We assume a uniform initial porosity of $\emptyset = .2$. In our simulator, the producer wells' bottom hole pressures are maintained at a constant value of 2800 psi and the injector well is steady at 7200 psi. The reservoir size is 2000ft by 2000ft by 600ft. The reservoir is divided into 40 feet intervals in the x and y directions. We assume that there are no variations along the z direction. The reservoir simulation parameters are summarized in Table 1

Reservoir specifications		Simulation parameters		Well properties	
Nx	40	Max number of N.R iterations	15	Injector location (x,y)	(23,20)
Ny	40	Maximum simulation time (years)	20	Producer 1 (x,y)	(24,5)
Nz	1	Relative error tolerance	1E+12	Producer 2 (x,y)	(20,35)
Length in x direction (ft)	2000	Fluid properties		Producers' bottom hole pressure	2800
Length in y direction (ft)	2000	Oil reference viscosity (cp)	3	Injector bottom hole pressure (p	7200
Length in z direction (ft)	600	Water reference viscosity (cp)	1		
Initial reservoir pressure (psi)	4200	Oil density (lbm/scf)	40		
Initial water saturation	0.1	mu oil	0.000002		
		Water density (lbm/scf)	62.238		

Table 1: Summary of simulation parameters

4.2 PID controller tuning

We tune the PID controller to obtain the optimal coefficients for our simulator. There are automatic tuning algorithms designed to easily determine the optimal PID coefficients. However, these tuning algorithms are not well suited for adaptive time-stepping methods. As a result, we manually tune the PID controller with a procedure consisting of

varying one PID coefficient while leaving the others constant and recording the performance of the system. This process is performed for all three PID coefficients with the objective of obtaining PID coefficients that would lead to a fast response while minimizing the likelihood of overshoots above the user-defined control target. The results obtained from the PID tuning are summarized in Table 2. We obtain the best results with K_p=.001, K_i=1.34 and K_d= .01. These coefficients are used throughout our study to compare the performance of the PID controller with the basic controller. In the next section, we discuss how each coefficient affect the response of the PID controller through a sensitivity analysis.

KP	KI	KD	Time-step	NR	Overshoot
0	0.01	0	Very slow	Very slow	Very slow
0	0.1	0	632	159	n/a
0	1	0	404	87	1
0	1.2	0	397	85	3
0	1.4	0	388	83	3
0.01	0.1	0	629	158	n/a
0.01	1.2	0	397	85	1
0.05	1.2	0.01	390	84	2
0.001	1.4	0	392	84	1
0.001	1.34	0.01	392	84	1

Table 2: Results of manual tuning of PID controller. N.R is the total number of Newton-Raphson iteration performed during the simulation and the overshoot is the number of times the controlled variable goes above the controller target.

4.3 Sensitivity analysis

We perform a sensitivity analysis to understand how each coefficient of the PID controller influence the time-step response. In Figure 11, we plot the time-step response with varying proportional coefficient values, leaving the integral and derivative coefficients constant. The time-step response is faster with increasing proportional coefficient. For coefficients above the critical value of 1, the PID controller becomes unstable and the reservoir simulator is rendered inefficient. This is in line with the general PID control theory as higher gains leads to less stability [19]. We perform a similar analysis on the derivative and the integral coefficients and illustrate the results in Figure 12 and Figure 13 respectively. In both cases, the results show that the time-step response is faster but less stable with increasing PID coefficients. We also noted that the integral coefficient was the most sensitive coefficient in our simulator. In fact, a change of K_i from .01 to .1 led to a drastic change in the time-step response as shown on Figure 12. The same change in K_d barely affected the response as shown in Figure 13. The change in K_p from .01 to .1 only affected the response moderately.

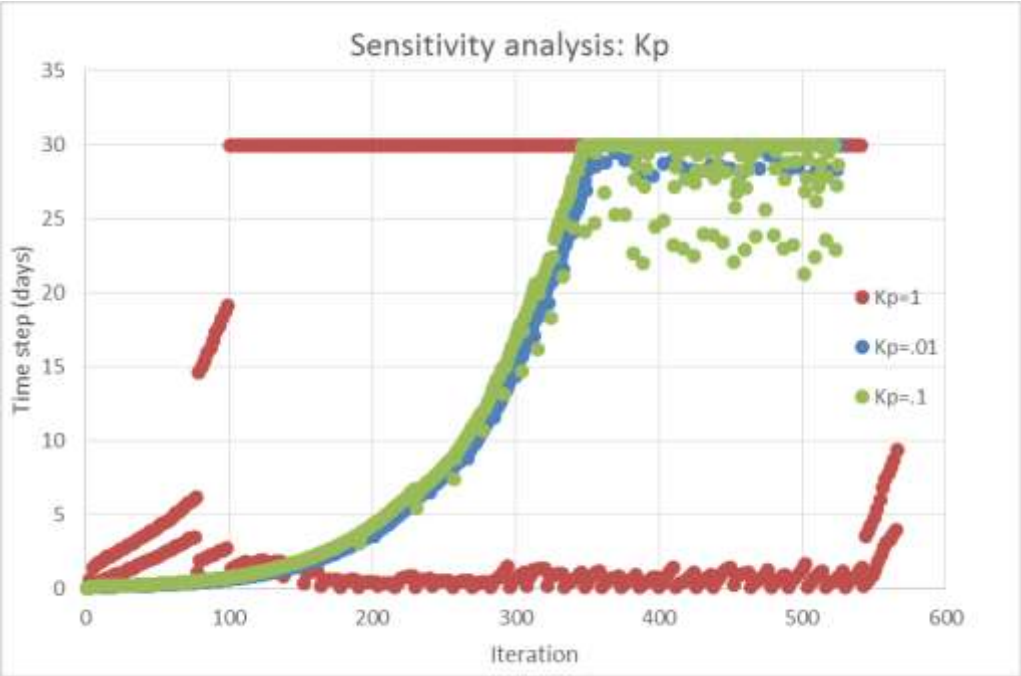

Figure 11: Proportional coefficient sensitivity analysis

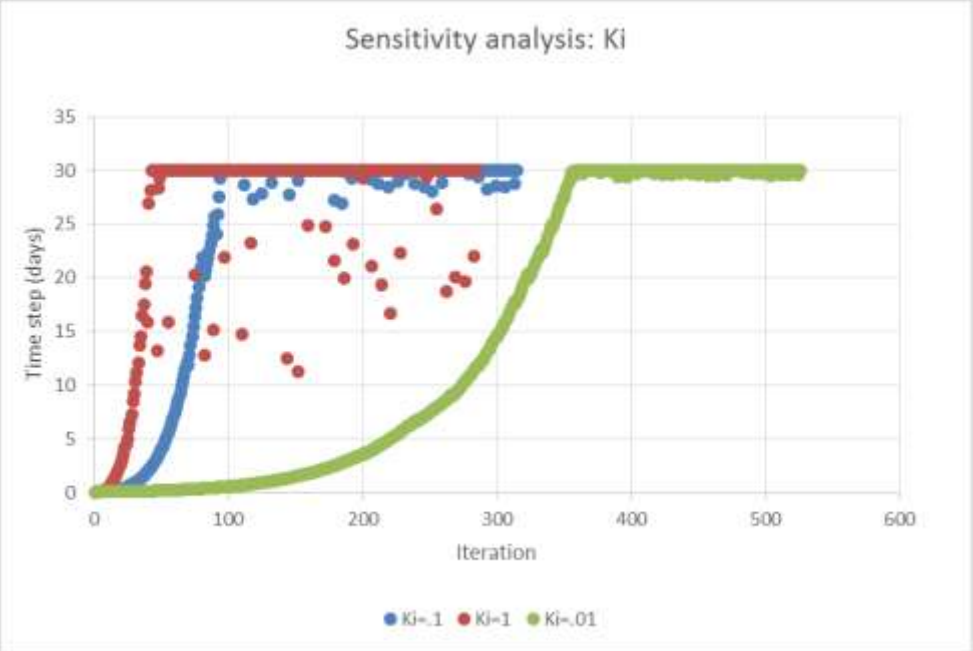

Figure 12: Integral coefficient sensitivity analysis

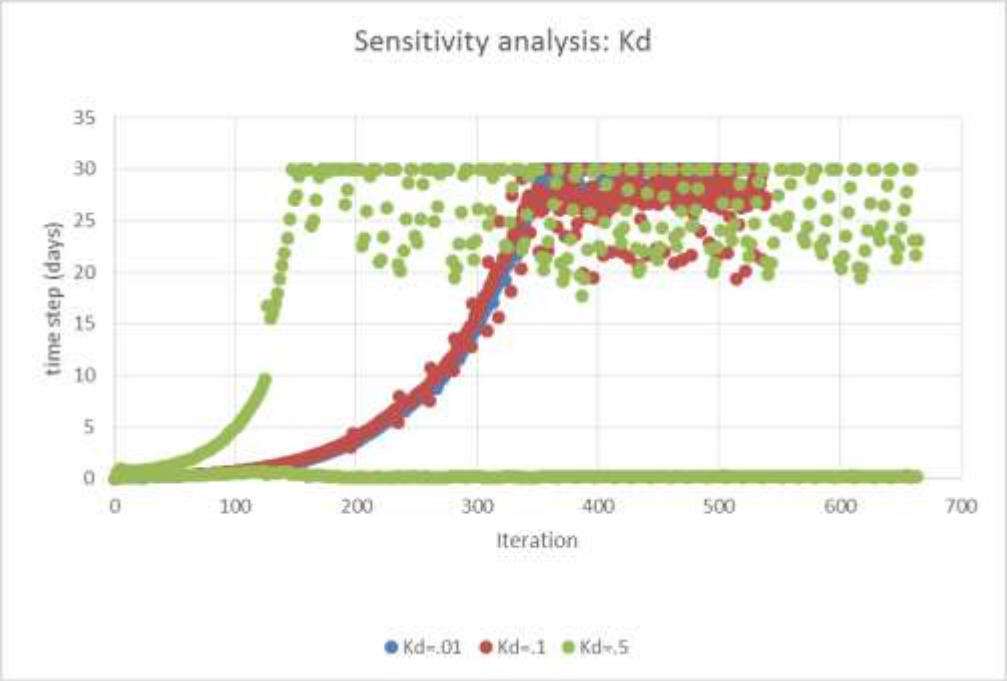

Figure 13: Derivative coefficient sensitivity analysis

4.4 Resolution control

For the resolution control system, the goal of the PID controller is to maintain the pressure change and saturation change below a limit set by the user. This controller does not reject non-compliant solutions, but takes corrective action at the next iteration. In control system terms, the pressure change and saturation change represent the controlled variables. The time-step used in the simulation is the manipulated variable and the reservoir simulator is the plant.

