İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

A COMPARISON OF LANCZOS DECOMPOSITION METHOD (LDM) AND CONVENTIONAL IMPLICIT TIME-STEPPING METHOD (ITSM) FOR NUMERICAL RESERVOIR SIMULATION

M.Sc. Thesis by
Koray YILMAZ

Department : Petroleum and Natural Gas Engineering

Programme : Petroleum and Natural Gas Engineering

Thesis Supervisor: Prof. Dr. Mustafa ONUR

JUNE 2009

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

A COMPARISON OF LANCZOS DECOMPOSITION METHOD (LDM) AND CONVENTIONAL IMPLICIT TIME-STEPPING METHOD (ITSM) FOR NUMERICAL RESERVOIR SIMULATION

M.Sc. Thesis by
Koray YILMAZ
(505051506)

Date of submission : **4 May 2009**
Date of defence examination: **3 June 2009**

Supervisor (Chairman) : **Prof. Dr. Mustafa Onur (ITU)**
Members of the Examining Committee : **Prof. Dr. Altuğ Şişman (ITU)**
Assis. Prof. Dr. Ö. İnanç Türeyen (ITU)

JUNE 2009

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

SAYISAL REZERVUAR SİMÜLASYONU İÇİN LANCZOS AYRIŞIM (LDM) YÖNTEMİ İLE GELENEKSEL ÖRTÜK ZAMAN ADIMLI (ITSM) YÖNTEMİN KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ
Koray YILMAZ
(505051506)

Tezin Enstitüye Verildiği Tarih : 04 Mayıs 2009
Tezin Savunulduğu Tarih : 03 Haziran 2009

Tez Danışmanı : Prof. Dr. Mustafa ONUR (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Altuğ Şişman (İTÜ)
Yrd. Doç. Dr. Ö. İnanç Türeyen (İTÜ)

HAZİRAN 2009

FOREWORD

I would like to thank my advisor, Prof. Dr. Mustafa Onur for his contribution to this thesis. I also would like to express my appreciations to my committee members, Prof. Dr. Altuğ Şişman and Assis. Prof. Dr. Ömer İnanç Türeyen for their help and suggestions.

Most importantly, I would like to express my thankfulness to my family and my friends for their constant support and love.

May 2009

Koray Yılmaz

Petroleum and Natural Gas Engineering

TABLE OF CONTENTS

Page

SYMBOLS ..... **viii**
LIST OF TABLES ..... **x**
LIST OF FIGURES ..... **xi**
SUMMARY ..... **xiii**
ÖZET ..... **xv**
1. INTRODUCTION **1**
1.1 Purpose of the Thesis 2
1.2 Scope of the Thesis 3
2. MATHEMATICAL DEVELOPMENT **5**
2.1 Continuity Equation In 2-D (*x*-*y*) Space 5
2.2 Construction of Structured Cartesian Grids 9
2.3 Initial and Boundary Conditions 11
2.4 Finite Difference Approximation To Two Dimensional Flow 12
2.4.1 Conventional implicit time-stepping method (ITSM) 13
2.4.2 Lanczos decomposition method (LDM) 20
3. SIMULATOR APPLICATIONS **29**
3.1 Verification of Simulators 30
3.1.1 Constant rate production case 31
3.1.2 Variable rate production case 41
4. CONCLUSIONS **45**
APPENDICES **49**
CURRICULUM VITA **51**

SYMBOLS

A	: Cross-Sectional Area, ft^2
B	: Formation Volume Factor, RB/STB
b	: Bulk
c_f	: Fluid Compressibility, psi^{-1}
c_r	: Rock Compressibility, psi^{-1}
c_t	: Total Compressibility, psi^{-1}
f	: Fluid
h	: Thickness of Reservoir Zone, ft
i	: Index Denoting x Direction Gridblock Number
j	: Index Denoting y Direction Gridblock Number
k	: Permeability, md
k_x	: Permeability in x Direction, md
k_y	: Permeability in y Direction, md
k_{effx}	: Effective Permeability in x Direction, md
k_{effy}	: Effective Permeability in y Direction, md
N_x	: Number of Gridblocks In x Direction
N_y	: Number of Gridblocks In Y Direction
N_w	: Total Number of Wells
n	: Last Time Step Taken
$n+1$	: Next Time Step Taken
p	: Pressure, psi
p_i	: Initial Pressure, psi
p_{wf}	: Bottom Hole Well Bore Flowing Pressure, psi
q	: Flow Rate, RB/D
r	: Rock
r_o	: Effective Well Block Radious, ft
r_w	: Well Radious, ft
S	: Skin Factor, Dimensionless
t	: Time, days
T_x	: Transmissiblity in x Direction, $cp.bbl.day^{-1}.psi^{-1}$
T_y	: Transmissiblity in y Direction, $cp.bbl.day^{-1}.psi^{-1}$
V	: Volume, ft^3
WI	: Well Index, (STB/D)/psi
x,y,z	: Space Dimensions, ft
$\underline{n}$	: Unit Outward Normal Vector
μ	: Viscosity, cp
Δ	: Difference Operator
Δp	: Pressure Drop, psi
Δx	: Gridblock Length in x Direction, ft
Δy	: Gridblock Length in y Direction, ft
Δt	: Length of Time Step

ϕ : Porosity, fraction
ρ : Density, lb_m/ ft^3
$\tilde{q}$: Well Flow Rate, STB/D
δ : Delta Function, ft^{-1}

LIST OF TABLES

Page

Table 3.1: Formation and fluid properties for validity 31
Table 3.2: Simulator using Lanczos decomposition vs. ECRIN software results (Number of gridblocks N_x = 11 and N_y = 11) 33
Table 3.3: Simulator using Lanczos decomposition vs. ECRIN software results (Number of gridblocks N_x = 21 and N_y = 21) 33
Table 3.4: Simulator using Lanczos decomposition vs. ECRIN software results (Number of gridblocks N_x = 51 and N_y = 51) 34
Table 3.5: Simulator using Lanczos decomposition vs. ECRIN software results (Number of gridblocks N_x = 67 and N_y = 67) 34
Table 3.6: CPU times from the simulators using Lanczos decomposition with full and sparse matrix storage of the matrix A. (Number of gridblocks N_x = 67 and N_y = 67) 35
Table 3.7: Simulator using ITSM vs. ECRIN software results (Number of gridblocks N_x = 11 and N_y = 11) 36
Table 3.8: Simulator using ITSM vs. ECRIN software results (Number of gridblocks N_x = 21 and N_y = 21) 36
Table 3.9: Simulator using ITSM vs. ECRIN software results (Number of gridblocks N_x = 51 and N_y = 51) 37
Table 3.10: Simulator using ITSM vs. ECRIN software results (Number of gridblocks N_x = 67 and N_y = 67) 37
Table 3.11: CPU times from the simulators using ITSM with full and sparse matrix storage of the matrix A. (Number of gridblocks N_x = 67 and N_y = 67) 38
Table 3.12: Comparing accuracy of solutions as a function of the total number of gridblocks used in simulation at 10.569 days..... 40
Table 3.13: Comparing CPU times from MATLAB and FORTRAN codes for ITSM and LDM for different gridblock numbers at t = 1000 days. 41

LIST OF FIGURES

Page

Figure 2.1 : Gridblock definition for a 2D *x-y* Cartesian coordinate system. 6
Figure 2.2 : An example of block-centered grid system for a cartesian *coordinate system (x-y).*.....10
Figure 2.3 : Coordinates for a block-centered grid.....10
Figure 2.4 : Definition of initial and boundary conditions in 2D reservoir (Dalton and Mattax, 1990).....12
Figure 2.5 : Ordering scheme used for a 2D reservoir simulation. 16
Figure 2.6 : Structure of the matrix A used in ITSM.....19
Figure 2.7 : Structure of the matrix A used in LDM 24
Figure 3.1: A schematic representation of reservoir/well system used for verification. 32
Figure 3.2. : Delta pressure and delta pressure derivative at the production well simulation results compared with ECRIN model results (21x21 and 67x67 gridblocks used for simulation) ……...39
Figure 3.3 : Flow rate history for variable production rate example……..42
Figure 3.4 : Comparison of flowing bottom-hole pressures computed from LDM, ITSM, and ECRIN, for variable production rate example. 43

A COMPARISON OF LANCZOS DECOMPOSITION METHOD (LDM) AND CONVENTIONAL IMPLICIT TIME-STEPPING METHOD (ITSM) FOR NUMERICAL RESERVOIR SIMULATION

SUMMARY

Reservoir simulations constitute a cornerstone to predict the flow of fluids through porous media. Various numerical models called simulators are developed to simulate performance of hydrocarbon reservoirs. These models are used in field development since production forecasts are necessary to make investment decisions. Nowadays, numerical simulators are widely used by reservoir engineers.

In this study, two new simulators were developed using Lanczos Decomposition Method (LDM) and Conventional Implicit Time-Stepping Method (ITSM). The study focuses on 2-D flow for slightly compressible fluid of constant viscosity with multiple wells. Derivation of the model equations was performed using continuity equation for both methods. The simulators were written using the MATLAB programming language. The simulators developed in this study are capable of assigning uniform and non uniform gridblock distribution; porosity and permeability distribution as well as developing various production and injection scenarios for single or multiple wells depending on different areas of application. Validity and accuracy of 2D flow simulator were examined by comparing simulation results with that of obtained from the commercial software called ECRIN. The results of the simulator were almost identical with the results obtained from the commercial software. During the model runs, the CPU time of the two simulators were compared. A special case was also studied for a single well with variable rate history using both ITSM and LDM written with FORTRAN.

SAYISAL REZERVUAR SİMÜLASYONU İÇİN LANCZOS AYRIŞIM (LDM) YÖNTEMİ İLE GELENEKSEL ÖRTÜK ZAMAN ADIMLI (ITSM) YÖNTEMİNİN KARŞILAŞTIRILMASI

ÖZET

Rezervuar simülasyonları gözenekli ortamdaki akışkan akışını modellemede önemli bir yer teşkil eder. Sayısal modellemeler hidrokarbon rezervuarlarının performansının belirlenmesinde kullanılmaktadır. İsimlendirme olarak simülator şeklinde adlandırılmaktadır. Bu modellemeler üretim verilerinden ve üretim tahminlerinden faydalanılarak üretim sahalarının geliştirilmesi için gerekli yatırım kararlarının oluşturulmasında önemli fayda sağlamaktadır. Günümüzde sayısal simulatörler rezervuar mühendisleri tarafından yaygın kullanıma sahip duruma gelmiştir.

Bu çalışmada, Sayısal Rezervuar Simülasyonu için Lanczos Ayrışım (LDM) Yöntemi ve Geleneksel Örtük Zaman Adımlı (ITSM) Yöntemleri rezervuar simülasyonunun geliştirilmesinde kullanılarak iki adet farklı simülator geliştirilmiştir. Her iki metod ile geliştirilen simülatörlerin sonuçları karşılaştırılmıştır. Geliştirilen simülatörler sabit sıkıştırılabilirlik ve akmazlığa sahip akışkan rezervuarlarında çoklu kuyu sistemiyle iki boyutlu akış sistemi için sayısal rezervuar simülasyonudurlar. Simülatörlerin matematiksel gelişimi süreklilik denkleminden çıkartılmış olup bu gelişim çalışmada detaylı olarak verilmektedir. Simülatörler, MATLAB programlama dili kullanılarak yazılmıştır. Her iki simülatörde değişken veya değişken olmayan hücre boyutları, farklı gözeneklilik ve geçirgenlik dağılımları ile birlikte çeşitli üretim ve enjeksiyon senaryoları ile tekil veya çoğul kuyu dizaynı kullanılarak farklı uygulamalarda kullanılabilecek fonksiyonlara sahiptir. Geliştirilen simülatörlerin sonuçlarının doğrulaması ticari yazılım ile üretilen sentetik veriler ile karşılaştırılarak yapılmıştır. Simülatör sonuçları ile ticari yazılım sonuçları uygunluk sağlamaktadır. Uygulama örneklemelerinde her iki simulatörün çalışma zamanları karşılaştırılarak hesaplanan sonuçların ticari yazılım sonuçları ile yakınsamaları da karşılaştırılmıştır. Ayrıca, değişken debili üretim tarihçesi kullanılarak örneklenen simülator sonuçları da çalışmada sunulmuştur.

1. INTRODUCTION

Reservoir numerical simulation is the modeling of the physical phenomena for the understanding of reservoir processes, which may be quite complex as it requires integration of knowledge from various disciplines such as mathematics, physics, computer and reservoir engineering. The main objective of reservoir simulation is to predict the performance of the gas or oil reservoir behavior under different operating conditions (Ertekin et.al., 2001).

The date for the reservoir simulation modeling goes back to 1940, since then it has been applied by the companies to the various reservoirs to predict the future reservoir production under a series of potential scenarios, such as drilling new wells, injecting various fluids or stimulation (Dalton and Mattax, 1990).

Reservoir simulations give the imaginary pictures of what a reservoir looks like under the surface of the earth. The more accurate the picture, the easier the engineer can get the product out of the ground and so then, the more profitable the well will be (Crichlow, 1977).

The reservoir simulation is crucial for an oil/gas field management. A large amount of capital cost may be necessary for the development of a field, therefore forecasting the oil and gas resources helps the investor to decide whether to develop a field or not and also to optimize the cost. For these reasons, reservoir simulation studies have been used for decades to better decide a field development.

Reservoir simulation studies are based on growing dataset with more complex physics and detailed models. The data used in the simulator depend on the characteristic and hydrocarbon potential of the reservoir. As the variation of the parameters in the simulator increases, more cells are needed to represent the properties across the model. An increase in the number of cells in the model results in numerical difficulties and the performance problems such as grooving of the process and complexity arise (Onur, 1997).

Following the development of the simulator, verification needs to be done using either analytical solutions or the results of another verified simulator to prove that the simulator runs accurately and can be used for field applications. Validation of the simulator, which is the last step in developing a reservoir simulator, is a key process to make sure that no errors have been made during the development of the simulator.

An issue of running a simulator is to achieve an acceptable CPU time. Depending on the developing hardware technologies, the challenge is to improve the efficiency of reservoir simulation software on a large number of processors.

