NEW SOLUTION FOR WELL TEST ANALYSIS IN RESERVOIRS WITH PERMEABILITY DISCONTINUITIES

ATSUSHI SAGAWA

A Thesis Submitted for the Degree of Doctor of Philosophy

Department of Petroleum Engineering, Heriot-Watt University

3 May, 2000

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEGDEMENTS

I wish to express my appreciation to Dr. David R. Davies and Professor Patrick W. M. Corbett for their advice and encouragement during the course of this research. I am indepted to Dr. Leif Larsen and Professor Michael A. Christie who reviewed this dissertation and participated in the examination committee. Geoquest RT is thanked for allowing use of Eclipse numerical simulator. All the authors of the paper [106] thank Norsk Hydro for the use of the well test data and Professor George Stewart for his review of the work. I also appreciate Professor George Stewart for allowing me to attend the course 'Advanced Welltest Analysis' held in Edinburgh Petroleum Services Limited, which provided me many ideas on strip-wellbore model.

Finally, I acknowledges the financial support of Japan National Oil Corporation and Teikoku Oil Corporation.

ABSTRACT

This study presents new methods to analyse transient pressure response of reservoirs with permeability discontinuities. In particular, vertical wells in reservoirs with a high permeability lens (Model A), and horizontal wells in reservoirs associated with a high permeability zone (Model B) have been analysed. A multi-layered composite model was used to analyse model A, and a new analytical model was developed for model B. Both models use the concept of pseudo-steady state interlayer crossflow. Type curves have been generated from both models to show various responses of such reservoirs.

Sensitivity studies have generated new insights into the typical pressure response of model A reservoirs. It was shown that the pressure response is somewhat similar to that of layered reservoirs, radial composite reservoirs and fractured reservoirs. The high permeability lens, if penetrated by the well, has a significant, positive impact on production. Field data was analysed to show how reservoir properties can be determined using this model.

Model B was developed using a strip source and Fourier-Laplace transforms. A novel analysis of linear systems having two permeability regions - both penetrated by a wellbore - was made. The results show that the presence of the high permeability zone complicates the pressure response, making it difficult to capture typical characteristic features. However, interpretation can be done by matching the synthetic data generated by this model with field data.

This dissertation is dedicated to my parents,

Yoshio and Masae Sagawa,

and the memory of my brother,

Tomonori Sagawa.

CHAPTER 1

INTRODUCTION

Most reservoirs show heterogeneities of rock properties not only vertically, but also horizontally. Such "three-dimensional" heterogeneities can be inferred from outcrops of ancient sedimentary rocks or modern sedimentary environments developing in all over the world. They occur according to the differences of grain size, grain shape, sorting, packing, or grain orientation [109]. The differences are controlled primarily by depositional environments, and secondarily by diagenetic processes or structural movements. In particular, the diagenetic processes are important for carbonate reservoirs.

Core logging is a direct method to identify reservoir heterogeneities, and to obtain rock properties, such as permeability and porosity. Wireline well logging indirectly gives us information on the rock properties along a wellbore, whether the well is vertical or horizontal. The core and the wireline logs of a vertical well reveal vertical facies succession and the transition of the rock properties. Similarly, such logs of a horizontal well may unveil the lateral transition of the facies and the rock properties.

These logging techniques examine geology only at the wellbore. By contrast, seismic surveys can cover large portions of the reservoir. In fluvial or deltaic systems, sand bodies are often discontinuous laterally. In such environments, sand body correlation using the well logs is often impossible since well intervals are larger than the lateral continuity of sand bodies. The seismic survey is a promising technology to understand the geology between wells, and has been increasingly used for reservoir characterisation and planning of horizontal wells [105].

Well test analysis is also useful for detecting heterogeneities if combined with geological knowledge obtained from other sources as described above [51]. However, if such information is not available, an analysis of well test data can be misleading due to the non-unique nature of the well test interpretation [76, 34]. Another difficulty for determining heterogeneous reservoir characters in well test analysis arises from the averaging process of pressure response [87]. Because the pressure propagation is a diffusion phenomenon, the pressure response is smoothed even though reservoirs have sharp discontinuities of rock or fluid properties. Thus, to obtain small-scale distribution of reservoir properties for a detailed reservoir simulation or an initial guess of stochastic reservoir modelling may

not be appropriate in well testing. However, it is still important to identify relatively large reservoir structures (large-scale permeability anomalies, structural faults, large-scale reservoir geometries, etc.). Such information on the reservoir properties or structures can be used for further detailed reservoir characterisation studies.

Transient pressure analysis of homogeneous reservoirs can be done using a simple semi-log pressure versus time plot. Even in the presence of heterogeneities, it is possible to analyse well test data by special plots (e.g., a pressure versus fourth-root plot for bilinear flow in a hydraulically fractured reservoir [19]). However, in that case, it is necessary that the flow regime is clearly recognised and a simple analytical expression describing the flow regime is available. If such conditions are not satisfied, interactive or automatic parameter estimation is performed by matching synthetic data with field data [3]. Either an analytical approach or a numerical approach should be adopted to generate synthetic well test history. In both cases, a reservoir model (the rock properties, the reservoir geometry, etc.) must be specified beforehand to generate a synthetic pressure history. However, insufficient information is often available from other studies. Thus, trial and error is necessary to reach a certain model.