4.4.1 Time-step controller limited by pressure target

The resolution controller simultaneously control the pressure change and the saturation change throughout the simulation. In this section, we make the pressure the limiting factor of the controller by setting the target saturation change dS_{target}=1 and the target pressure change dP_{target}= 80psi. The goal is to evaluate the effectiveness of the controller at regulating the pressure without interference from the saturation component. In our simulations, the time-steps are not bounded and allowed to take any value assigned by the controllers. The idea is to allow the controllers to operate freely in order to fully understand and assess their performance throughout the reservoir simulation. The time-step response that we obtain for the base case using the control parameters described above is shown on Figure 14

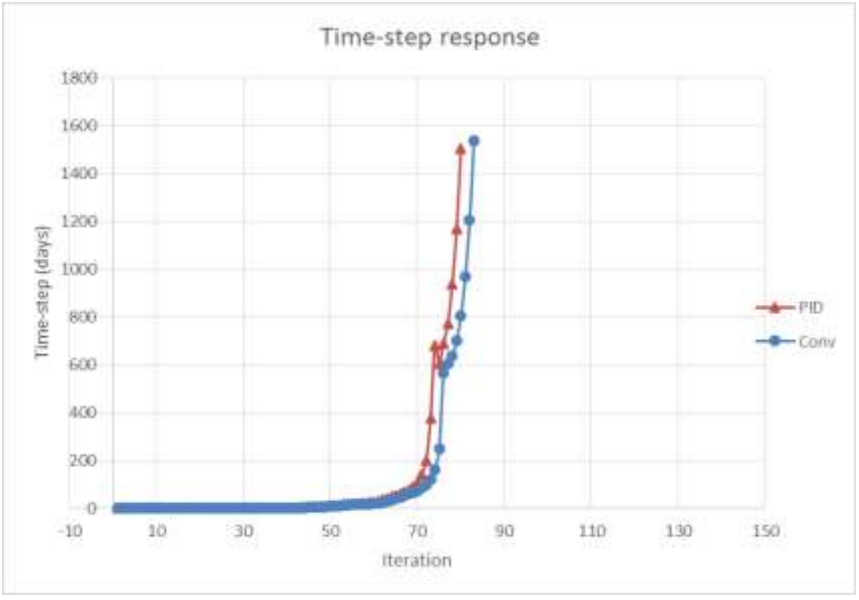

Figure 14: Base case time step response PID vs basic

We can see that the PID controller has a faster response than the basic controller. In fact, throughout the reservoir simulation, the time-step plot of the PID controller is above the one of the basic controller. We also notice a drop in the time-step of the PID controller after 75 time iterations. Around the same time, we observe a reduction in the rate of increase of the basic time-step response. In order to explain this occurrence, we plot of the pressure change versus the simulation time in Figure 15.

Figure 15: Pressure change versus simulation time (base case with pressure limiting factor)

From this plot, we see that the pressure change remains just below the target throughout the simulation for both controllers. After 1000 days, we notice a sudden drop in the

pressure change that results in a time-step increase by the PID and the basic controllers to counteract the drop. The increase of the PID time-step is much more drastic than the one of the basic controller and it results in an overshoot of the target by about 8%. The PID controller then corrects the overshoot by reducing the time-step, hence the drop in time-step observed after 75 iterations. The basic controller doesn't overshoot and we do not see any reduction in its time-step throughout the simulation. Instead, the basic controller considerably reduces the rate of increase of the time-step as the pressure change reaches its target at around 1500 days.

As the reservoir system transitions from the transient to the steady-state region, the saturation change and the pressure change are considerably reduced. Thus, both controllers significantly increase the time-step to maintain the system close to target.

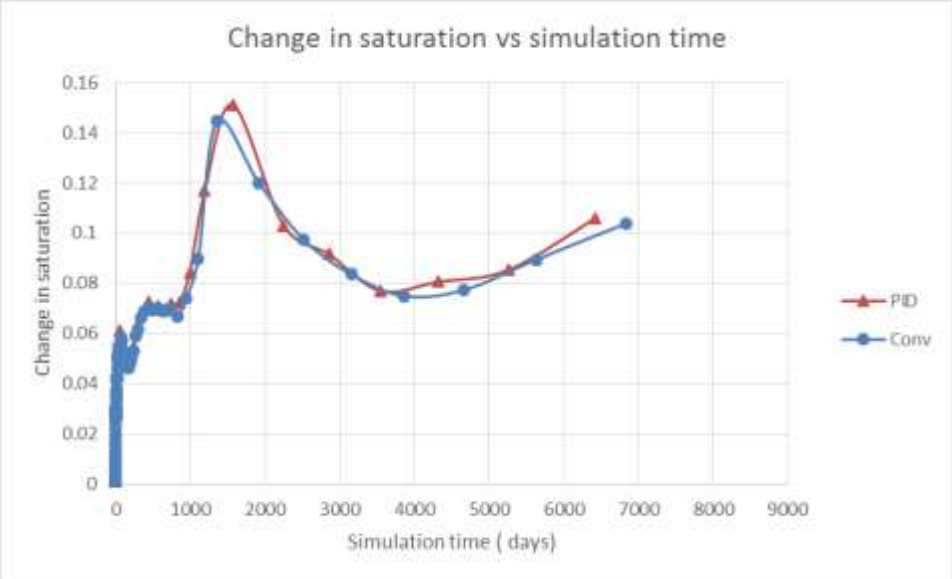

Figure 16: Saturation change vs simulation (Base case with pressure as limiting factor)

In fact from Figure 15, we observe after 3000 days that the pressure change of the PID and the basic controller are well below their target and drifting even lower with time. In order to maintain the pressure change close to the target, the two controllers drastically increase the rate of increase of the time-step, resulting in a near vertical time-step response. The PID controller has a faster time-step response and is able to maintain the pressure change closer to the target than the basic controller as seen in Figure 15. The use of higher time-steps by the PID controller results in a lower overall computation cost and a higher risk of overshoot.

We plot the saturation change as a function of the simulation time in Figure 16. The saturation change increases until it reaches a maximum value of .15 at around 1500 days. After that, the saturation change drops and stabilizes. The peak in the saturation change coincide with the sudden drop in pressure change recorded in Figure 15. As discussed earlier, the controllers are not limited by the saturation change and the saturation change plot generated is a result of the controllers' response to the pressure change requirements.

The performance of the two controllers is summarized in Table 3. The results show that the PID controller generally has a lower time-step, total number of Newton-Raphson iterations and computation time compare to the basic controller. As we observe in Figure 17, the computation time of the PID controller is generally lower than the one of the basic controller except for a simulation time of 2 years. The percentage reduction of the computation time varies from -7% to 30%.

simulation time (years)	1	2	3	4	5	20
Conventional method						
Computation time (min)	0.647	0.731	0.89	0.775	1.09	0.95
Total number of time steps	65	71	73	75	75	82
Number of NR iterations	268	298	308	319	319	354
Ratio NR/total time steps	4.123077	4.197183	4.219178	4.253333	4.253333	4.317073
PID						
Computation time (min)	0.64	0.78	0.74	0.751	0.759	0.82
Total number of time steps	63	68	71	72	73	79
Number of NR iterations	259	284	299	304	310	340
Ratio NR/total time steps	4.111111	4.176471	4.211268	4.222222	4.246575	4.303797
Comparison						
% reduction in NR iteration	3.36%	4.70%	2.92%	4.70%	2.82%	3.95%
% reduction in time steps	3.08%	4.23%	2.74%	4.00%	2.67%	3.66%
% comp time reduction	1.08%	-6.70%	16.85%	3.10%	30.37%	13.68%

Table 3: Performance of PID vs basic (base case with pressure limiting factor)

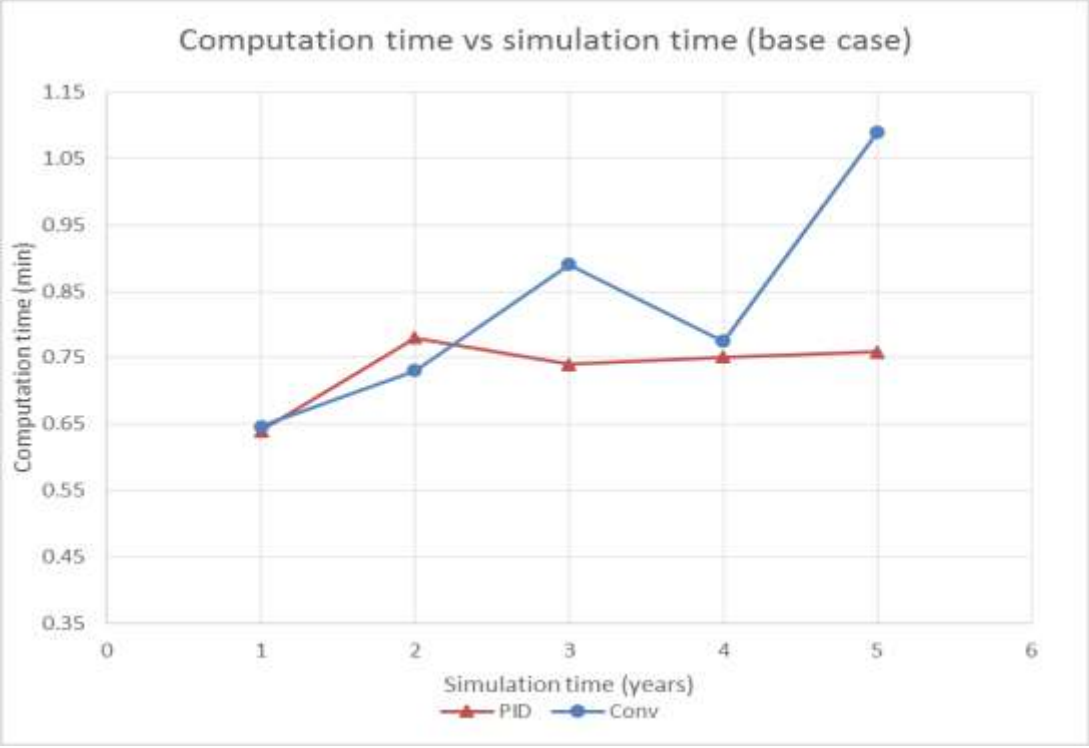

Figure 17: Computation time vs simulation time (base case with pressure as limiting factor)

We note a percentage reduction in the total number of Newton-Raphson iterations of the PID by nearly 4% for a simulation time of 20 years. Similarly, we also note a 4% reduction in the total number of time-steps for the same simulation time.