As is well known [see for example, Ertekin et al. (2001)], the implicit time-stepping method is the most commonly used method for reservoir simulation as it does not suffer from stability problems for single-phase flow problems. The method is based on the discretization of both spatial and time derivatives in partial differential equations to be solved for reservoir simulation.

However, in recent years, a few studies have proposed the use of the Lanczos decomposition method in reservoir simulation studies (Druskin and Knizhnerman, 1994a; Knizhnerman, et al., 1994; Druskin and Knizhnerman; 1998; Alpak et al., 2003). The attractiveness of this method appears to be the avoiding of time stepping in simulation and allows directly the computation of reservoir pressures at a given time as the method uses a semi-discrete formulation of the partial differential equations governing fluid flow in porous media. As in the cited works considered the use of the spectral Lanczos method, in this thesis, we consider single-phase flow of slightly compressible fluid of constant viscosity. However, unlike the works of Knizhnerman et al. (1994) and Alpak et al. (2003), who have applied the Lanczos method for 2D r-z flow of a single-well slightly compressible fluid of constant compressibility and viscosity, we consider 2D x-y flow of a slightly compressible fluid of constant compressibility and constant viscosity. Our formulation given here is quite general and can be used for multi-well systems as well as variable flow rate histories at the wells.

1.1 Purpose of the Thesis

To date, in the petroleum engineering literature, to the best of the author's knowledge, there is no work published that compares the performances (in terms of computational aspects as well as CPU times) of the Lanczos method and the conventional implicit-time

stepping method. Therefore, the main objective of this work is to provide a comparison of the spectral Lanczos and implicit time-stepping methods for 2D x-y flow of a slightly compressible fluid of constant viscosity and compressibility in a closed rectangular reservoir.

1.2 Scope of the Thesis

The thesis consists of four chapters including this introduction chapter. In Chapter II, formulations for both conventional implicit time-stepping and Lanczos methods for a 2D x-y simulation of slightly compressible fluid of constant viscosity and compressibility in a closed rectangular reservoir with multiple wells are provided. In Chapter III, we verify the simulators developed using both implicit time stepping and Lanczos methods by comparing the results obtained from our simulators by the results of the analytical solutions given in ECRIN commercial well-test software for various cases. In addition, the results of the two methods were compared in terms of CPU times and efficiencies of each solution method in Chapter III. Conclusions and recommendations are given in Chapter IV.

2. MATHEMATICAL DEVELOPMENT

Mathematical models refer to physical processes such as the flow of fluids in porous media. Isothermal fluid flow in a reservoir can be characterized with partial differential equations (PDEs) based on conservation of mass, which are converted into a numerical model by using finite difference, integrated finite difference, finite volume, or finite element approaches. Most reservoir simulators are based on the finite difference or integrated finite difference approaches as these approaches are much easier to implement than are finite element methods.

The PDEs derived during the formulation step, if solved analytically, would give reservoir pressure, fluid saturations (in case multi-phase flow), and well flow rates as continuous functions of space and time. Because of the highly nonlinear nature of the PDEs and the need to handle heterogeneities in rock property fields (e.g., nonuniform permeability and porosity), analytical techniques are difficult, if not impossible, to solve.

In this section, the development of the mathematical model for a 2-D x-y linear flow of slightly compressible fluid of constant viscosity and constant compressibility with multiple wells is presented.

2.1 Continuity Equation In 2-D (*x*-*y*) Space

We would like to solve the following mass balance equation, which is obtained from application of the law of conservation of mass and divergence theorem:

$$-\int_{\Omega}(\frac{\underline{v}}{B}.\underline{n})d\Omega-\int_{V_b}\tilde{q}_{i,j}dV=\frac{1}{5.615}\int_{V_b}\frac{\partial}{\partial t}\left(\frac{\phi}{B}\right)_{i,j}dV \qquad \textbf{(2.1)}$$

where $\underline{v}$ is the vector of fluid velocity in RB/ft^2-day, B is formation volume factor in RB/STB, Ω represents the boundary area of the volume in ft^2, $\underline{n}$ represents the unit outward normal vector to dΩ, $\tilde{q}$ represents the source/sink terms (in unit of STB/D per unit volume of the reservoir) which can be a function of space and time. For an injection

(source) well, $\tilde{q}$ is negative, and for a production (sink) well, $\tilde{q}$ is positive. ϕ represents the porosity (unitless). V_b represents the bulk volume (pore volume + solid volume) of the system.

As we consider 2D x-y flow, the velocity vector $\underline{v}$, has two components v_x and v_y. A grid block for a cartesian coordinate system is given in Figure 2.1. The gridblock be defined with i and j "dummy" index. The center of a gridblock in the system can be indicated as (x_i, y_j) as shown in Figure 2.1:

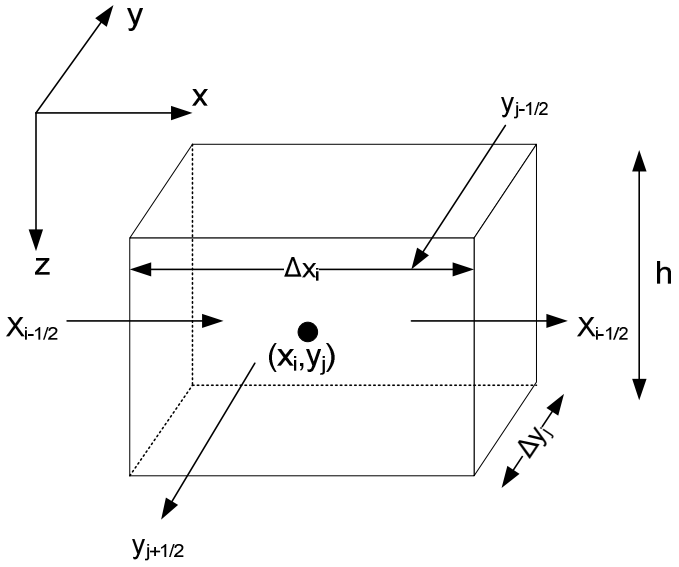


Figure 2.1 : Gridblock definition for a 2D *x-y* Cartesian coordinate system.

Bulk volume $(V_b)_{i,j}$ of each gridblock defined in a Cartesian coordinate system (x,y) is calculated from the gross thickness h of each gridblock and the gridblock lengths Δx, Δy along the x and y axes as follows:

$$\left(V_b\right)_{i,j} = \Delta x_i \Delta y_j h \tag{2.2}$$

Multiply and divide eq. 2.1 by volume of the gridblock $(V_b)_{i,j}$

$$-\int_{\Omega} (\frac{\underline{v}}{B}.\underline{n})\, d\Omega - (V_b)_{i,j} \frac{1}{(V_b)_{i,j}} \int_{V_b} \tilde{q}_{i,j}\, dV = \frac{(V_b)_{i,j}}{5.615}\left[\frac{1}{V_b}\int_{V_b} \frac{\partial}{\partial t}(\frac{\phi}{B})_{i,j}\, dV\right] \tag{2.3}$$

Note that:

$$\frac{1}{V_b}\int_{V_b}\frac{\partial}{\partial t}\frac{\phi}{B}dV=\frac{\partial}{\partial t}\left(\frac{\phi}{B}\right) \tag{2.4}$$

which represents the volumetric average of the time derivative of ϕ/B over the bulk volume V_b. Since ϕ and B are not functions of time, the derivation term is taken out of the integral as shown in Eq. 2.5.

$$\frac{1}{V_b}\int_{V_b}\frac{\partial}{\partial t}\left(\frac{\phi}{B}\right)_{i,j}dV=\frac{\partial}{\partial t}\left(\frac{\phi}{B}\right)_{i,j} \tag{2.5}$$

Similarly the sink/source term in Eq. 2.3 for the gridblock (i,j) is defined as,

$$\frac{(V_b)_{i,j}}{5.615}\left[\frac{1}{V_b}\int_{V_b}\tilde{q}_{i,j}dV\right]=\tilde{q}_{i,j}\Delta x_i\Delta y_j h \tag{2.6}$$

Note that Eq. 2.6 actually represents $q_{i,j}$, the volumetric flow rate in STB/D and positive for a production well located in gridblock, i.e.,

$$V_b\left[\frac{1}{V_b}\int_{V_b}\tilde{q}dV\right]=\tilde{q}_{i,j}\Delta x_i\Delta y_j h=q_{i,j} \tag{2.7}$$

The time derivate in the right-hand-side (RHS) of Eq. 2.4, by using chain rule and product rule for derivates can be written as,

$$\frac{\partial}{\partial t}\left(\frac{\phi}{B}\right)=\frac{\phi}{B}\left[\frac{1}{\phi}\frac{d\phi}{dp}+B\frac{d(1/B)}{dp}\right]\frac{\partial p}{\partial t} \tag{2.8}$$

Using the definitions of isothermal compressibilities of pore and fluid; which are given by Eqs. 2.9 and 2.10,

$$c_f=B\frac{d(1/B)}{dp}=-\frac{1}{B}\frac{dB}{dp} \tag{2.9}$$

and

$$c_r = \frac{1}{\phi}\frac{d(\phi)}{dp} \tag{2.10}$$

we can express Eq. 2.8 as:

$$\frac{\partial}{\partial t}\left(\frac{\phi}{B}\right) = \frac{\phi c_t}{B}\frac{\partial p}{\partial t} \tag{2.11}$$

where c_t is defined as the total compressibility of the system (rock + fluid),

$$c_t = c_r + c_f \tag{2.12}$$

and will be treated as constant due to our assumption of slightly compressible fluid. Note also that for liquid flow, the change of B (or equivalently the density of liquid) with pressure is small and hence, B can be treated as constant.

Now, we focus on expressing the first term on the left-hand-side of Eq. 2.3 for the Cartesian gridblock shown in Figure 2.1. Note that there are four faces and the unit outward normal vectors for these faces. For the gridblock shown in Figure 2.1, this term is expressed for 2-D x-y linear flow as given in Equation 2.13:

$$\begin{aligned}\int_{\Omega}\left(\frac{\underline{\underline{v}}}{B}.\underline{n}\right)d\Omega &= \sum_{l=1}^{4}\int_{\Omega_l}\left(\frac{\underline{\underline{v}}}{B}.\underline{n}_l\right)d\Omega_l = \sum_{l=1}^{4}\Omega_l\left[\frac{1}{\Omega_l}\int_{\Omega_l}\left(\frac{\underline{\underline{v}}}{B}.\underline{n}_l\right)d\Omega_l\right] \\ &= \left[\left(\frac{v_x}{B}\right)_{i+1/2,j}\Delta y_j h - \left(\frac{v_x}{B}\right)_{i-1/2,j}\Delta y_j h + \left(\frac{v_y}{B}\right)_{i,j+1/2}\Delta x_i h - \left(\frac{v_y}{B}\right)_{i,j-1/2}\Delta x_i h\right]\end{aligned} \tag{2.13}$$

Here, as mentioned previously, v_x, and v_y are the components of $\underline{\underline{v}}$ in x and y directions, respectively and given by Darcy's equation:

$$v_x = -1.127\times10^{-3}\frac{k_x}{\mu}\frac{\partial p}{\partial x} \tag{2.14}$$

$$v_y = -1.127\times10^{-3}\frac{k_y}{\mu}\frac{\partial p}{\partial y} \tag{2.15}$$

where k_x and k_y denote the permeabilities in the x and y directions, respectively and these permeabilities need not be taken as equal to model anisotropy in permeability. It is

also worth noting that throughout this thesis, we will neglect the gravity effects as we assume a horizontal rectangular reservoir.

Using Eqs. 2.2, 2.7, 2.11, 2.13, 2.14 and 2.15 in Eq. 2.3 gives:

$$
\begin{aligned}
&-c_1\left[\left(-\frac{k_x}{\mu B}\frac{\partial p}{\partial x}\right)_{i+1/2,j}\Delta y_j h-\left(-\frac{k_x}{\mu B}\frac{\partial p}{\partial x}\right)_{i-1/2,j}\Delta y_j h\right]\\
&-c_1\left[\left(-\frac{k_y}{\mu B}\frac{\partial p}{\partial y}\right)_{i,j+1/2}\Delta x_i h-\left(-\frac{k_y}{\mu B}\frac{\partial p}{\partial x}\right)_{i,j-1/2}\Delta x_i h\right]\\
&-q_{i,j}=\frac{\Delta x_i\Delta y_j h}{c_2}\left(\frac{\phi c_t}{B}\frac{\partial p}{\partial t}\right)_{i,j}
\end{aligned}
\qquad \textbf{(2.16)}
$$

where $c_1 = 1.127\times10^{-3}$ and $c_2 = 5.615$.

It should be noted that if we divide the reservoir into N_x gridblocks in the x-direction and N_y gridblocks in the y-direction, then we will have a total of N_xN_y gridblocks (cells) in the reservoir and Eq. 2.16 should be applied for each of these gridblocks. Next, we discuss the construction of grid system to be used for approximating Eq. 2.16 to solve pressures.

2.2 Construction of Structured Cartesian Grids

The purpose of the grid system is to divide the reservoir into blocks to which the representative rock properties can be assigned. Historically, block-centered finite differences methodology has been used to develop reservoir simulators. In a block-centered grid, gridblocks with known dimensions are superimposed over the reservoir. For a rectangular coordinate system, the gridpoints are defined as the centers of these gridblocks. It is necessary to introduce gridblock construction for characterized fluid flow of the model. In this study, the block-centered grid system in Cartesian coordinates is used.

As mentioned previously, we define N_x and N_y to set up the gridblocks in the x and y directions respectively. Since only 2-D x-y simulation is considered in this study, each gridblock having thickness equal to formation thickness, h, is used (Aziz, 1979). The type of grid shape can be considered uniform or non uniform in any directions. Typical 2D, nonuniform, rectangular block-centered finite difference grid is shown in Figure 2.2. Also, coordinate axis for block centered grid is illustrated in Figure 2.3.