Many analytical solutions for various heterogeneous reservoirs have been developed - e.g., faulted reservoirs [124, 6, 1], lateral composite reservoirs [11, 15, 64], radial composite reservoirs [107, 86, 101], layered reservoirs without crossflow (commingled reservoirs) [73, 117], layered reservoirs with crossflow (crossflow reservoirs) [103, 56, 84], naturally fractured reservoirs [120, 27, 44], hydraulically fractured reservoirs [94, 48, 19], compartmentalised reservoirs [35, 112, 99], etc.

Although analytical solutions introduce some assumptions (e.g., constant reservoir thickness throughout the reservoir, no-deviation wellbore, etc.), they provide us useful information if their limitations are well-understood. In numerical simulations adopting the finite difference method (FDM) or the finite element method (FEM), the grid blocks around the wellbore and also time steps should be small enough in order to capture very large transient at very early times [97]. Such grid refinement is also necessary if large permeability contrasts exist between adjacent grid blocks where the pressure gradient is

very large. However, this is not required for analytical solutions. Another benefit of the analytical solutions is the ease of reservoir modelling since the gridding of the reservoir model is no longer necessary. In addition, an automatic type-curve matching algorithm can be easily incorporated into the analytical solutions. Numerical simulations are required if a reservoir geometry or a distribution of reservoir properties is roughly known, but is so complicated that an analytical solution is not available. If reservoir fluid is multi-phase, numerical well test simulations are of great use.

In practice, it is often assumed all the layers detected by the core or the well logging are continuous to the same lateral distance due to the lack of any other data. This is actually a layered reservoir system, and pressure analysis of such a system can be done using many commercial welltest software packages. However, in fluvial or deltatic systems, it is possible that some "layers" are embedded in the reservoir. The objective of the first part of this thesis is to examine the well test response of such reservoirs (called "lens reservoirs" in this study).

Pressure analysis of the lens reservoirs has not been studied in the past despite their frequency of occurrence. One of a few papers we can find is the work by Corbett et al [22]. In this study, the pressure behaviour of a high permeability lens penetrated by a vertical well in a low permeability reservoir matrix was investigated. It was shown that pseudo-radial flow period can be observed at late times and the pseudo-skin obtained from the analysis of late-time pressure data becomes small as the lens permeability or the size of the high permeability lens increases. Some similarities to a fractured reservoir case were observed, and the lens was termed "a pseudo-fracture channel".

It is understandable that the lens reservoirs are general cases of layered reservoirs, composite reservoirs, and horizontally fractured reservoirs. The knowledge of these reservoirs is helpful in analysing lens reservoirs. The lens reservoir will behave as a layered reservoir if the external limit of the lens approaches the external limit of the reservoir. If the thickness of the lens approaches the thickness of the whole reservoir, pressure response may be similar to that of the composite reservoirs. The pressure response of horizontally fractured reservoirs can be obtained if the permeability contrast between the lens and the

reservoir becomes large. Well productivity will be improved when the high permeability lens is penetrated by a well, just like the fractured well case.

In this thesis, the work by Corbett et al. [22] is extended, and new insights are presented by close observation of pressure and flow rate behaviours. An analytical solution developed by Gomes and Ambastha [41] is used to generate the lens reservoir response. This has the several advantages over the numerical simulation as discussed above. The solution makes use of "semi-permeable wall model" developed by Polubarinove-Kocina [92], and applied to petroleum problems by Gao [38] and Gao and Deans [39]. The model makes the mathematical problem manageable by reducing the dimensions of partial differential equations.

Another objective of this research is a horizontal well case where the well penetrates several regions having different rock properties, similar to the lens reservoir case. Horizontal wells are being widely applied in all over the world. Technological advance on horizontal well drilling has made it possible to drill longer horizontal wells, and its application is increasing since it is often economical to produce from several reservoir regions with only one well. Numerous papers on pressure analysis of a horizontal well in homogeneous reservoirs (e.g., Refs. [42, 20, 24, 88, 100, 85, 62]) and in layered reservoirs (Refs. [116, 63]) have been published. A good summery of horizontal well pressure analysis can be found in Ref. [61]. However, none of them considered the well penetrating several regions with different rock properties.

The situation is the same in the literature on a slanted well in layered reservoirs, which may include the horizontal well as a limiting case. Kuchuk and Wilkinson [65] presented an analytical solution of a slanted well, but only in commingled reservoirs. Lu [75] developed an analytical solution of a slanted well in crossflow reservoirs based on transmission and reflection method presented by Kuchuk and Habashy [63]. However, it is difficult to apply his method to more general problems since the transmission and reflection method is not suitable to describe more than one dimensional wave propagation. Larsen [70] also presented an analytical method to calculate the pressure response using the semi-permeable wall model and composite-fracture formulation. In order to obtain the

boundary condition at the wellbore, a fracture-wellbore correction term calculated from a uniform-flux fracture solution and a deviated well solution was used in his model. In the horizontal well case where the well penetrates several heterogeneous regions, it is quite difficult to obtain the correction term for converting a uniform-flux fracture solution to a horizontal well solution. This hampers the application of his model to the vertically and horizontally heterogeneous reservoirs.