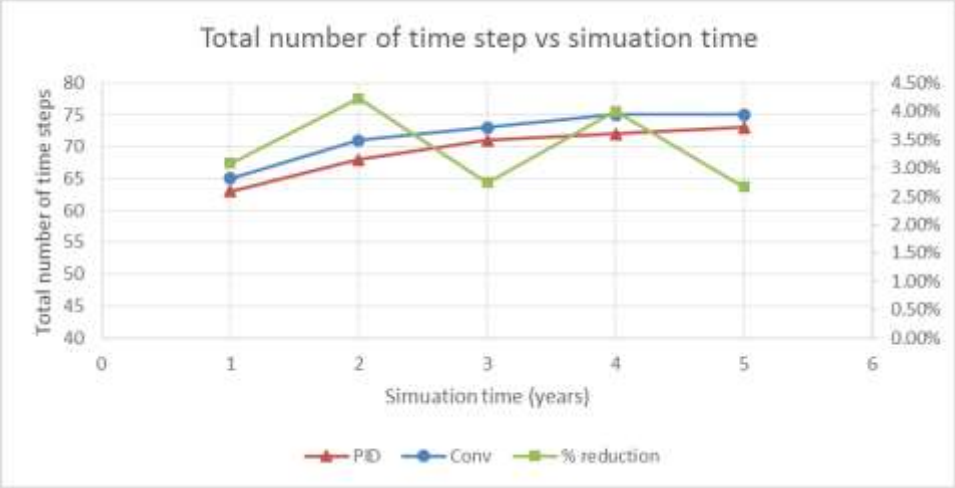

Figure 18: Total number of time-step vs simulation time for base case

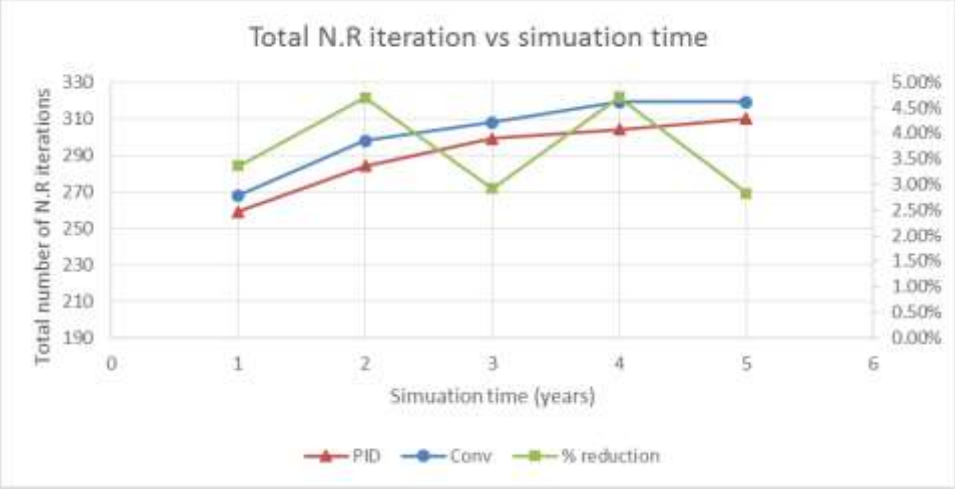

Figure 19: N.R iteration vs simulation time for base case

We compute and plot the ratio between the total number of Newton-Raphson (N.R) iterations and the total number of time-steps on Figure 20 .We see that ratio the of N.R iterations to number of time-steps is consistently higher for the PID controller. This trend is expected as the average time-step generated by the PID controller is slightly larger than the one from the basic controller. As a result, it takes more iterations for the Newton-Raphson to converge to the solution when using the PID controller. It is also important to note that even though the average number of N.R iteration per time-step is higher for the PID controller, we obtain a lower total number of N.R iterations and computation time using this method. As a result, the overall computational cost of using the PID controller is lower than the one for the basic controller.

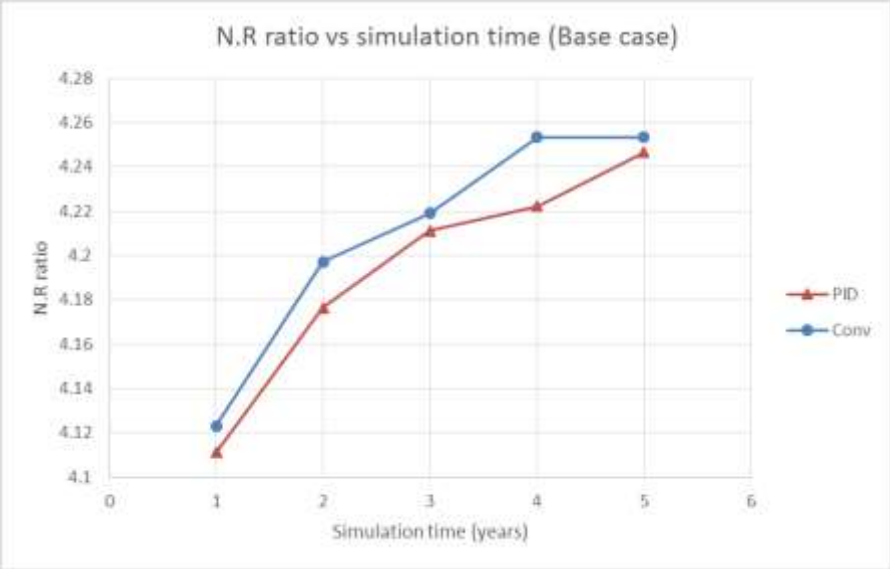

Figure 20: N.R ratio vs simulation time for base case

In Figure 21, we plot the oil production rate of production well number 2. The results obtained using the PID controller are compared with the one obtained using the basic controller. As we see from the figure, we get the same production rate with the two adaptive time-stepping methods. Despite the lower number of time-steps and computation time obtained using the PID controller, the accuracy of the solution has not been affected.

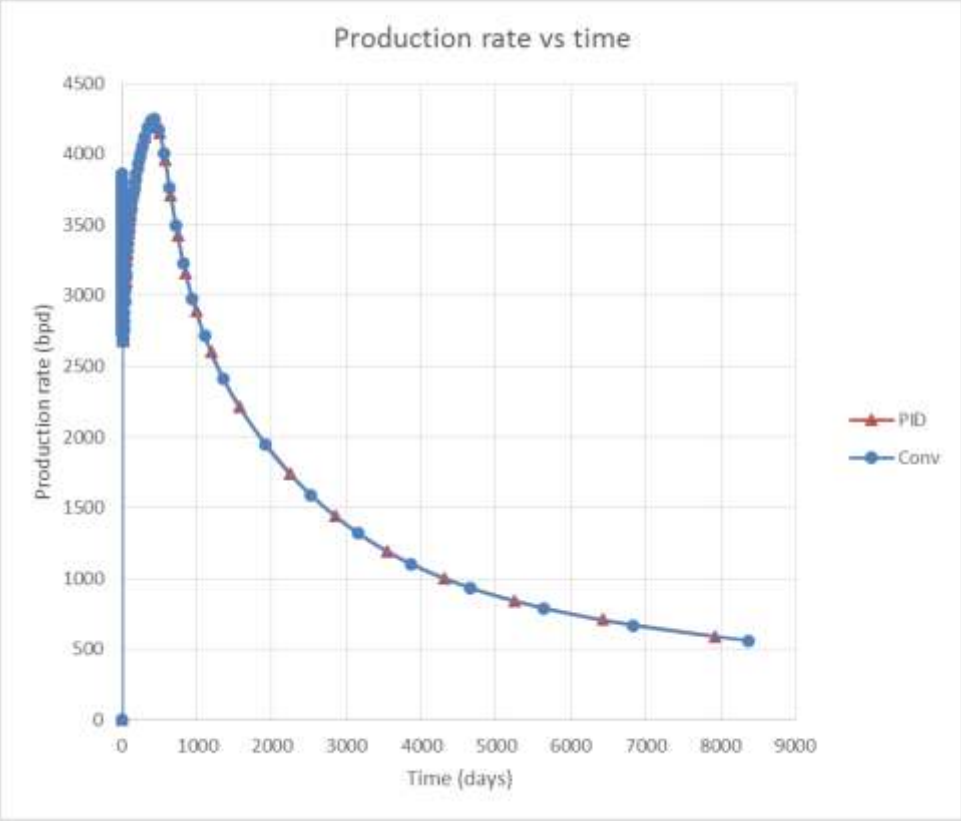

Figure 21: Production rate vs simulation time for production well 2 (base case with pressure limiting factor)

4.4.2 Effect of disturbance on the pressure-limited controller

In this section, we introduce disturbances in reservoir simulator in the form of sudden changes in bottom hole pressures of the producer and injector wells and record the performance of the controllers. The bottom hole pressures vary within a range of 300 psi from the nominal values and a period of 200 days.

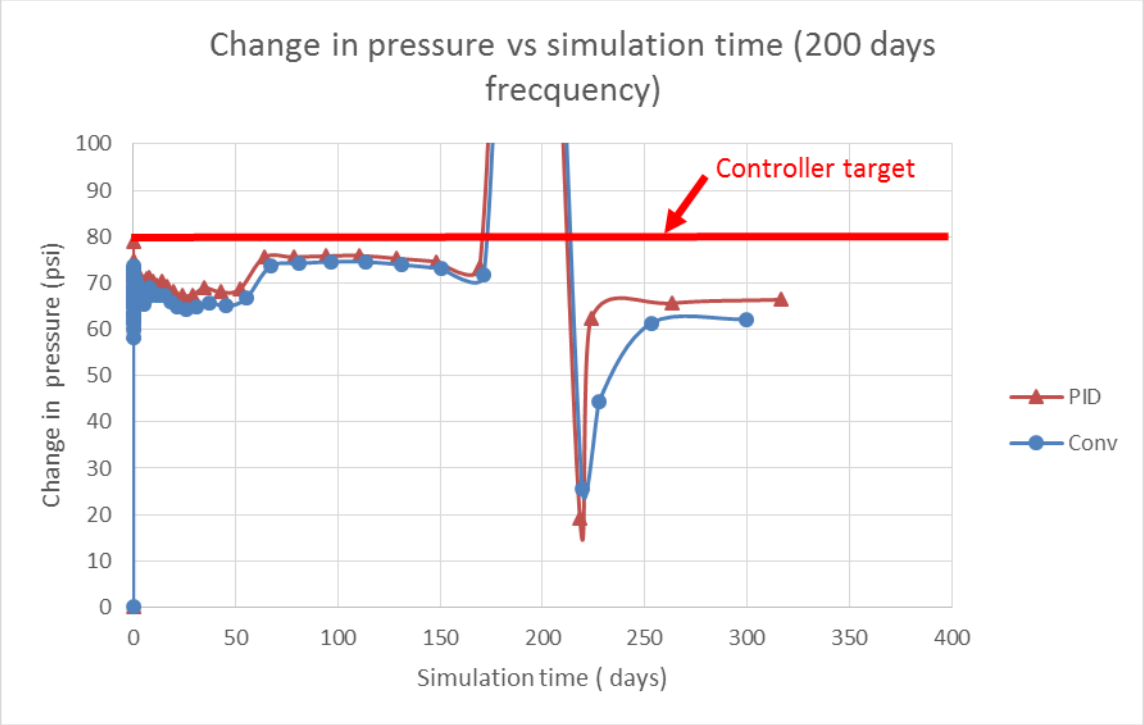

Figure 22: Pressure change with 300 psi disturbance (early times)

In Figure 22, we plot the pressure change at early times with a sudden change in the bottom hole pressure. As the bottom hole pressure change occur, the pressure change curve shoots up above the controller target. As a result, the time-step controller takes

corrective action by dropping the time-step of the reservoir simulation as shown on Figure 23.