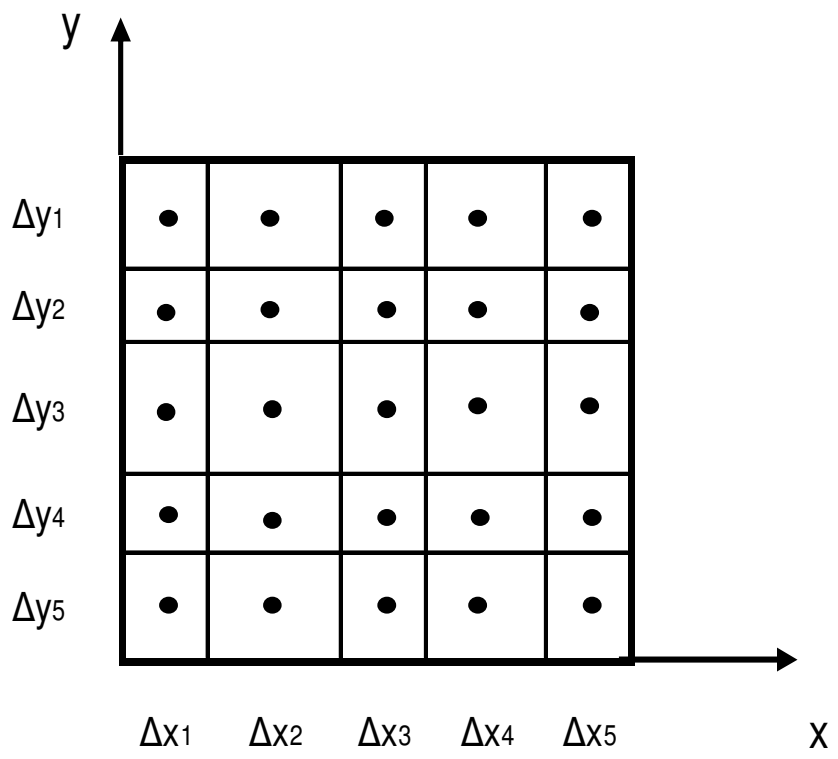


Figure 2.2 : An example of block-centered grid system for a cartesian coordinate system (x-y).

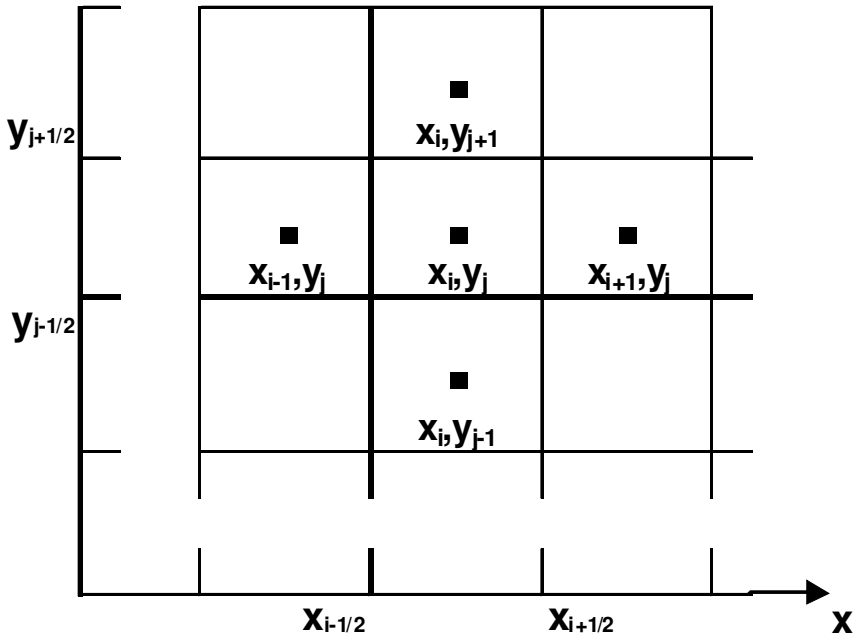


Figure 2.3 : Coordinates for a block-centered grid.

We let $x_{1/2}$ =0 and $x_{Nx+1/2}$ =L_x and similarly $y_{1/2}$ =0 and $y_{Ny+1/2}$ =L_y to denote the boundaries of the reservoir. L_x is the total length of the reservoir in the x-direction and L_y is the total length of the reservoir in the y-direction. The center of each grid block at any directions is defined by

$$x_i = \frac{x_{i-1/2} + x_{i+1/2}}{2}, (i = 1,2,.....,N_x) \quad \textbf{(2.17)}$$

$$y_i = \frac{y_{j-1/2} + y_{j+1/2}}{2}, (j = 1,2,.....,N_y) \quad \textbf{(2.18)}$$

The sums of the lengths of gridblocks in the *x* and *y* directions should sum up to L_x and L_y, and L_y, respectively:

$$\sum_{i=1}^{N_x} \Delta x_i = L_x \quad \textbf{(2.19)}$$

$$\sum_{y=1}^{N_y} \Delta y_j = L_y \quad \textbf{(2.20)}$$

2.3 Initial and Boundary Conditions

In order to describe a physical problem, a set of initial conditions must be specified. The well-known approach is to assume that the initial reservoir pressure is constant at a reference time, t= 0.

The initial condition (IC) used is given by:

$$p(x, y, t = 0) = p_0 = \text{constant} \quad \textbf{(2.21)}$$

Other conditions that must be specified are boundary conditions. The form of the flow equation (Eq. 2.16) represented above applies to block central gridblocks, and all the grids at the boundaries of the reservoir model have boundary conditions. Here, we have the most commonly encountered boundary condition which is no-flow (or also called Neumann) boundary conditions applied along the model (see Figure 2.4). The model with no-flow boundaries considers the reservoir boundary sealed for the flow.

The Boundary Conditions (BC) are:

$$\left(\frac{\partial p}{\partial x}\right)_{x=0,y} = \left(\frac{\partial p}{\partial x}\right)_{x=L_x,y} = \left(\frac{\partial p}{\partial y}\right)_{x,y=0} = \left(\frac{\partial p}{\partial y}\right)_{x,y=L_y} = 0 \qquad \textbf{(2.22)}$$

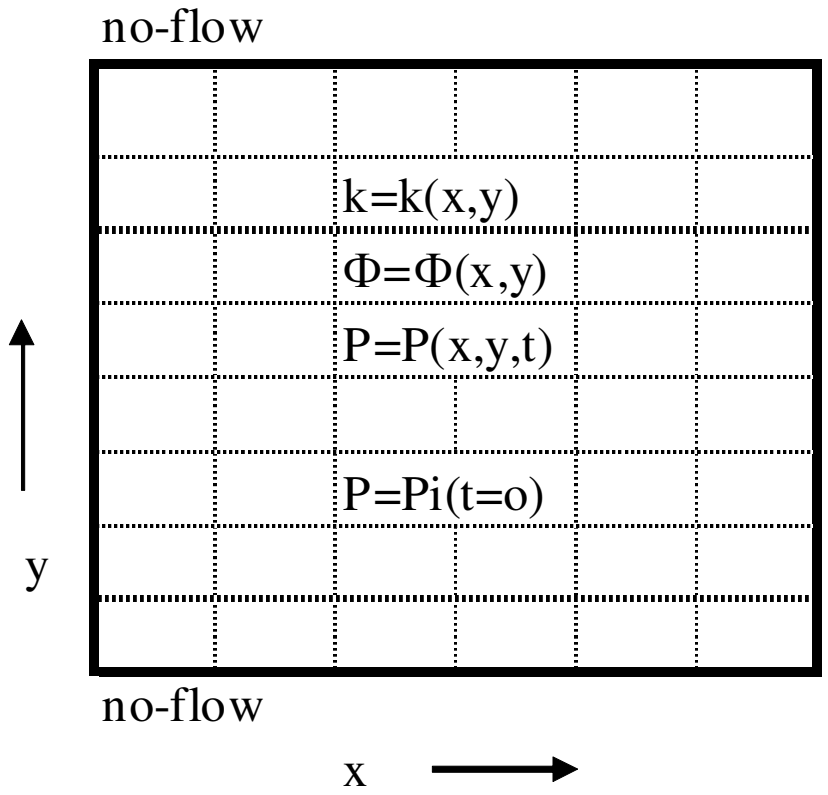


Figure 2.4 : Definition of initial and boundary conditions in 2D reservoir (Dalton and Mattax, 1990).

2.4 Finite Difference Approximation To Two Dimensional Flow

The finite difference method begins with the discretization of space (as discussed in Section 2.2) and time. When we consider a time stepping method, we discretize the time axis as: $\Delta t^{n+1}=t^{n+1}-t^n$ for n=0,1,2,….,t-1,t where t is an integer. In the next two subsections, we give basic formulations of implicit time-stepping method (ITSM) and Lanczos decomposition method (LDM).

2.4.1 Conventional implicit time-stepping method (ITSM)

Here, we explain the simulation for simulation of single-phase flow of slightly compressible fluid of constant viscosity by using Conventional Implicit Time-Stepping method. This method assumes that the parameters in the flow equation are constant for each time step. Besides, the pressures at the new time level are unknown. In the scheme illustrated below, we have to begin with the finite difference formulation of Eq. 2.16 by approximating partial derivatives of pressure in the x and y derivatives at block boundaries defined by $(x_{i\pm1/2}, y_j)$ and $(x_i, y_{j\pm1/2})$ at the time level t^{n+1},

$$\left(\frac{k_x}{\mu B}\frac{\partial p}{\partial x}\right)^{n+1}_{i-1/2,j} = \left(\frac{k_x}{\mu B}\right)_{i-1/2,j}\left(\frac{\partial p}{\partial x}\right)^{n+1}_{i-1/2,j} \approx 2\left(\frac{k_x}{\mu B}\right)_{i-1/2,j}\frac{\left(p^{n+1}_{i,j}-p^{n+1}_{i-1,j}\right)}{\left(\Delta x_i+\Delta x_{i-1}\right)} \tag{2.23}$$

$$\left(\frac{k_x}{\mu B}\frac{\partial p}{\partial x}\right)^{n+1}_{i+1/2,j} = \left(\frac{k_x}{\mu B}\right)_{i-1/2,j}\left(\frac{\partial p}{\partial x}\right)^{n+1}_{i+1/2,j} \approx 2\left(\frac{k_x}{\mu B}\right)_{i+1/2,j}\frac{\left(p^{n+1}_{i+1,j}-p^{n+1}_{i,j}\right)}{\left(\Delta x_i+\Delta x_{i+1}\right)} \tag{2.24}$$

$$\left(\frac{k_y}{\mu B}\frac{\partial p}{\partial y}\right)^{n+1}_{i,j-1/2} = \left(\frac{k_y}{\mu B}\right)_{i,j-1/2}\left(\frac{\partial p}{\partial y}\right)^{n+1}_{i,j-1/2} \approx 2\left(\frac{k_y}{\mu B}\right)_{i,j-1/2}\frac{\left(p^{n+1}_{i,j}-p^{n+1}_{i,j-1}\right)}{\left(\Delta y_j+\Delta y_{j-1}\right)} \tag{2.25}$$

$$\left(\frac{k_y}{\mu B}\frac{\partial p}{\partial y}\right)^{n+1}_{i,j+1/2} = \left(\frac{k_y}{\mu B}\right)_{i,j+1/2}\left(\frac{\partial p}{\partial y}\right)^{n+1}_{,i,j+1/2} \approx 2\left(\frac{k_y}{\mu B}\right)_{i,j+1/2}\frac{\left(p^{n+1}_{i,j+1}-p^{n+1}_{i,j}\right)}{\left(\Delta y_j+\Delta y_{j+1}\right)} \tag{2.26}$$

The time axis is divided into consecutive points as defined by

$$0=t^0<t^1<t^2<\cdots<t^{N+1}=t \text{ and } \Delta t^{n+1}=t^{n+1}-t^n \tag{2.27}$$

The partial derivative of p with respect to time is given in Eq. 2.16 as:

$$\left(\frac{\partial p}{\partial t}\right)^{n+1}_{i,j} = \frac{p^{n+1}_{i,j}-p^{n}_{i,j}}{\Delta t^{n+1}} \tag{2.28}$$

Using Eqs. 2.23-2.26 and Eq. 2.28 in Eq. 2.16 and after doing some arrangement and using the definitions of transmissibility (T) and volumetric term ($\tilde{V}$) gives:

$$Tx_{i+1/2,j}\left[p_{i+1,j}^{n+1}-p_{i,j}^{n+1}\right]-Tx_{i-1/2,j}\left[p_{i,j}^{n+1}-p_{i-1,j}^{n+1}\right]+$$
$$Ty_{i,j+1/2}\left[p_{i,j+1}^{n+1}-p_{i,j}^{n+1}\right]-Ty_{i,j-1/2}\left[p_{i,j}^{n+1}-p_{i,j-1}^{n+1}\right]-q_{i,j}^{n+1}=\tilde{V}_{i,j}^{n+1}\left[p_{i,j}^{n+1}-p_{i,j}^{n}\right] \quad \textbf{(2.29)}$$

where

$$Tx_{i\pm1/2,j}=\frac{2c_1}{B\mu}\frac{k_{x_{i\pm1/2,j}}\Delta y_j h}{(\Delta x_i+\Delta x_{i\pm1})} \quad \textbf{(2.30)}$$

$$Ty_{i,j\pm1/2}=\frac{2c_1}{\mu B}\frac{k_{y_{i,j\pm1/2}}\Delta x_i h}{(\Delta y_j+\Delta y_{j\pm1})} \quad \textbf{(2.31)}$$

and

$$\tilde{V}_{i,j}^{n+1}=\frac{\Delta x_i\Delta y_j h}{c_2\Delta t^{n+1}}\frac{\phi_{i,j}c_t}{B} \quad \textbf{(2.32)}$$

It is worth noting that we assume viscosity (μ) and formation volume factor (B) are constant; i.e., they do not change either in space or time.