An analytical solution for the pressure analysis of a horizontal well in heterogeneous linear reservoirs was developed to generate synthetic pressure and fractional flow rate responses. The solution adopts the semi-permeable wall model used for the lens reservoir model and strip-wellbore model presented by Goode and Thambynayagam [42]. Unlike the method of the wellbore-reservoir connection in Larsen's paper [70], the horizontal strip-wellbore is directly incorporated in the model as a fracture since the pressure behaviour of a horizontal well is quite similar to that of a fractured well. An early-time linear-flow behaviour, which may be seen for a stimulated well in a carbonate reservoir, can be simulated unlike the line source approximation of a wellbore.

In this thesis, the pressure and the flow rate responses of heterogeneous reservoirs where a vertical or horizontal well may penetrate several regions with different rock properties are discussed. The effects of the size and rock properties of the regions are investigated, and the method for analysing well test data is proposed for both of the reservoirs. The effect of a high permeable sand body penetrated by a well have not been reported fully before. The recognition of the position and the size of heterogeneous sand bodies is important for planning a well trajectories for future drilling as well as for the prediction of future well performance.

In Chapter 2, the geology of heterogeneous reservoirs which may give rise to the lens reservoirs and the heterogeneous linear reservoirs is discussed. In addition, papers on pressure analyses related to those reservoirs are reviewed. Chapter 3 discusses the pressure transient response and the fractional flow rate response in a new type of reservoirs including a high permeability lens intersected by a vertical well. The method to calculate a well productivity is presented. Finally, a field case is analysed for estimating various reservoir

properties. In Chapter 4, an analytical model for the elongated heterogeneous linear reservoir is developed. The pressure and the flow rate behaviours of reservoirs with a high permeability zone are discussed. Since the model developed in Chapter 4 is quite general, the further applications of the model are outlined in Chapter 5. The computer program used to generate the synthetic pressure or flow rate for the lens reservoirs is given in Appendix A. In Appendix B, the computer program developed for a horizontal well in a heterogeneous linear reservoir is provided.

CHAPTER 2

BACKGROUND WORK

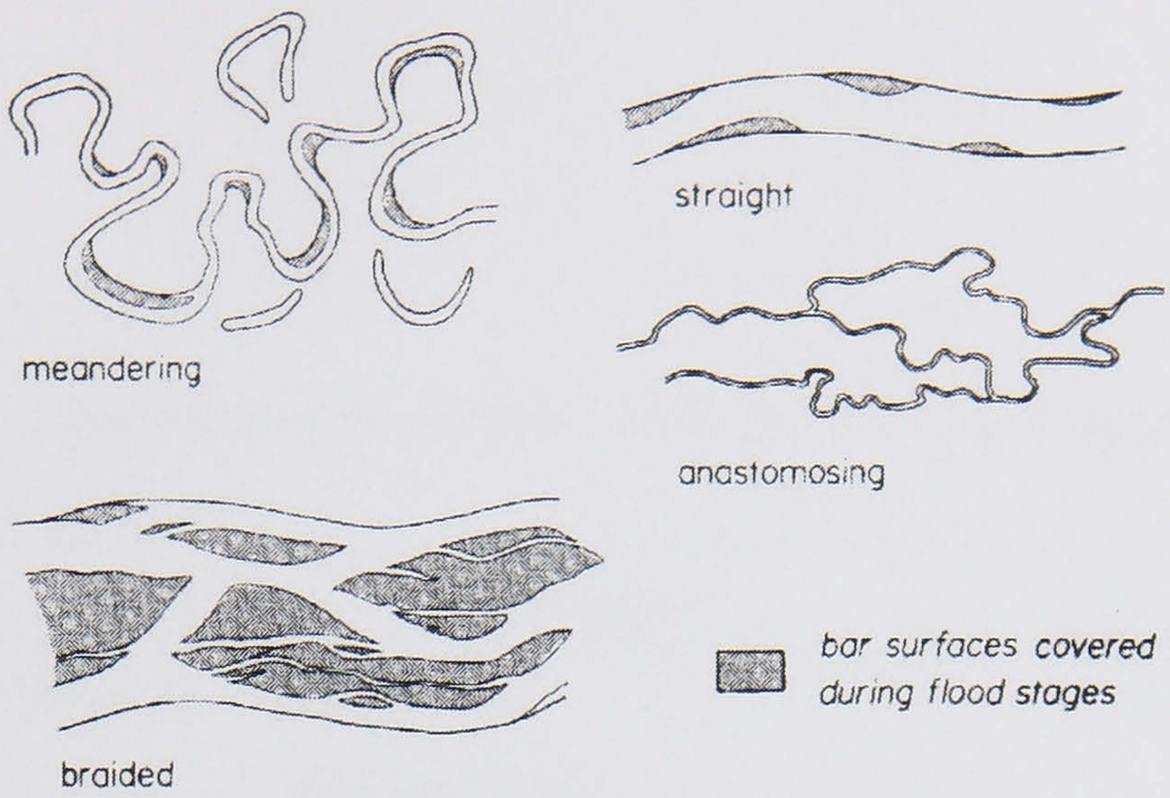


Figure 2.1: Principal types of rivers [80].

In the first section, the geological background of lens reservoirs and heterogeneous linear reservoirs are discussed by referring the published literature on aspects of sedimentological geology. Although numerous papers on pressure analysis in heterogeneous reservoirs have been published, the selected literature related to the semi-permeable wall model, the strip-wellbore model, the lens reservoirs, and the linear heterogeneous reservoirs is reviewed in the second section.