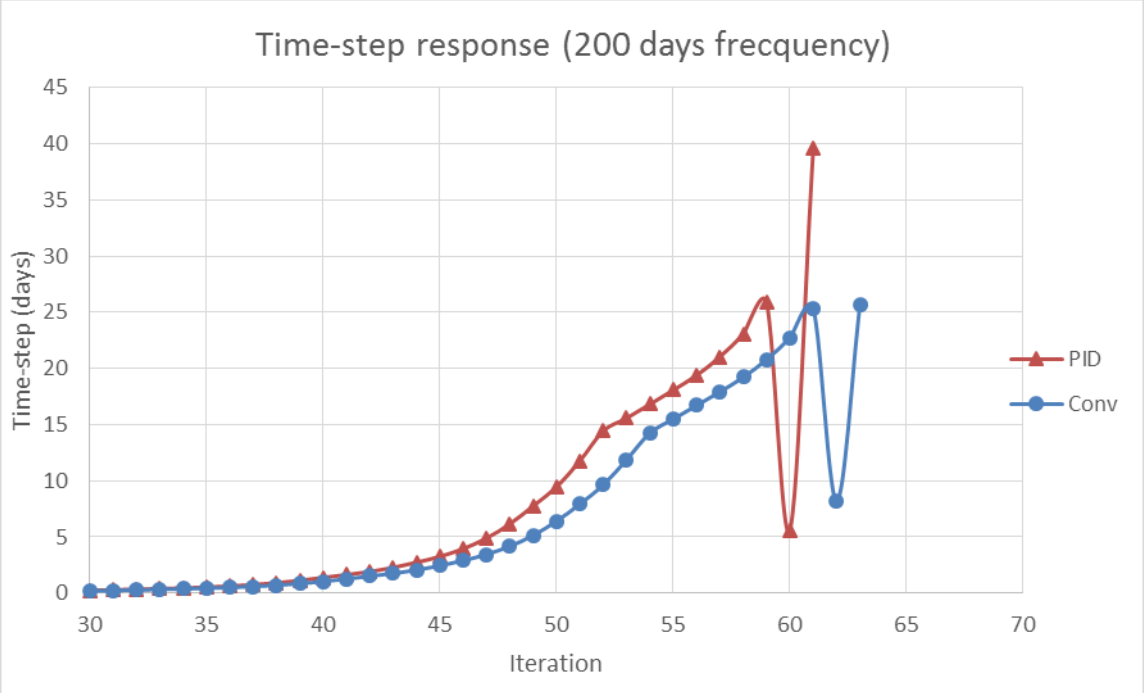

Figure 23: Time-step response with occurrence of disturbance (early times)

The controllers respond to the disturbance by drastically reducing the time-step in the simulation with the PID controller inducing a more significant drop in time-step than the basic controller. The reduction in time-step by both controllers leads to a significant undershoot of the pressure change target. In order to correct for the newly created undershoot, the controllers switch directions again and now increase the time-step creating a v-shaped time-step response to the disturbance. As the system stabilizes, we

see in Figure 22 that the PID controller pressure change is closer to the target than the basic controller. The proximity of the PID controller to the target pressure change is due to the higher time-steps used that also leads to a lower computational cost. In fact, with the PID controller we get a lower total number of time-steps, N.R iterations and computation time as illustrated in Table 4.

Disturbance amplitude (psi)	300	400	500
Conventional method			
Computation time (min)	1.599	1.62	2.093
Total number of time steps	147	156	203
Number of NR iterations	629	664	854
Ratio NR/total time steps	4.278912	4.25641	4.206897
PID			
Computation time (min)	1.3	1.55	1.86
Total number of time steps	125	150	173
Number of NR iterations	536	637	736
Ratio NR/total time steps	4.288	4.246667	4.254335
Comparison			
% reduction in NR iteration	14.79%	4.07%	13.82%
% reduction in time steps	14.97%	3.85%	14.78%
% comp time reduction	18.70%	4.32%	11.13%

Table 4: Effect of disturbance amplitude on time-step controllers

As it can be seen in Table 4, we get a total number of N.R iterations reduction with the PID controller of 15%, 4% and 14% for disturbance ranges of 300 psi, 400psi and 500 psi respectively. We do not observe a clear trend in the computational cost reduction with increasing disturbance range.

4.4.3 Effect of pressure target on the reservoir system

In this section, we vary the pressure target of the controllers and record the performance. The results are summarized in Table 5. We test a total of 4 target pressures for the time-step controllers: 30, 40, 50, and 80 psi. The results show that as the pressure change target increases, so thus the percentage reduction in total N.R iterations, time-steps and computation time of the PID controller compare to the basic controller.

Controller pressure target (psi)	80	50	40	30
Conventional method				
Computation time (min)	0.95	1	1.219	1.58
Total number of time steps	82	124	152	199
Number of NR iterations	354	494	599	756
Ratio NR/total time steps	4.317073	3.983871	3.940789	3.798995
PID				
Computation time (min)	0.82	0.9945	1.21	1.59
Total number of time steps	79	121	149	196
Number of NR iterations	340	483	586	746
Ratio NR/total time steps	4.303797	3.991736	3.932886	3.806122
Comparison				
% reduction in NR iteration	3.95%	2.23%	2.17%	1.32%
% reduction in time steps	3.66%	2.42%	1.97%	1.51%
% comp time reduction	13.68%	0.55%	0.74%	-0.63%

Table 5: Effect of pressure target on reservoir performance

This trend is observed because as the pressure target decreases, there is less room for the time-step of the PID controller to increase, as a result, its speed is limited to a small time-step interval , hence a lower reduction in computation time, N.R iterations, and

time-steps. In Figure 24 we plot the total number of time-steps as a function of the pressure target. We see that the percentage reduction in total time-steps rise with increase in the pressure target of the controller. In other words, the PID controller becomes more computationally efficient as the pressure change target of the controller is raised. The same trend is observed for the total number of N.R iterations as illustrated in Figure 25.

Based on the data recorded, we conclude this section by establishing the correlation between the increase of the pressure target and the lowering of the computational cost of the PID controller relative to the basic method.

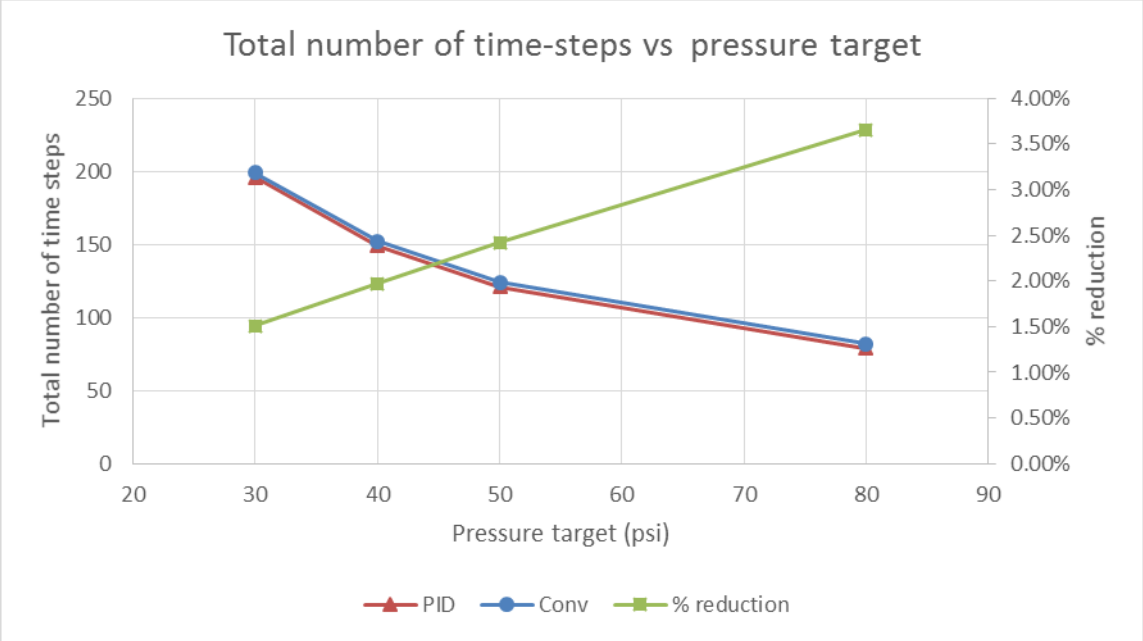

Figure 24: Total number of time-step vs pressure target

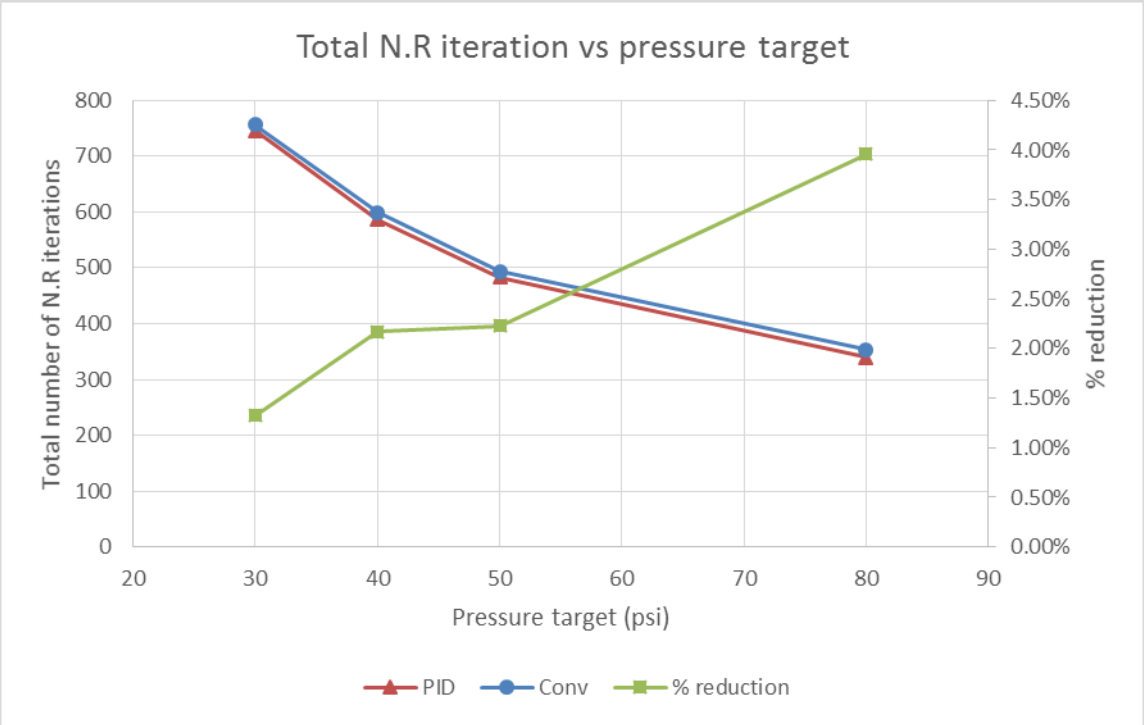

Figure 25: Total number of N.R iterations as a function of pressure target

4.4.4 Time-step controller limited by saturation target

In this section, we make the saturation change the limiting factor by setting $dS_{target} = .05$ and $dP_{target} = 250\ psi$. In this scenario, we are able to study the controllers' ability to regulate the saturation change without any interference from the pressure component.