In Eq. 2.30 and 2.32, $k_{x_{i\pm1/2,j}}$ and $k_{y_{i,j\pm1/2}}$ are permeabilities at the gridblock boundaries and the proper method of computing these permeabilities form the gridblock permeabilities is based on the "harmonic" average. Therefore, we compute these permeabilities from the following equations:

$$k_{x_{i\pm1/2,j}}=\frac{k_{x_{i,j}}k_{x_{i\pm1,j}}\left(\Delta x_i+\Delta x_{i\pm1}\right)}{k_{x_{i,j}}\Delta x_{i\pm1}+k_{x_{i\pm1,j}}\Delta x_i} \quad \textbf{(2.33)}$$

$$k_{y_{i,j\pm1/2}}=\frac{k_{y_{i,j}}k_{y_{i,j\pm1}}\left(\Delta y_j+\Delta y_{j\pm1}\right)}{k_{y_{i,j}}\Delta y_{j\pm1}+k_{y_{i,j\pm1}}\Delta y_j} \quad \textbf{(2.34)}$$

We can rearrange Eq. 2.29 as:

$$-Tx_{i-1/2,j}p_{i-1,j}^{n+1}+\left[\left(\sum T_{i,j}\right)+\tilde{V}_{i,j}^{n+1}\right]p_{i,j}^{n+1}-Tx_{i+1/2,j}p_{i+1,j}^{n+1}$$
$$-Ty_{i,j-1/2}p_{i,j-1}^{n+1}-Ty_{i,j+1/2}p_{i,j+1}^{n+1}=-q_{i,j}^{n+1}+\tilde{V}_{i,j}^{n+1}p_{i,j}^{n} \quad \textbf{(2.35)}$$

for i =1,2,…,N_x, and j = 1,2,…,N_y. In Eq. 2.35, the $\sum T_{i,j}$ is given by:

$$\sum T_{i,j} = Tx_{i-1/2,j} + Tx_{i+1/2,j} + Ty_{i,j-1/2} + Ty_{i,j+1/2} \quad \textbf{(2.36)}$$

Eq. 2.35 is evaluated in an implicit manner subject to the initial condition given in Eq. 2.21 and boundary conditions given by Eq. 2.22.

This is equivalent to setting x-transmissibilities to zero,

$$Tx_{1/2,j} = 0, \qquad \text{for } t>0, \text{ and all } j \quad \textbf{(2.37)}$$

$$Tx_{Nx+1/2,j} = 0, \qquad \text{for } t>0, \text{ and all } j \quad \textbf{(2.38)}$$

y-transmissibilities are set the zero as shown,

$$Ty_{i,1/2,k} = 0, \qquad \text{for } t>0 \text{ and all } i \quad \textbf{(2.39)}$$

$$Ty_{i,Ny+1/2,k} = 0, \qquad \text{for } t>0, \text{ and all } i \quad \textbf{(2.40)}$$

Consequently, several points were indicated through the finite difference approximation. We will use the gridblock ordering shown in Figure 2.5 to order our unknown pressures when solving Eq. 2.35 implicitly. Figure 2.5 represents an example case with 5 gridblocks in the x- (or i) direction and 5 gridblocks in the y- (or j) direction. In this ordering, the gridblock index l is related to (i,j) indices of each gridblock by the following equation:

$$l = i + (j-1) \times N_x; \text{ for } i = 1,2,\cdots N_x,\ j = 1,2,\cdots,N_y \quad \textbf{(2.41)}$$

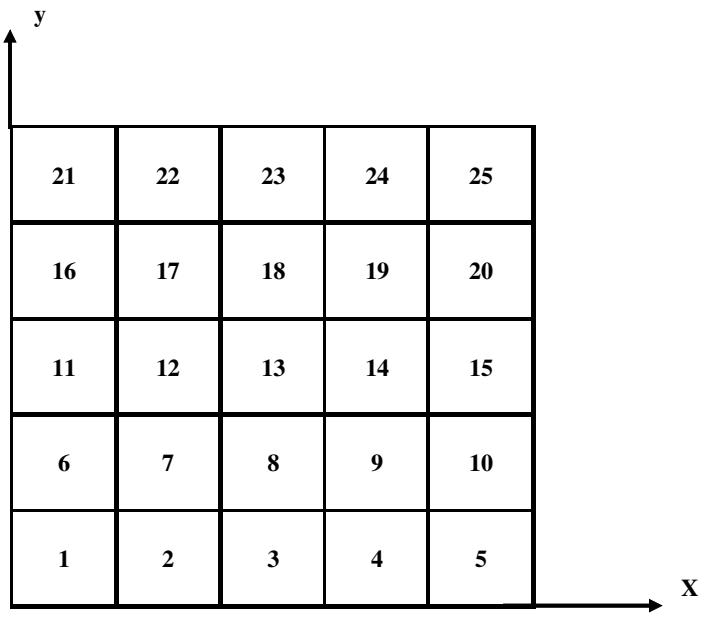


Figure 2.5 : Ordering scheme used for a 2D reservoir simulation.

Here, we show how one expresses Eq. 2.35 for each gridblock by incorporating the boundary conditions (see Eqs. 2.37 to 2.40) at any time step.

For i=1, j=1, Eq. 2.35 is used as:

$$\left[T_{x3/2,1}+T_{y1,3/2}+\tilde{V}_{1,1}^{n+1}\right]p_{1,1}^{n+1}-Tx_{3/2,1}p_{2,1}^{n+1}-Ty_{1,3/2}p_{1,2}^{n+1}=-q_{1,1}^{n+1}+\tilde{V}_{1,1}^{n+1}p_{1,1}^{n} \tag{2.42}$$

For i=N_x, j=1, Eq. 2.35 is used as:

$$\begin{aligned}&-T_{xNx-1/2,1}p_{Nx-1,1}^{n+1}+\left[Tx_{Nx-1/2,1}+Ty_{Nx,3/2}+\tilde{V}_{Nx,1}^{n+1}\right]p_{Nx,1}^{n+1}-Ty_{Nx,3/2}p_{Nx,2}^{n+1}\\&=-q_{Nx,1}^{n+1}+\tilde{V}_{Nx,1}^{n+1}p_{Nx,1}^{n}\end{aligned} \tag{2.43}$$

For i=1, j=N_y, Eq. 2.35 is used as:

$$\begin{aligned}&\left[Tx_{3/2,Ny}+Ty_{1,Ny-1/2}+\tilde{V}_{1,Ny}^{n+1}\right]p_{1,Ny}^{n+1}-Tx_{3/2,Ny}p_{2,Ny}^{n+1}-Ty_{1,Ny-1/2}p_{1,Ny-1}^{n+1}\\&=-q_{1,Ny}^{n+1}+\tilde{V}_{1,Ny}^{n+1}p_{1,Ny}^{n}\end{aligned} \tag{2.44}$$

For i=N_x, j=N_y

$$\begin{aligned}&-Tx_{Nx-1/2,Ny}p_{Nx-1,Ny}^{n+1}+\left[T_{xNx-1/2,Ny}+T_{yNx,Ny-1/2}+\tilde{V}_{Nx,Ny}^{n+1}\right]p_{Nx,Ny}^{n+1}-Ty_{Nx,Ny-1/2}p_{Nx,Ny-1}^{n+1}\\&=-q_{Nx,Ny}^{n+1}+\tilde{V}_{Nx,Ny}^{n+1}p_{Nx,Ny}^{n}\end{aligned} \tag{2.45}$$

We define the N_xN_y-dimensional vector (denoted by p^{n+1}) of unknown pressures in the LHS of Eqs. 2.42-2.45 and the N_xN_y-dimensional vector (denoted by d^n) of known quantities in the RHS of Eqs. 2.42-2.45, based on the ordering scheme of Eq. 2.41 (also see Figure 2.5):

$$p^{n+1} = \begin{pmatrix} p_{1,1}^{n+1} \\ p_{2,1}^{n+1} \\ p_{3,1}^{n+1} \\ . \\ . \\ . \\ p_{N_x,1}^{n+1} \\ p_{1,2}^{n+1} \\ p_{2,2}^{n+1} \\ p_{3,2}^{n+1} \\ . \\ . \\ . \\ p_{N_x,N_y}^{n+1} \end{pmatrix} \qquad \textbf{(2.46)}$$

and

$$d^n = \begin{pmatrix} \tilde{V}_{1,1}^{n+1} p_{1,1}^n - q_{1,1}^{n+1} \\ \tilde{V}_{2,1}^{n+1} p_{2,1}^n - q_{2,1}^{n+1} \\ \tilde{V}_{3,1}^{n+1} p_{3,1}^n - q_{3,1}^{n+1} \\ . \\ . \\ . \\ \tilde{V}_{N_{x-1},N_y}^{n+1} p_{N_{x-1},N_y}^n - q_{N_{x-1},N_y}^{n+1} \\ \tilde{V}_{N_x,N_y}^{n+1} p_{N_x,N_y}^n - q_{N_x,N_y}^{n+1} \end{pmatrix} \qquad \textbf{(2.47)}$$

It should be noted that the well flow rate terms are non-zero if the gridblock contains a source (injector) or a sink (a producer) in the RHS of Eq. 2.47.

Using the definitions of Eqs. 2.46 and 2.47, the system of linear equations given by Eqs. 2.42-2.45 can be written in the following matrix-vector form:

$$A^{n+1} p^{n+1} = d^n \tag{2.48}$$

where A is the $N_x N_y \times N_x N_y$ dimensional-coefficient matrix having a sparse and symmetric structure as shown in Figure 2.6. In Figure 2.6, "C" represents the diagonal elements, which are computed from:

$$C = \left[\left(\sum T_{i,j}\right) + \tilde{V}_{i,j}^{n+1}\right] = Tx_{i-1/2,j} + Tx_{i+1/2,j} + Ty_{i,j-1/2} + Ty_{i,j+1/2} + \tilde{V}_{i,j}^{n+1} \tag{2.49}$$

Solution methods for the problem given by Eq. 2.48 are either direct or iterative. In a direct method, the algorithm produces an exact solution and gives a precise answer for the corresponding number of step. Gaussian elimination is one of the oldest and most popular direct-solution methods among several other direct-solution techniques. Iterative method approaches a solution using iterations. In contrast to direct method, which attempt to solve an equation approximately from an initial guess. Conjugate gradient type methods are very popular iterative procedures. In this thesis, the MATLAB code given uses a direct method to solve the matrix problem. Although it provides accurate results, as it requires the storage of full matrix A, it is not used for large size problems. For large size problems, iterative methods are more advantageous to use than direct methods. For large size problems, we use the iterative method of Welty and Meijerink (1981) as to be discussed in Chapter III. Also efficient storage schemes should be used to store the matrix for large problems.

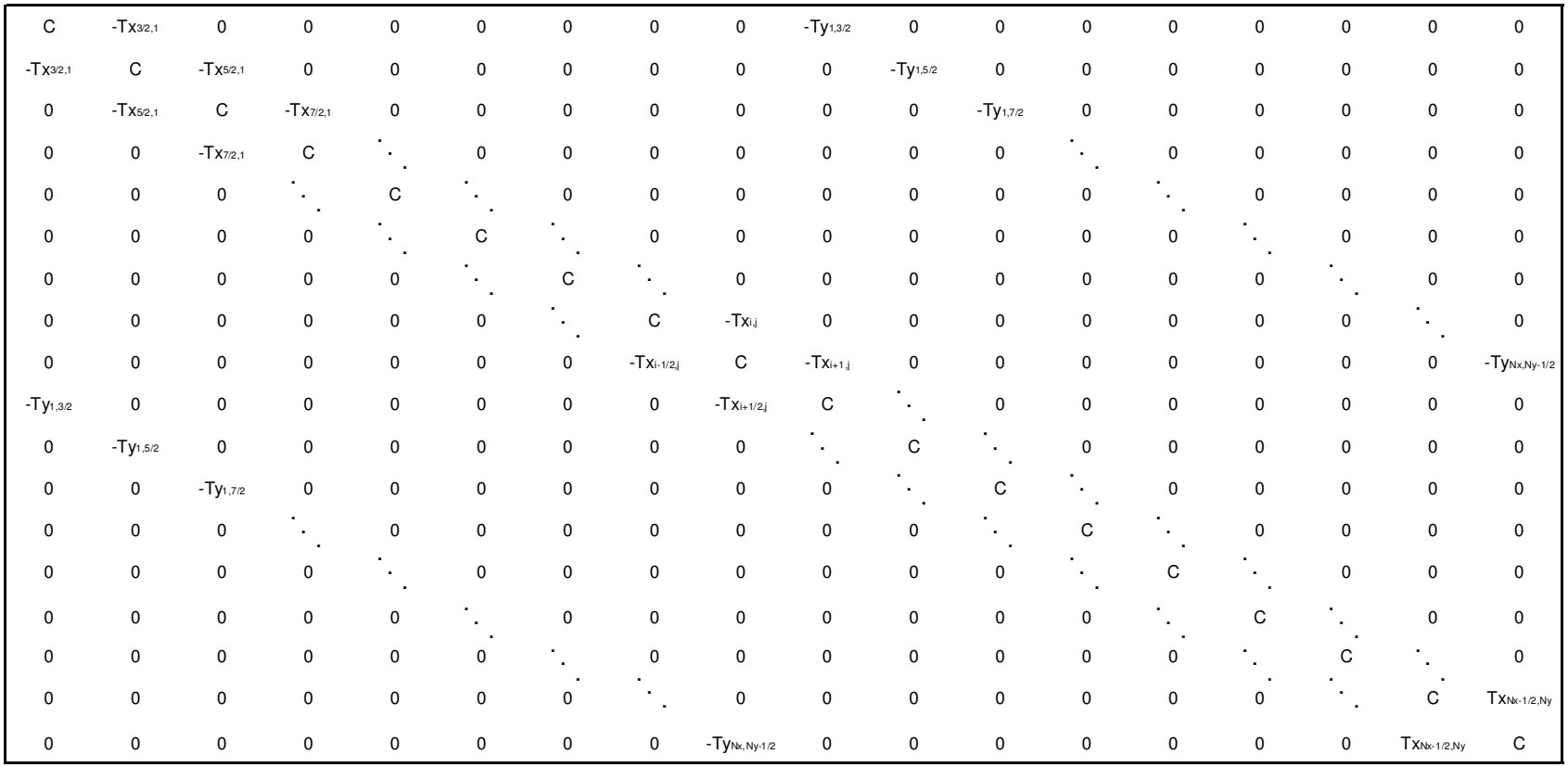


Figure 2.6 : Structure of the matrix A used in ITSM.