2.1 Geological Background

2.1.1 Lens reservoirs

Le Blanc summarised distribution and continuity of sandstone reservoirs referring to some 250 selected papers [71, 72]. The classification of modern environments of sand deposition by Le Blanc is given in Table 2.1. Among them, lens reservoirs can be often found in fluvial environments, in particular, in braided and meandering stream environments.

Braided rivers develop in the distal parts of alluvial fans where enough sediment is supplied [12]. Braiding occurs due to rapid, large fluctuations in river discharge, which causes mid-channel bars of coarse sediment to form (Fig. 2.1). A large variety of vertical and lateral facies successions can be generated by the lateral and vertical accretions of

Table 2.1: Classification of modern environments of sand deposition [72].

Continental	
Alluvial (Fluvial)	Alluvial fan Braided stream Meandering stream (includes flood basins between meander belts) Aeolian (can occur at various positions within continental and transitional environments)
Transitional	
Deltaic	Birdfoot-lobate (fluvial dominated) Cuspate-arcuate (wave and current dominated) Estuarine (with strong tidal influence)
Coastal interdeltaic	Barrier island (includes barrier islands, lagoons behind barriers, tidal channels, and tidal deltas) Chenier plain (includes mud flats and cheniers) Transgressive marine
Marine	
Note:	Sediments deposited in shallow marine environments, such as deltas and barrier islands, are included under the transitional group of environments above.
Deep marine	

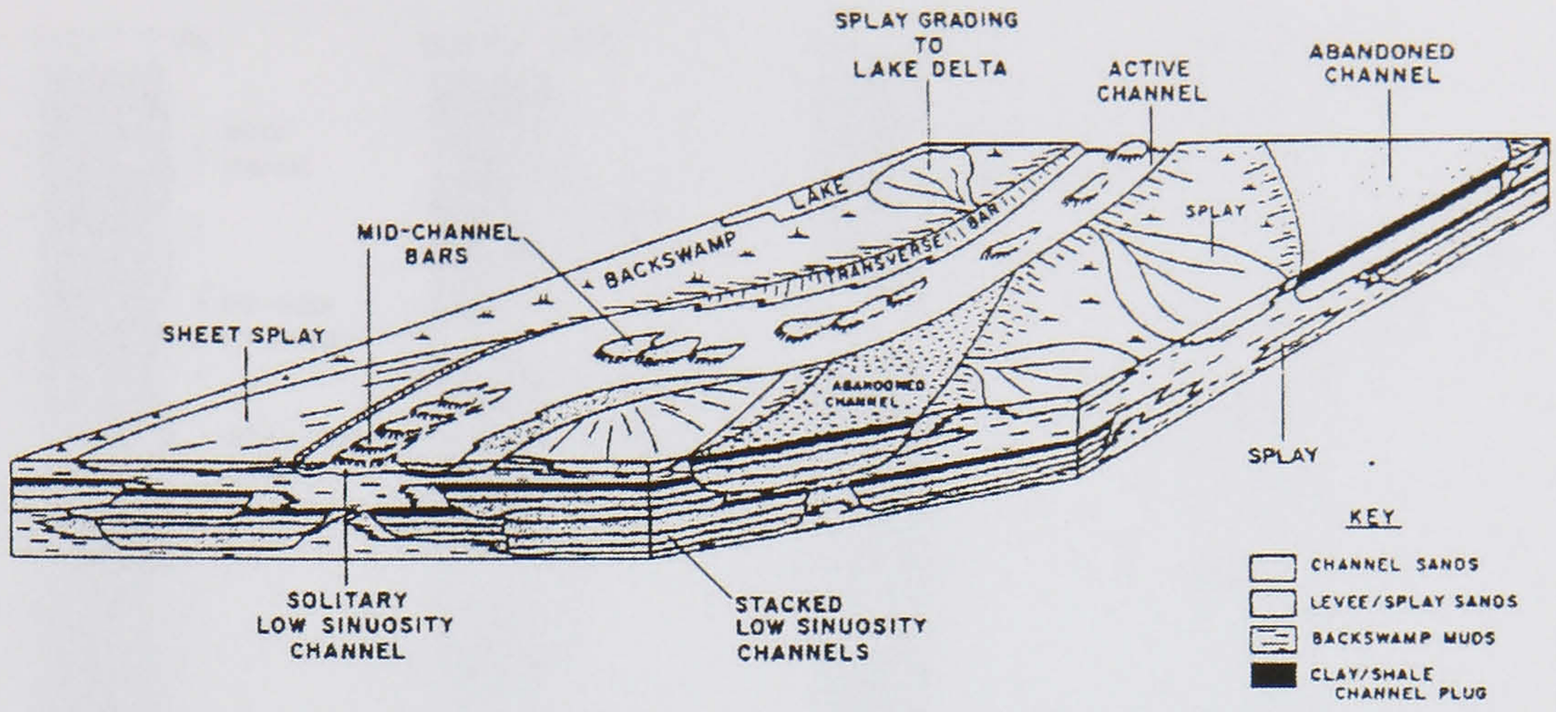


Figure 2.2: Channel characteristics and environmental relationship in the braided stream system [25].

the braided river deposits. River channels may be filled by aggradation under decreasing current velocity. During periods of flooding, beds are superimposed and new channels will develop (Fig. 2.2).

Typical stratigraphic models for braided streams are shown in Fig. 2.3. The Scott-type model consists mainly of roughly horizontally bedded gravels and minor sand wedges, while the Bijou Creek-type model consists of superimposed flood sand sediments. The Donjek-type and the Platte-type models include fining-upward sequences which consist of sands and gravels.