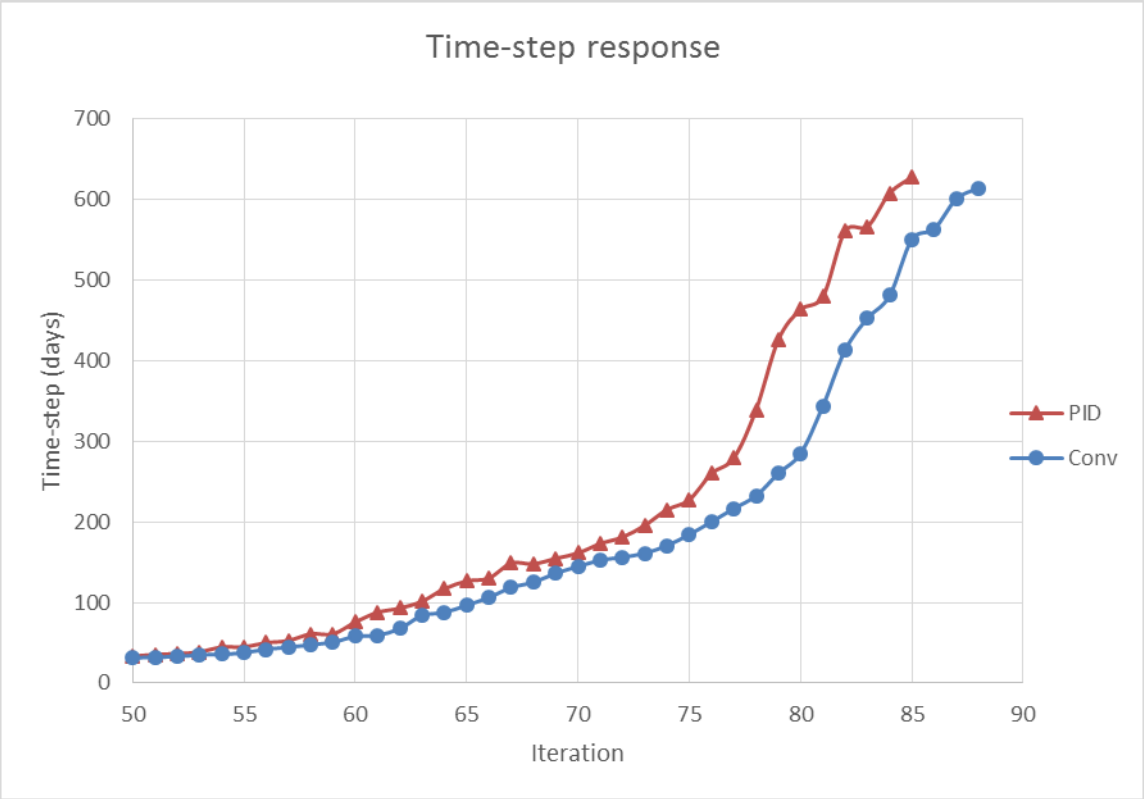

Figure 26: Time-step response (base case with saturation as limiting factor)

In Figure 26, we graph the time-step response of the PID controller and the basic with the saturation change as the limiting factor. We observe that the response from the PID controller is faster than the basic controller. In fact, the PID time-steps remains consistently above the basic controller time-step throughout the simulation.

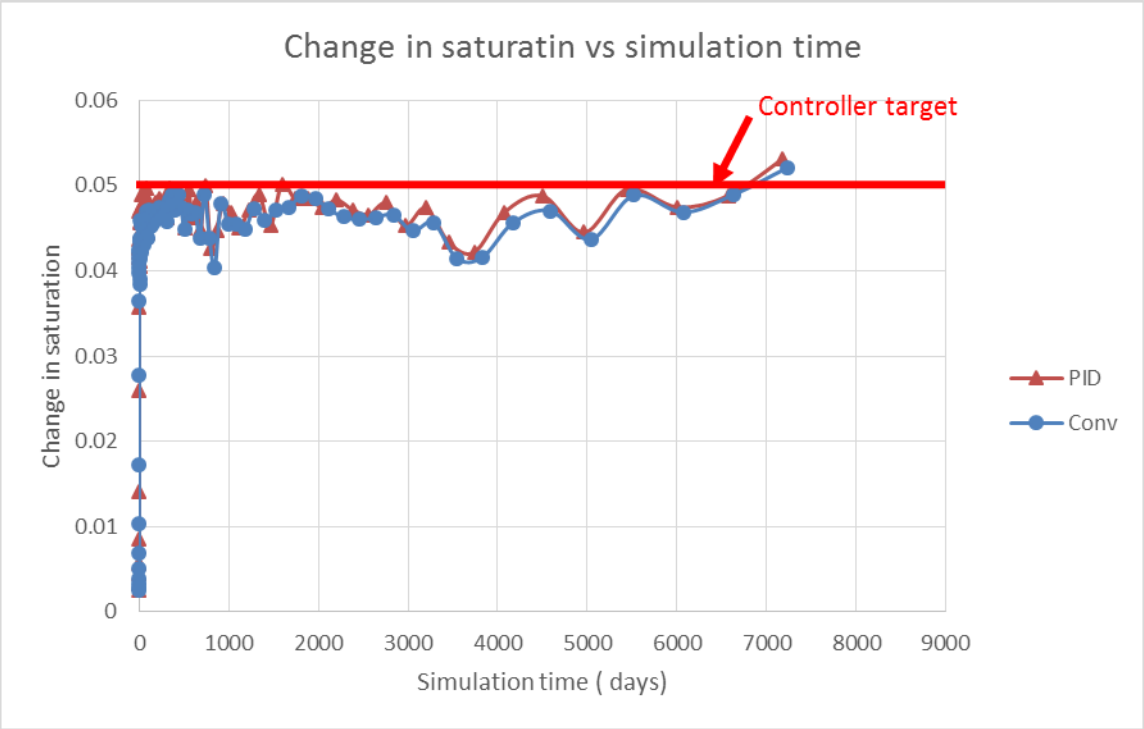

Figure 27: Saturation change vs simulation time (base case with saturation as limiting factor)

On Figure 27, we plot the change of saturation as a function of the simulation time for the PID controller and the basic controller. We note a lot of oscillations at early time with the saturation change mainly remaining below the controller target. As the system switches from transient phase to steady-state, we see a drop in the saturation change that is eventually corrected by the time-step controllers. Unlike the pressure control case, the controller is able to easily maintain the saturation change close to the target set by the user. We also note an overshoot of the target at the last iteration in both the basic and PID controllers. In Table 6, we sumarize the performance results from the basic and the

PID controller. The computation results in Table 6 confirms that the PID controller is less computationally costly than the basic controller.

simulation time (years)	1	2	3	4	5	20
Conventional method						
Computation time (min)	0.429	0.518	0.55	0.58	0.613	0.79
Total number of time steps	49	59	64	67	70	87
Number of NR iterations	216	266	290	305	320	404
Ratio NR/total time steps	4.408163	4.508475	4.53125	4.552239	4.571429	4.643678
PID						
Computation time (min)	0.399	0.482	0.522	0.565	0.581	0.728
Total number of time steps	47	56	61	64	67	84
Number of NR iterations	208	253	277	292	307	392
Ratio NR/total time steps	4.425532	4.517857	4.540984	4.5625	4.58209	4.666667
Comparison						
% reduction in NR iteration	3.70%	4.89%	4.48%	4.26%	4.06%	2.97%
% reduction in time steps	4.08%	5.08%	4.69%	4.48%	4.29%	3.45%
% comp time reduction	6.99%	6.95%	5.09%	2.59%	5.22%	7.85%

Table 6: Performance of PID controller vs basic (base case saturation limiting factor)

In fact we get a reduction in total N.R iterations ranging from 3% to 5% with the PID controller compared to the basic controller. The same trend is observed for the total time-step and the computation time. In both cases, the PID controller has lower values with the percentage reduction in computation time reaching nearly 8%. We can conclude that the PID controller is more efficient at regulating the saturation change given the conditions used in this section.

4.4.5 Effect of saturation target on the reservoir system

In this section, we vary the saturation target and record the effect on the controllers' performance. The results obtained are summarized in Table 7

Controller saturation target	0.05	0.025	0.01
Conventional method			
Computation time (min)	0.79	1.42	4.49
Total number of time steps	87	172	435
Number of NR iterations	404	691	1735
Ratio NR/total time steps	4.643678	4.017442	3.988506
PID			
Computation time (min)	0.728	1.4	4.22
Total number of time steps	84	169	432
Number of NR iterations	392	676	1722
Ratio NR/total time steps	4.666667	4	3.986111
Comparison			
% reduction in NR iteration	2.97%	2.17%	0.75%
% reduction in time steps	3.45%	1.74%	0.69%
% comp time reduction	7.85%	1.41%	6.01%

Table 7: Performance of controllers with varying saturation target

The results that we obtained are similar the pressure target case. We observe in Table 7 that, as the saturation target drops, the percentage drop in total N.R iterations and time-step drops as well. The advantages of the PID controller are more pronounced when the saturation target is higher.

4.4.6 Effect of disturbance on the saturation-limited controller

In this section, we discuss the results that we obtain by introducing disturbances in the form of sudden change in bottom hole pressures and record the reaction of the PID and the basic controllers.