In 2D x-y simulations, we usually solve pressure values in the gridblocks. They do not represent the wellbore pressure where the well is located (Onur, 1997). Peaceman's well model approach calculates bottom hole pressure where the well is located in the gridblock. Peaceman's equation is also effective and generally the accepted approach for most simulator applications. Most of simulators apply Peaceman approach. The following equation 2.50 developed by Peaceman is given by (Peaceman, 1983):

$$q_{i,j}^{n+1} = (WI)_{i,j}(p_{i,j}^{n+1} - p_{wfi,j}^{n+1}) \tag{2.50}$$

and the well index [in (STB/D)/psi] is given by

$$(WI)_{i,j} = \frac{2\pi c_1 h\sqrt{k_{x,i,j}k_{y,i,j}}}{\mu\left[\ln(\frac{r_{o,i,j}}{r_{w,i,j}}) + S_{i,j}\right]} \tag{2.51}$$

In Eq. 2.51, $S_{i,j}$ refers to the well skin factor, and $r_{oi,j}$ is the radius where $p_{i,j}$ occurs. It is given by,

$$r_{o,i,j} = \frac{0.28073\Delta x_i\sqrt{1+\frac{k_{x,i,j}}{k_{y,i,j}}}(\frac{\Delta y_j}{\Delta x_i})^2}{1+\sqrt{\frac{k_{x,i,j}}{k_{y,i,j}}}} \tag{2.52}$$

In this thesis, we consider only the cases where flow rate histories of wells are prescribed. So, we first solve the matrix problem given by Eq. 2.48 for gridblock pressures and then use Eq. 2.50 to solve the flowing bottom-hole pressures at the well locations as follows:

$$p_{wfi,j}^{n+1} = p_{i,j}^{n+1} - \frac{q_{i,j}^{n+1}}{(WI)_{i,j}} \tag{2.53}$$

2.4.2 Lanczos decomposition method (LDM)

In previous chapter, the formulation of the ITSM for simulation of the single-phase flow of slightly compressible fluid and constant viscosity was given. As mentioned

previously, our main objective in this thesis is to investigate whether the Lanczos Decomposition Method (LDM) proposed by Druskin and Knizhnerman, 1994 and then applied by Knizhnerman, et al., 1994 and Alpak et al., 2003 for 2D r-z single-phase flow of slightly compressible fluid of constant viscosity and compressibility, single-well problems. Here, we provide the formulation for a 2D x-y single-phase, but for multi-well systems. It is important to note that the formulation provided by Knizhnerman, et al., 1994 and Alpak et al., 2003 is only valid for a constant rate production at the well. To generate flowing wellbore pressures for prescribed variable rate production at the well, they use a two-step procedure: First, they use the Lanczos method to solve wellbore pressure for a single-well constant rate problem and then an external procedure based on the method of superposition to generate the wellbore pressures for prescribed variable rate cases. The formulation we provide does not require a two-step procedure and directly provides the solution for the cases where each well produces with a prescribed variable flow rate history.

The Lanczos method uses the semi-discrete form of the continuity equation as given by Eq. 2.16. Unlike the implicit time-stepping method (ITSM) discussed in the previous section, the Lanczos method does not consider the discretization of the pressure derivative with time in the RHS of Eq. 2.16, but the spatial derivatives of pressure in the LHS of Eq. 2.16 are discretized as in the case of the ITSM method. So, the semi-discrete form of Eq. 2.16 with the spatial derivatives discretized based on a structured Cartesian block centered grid system (Section 2.2) can be written as:

$$T_{x,i+1/2,j}\left(p_{i+1,j}-p_{i,j}\right)-T_{x,i-1/2,j}\left(p_{i,j}-p_{i-1,j}\right)+T_{y,i,j+1/2}\left(p_{i,j+1}-p_{i,j}\right)$$
$$-T_{y,i,j-1/2}\left(p_{i,j+1}-p_{i,j}\right)-q_{i,j}(t)=\bar{V}_{i,j}\frac{dp_{i,j}}{dt} \qquad \textbf{(2.54)}$$

where

$$\bar{V}_{i,j}=\frac{\Delta x_i \Delta y_j h}{c_2}\frac{\phi_{i,j} c_t}{B} \qquad \textbf{(2.55)}$$

for $i = 1,2,\ldots,N_x$, $j = 1,2,\ldots,N_y$.

We divide both sides of Eq. 2.54 by equation by $\bar{V}_{i,j}$ and rearrange the resulting equation to obtain Eq. 2.56:

$$\frac{dp_{i,j}}{dt}-\frac{T_{x,i+1/2,j}}{\bar{V}_{i,j}}\left(p_{i+1,j}-p_{i,j}\right)+\frac{T_{x,i-1/2,j}}{\bar{V}_{i,j}}\left(p_{i,j}-p_{i-1,j}\right)+$$
$$-\frac{T_{y,i,j+1/2}}{\bar{V}_{i,j}}\left(p_{i,j+1}-p_{i,j}\right)+\frac{T_{y,i,j-1/2,k}}{\bar{V}_{i,j,k}}\left(p_{i,j}-p_{i,j-1}\right)=-\frac{q_{i,j}(t)}{\bar{V}_{i,j}} \quad \textbf{(2.56)}$$

or could be rewritten as:

$$\frac{dp_{i,j}}{dt}-\frac{T_{x,i-1/2,j}}{\bar{V}_{i,j}}p_{i-1,j}+\frac{\left(\sum T_{i,j}\right)}{\bar{V}_{i,j}}p_{i,j}-\frac{T_{x,i+1/2,j}}{\bar{V}_{i,j}}p_{i+1,j}-\frac{T_{y,i,j-1/2}}{\bar{V}_{i,j}}p_{i,j-1}$$
$$-\frac{T_{y,i,j+1/2}}{\bar{V}_{i,j}}p_{i,j+1}=-\frac{q_{i,j}(t)}{\bar{V}_{i,j}} \quad \textbf{(2.57)}$$

We define the N_xN_y-dimensional vector (denoted by p) of unknown pressures in the LHS of Eq. 2.57 and the N_xN_y-dimensional vector (denoted by r) of known quantities in the RHS of Eq. 2.57, based on the ordering scheme of Eq. 2.41 (also see Figure 2.5):

$$p(t)=\begin{pmatrix} p_{1,1}(t) \\ p_{2,1}(t) \\ p_{3,1}(t) \\ . \\ . \\ . \\ p_{N_x,1}(t) \\ p_{1,2}(t) \\ p_{2,2}(t) \\ p_{3,2}(t) \\ . \\ . \\ . \\ p_{N_x,N_y}(t) \end{pmatrix} \quad \textbf{(2.58)}$$

and

$$r(t) = -\begin{pmatrix} q_{1,1}(t)/\bar{V}_{1,1} \\ q_{2,1}(t)/\bar{V}_{2,1} \\ q_{3,1}(t)/\bar{V}_{3,1} \\ . \\ . \\ . \\ q_{N_{x-1},N_y}(t)/\bar{V}_{N_{x-1},N_y} \\ q_{N_x,N_y}(t)/\bar{V}_{N_x,N_y} \end{pmatrix} \qquad \textbf{(2.59)}$$

Using the definitions given by Eqs. 2.58 and 2.59, we can express Eq. 2.57 in the following matrix-vector form:

$$\frac{dp(t)}{dt} + Ap(t) = r(t) \qquad \textbf{(2.60)}$$

In Eq. 2.60, A is the $N_xN_y \times N_xN_y$ dimensional-coefficient matrix having a sparse structure as shown in Figure 2.7. In Figure 2.7 the diagonal elements denoted by the letter C is given by $\left(\sum T_{i,j}\right)/\bar{V}_{i,j}$.

A few remarks are in order for the matrix A of the system of equations (Eq. 2.60) as shown in Figure 2.7: (i) The matrix A is not necessarily a symmetric matrix. It is a symmetric matrix if and only if the porosity field (ϕ) is uniform (homogeneous) and the grid is uniform in the x and y directions (see definition of $\bar{V}_{i,j}$ given by Eq. 2.55); (ii) It is a singular matrix (not rigorously shown here) because of the Neumann boundary conditions we use. One can use the transformation proposed by Knizhmerman et al.1994 to derive a formulation where the matrix A in Eq. 2.60 will be symmetric. The singularity of the matrix A due to Neumann boundary conditions seems not to impose a difficulty for solving Eq. 3.6 by the LDM method. Although not shown here, it can be shown that the matrix A in Eq. 2.60 will be nonsingular if we consider Dirichlet (constant pressure) boundary conditions, which are not considered in this thesis.

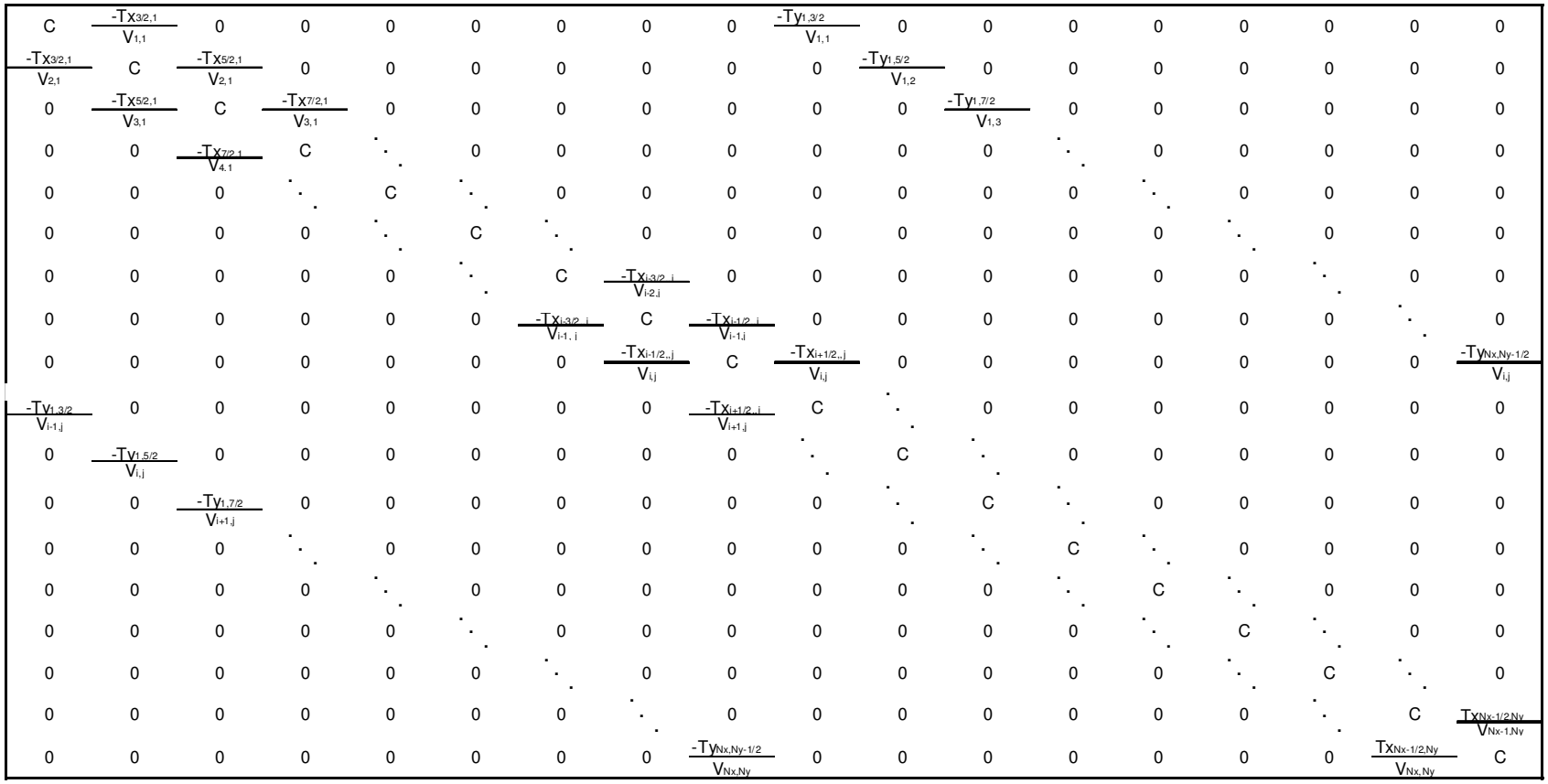


Figure 2.7 : Structure of the matrix *A* used in LDM

It is important to note that Eq. 2.60 is a first order, non-homogeneous linear ordinary differential equation and its solution is given by (Ross, 1974):

$$p(t) = e^{-tA} p^0 + \int_0^t e^{-(t-\tau)A} r(\tau)\, d\tau \qquad \textbf{(2.61)}$$

where p^0 is the $N_x \times N_y$ dimensional vector of initial gridblock pressure.

Eq. 2.61 is quite general in that it applies to cases where each source/sink has a variable flow rate history. Before we discuss how we evaluate Eq. 2.61 for the general case where the source/sink vector $r(t)$ can change with time t, we consider a the simple case where the vector r is constant, i.e., it does not change with time t. This assumption indicates that each source/sink starts operating at $t = 0$ with a constant prescribed flow rate.

Now, we focus on the solution for a relatively simple case where the vector r in Eq. 2.61 is constant over the time interval $[0,t]$. For this case, we obtain the following equation after performing the integration in Eq. 2.61:

$$p(t) = e^{-tA} p^0 + t\varphi(-tA) r \qquad \textbf{(2.62)}$$

where

$$\varphi(\tau A) = \frac{\left(e^{\tau A} - 1\right)}{\tau A} \qquad \textbf{(2.63)}$$

where $\tau = -t$ or $-\tau = t$.

Sidje (1998) provides computational algorithms contained in a publically available package called EXPOKIT for evaluating the action of the matrix exponential on an arbitrary vector. EXPOKIT provides a set of routines written in Fortran 77 aimed at computing matrix exponentials and also EXPOKIT is available in MATLAB. More precisely, it computes either a small matrix exponential in full, the action of a large sparse matrix exponential on an operand vector, or the solution of a system of linear ODEs with constant inhomogeneity. The backbone of the sparse routines consists of Krylov subspace projection methods [Arnoldi (for nonsymmetric matrix case) and

Lanczos (for symmetric matrix case) processes, see Saad, 1992], and hence EXPOKIT is capable of coping with sparse matrices of large dimensions. At this point, it is important to note that the Lanczos method uses Krylov subspaces and hence the solution depends on the dimension (or basis) of the Krylov subspace (denoted by *m*) method at a given time step. Although there are formulas derived for predicting the appropriate value of *m* to be used for a given time step (see for example, Druskin and Knizhnermann, 1995, and Sidje (1998)), these formulas require that we compute the norm of the matrix *A*. This requires additional computation work for computing the norm of the matrix *A* shown in Figure 2.7.