As Sneider et al. [110] discussed in their paper, porosity and permeability of any sandstone or conglomerate generally correlate with grain size and sorting. Fig. 2.4 shows experimental data on the effects of them in unconsolidated clay-free sands performed by Beard and Weyl [9]. Finer and better sorted sands often show higher porosities. Permeability increases as the grain size becomes larger and the sorting becomes better. Similar relations can be found in actual reservoir rocks [110, 121]. Thus, it can be expected that the contrasts of grain size as shown in Fig. 2.3 give rise to permeability contrasts which may affect well test response.

Mid-channel bars often form longitudinal or linguoid bars. The longitudinal bars are oriented with their long axis roughly parallel to current flow, while the linguoid bars are

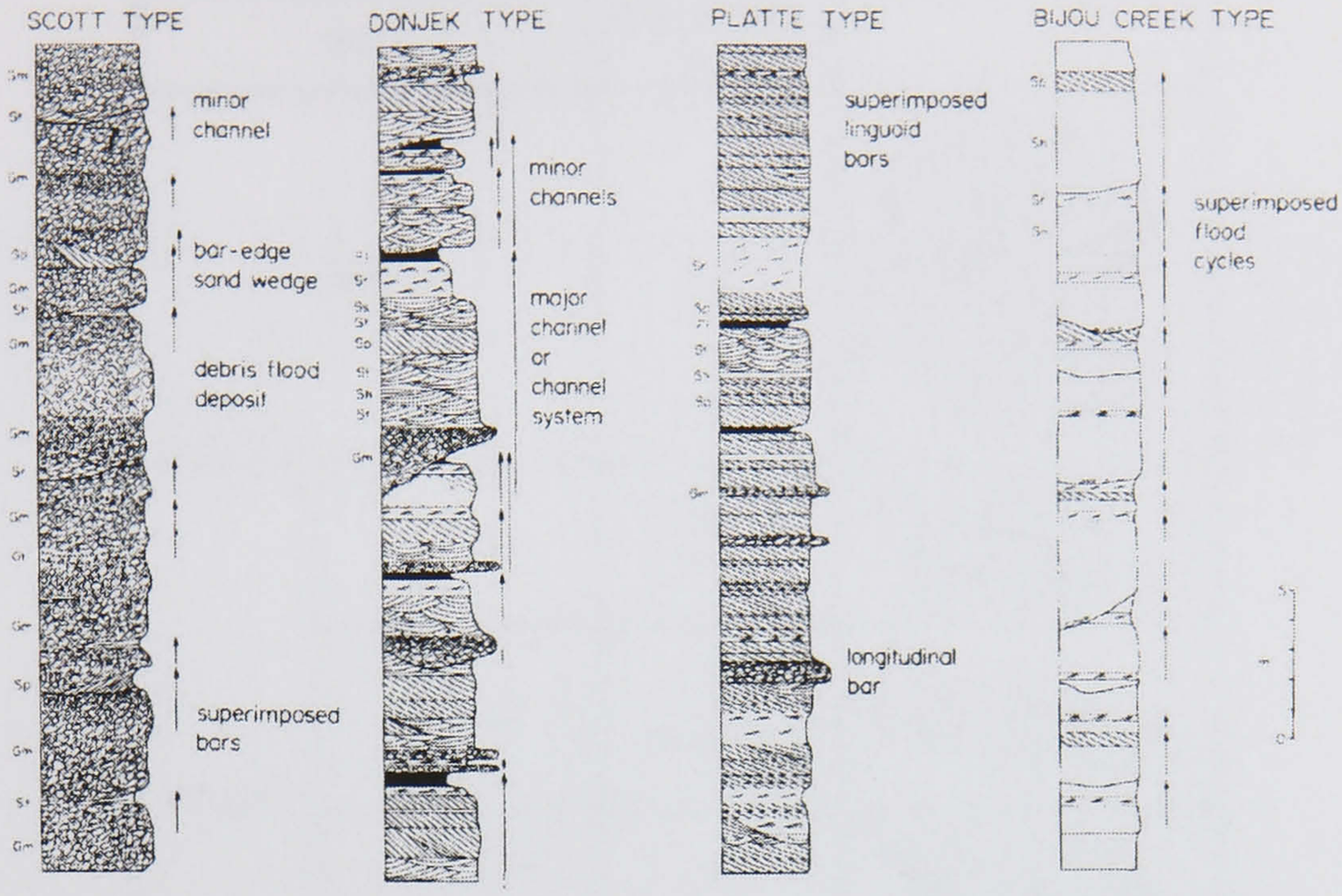


Figure 2.3: General stratigraphic models for the sandy braided stream [80].