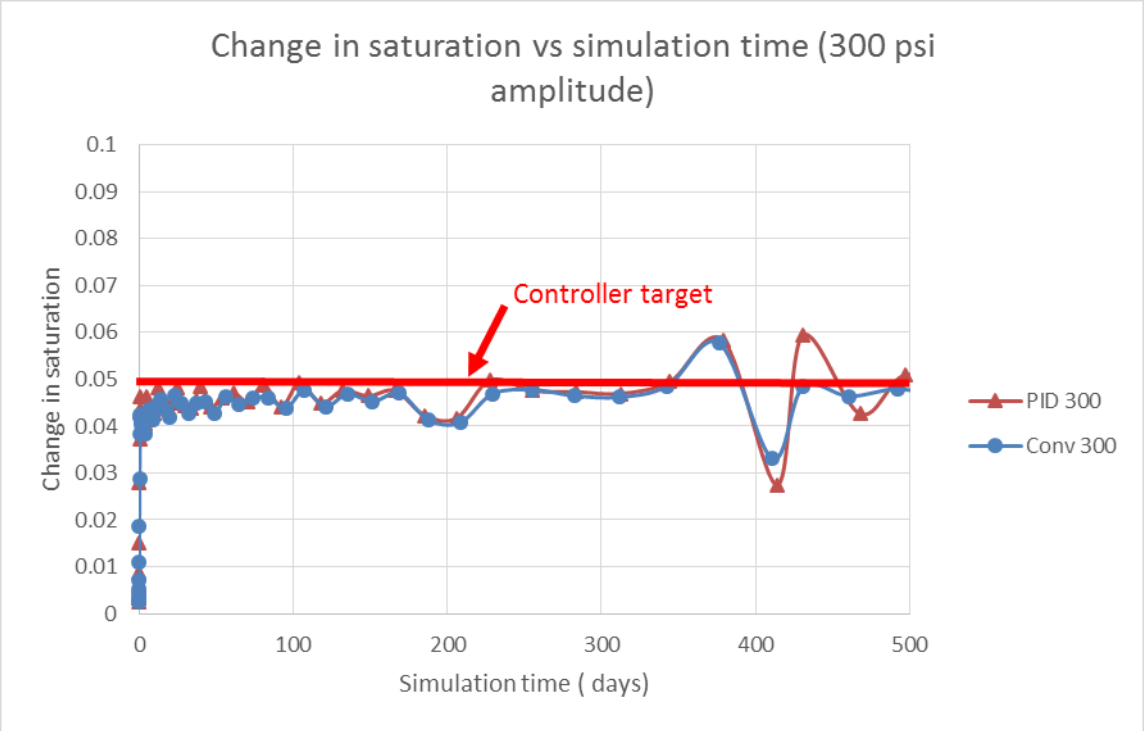

Figure 28: Saturation change with disturbance (saturation limiting factor)

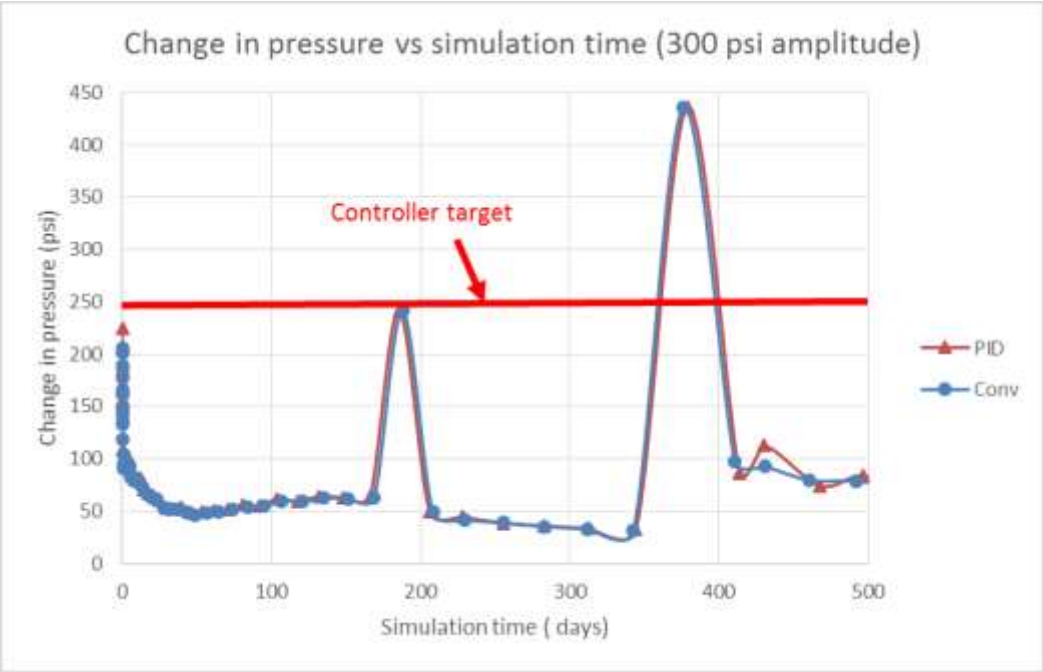

Figure 29: Pressure change with disturbance (saturation limiting factor)

In Figure 28 and Figure 29, we present the saturation change and pressure change versus simulation time with a sudden change in the bottom hole pressure of the producer and injector wells. The first disturbance occurs after 190 iterations, but has a mild reaction from the controllers because it does not cross the pressure target line as seen in Figure 29. We have a second disturbance after around 430 iterations. This time, the target pressure change is exceeded, as a result the controllers take drastic corrective action as illustrate in Figure 30.

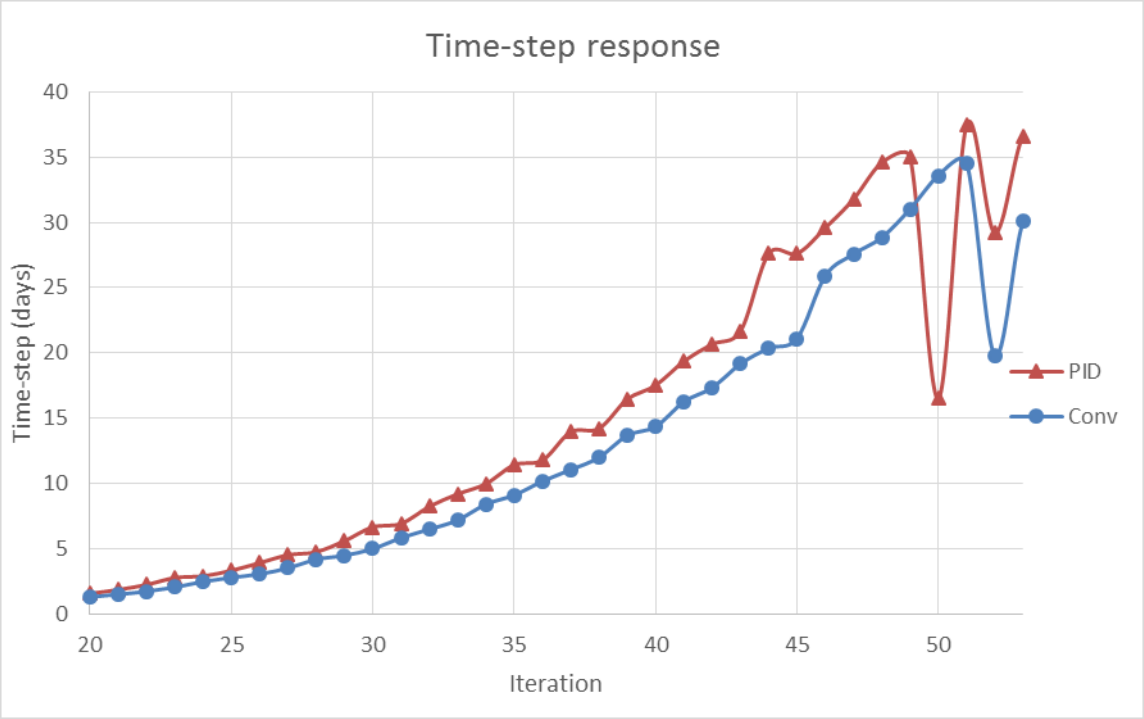

Figure 30: Time-step response with presence of disturbance (saturation as limiting factor)

As can be observed, we record a significant drop in the time-step of the PID and the basic controller. We note that the PID controller forces a steeper drop in the time-step than the basic controller. As a result, the saturation change drops significantly before being pulled back up by the controller. The pullback effect of the PID controller leads to a 20% overshoot of the saturation target. At the same time, we see that the basic controller has a more smooth reaction and as result avoid overshooting the target saturation. Again, we see that with the more responsive PID controller has a faster reaction than the basic controller, but at the same time can lead to overshoot depending on the amplitude of the disturbance and the response of the reservoir system. It is

important to note that the basic controller also overshoots the target, but the chances of overshoot are greater with the PID controller.

Disturbance amplitude (psi)	300	400	500
Conventional method			
Computation time (min)	0.837	0.898	0.959
Total number of time steps	93	104	111
Number of NR iterations	424	480	515
Ratio NR/total time steps	4.55914	4.615385	4.63964
PID			
Computation time (min)	0.805	0.809	0.8259
Total number of time steps	91	93	93
Number of NR iterations	422	428	427
Ratio NR/total time steps	4.637363	4.602151	4.591398
Comparison			
% reduction in NR iteration	0.47%	10.83%	17.09%
% reduction in time steps	2.15%	10.58%	16.22%
% comp time reduction	3.82%	9.91%	13.88%

Table 8: Performance of PID controller vs Basic controller with varying disturbance amplitude (saturation limiting factor)

The performance of the PID controller limited by saturation compared to the basic controller is summarized in Table 8. The data shows that the increase in amplitude of the disturbance correlates with a relatively faster computation time, lower time-step and N.R iterations of the PID controller compared to the basic controller. We also recorded the data showing the performance of the PID and the basic controller with increasing disturbance frequency in Table 9. The data shows that as the frequency of the disturbance increases, the PID controller becomes faster compare to the basic method. In

fact, with a period of 200 days, we get about .5% reduction in the total number of N.R iterations. This number goes up to 8.67% with a period of 50 days. A similar trend is observed with the computation time and the total number of time-steps.

Disturbance period (days)	200	100	50
Conventional method			
Computation time (min)	0.837	0.959	1.1
Total number of time steps	93	111	128
Number of NR iterations	424	509	588
Ratio NR/total time steps	4.55914	4.585586	4.59375
PID			
Computation time (min)	0.805	0.94	0.997
Total number of time steps	91	109	115
Number of NR iterations	422	504	537
Ratio NR/total time steps	4.637363	4.623853	4.669565
Comparison			
% reduction in NR iteration	0.47%	0.98%	8.67%
% reduction in time steps	2.15%	1.80%	10.16%
% comp time reduction	3.82%	1.98%	9.36%

Table 9: Performance of PID and basic with varying disturbance frequency (saturation as limiting factor)

4.4.7 Robustness of the PID controller with permeability changes

In this section, we test the ability of the PID to effectively operate in various reservoir properties. To this end, we multiply the base case permeability of the reservoir simulator by a factor and record the performance of the PID controller and compare its

performance to the basic controller. By increasing the permeability of the simulator, we essentially increase the magnitude of the pressure changes and saturation that occur throughout the simulation. The goal is to assess the robustness of the PID controller under varying conditions. The results are presented in Table 10.