For our investigation and the results to be presented later, we have used EXPOKIT package as implemented in MATLAB to perform the computations given by Eqs. 2.61, 2.62 and 2.63, depending on the case considered. In EXPOKIT, one specifies the basis of Krylov subspace (m) and the accuracy tolerance (tol) on the solution *p*. The default values of *m* and tol in EXPOKIT are 30 and 10^{-7}. It is also worth noting that Sidje (1998) states that in reality, due to stability and accuracy requirements, for example, $p(t)$ from Eq. 2.62 is not computed for a given time *t* in one stretch, and uses a time-stepping strategy along with error estimation which is embedded within the process. Typically, the algorithm for Eq. 2.62 evolves with the integration scheme given by:

$$\begin{aligned} &p(t=0) = p^0 \\ &p(t_{k+1}) = \tau_k \varphi(-\tau_k A)\left[-Ap(t_k) + r\right] + p(t_k); \text{ for } k = 0,1,2,..,s \end{aligned} \qquad \textbf{(2.64)}$$

where

$$\tau_k = t_{k+1} - t_k, \quad 0 = t_0 < t_1 < \cdots < t_s < t_{s+1} = t \qquad \textbf{(2.65)}$$

Now, we discuss how we evaluate Eq. 2.61 for the general case where the source/sink vector r changes with time. To compute the solution $p(t)$ from Eq. 2.61, we will assume that flow rate at each well is represented as stepwise constant over time because flow rates are typically reported as stepwise constant for pressure transient test applications. Then, by considering a time discretization of the interval

[0,*t*] as $0 = t_0 < t_1 < t_2 < \cdots < t_n < t_{n+1} = t$, then we can compute the solution at time *t* from Eq. 2.66:

$$p(t_{n+1}) = e^{-t_{n+1}A} p^0 + \sum_{k=1}^{n+1} \left[(t_{n+1} - t_{k-1}) \varphi\left(-(t_{n+1} - t_{k-1}) A\right) - (t_{n+1} - t_k) \varphi\left(-(t_{n+1} - t_k) A\right) \right] r(t_k) \quad \textbf{(2.66)}$$

A more computationally attractive form. can be obtained from Eq. 2.66 and this form is given by Eq. 2.67:

$$p(t_{n+1}) = e^{-t_{n+1}A} p^0 + t_{n+1} \varphi(-t_{n+1} A) r(t_1) + \sum_{k=1}^{n} (t_{n+1} - t_k) \varphi\left(-(t_{n+1} - t_k) A\right) \left[r(t_{k+1}) - r(t_k) \right] \quad \textbf{(2.67)}$$

As in the ITSM, we compute the flow bottom hole pressure at the source/sink locations by using the formula given by Peacemean (1983); i.e.,

$$p_{wf_{i,j}}(t) = p_{i,j}(t) - \frac{q_{i,j}(t)}{(WI)_{i,j}} \quad \textbf{(2.68)}$$

Where the well index $(WI)_{i,j}$ is given by Eq. 2.51.

3. SIMULATOR APPLICATIONS

The modeling of a reservoir system can be via a computer simulator. The simulators attempt to find solutions to problems that enable the prediction of the behavior of the system from a set of parameters and initial conditions.

The problem under study is an initial boundary value problem that is solved numerically using both implicit time-stepping finite difference method and the Lanczos method. Computer simulators for both methods were written by MATLAB (**MAT**rix **LAB**oratory). MATLAB is an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. This language is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. Because of efficiency on large scale computing, MATLAB has become a rival choice of language for scientific and high performance computing on modern supercomputers. Computer programs are represented in Appendix A and B.

The user must input some of the variables orderly, these are,

i. The reservoir length, L_x by ft
ii. The reservoir width L_y, by ft
iii. The reservoir height h, by ft
iv. Initial pressure value, p_0 by psia
v. Total compressibility, c_t of reservoir rock and fluid, psi^{-1}
vi. Fluid viscosity, μ, cp
vii. Number of x-axis direction gridblocks, Nx
viii. Number of y-axis direction gridblocks, Ny
ix. Multiple Production or Injection Wells (sink or source), N_w
x. Well radius r_w, by ft
xi. Skin factor at each well l, S_l, l=1,2,…,N_w,

Additionally, the user has some options,

i. The user can assign different gridblock lengths which can be uniform or nonuniform.
ii. Inputting the permeabilities k_x and k_y with variable values for each gridblock separately, or inputting same permeability for all gridblocks. And then, the user can change permeability in a specific location after assigned same value for all gridblocks.
iii. Inputting the porosity for each gridblock separately with variable values, or inputting same porosity for all gridblocks. And then, the user can change porosity in a specific location after assigned same value for all gridblocks.
iv. Inputting single well in a location or multiple wells in various locations, with different flow rate and well radius.

3.1 Verification of Simulators

The simulators are validated by comparing the simulation results with the results from a commercial software (ECRIN). The comparison also show the two different simulators which were written by Lanczos decomposition and conventional time-stepping methods. We compare the simulators for the cases where we have single-well constant-rate production and with variable rate production. In some of the comparisons given in the following section for the constant rate case, we fully store the coefficient matrix *A* required in both the ITSM and LDM to show the advantage of storing only the nonzeros of the matrix in computations. We provide comparisons of the ITSM and LDM when computations are performed with full storage and sparse storage of matrix *A* (i.e., only storing nonzeros and carrying out the computations with such a stored matrix). When using the MATLAB code written for LDM, we consider SPARSE(*A*) function to store only the nonzero elements of the matrix *A*. In cases where we store the full matrix *A*, we were not able to run "large size" problems due to memory requirements of the computer we used. The the largest grid system that could be considered was equal to 67×67. As the codes in Fortran 77 written by Dr. Mustafa Onur for both LDM and ITSM were based on the efficient storage method based on compressed row storage for storing the matrix *A* used in the ITSM (Figure 2.6) and LDM (see Figure 2.7), we also compared the

performance of the solutions in terms of matrix storage. Dr. Mustafa Onur's Fortran 77 code for the ITSM is based on the iterative solver method of Symmetric Strongly Implicit Procedure (SSIP) described by Welty and Meijerink (1981) and uses a compressed row storage scheme for storing the coefficient matrix. It should be noted that when row storage scheme is used, we store only the nonzero entries of the coefficient matrix *A*. The performance comparisons of the MATLAB (using full storage and direct solver for the ITSM and MATLAB codes for the LDM) and FORTRAN codes (using efficient storage and iterative solver for the ITSM and publically available FORTRAN codes of Sidje (1988) for the LDM) for both the ITSM and LDM are also given in this section.

3.1.1 Constant rate production case

In this verification, reservoir and fluid properties are given in Table 3.1 and location of the well in the reservoir system is illustrated in Figure 3.1. (Note that the well is a sink (a production) well located at the center of the reservoir.) Furthermore, pressure and pressure derivative values (only for this case) were calculated for four different cases: reservoir was divided into 11x11, 21x21, 51x51 and 67x67 gridblocks. In each case, one production well was located at the center of reservoir.

Table 3.1: Formation and fluid properties for validity

Reservoir Length in x-direction, L_x , ft	3000
Reservoir Length in y-direction , L_y , ft	3000
Reservoir Height , h , ft	100
Initial Reservoir Pressure, p_0, psia	3500
Reservoir Porosity, ϕ (fraction)	0.15
Reservoir Permeability in x direction, k_x ,md	100
Reservoir Permeability in y direction, k_y ,md	100
Fluid viscosity, μ cp	1
Total Compressibility, c_t , psi^{-1}	0.00002
Number of wells	1 production
Well Production/injection Rate, q, STB/D	1000
Formation Volume factor, B, RB/STB	1.0
Well Radius, r_w, ft	0.354
Skin Factor at the well, S	0

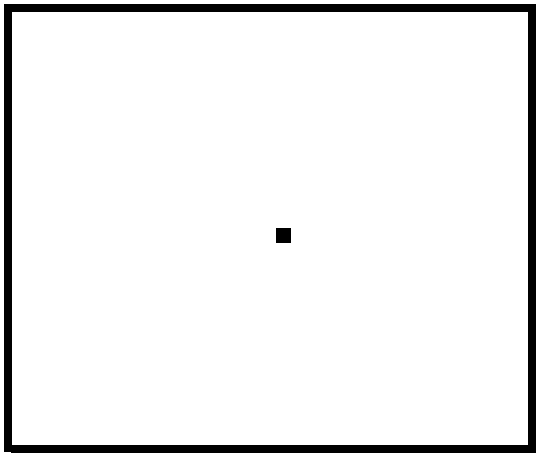


Figure 3.1: A schematic representation of reservoir/well system used for verification.

Two different simulator applications were used to find pressures. The first code was written by conventional implicit time-stepping, and the second code was developed for Lanczos decomposition method. Using the input reservoir, fluid and formation data (Table 3.1), the simulators were run and the results obtained from the simulators of this study were compared with the results obtained the commercial software called ECRIN.

Tables 3.2-3.5 compare how bottom-hole pressure at the well computed from our simulator based on the Lanczos method compare with the corresponding results from ECRIN software. All results given in Table 3.2-3.5 were generated by using full storage of the matrix A of the LDM (see Figure 2.7). In each table, we consider different gridblock numbers to simulate the system shown in Figure 3.1. For instance, Table 3.2 is for a total of 121 gridblocks, i.e., $N_x = N_y = 11$, whereas Table 3.5 gives the results for a total of 4489 gridblocks, i.e., $N_x = N_y = 67$. Also, these tables presents CPU times spent by our simulator based on the Lanczos method to compute the flowing bottom hole pressure at a given time step. As mentioned previously, we use the default value of Krylov basis m equal to 30 and the tolerance value of tol = 10^{-7} were used to obtain the solutions given in Tables 3.2-3.5 CPU measurements per time step were made using TIC and TOC functions in MATLAB. The same functions could also be used in FORTRAN programs. These functions are used to find the time to run segments of the codes and hence to compare the speed of different implementations of the programs.

Table 3.2: Simulator using Lanczos decomposition vs. ECRIN software results (Number of gridblocks $N_x = 11$ and $N_y = 11$)

Time (days)	p_{wf} at the production well (LDM)	p_{wf} at the production well (ECRIN)	Relative Error,%	Simulator CPU Time (seconds)
0.0013	3428.654	3440.241	3.38E-01	1.88E-01
0.0025	3428.346	3435.355	2.04E-01	3.13E-02
0.0053	3427.673	3430.058	6.96E-02	3.13E-02
0.0106	3426.459	3425.183	3.72E-02	3.13E-02
0.1057	3412.781	3408.928	1.13E-01	3.13E-02
1.0569	3392.980	3392.665	9.29E-03	3.13E-02
10.569	3368.818	3368.842	6.95E-04	3.13E-02
102.076	3178.517	3178.990	1.49E-02	1.56E-02
252.076	2866.573	2867.750	4.11E-02	3.13E-02
502.076	2346.665	2349.018	1.00E-01	3.13E-02
752.076	1826.758	1830.213	1.89E-01	1.56E-02
902.076	1514.814	1519.015	2.77E-01	3.13E-02
1000	1311.167	1315.859	3.58E-01	1.56E-02

Table 3.3: Simulator using Lanczos decomposition vs. ECRIN software results (Number of gridblocks $N_x = 21$ and $N_y = 21$)

Time (days)	p_{wf} at the production well (LDM)	p_{wf} at the production well (ECRIN)	Relative Error,%	Simulator CPU Time (seconds)
0.0013	3436.981	3440.241	9.49E-02	9.53E-01
0.0025	3435.919	3435.355	1.64E-02	9.53E-01
0.0053	3433.727	3430.058	1.07E-01	9.69E-01
0.0106	3430.189	3425.183	1.46E-01	9.69E-01
0.1057	3409.920	3408.928	2.91E-02	9.69E-01
1.0569	3392.750	3392.665	2.52E-03	9.69E-01
10.569	3368.803	3368.842	1.14E-03	9.69E-01
102.076	3178.502	3178.990	1.53E-02	9.53E-01
252.076	2866.558	2867.750	4.16E-02	9.53E-01
502.076	2346.651	2349.018	1.01E-01	9.53E-01
752.076	1826.743	1830.213	1.90E-01	9.69E-01
902.076	1514.799	1519.015	2.78E-01	9.69E-01
1000	1311.152	1315.859	3.59E-01	9.53E-01

Table 3.4: Simulator using Lanczos decomposition vs. ECRIN software results (Number of gridblocks $N_x = 51$ and $N_y = 51$)

Time (days)	p_{wf} at the production well (LDM)	p_{wf} at the production well (ECRIN)	Relative Error,%	Simulator CPU Time (seconds)
0.0013	3444.780	3440.241	1.32E-01	7.66E-01
0.0025	3440.369	3435.355	1.46E-01	6.72E-01
0.0053	3433.727	3430.058	1.07E-01	7.19E-01
0.0106	3427.082	3425.183	5.54E-02	1.42E+00
0.1057	3409.067	3408.928	4.09E-03	1.97E+00
1.0569	3392.682	3392.665	5.08E-04	5.92E+00
10.569	3368.798	3368.842	1.28E-03	1.60E+01
102.076	3178.498	3178.990	1.55E-02	5.22E+01
252.076	2866.553	2867.750	4.17E-02	1.10E+02
502.076	2346.646	2349.018	1.01E-01	2.04E+02
752.076	1826.738	1830.213	1.90E-01	3.01E+02
902.076	1514.794	1519.015	2.79E-01	3.61E+02
1000	1311.148	1315.859	3.59E-01	3.94E+02

Table 3.5: Simulator using Lanczos decomposition vs. ECRIN software results (Number of gridblocks $N_x = 67$ and $N_y = 67$)

Time (days)	p_{wf} at the production well (LDM)	p_{wf} at the production well (ECRIN)	Relative Error,%	Simulator CPU Time (seconds)
0.0013	3445.316	3440.241	1.47E-01	1.78E+00
0.0025	3439.531	3435.355	1.21E-01	1.80E+00
0.0053	3432.296	3430.058	6.52E-02	3.30E+00
0.0106	3426.156	3425.183	2.84E-02	6.55E+00
0.1057	3409.007	3408.928	2.33E-03	1.91E+01
1.0569	3392.676	3392.665	3.38E-04	6.07E+01
10.569	3368.798	3368.842	1.30E-03	2.25E+02
102.076	3178.497	3178.990	1.55E-02	4.63E+02
252.076	2866.553	2867.750	4.18E-02	8.58E+02
502.076	2346.645	2349.018	1.01E-01	1.27E+03
752.076	1826.738	1830.213	1.90E-01	1.57E+03
902.076	1514.794	1519.015	2.79E-01	1.67E+03
1000	1311.147	1315.859	3.59E-01	2.12E+03

To show the computational improvement if one uses a sparse storage of the matrix A in LDM, we present the CPU times obtained for LDM with full storage and sparse storage of the matrix A in Table 3.6. Table 3.6 presents only the results for the case

where the grid system is 67×67. As can be seen from Table 3.6, using a sparse storage of the matrix A in the LDM significantly reduces the computational times at given time.