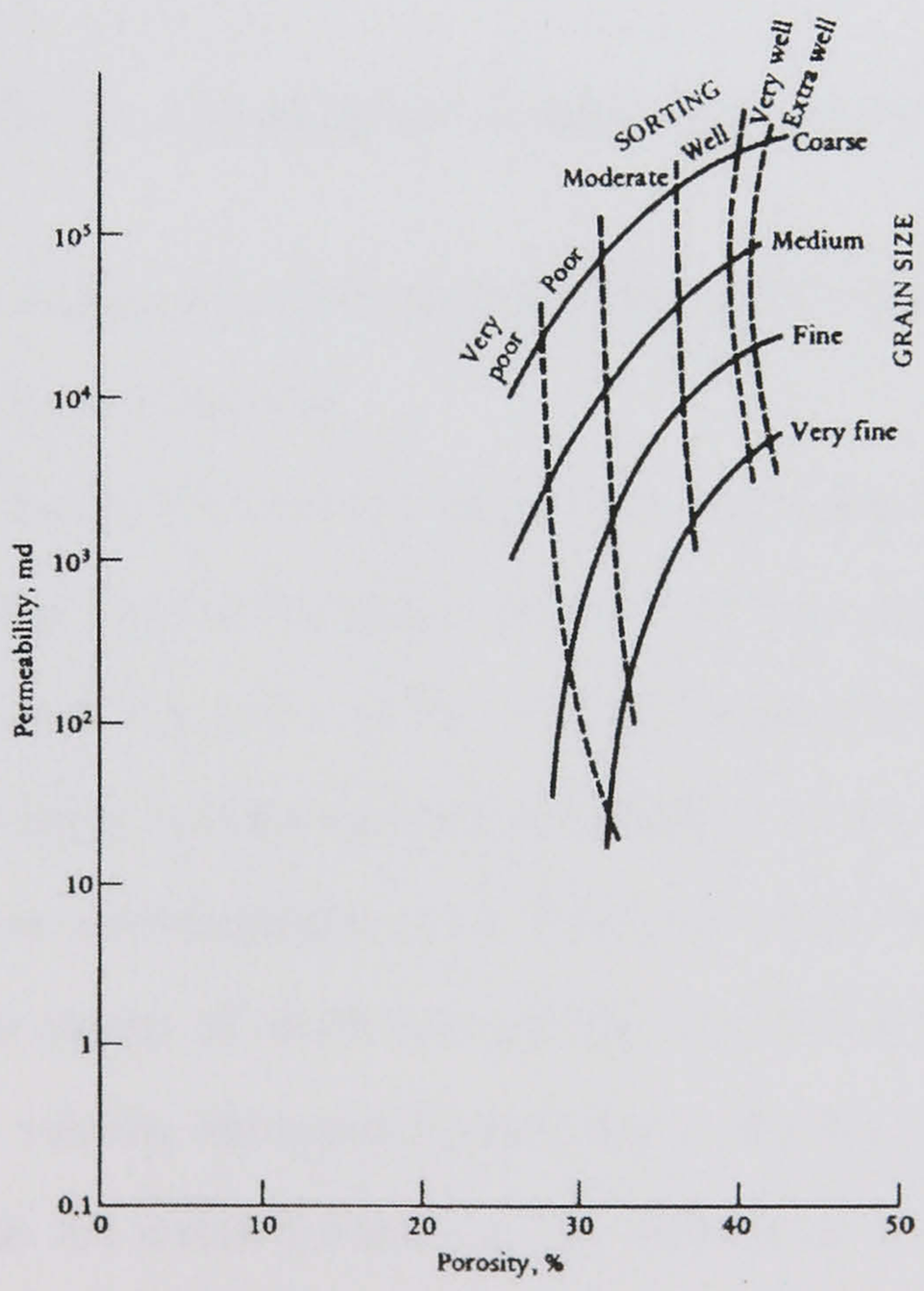


Figure 2.4: Porosity / permeability plot of various grain size and sorting mixes of unconsolidated clay-free sands [9, 83].

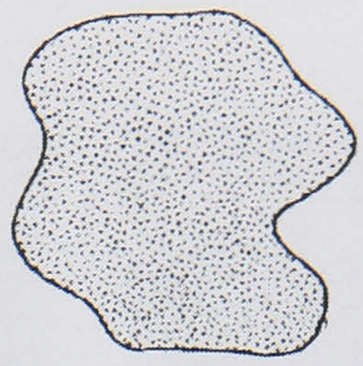
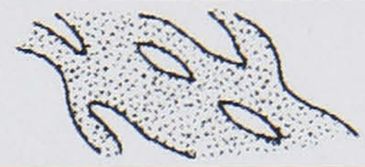
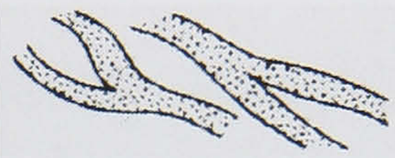
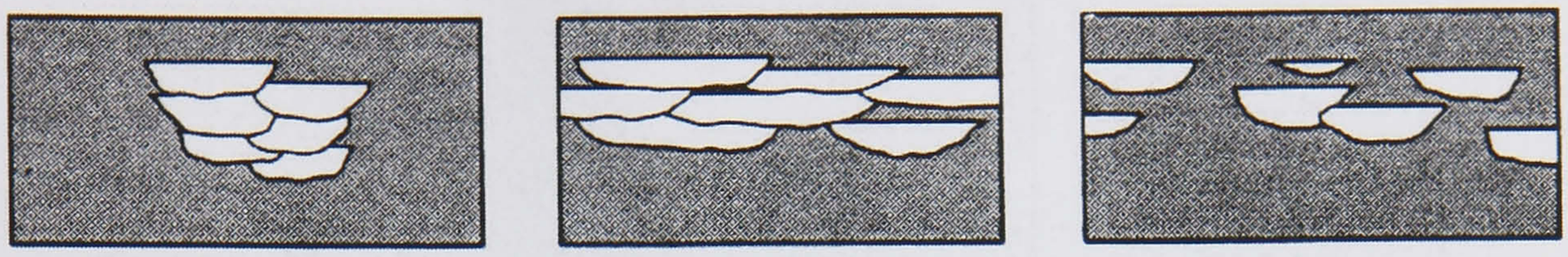
Name		Length-width ratio	
Sheet		~ 1-1	
Elongate	Belt	Sheet with holes	
	Dendroid	> 3-1 bifurcating	
	Ribbon, or shoestring	> 3-1	
	Pod	< 3-1	

Figure 2.5: Classification of sand body geometry [93].

rather lobate in areal configuration. Potter [93] classified the continuity of sands according to their length-width ratios (Fig. 2.5).