Permeability factor	0.01	0.1	1	10	100
Conventional method					
Computation time (min)	0.448	0.64	0.78	1.0258	5.07
Total number of time steps	52	74	87	118	583
Number of NR iterations	211	331	404	535	2484
Ratio NR/total time steps	4.057692	4.472973	4.643678	4.533898	4.26072
PID					
Computation time (min)	0.425	0.613	0.728	1.011	4.94
Total number of time steps	50	71	84	115	581
Number of NR iterations	207	319	392	521	2463
Ratio NR/total time steps	4.14	4.492958	4.666667	4.530435	4.239243
Comparison					
% reduction in NR iteration	1.90%	3.63%	2.97%	2.62%	0.85%
% reduction in time steps	3.85%	4.05%	3.45%	2.54%	0.34%
% comp time reduction	5.13%	4.22%	6.67%	1.44%	2.56%

Table 10: Performance of controller with varying permeability (saturation as limiting factor)

We observe much lower time-step for the PID controller than the basic controller. Excluding the permeability factor of .01 data, we observe that the PID controller becomes less efficient relative to the basic controller with increasing permeability factor. In fact, we note a percentage reduction in the total number of N.R iterations of 4.05% with a permeability factor of .1. This number drops to .34% with a permeability factor of

100. This trend can be explained using Figure 31. In fact, from Figure 31, we see that the time-step of the two methods stays within the range of 5 to 15 days. As the permeability of the system increases, so thus the variations of pressure and saturations. As a result, the controllers have to use relatively small time-steps to keep the variations within the user-defined tolerance. Since the time-steps does not have room to increase in this simulation, the fast response of the PID controller becomes less utilized, as the time-step remains within a relatively small range.

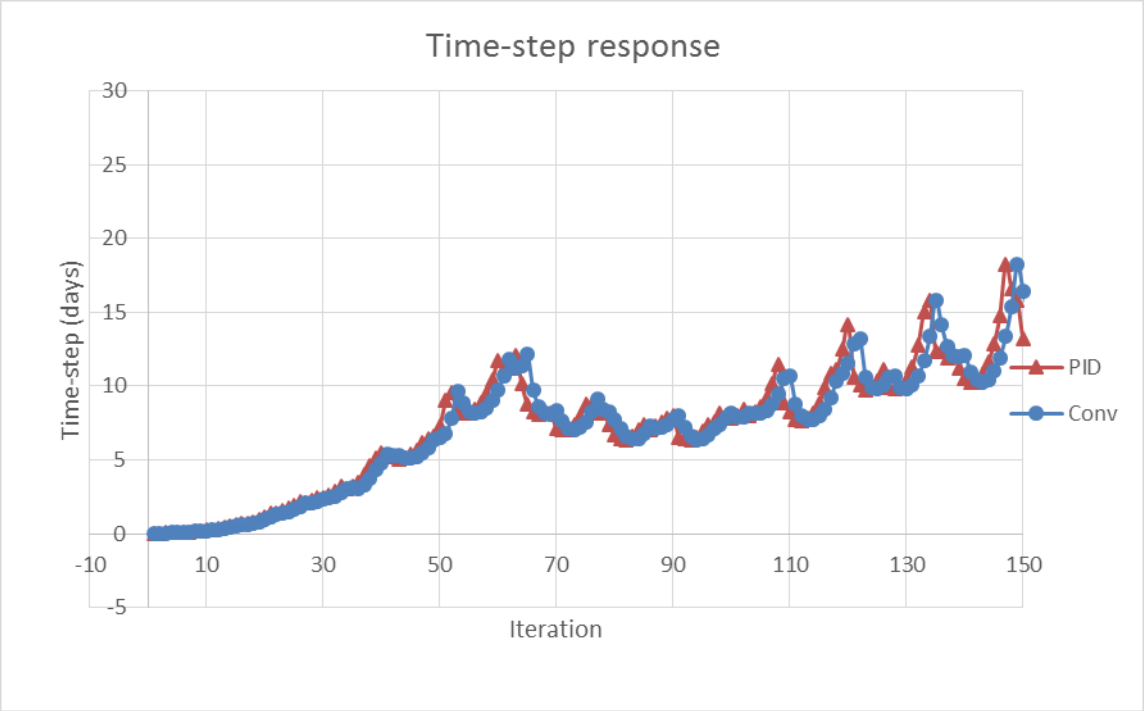

Figure 31: Time-step response PID controller (perm x100)

In Figure 32 we plot the saturation change during the simulation of a system with a permeability factor of 100. From the graph, we see that both methods constantly overshoot the target. This trend is due to the fact that the reservoir system has a quick response and small changes in time-steps leads to large pressure and saturation variations. The PID controller can be adjusted to eliminate these overshoot by lowering the PID coefficients. This action will also lead to an increase in the computation cost of the reservoir simulation. Again, we see that the adaptive time-stepping scheme is a fine balance between speed and accuracy.

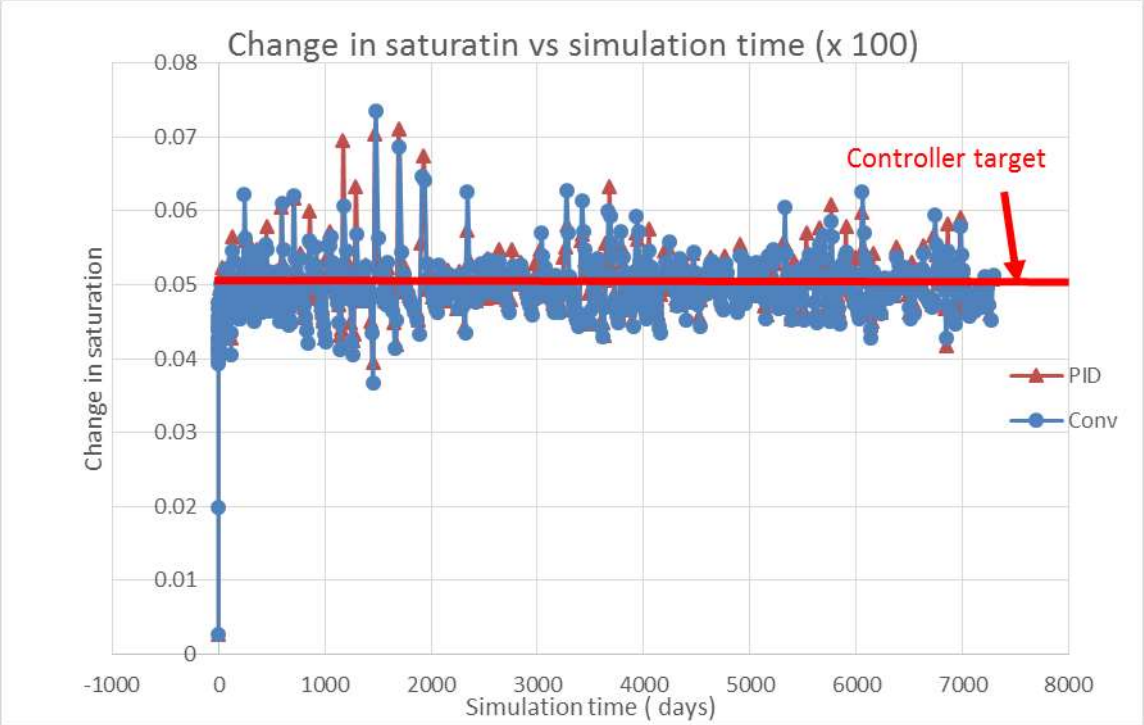

Figure 32: Saturation change vs simulation time (perm x100)

4.4.8 Effect of well location

In this section, we change the well locations from the base case and record the performance of the reservoir simulator. The injector location remains the same. However, the location of the producer wells are change to (12, 20) and (32, 20).

In Table 11, we summarize the performance of the PID controller and compare it to the basic controller. The results show that the PID controller has a lower computation time, total number of N.R iterations and time-steps. From Figure 34, we can observe that the time-steps of the PID controller are consistently larger than the ones of the basic controller. We also note sharper variation in the time-step response of the PID controller than the basic controller. Notably, after about 68 iterations, the PID controller has a sharp drop in time-step whereas the basic only exhibit a reduction in the rate of increase of the time-step. The PID controller has a faster response than the basic controller, but at the same time this aggressive response increases the chances of overshoots during the simulation.

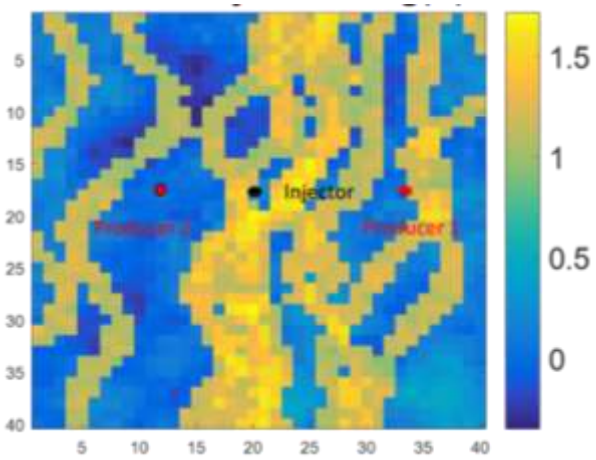

Figure 33: Permeability data with new well location. The image represents log (k) where k is in millidarcies

simulation time (years)	1	2	3	4	5	20
Conventional method						
Computation time (min)	0.38	0.438	0.47	0.486	0.501	0.658
Total number of time steps	45	50	54	56	58	73
Number of NR iterations	193	218	238	248	258	329
Ratio NR/total time steps	4.288889	4.36	4.407407	4.428571	4.448276	4.506849
PID						
Computation time (min)	0.366	0.412	0.454	0.488	0.4921	0.618
Total number of time steps	43	48	51	54	55	70
Number of NR iterations	186	211	226	241	246	318
Ratio NR/total time steps	4.325581	4.395833	4.431373	4.462963	4.472727	4.542857
Comparison						
% reduction in NR iteration	3.63%	3.21%	5.04%	2.82%	4.65%	3.34%
% reduction in time steps	4.44%	4.00%	5.56%	3.57%	5.17%	4.11%
% comp time reduction	3.68%	5.94%	3.40%	-0.41%	1.78%	6.08%