Table 3.6: CPU times from the simulators using Lanczos decomposition with full and sparse matrix storage of the matrix *A*. (Number of gridblocks N_x = 67 and N_y = 67)

Time (days)	CPU Time for LDM with full matrix storage (seconds)	CPU Time for LDM with sparse matrix storage (seconds)
0.0013	1.78E+00	9.26E-02
0.0025	1.80E+00	3.72E-02
0.0053	3.30E+00	2.45E-02
0.0106	6.55E+00	5.24E-02
0.1057	1.91E+01	9.87E-02
1.0569	6.07E+01	3.01E-01
10.569	2.25E+02	9.57E-01
102.076	4.63E+02	3.66E+00
252.076	8.58E+02	7.37E+00
502.076	1.27E+03	1.39E+01
752.076	1.57E+03	2.00E+01
902.076	1.67E+03	2.39E+01
1000	2.12E+03	2.65E+01

Tables 3.7-3.10 compare how the bottom-hole pressure at the well computed from our simulator based on the implicit-time stepping method (ITSM) with the corresponding results from ECRIN software. The results given in Tables 3.7-3.10 were generated by using the full storage of the matrix A in direct solver.

Table 3.7: Simulator using ITSM vs. ECRIN software results (Number of gridblocks $N_x = 11$ and $N_y = 11$)

Time (days)	p_{wf} at the production well (ITSM)	p_{wf} at the production well (ECRIN)	Relative Error,%	Simulator CPU Time (seconds)
0.0013	3428.681	3440.241	3.37E-01	9.52E-04
0.0025	3428.375	3435.355	2.04E-01	1.87E-03
0.0053	3427.710	3430.058	6.85E-02	2.75E-03
0.0106	3426.527	3425.183	3.92E-02	3.62E-03
0.1057	3415.252	3408.928	1.85E-01	4.57E-03
1.0569	3396.314	3392.665	1.07E-01	5.51E-03
10.569	3369.407	3368.842	1.68E-02	6.45E-03
102.076	3178.524	3178.990	1.47E-02	7.39E-03
252.076	2866.573	2867.750	4.10E-02	8.32E-03
502.076	2346.666	2349.018	1.00E-01	9.47E-03
752.076	1826.758	1830.213	1.89E-01	1.05E-02
902.076	1514.814	1519.015	2.77E-01	1.14E-02
1000	1311.168	1315.859	3.58E-01	1.25E-02

Table 3.8: Simulator using ITSM vs. ECRIN software results (Number of gridblocks $N_x = 21$ and $N_y = 21$)

Time (days)	p_{wf} at the production well (ITSM)	p_{wf} at the production well (ECRIN)	Relative Error,%	Simulator CPU Time (seconds)
0.0013	3437.091	3440.241	9.17E-02	7.12E-03
0.0025	3436.048	3435.355	2.02E-02	1.43E-02
0.0053	3433.947	3430.058	1.13E-01	2.14E-02
0.0106	3430.639	3425.183	1.59E-01	2.88E-02
0.1057	3413.644	3408.928	1.38E-01	3.59E-02
1.0569	3395.689	3392.665	8.91E-02	4.31E-02
10.569	3369.356	3368.842	1.53E-02	5.03E-02
102.076	3178.509	3178.990	1.52E-02	5.75E-02
252.076	2866.558	2867.750	4.16E-02	6.47E-02
502.076	2346.651	2349.018	1.01E-01	7.19E-02
752.076	1826.743	1830.213	1.90E-01	7.91E-02
902.076	1514.799	1519.015	2.78E-01	8.63E-02
1000	1311.153	1315.859	3.59E-01	9.34E-02

Table 3.9: Simulator using ITSM vs. ECRIN software results (Number of gridblocks $N_x = 51$ and $N_y = 51$)

Time (days)	p_{wf} at the production well (ITSM)	p_{wf} at the production well (ECRIN)	Relative Error,%	Simulator CPU Time (seconds)
0.0013	3445.711	3440.241	1.59E-01	6.78E-01
0.0025	3441.500	3435.355	1.79E-01	1.36E+00
0.0053	3435.342	3430.058	1.54E-01	2.07E+00
0.0106	3428.888	3425.183	1.08E-01	2.79E+00
0.1057	3412.032	3408.928	9.10E-02	3.51E+00
1.0569	3395.494	3392.665	8.33E-02	4.22E+00
10.569	3369.340	3368.842	1.48E-02	4.93E+00
102.076	3178.504	3178.990	1.53E-02	5.66E+00
252.076	2866.553	2867.750	4.17E-02	6.37E+00
502.076	2346.646	2349.018	1.01E-01	7.09E+00
752.076	1826.739	1830.213	1.90E-01	7.80E+00
902.076	1514.794	1519.015	2.79E-01	8.52E+00
1000	1311.148	1315.859	3.59E-01	9.19E+00

Table 3.10: Simulator using ITSM vs. ECRIN software results (Number of gridblocks $N_x = 67$ and $N_y = 67$)

Time (days)	p_{wf} at the production well (ITSM)	p_{wf} at the production well (ECRIN)	Relative Error,%	Simulator CPU Time (seconds)
0.0013	3447.043	3440.241	1.97E-01	3.24E+00
0.0025	3441.403	3435.355	1.76E-01	6.55E+00
0.0053	3434.366	3430.058	1.25E-01	9.76E+00
0.0106	3427.929	3425.183	8.01E-02	1.30E+01
0.1057	3411.865	3408.928	8.61E-02	1.62E+01
1.0569	3395.479	3392.665	8.29E-02	1.94E+01
10.569	3369.339	3368.842	1.48E-02	2.27E+01
102.076	3178.503	3178.990	1.53E-02	2.59E+01
252.076	2866.553	2867.750	4.18E-02	2.94E+01
502.076	2346.646	2349.018	1.01E-01	3.26E+01
752.076	1826.738	1830.213	1.90E-01	3.58E+01
902.076	1514.794	1519.015	2.79E-01	3.91E+01
1000	1311.147	1315.859	3.59E-01	4.23E+01

To show the computational improvement if one uses a sparse storage of the matrix *A* in ITSM, we present the CPU times obtained for ITSM with full storage and sparse storage of the matrix *A* in Table 3.11. Table 3.11 presents only the results for the case where the grid system is 67×67. As can be seen from Table 3.11, using a sparse storage of the matrix *A* in the ITSM also significantly reduces the computational time at a given time.

Table 3.11: CPU times from the simulators using ITSM with full and sparse matrix storage of the matrix *A*. (Number of gridblocks $N_x = 67$ and $N_y = 67$)

Time (days)	CPU Time for ITSM with full matrix storage (seconds)	CPU Time for ITSM with sparse matrix storage (seconds)
0.0013	3.24E+00	1.60E-02
0.0025	6.55E+00	3.15E-02
0.0053	9.76E+00	4.64E-02
0.0106	1.30E+01	6.17E-02
0.1057	1.62E+01	7.73E-02
1.0569	1.94E+01	9.32E-02
10.569	2.27E+01	1.09E-01
102.076	2.59E+01	1.27E-01
252.076	2.94E+01	1.43E-01
502.076	3.26E+01	1.58E-01
752.076	3.58E+01	1.74E-01
902.076	3.91E+01	1.91E-01
1000	4.23E+01	2.07E-01

In summary, the results of Tables 3.2-3.11 indicate that in general, the LDM requires more CPU times than does ITSM. Moreover, as the number of gridblock increases the LDM requires too much CPU times as compared to the ITSM. So, in terms of computational times, it is clear that the ITSM is far superior to the LDM. Also, apparent from the CPU times reported in Tables 3.2-3.9, CPU times for both LDM and ITSM increase as the size of the time step increases. However, the increase in CPU times demanded by LDM at large time steps are far more than those demanded by the ITSM.

In Figure 3.2, we compare pressure changes (defined as $\Delta p = p_0 - p_{wf}(t)$ and pressure-derivative defined as $d\Delta p/d\ln t$, typically used for pressure transient analysis) computed from our simulators based on the LDM and ITSM and from the analytical

solution of ECRIN for two cases where $N_x = N_y = 21$ and $N_x = N_y = 67$. As can be seen both simulators give almost identical responses, but they predict quite different pressure and derivative responses at very early times from those of ECRIN's analytical solution. The reason for that the simulator pressures at the wellblock suffer from gridblock storage effect, i.e, actually production at the well occurs due to the expansion of fluid and rock in the wellblock. This behavior is reflected as the "apparent" wellbore storage effect at the well. This could be reduced if one reduces the volume of wellblock, which is equivalent to reducing the size of the well gridblock. As can be seen, using more gridblocks (see results for the case where $N_x = N_y = 67$), decreases the wellblock storage effect on pressure and derivative responses.

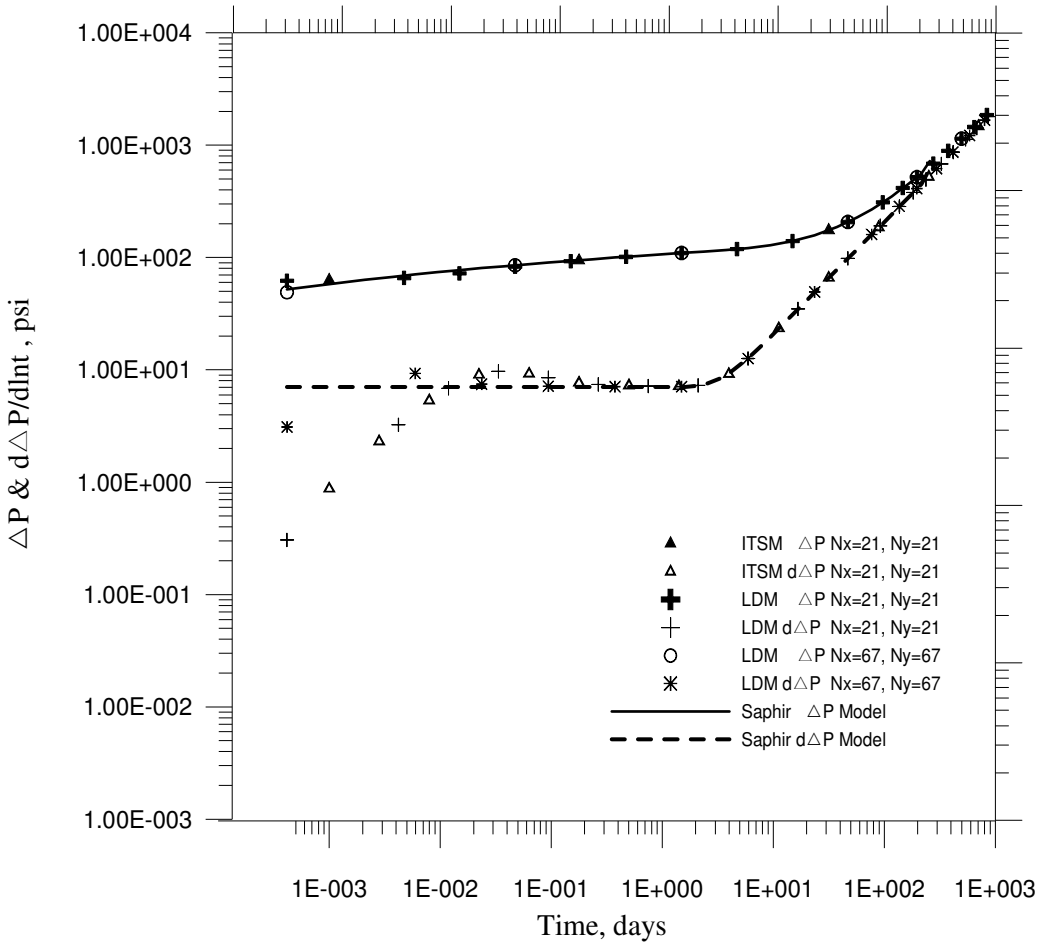


Figure 3.2 : Delta pressure and delta pressure derivative at the production well simulation results compared with ECRIN model results (21x21 and 67x67 gridblocks used for simulation).

In Table 3.12, we compare the accuracy of simulator flowing bottom-hole pressure at the well for both ITSM and LDM at a simulation time of 10.569 days in terms of relative error (reference to the pressure solution of ECRIN's analytical solution) for four different gridblock sizes. It looks like the LDM method predicts slightly more accurate than does the ITSM. However, for all practical purposes, the differences are negligible.