The length-width ratios of the longitudinal bars are generally classified to ribbon or shoestring type, however, those of the lingoid bars are sheet or pod type. Vertical or lateral stacking of channels (as in Fig. 2.6 A or B) with the longitudinal bars or the lingoid bars produces a large sand body containing high permeability lenses as shown in Fig. 2.2.

In meandering river environments, point bars and levees develop along the channel (Fig. 2.7). Point bars consist of sands and gravels, and develop on the insides of meander bends. Since flow velocity decreases upward due to the helicoidal secondary currents, finer-grained materials are swept further up the surface to show typical fining-upward sequence [21] (Fig. 2.8). Levees commonly develop on the outer margins of bends during flooding periods. They are generally thinner and finer grained than point bar sediments [26]. Outer banks of the bends are predominantly sites of erosion and inner banks



A B C

Figure 2.6: Classification of sand body continuity (A) vertically stacked (multi-storey) (B) laterally stacked (C) isolated [50].

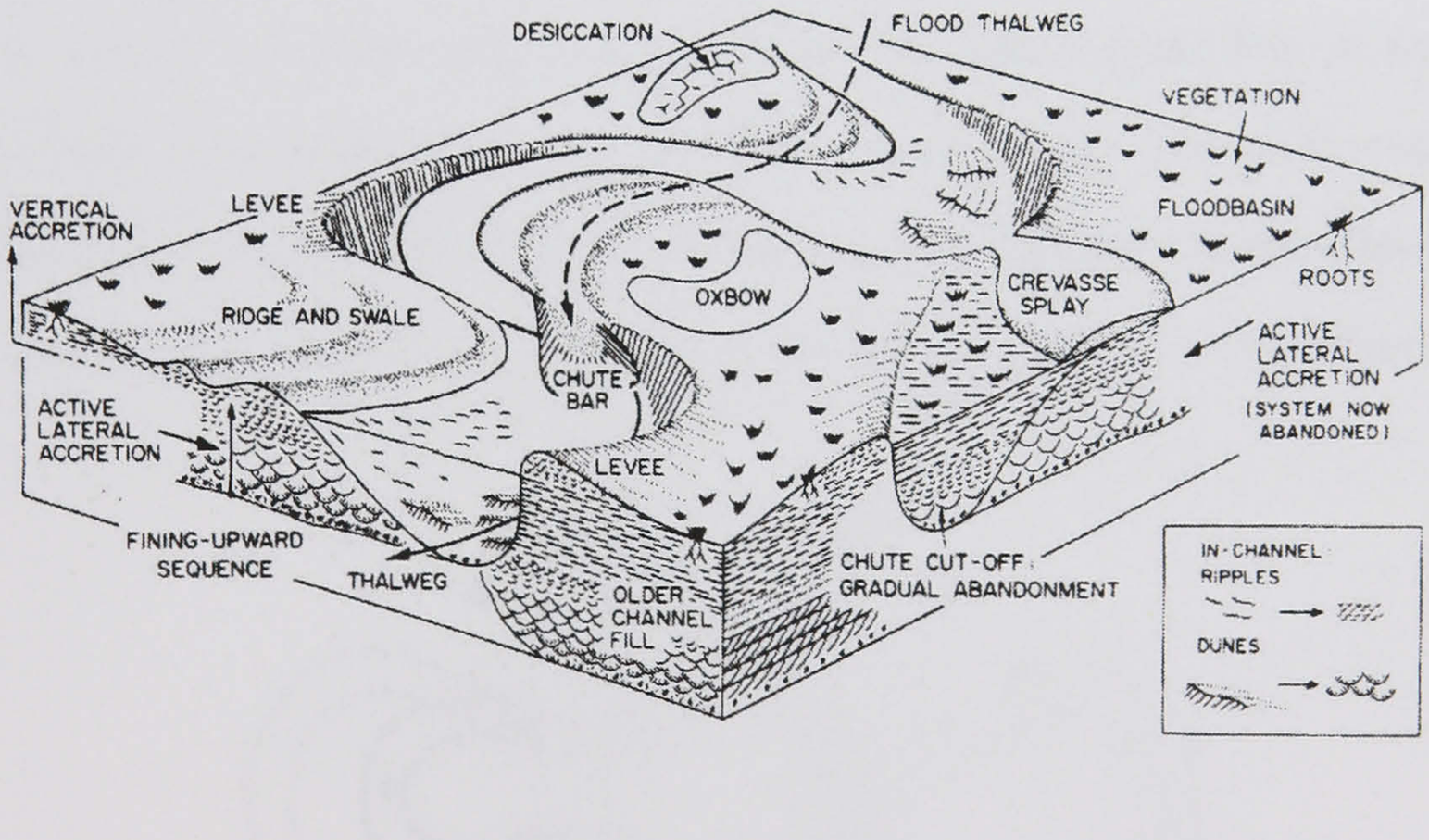


Figure 2.7: Channel characteristics and environmental relationship in the meandering stream system [119].