Table 11: Performance of PID and basic controller with different well location

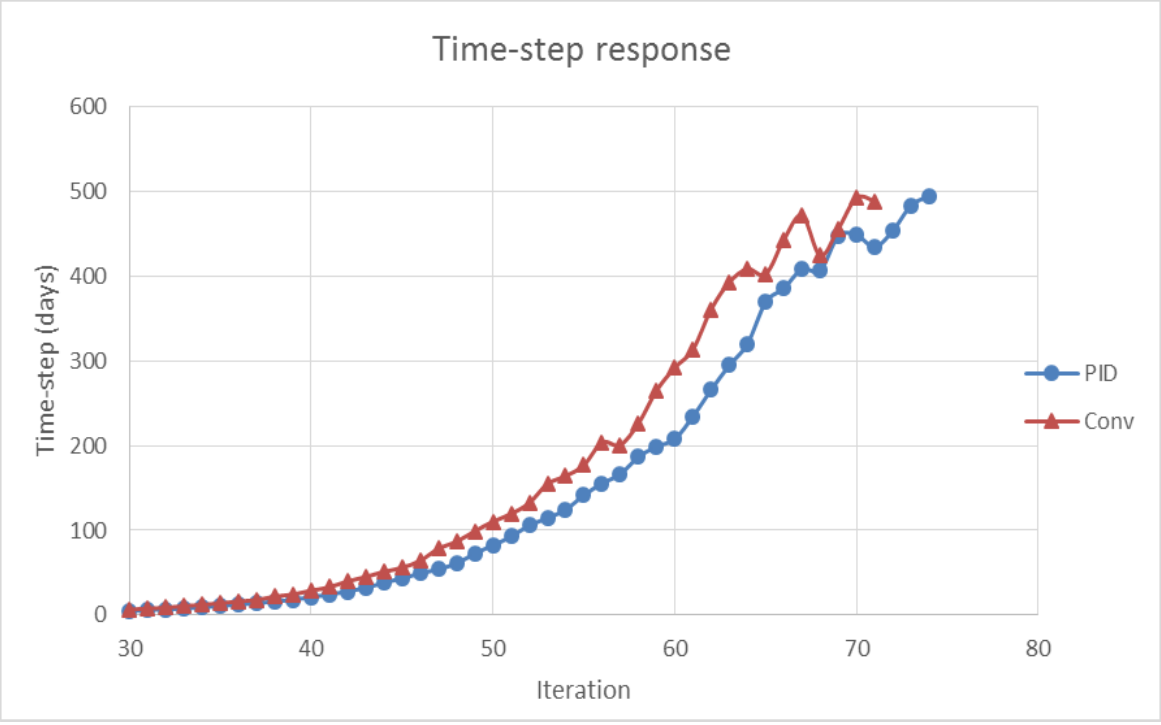

Figure 34: Time-step response of PID with different well location

4.5 Error control algorithm

We test the PID controller to adjust the time-step based on the error generated during the Newton-Raphson iteration. In case a solution is not within the defined tolerance, the PID controller would adjust the time-step such that the next step has a better chance of convergence to a solution within tolerance. In order to test this controller, we had to tighten the requirement for convergence to force steps to be rejected. As a result, we lower the maximum number of N.R iteration per time-step from 15 to 3. This change forces the system to reject most of the iterations and use the error control PID algorithm. The basic controller in this case consist of cutting the value of the previous time-step by half until the solution is within tolerance. We compare this algorithm with a PID controller using the characteristic equation 3.10 described in chapter 3.We were not able to obtain meaningful results with this algorithm as the system never converged. In fact, as we see from the equation below, the characteristic equation of the PID controller involves the error of previous iterations. During the Newton-Raphson iteration, the error of previous iteration is higher than current one as the scheme reduces the error in the solution. If the previous error is higher than the current one, then the proportional term would be higher than 1. In fact, $\left(\frac{e_{n-1}}{e_n}\right)^{K_p} > 1 \; if \; e_{n-1} > e_n$. Similarly, the derivative term could also be greater than one if $e_n e_{n-2} > e_{n-1}^2$. As a result, depending on the PID coefficient, the algorithm of the time-step after a rejected step could lead to an increase in the time-step that would reduce even further the chances of convergence of the reservoir simulator. As a result, this technique is not appropriate to control within the Newton-Raphson iteration loop.

CHAPTER V

CONCLUSION AND FUTURE WORK

In conclusion, we can say that the PID resolution algorithm is an effective adaptive time-stepping technique. In fact, it has proven to be a reliable and robust technique with superior performance when the proper tuning is done. The PID controller was tested using a broad range of control parameters and initial conditions. The results show that the PID controller can reduce the computational cost of the simulation while maintaining the same level of accuracy as the basic method. The tuning of the PID controller is the most important step in the implementation of this adaptive time-stepping method. A good tuning can result in an optimal performance of the simulator, but at the same time, if the tuning is not performed properly, the controller can become inefficient and lead to instabilities in the reservoir simulator. There are automatic tuning algorithm, but they do not work well for adaptive time-step purposes. Future work would have to focus on designing automatic tuning algorithms specifically tailored to work for adaptive time-stepping purposes to avoid time-consuming manual tuning.

The test of the PID error control algorithm showed that the PID controller is not suited to control the error within the Newton-Raphson loop after a step is rejected. In fact, the PID controller created instabilities in the reservoir simulator and we were not able to run the simulation efficiently. As a result, we do not recommend the use of this controller in the settings described in this study.

5.1 Future work

We were able to identify areas of improvement and topic of research that will help learn more about the PID controller and assess its effectiveness. We noted potential future work in the topics mentioned below:

1. The resolution controller should be tested on a larger scale reservoir simulator. This test will allow us to further assess the robustness of the controller at adjusting the time-step during the time discretization process.

2. We also suggest investigating the effectiveness of the controller at controlling the convergence of the Newton-Raphson iteration. The PID controller could be tested for adjusting the time-step for the purpose of controlling the convergence and preventing rejected steps. This convergence control could potentially help reduce the computational cost of the simulation.

3. Future work will also have to include automatic PID controller tuning algorithm to avoid time-consuming manual tuning. It will also be interesting to look into the ability of the controller to self-adjust during the simulation in order to optimize its performance.

REFERENCES

[1] Aziz, K., & Settari, A. (n.d.). Petroleum reservoir simulation.

[2] Bender, J., Erleben, K., & Teschner, M. (2010). Boundary handling and adaptive time-stepping for PCISPH. In *Workshop on virtual reality interaction and physical simulation VRIPHYS*.

[3] Bruce, W. (1943). Pressure Prediction for Oil Reservoirs. *Transactions of the AIME, 151*(01), 73-85.

[4] Courant, R., & John, F. (1989). *Introduction to Calculus and Analysis : Volume I. by Richard Courant, Fritz John*. New York, NY : Springer New York, 1989.

[5] Eidson, B. (2010). *An Experimental Evaluation of the Delta Operator in Digital Control* (Unpublished master's thesis). Auburn University.

[6] Edwards, D., Gunasekera, D., Morris, J., & Shaw, G. (2012). Reservoir simulation: Keeping pace with Oilfield complexity. *Oilfield Review Winter, 23*(4). Retrieved from http://www.slb.com/~/media/Files/resources/oilfield_review/ors11/win11/01_reservoir_sim.pdf

[7] *Effect of tuning the PID controller*. (n.d.). In *Http://www.ipsenharold.com/*. Retrieved June 12, 2016.

[8] Geiser, J., & Fleck, C. (2009). Adaptive Step-Size Control in Simulation of Diffusive CVD Processes. *Mathematical Problems in Engineering, 2009*, 34-34. doi:10.1155/2009/728105

[9] Gentle, J. E. (1998). *Numerical Linear Algebra for Applications in Statistics. by James E. Gentle*. New York, NY : Springer New York, 1998.

[10] Gustafson, K. (1991). Control theoretic for stepsize selection in explicit Runge-Kutta methods. *ACM Transactions on Mathematical Software, 17*(4), 533-554. doi:10.1145/210232.210242

[11] Gustafsson, K.)., Lundh, M.)., & Söderlind, G.). (1988). A PI stepsize control for the numerical solution of ordinary differential equations. *Bit, 28*(2), 270-287. doi:10.1007/BF01934091

[12] Gustafson, K., & Soderlind, G. (1997). Control strategies for the Iterative Solution of Nonlinear Equations in ODE Solvers. *SIAM Journal on Scientific Computing, 18*(1), 23-40. doi:10.1137/S1064827595287109

[13] Ilie, S.)., & Morshed, M.). (2015). Adaptive time-stepping using control theory for the chemical langevin equation. *Journal Of Applied Mathematics, 2015*doi:10.1155/2015/567275

[14] Jensen, O. (1980). An Automatic Timestep Selection For Reservoir Simulation. In *SPE annual Technical Conference and Exhibition* (pp. 21-24). Dallas, TX: Society of Petroleum engineers.

[15] Johnson, M. A., Moradi, M. H., & Crowe, J. (2005). PID control: new identification and design methods. Michael A. Johnson and Mohammad H. Moradi (editors) ; with J. Crowe [and others]. New York : Springer, 2005.

[16] Li, C., & Zeng, F. (2015). *Numerical methods for fractional calculus. Changpin Li, Fanhai Zeng*. Boca Raton : CRC Press, [2015].

[17] MATLAB & SIMULINK student version (Version R2015a.Ink) [Computer software]. (2015). Natick, MA: Mathworks.

[18] Nolen, J. S. (1973, January 1). Numerical Simulation Of Compositional Phenomena In Petroleum Reservoirs. Society of Petroleum Engineers. doi:10.2118/4274-MS

[19] O'Dwyer, A. (2009). *Handbook of PI and PID controller tuning rules. 3rd ed. Aidan O'Dwyer*. London : Imperial College Press : Disbributed in the UK by World Scientific ;, 2009.

[20] Osako, I. (2003). *Timestep Selection during streamline Simulation via Transverse Flux Correction* (Unpublished master's thesis). Texas A&M.

[21] Soderlind, G. (2003). Digital Filters in Adaptive Time-Stepping. *ACM Transactions On Mathematical Software, 29*(1), 1.

[22] Söderlind, G., & Wang, L. (2006). Adaptive time-stepping and computational stability. *Journal Of Computational And Applied Mathematics,185*(Special Issue: International Workshop on the Technological Aspects of Mathematics), 225-243. doi:10.1016/j.cam.2005.03.008

[23] Thomee, V. (2001). From finite differences to finite elements: A short history of numerical analysis of partial differential equations. *Journal of Computational and Applied Mathematics, 128*(1-2), 1-54. doi:10.1016/S0377-0427(00)00507-0

[24] Valli, A., Carey, G., & Coutinho, A. (2002). Control strategies for timestep selection in simulation of coupled viscous and heat transfer. *Communications in Numerical Methods in Engineering, 18*(2), 131-139. doi:10.1002/cnm.475

[25] Yu, C. (1999). *Autotuning of PID Controllers : Relay Feedback Approach. by Cheng-Ching Yu*. London : Springer London, 1999.