Table 3.12: Comparing accuracy of solutions as a function of the total number of gridblocks used in simulation at 10.569 days

Number of Gridblocks	Relative Error,% (LDM)	Relative Error,% (ITSM)
11x11	6.95E-04	1.68E-02
21x21	1.14E-03	1.53E-02
51x51	1.28E-03	1.48E-02
67x67	1.30E-03	1.48E-02

All results given above for the ITSM and LDM were generated from the MATLAB code using full and sparse matrix storage options and direct solver for the ITSM and MATLAB codes for the LDM. To further demonstrate how important is to efficiently store the coefficient matrix A to run large size problems, we report some of the results obtained from Fortran77 codes written by Dr. Onur for the ITSM and LDM, which use efficient storage and iterative solver for the ITSM and publically available FORTRAN codes of Sidje (1988) for the LDM. Table 3.13 presents the computed CPU times from MATLAB and Dr. Onur's Fortran77 codes for both the LDM and ITSM as a function of total number of gridblocks considered in simulation at a simulation time of t = 1000 days. The CPU times reported in Table 3.13 represent the total time spent to compute the pressure at 13 time points as given in the first columns of Tables 3.2-3.11. All runs for LDM were made by using a Krylov basis of m = 30. For the purpose of completeness, we report that the number of nonzero elements in the coefficient matrix A for the case of Table 3.11 where the grid system

of 151x151 is 113404, whereas the total elements of the matrix *A* including zero elements is equal to 5.2×10^8. As can be seen from the results of Table 3.12, storing the coefficient matrix A efficiently not only improves computational times but also allows running large size grid problems.

Table 3.13: Comparing CPU times from MATLAB and FORTRAN codes for ITSM and LDM for different gridblock numbers at t = 1000 days.

Number of Gridblocks	CPU Time, seconds (LDM, MATLAB with sparse storage)	CPU Time, seconds (ITSM, MATLAB with sparse storage)	CPU Time, seconds (LDM, FORTRAN)	CPU Time, seconds (ITSM, FORTRAN)
11x11	0.025	0.005	0.6406	0.00E-00
21x21	0.064	0.014	2.828	0.0156
51x51	22.334	0.091	35.531	0.156
67x67	26.468	0.207	83.859	0.359
101x101	-*	-*	342.94	1.141
151x151	-*	-*	1534.56	3.984
251x251	-*	-*	-*	19.406

*.Memory limit of the programs exceeded.

3.1.2 Variable rate production case

In this section, we compare the LDM and ISTM methods in terms of both computational efficiency and accuracy for a single-well producing at a variable rate-production history. The reservoir and fluid properties are the same as given in Table 3.1 and location of the well in the reservoir system is the same as illustrated in Figure 3.2. The flow rate history at the well is shown in Figure 3.3. As can seen from Figure 3.3, we consider two production periods with increasing flow rates followed by shut-

in (zero flow rate) period. We use a 67×67 grid system to simulate the pressures. For the LDM, a Krylov basis of m = 30 is used. For all results given here, we use the Fortran77 codes developed by Dr. Mustafa Onur.

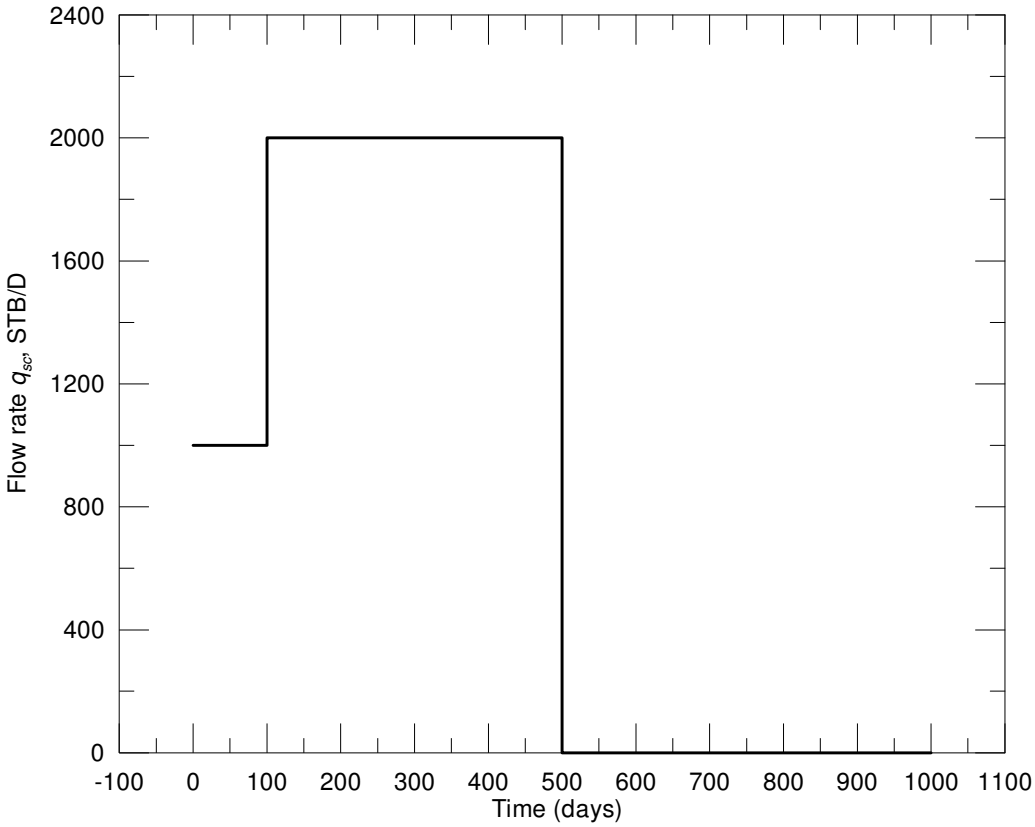


Figure 3.3 : Flow rate history for variable production rate example.

The total number of logarithmically spaced time points used for both simulators are the same and equal to 150. We simulate pressures in the time interval from 1×10^{-3} to 1×10^{3} days. We have deliberately used smaller time steps at times where the flow rate changes to accurately capture the pressure changes near these times. Figure 3.4 compares the flowing bottom-hole pressures computed from LDM, ITSM and ECRIN's analytical solution. As can be seen from Figure 3.4, the pressures from LDM and ITSM agree (or match) quite well with the corresponding pressures computed from the analytical solution of ECRIN's software. Although LDM and ITSM provide essentially the same values of flowing bottom-hole pressures, their computational efficiency or performance is quite different. The total CPU spent to obtain the solution (shown as circular data points in Figure 3.4) by the LDM for a

total of 150 time points was equal to 652 seconds, while the total CPU spent to obtain the solution (shown as triangular data points in Figure 3.4) was only 2.14 seconds. For both methods, we use the same tolerance on the accuracy of the pressure solution. Note that LDM is about 300 times slower than the ITSM for this example. Similar to the constant rate simulation cases, this example clearly shows that the LDM is also not computationally efficient for simulating variable rate cases and cannot beat the ITSM. Hence, the LDM cannot be an alternative to the ITSM for reservoir simulation problems studied here.

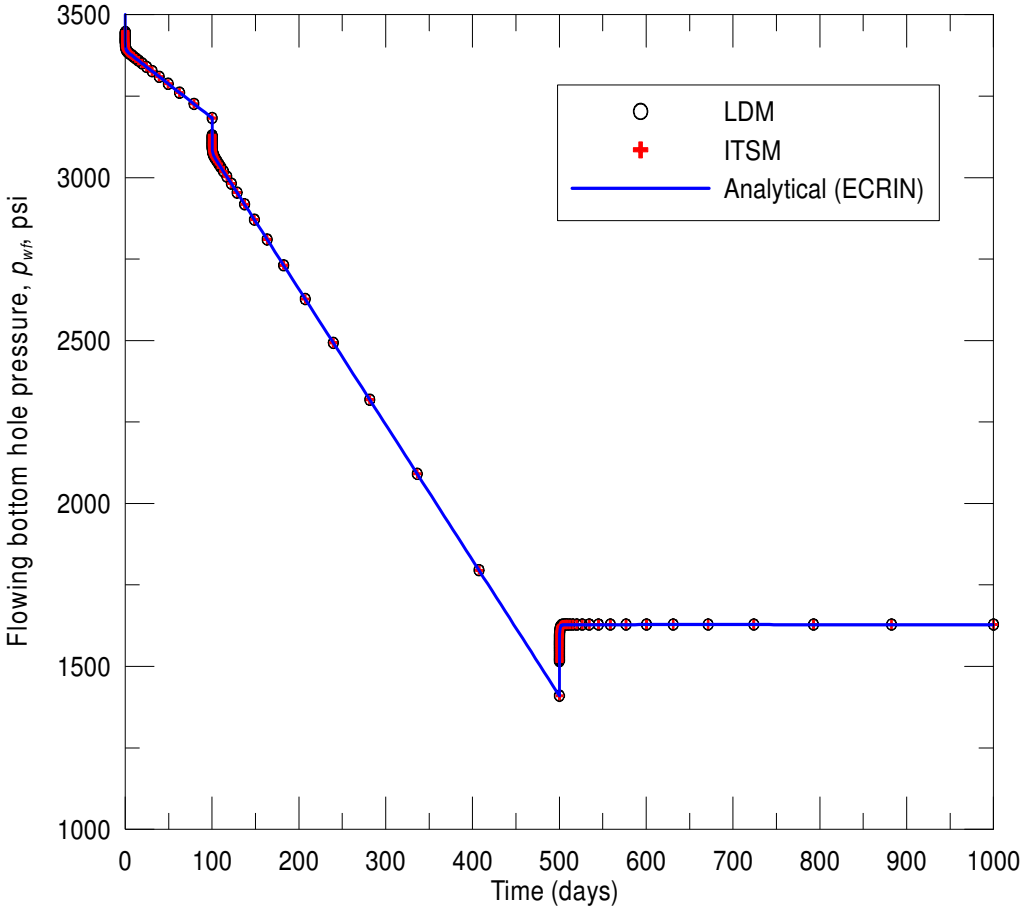


Figure 3.4 : Comparison of flowing bottom-hole pressures computed from LDM, ITSM, and ECRIN, for variable production rate example.

4. CONCLUSIONS

In this study, 2-D numerical reservoir simulation for slightly compressible fluid of constant viscosity and constant compressibility with multiple wells was developed with two different mathematical methods; namely, Lanczos Decomposition Method (LDM) and Conventional Implicit Time-stepping Method (ITSM). The simulators were written with MATLAB programming language.

Validity and accuracy of 2-D flow simulator were examined by comparing the simulation results with that of obtained using ECRIN commercial software. Simulation results give fairly good match ECRIN results.

In this study, the grid block size did not exceed over 67x67 since the simulations were performed with full matrix storage which uses a high memory in the system.

For the simulator developed with the LDM, the accuracy decreases as the number of grid blocks increases. For instance, pressure values for Nx=67 Ny=67 grids show less accuracy against those of 11x11 and 21x21. On the contrary, the ITS Method give more accurate results as the number of grids increases.

In the LD Method, the CPU time required per time step to solve the model is greater than the elapsed CPU time for conventional ITS method solution and the CPU time increases as the length of the time step becomes larger.

Although the coefficient matrix was formed using full storage and direct solver for the ITSM, the simulator developed with this method works more efficiently compared to the one developed with the LDM.

The simulator developed with the ITSM using FORTRAN codes which use efficient storage and iterative solver is much more efficient than the LDM using sparse storage matrix solver.

Similar to the constant rate simulation cases, the LDM written with FORTRAN for variable flow rate was not efficient compared to ITSM. Therefore, the LDM can not be an alternative to the ITSM for the simulations performed in this study.

REFERENCES

Alpak, F., Torres-Verdin., C., Sepehrnoori, K., Fang, S., Knizhnerman, L., 2003. An extended Krylov subspace method to simulate single-phase fluid flow phenomena in axisymmetric and anisotropic porous media, *Soc. Pet. Eng. J.*, 40, pp. 121-144.

Aziz, K. and Settari, A., 1979. Petroleum Reservoir Simulation, Applied Science Publishers Ltd., Wilmette, IL.

Crichhlow, H.B., 1977. Modern Reservoir Engineering – A Simulation Approach, Prentice-Hall Inc., Englewood Cliffs, NJ.

Dalton R., and Mattax, C., 1990. Reservoir Simulation, Monoghraphy **13**, SPE, Henry L. Doherty Series, Richardson, Texas, USA, pp. 1-5, 133-139.

Druskin, V. and Knizhermann., 1995. Krylov subspace approximations of eigenpairs and matrix functions in exact and computer arithmetic. Num. Lin. Alg. Appl., **2**, pp. 205-217.

ECRIN Ver.4.10.02., 2008. software, Sophia Antipolis, France: Kappa Engineering.

Ertekin, T., Abou-Kassem, J., and King, G., 2001. Basic Applied Reservoir Simulation, 1st edition, SPE, Henry L. Doherty Series, Richardson, Texas, USA, pp. 1-10, 75-104.

Knizhnerman, L., Druskin, V., Liu, Q., and Kuchuk, F., 1994**.** "Spectral Lanczos Decomposition Method for Solving Single-Phase Fluid Flow Porous Media," Numerical Methods for Partial Differential Equations, **10**, pp. 569-580.

MATLAB Ver.R2008a., 2008. software, MA, USA: The MathWorks, Inc.

Onur, M., 1997 "Numerical Reservoir Simulation'', ITU Mines Faculty, Petroleum and Natural Gas Engineering, Istanbul, Turkey, 1997-2002.

Peaceman, D.W., 1983. Interpretation of Well-Block Pressures in Numerical Reservoir Simulation With Nonsquare Grid Blocks and Anisotropic Permeability, *Soc. Pet. Eng. J.*, June, pp. 531-569.

Ross, L.S., 1974. Differential Equations, 2nd edition, John Wiley & Sons, Inc., New York, p. 46.

Saad, Y., 1992. Numerical Methods for Large Eigenvalue Problems. John Willey & Sons, Manchester University Press.

Sidje, R. B., EXPOKIT, 1998. A Software Package for Computing Matrix Exponentials, ACM Trans., Math. Softw., 24, pp. 130-156.

Welty, D. H. and Meijerink, J. A., 1981. An Improved Formulation and Solution Method for Single Phase Flow Problems, *Soc. Pet. Eng. J.*, June, p.p. 289-295.

APPENDICES

APPENDIX A : Computer Program ‘’LDM simulator Matlab code‘’

APPENDIX B : Computer Program ‘’EXPOKIT solver MATLAB code’’

APPENDIX C : Computer Program ‘ITSM simulator Matlab code’’



CURRICULUM VITA

Candidate's full name : Koray Yılmaz

Place and date of birth : Ankara 11.25.1982

Permanent Address : Arif Molu Anadolu Teknik Lisesi Loj. No:4
Kocasinan/Kayseri

University attended : Istanbul Technical University