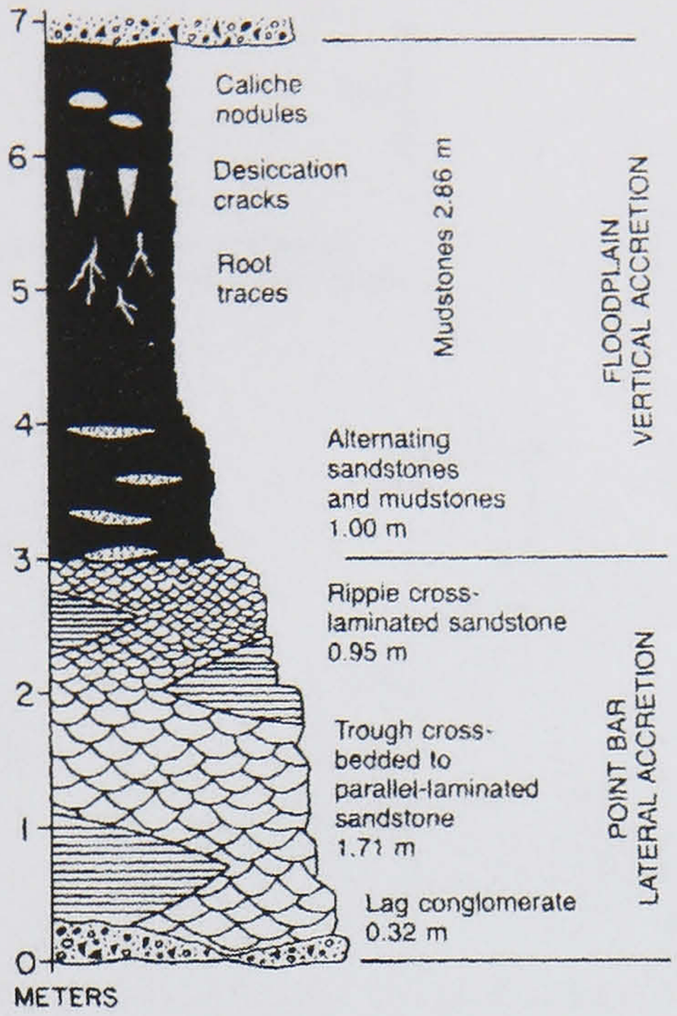


Figure 2.8: Fining-upward sequence in the meandering river deposit [119].

are predominantly sites of sediment accumulation, which causes lateral accretion as shown in Fig. 2.7.

The meandering of the river commonly gives rise to the cut-off of the river as shown in Fig. 2.9. Chute cut-off develops a new channel through a point bar on the inside of a meander bend, and isolates the part of the bar as an island. The more common neck cut-off takes place in the final stage of meander-loop development. It abandons the entire meander bend at the neck of it, and isolates the bar between two river channels. Since

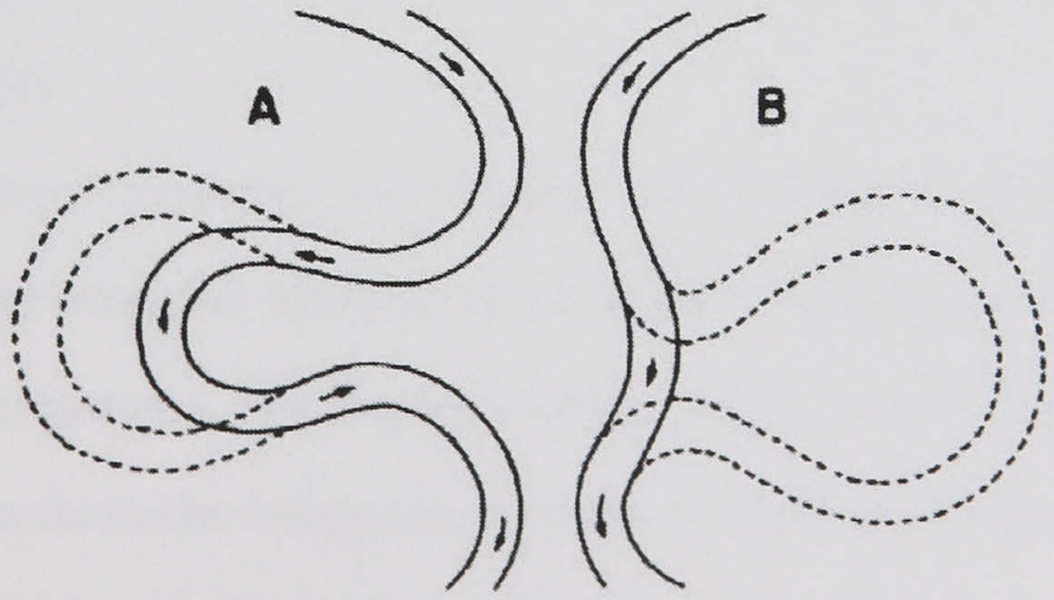


Figure 2.9: Types of channel cut-off (A) chute cut-off (B) neck cut-off [5].

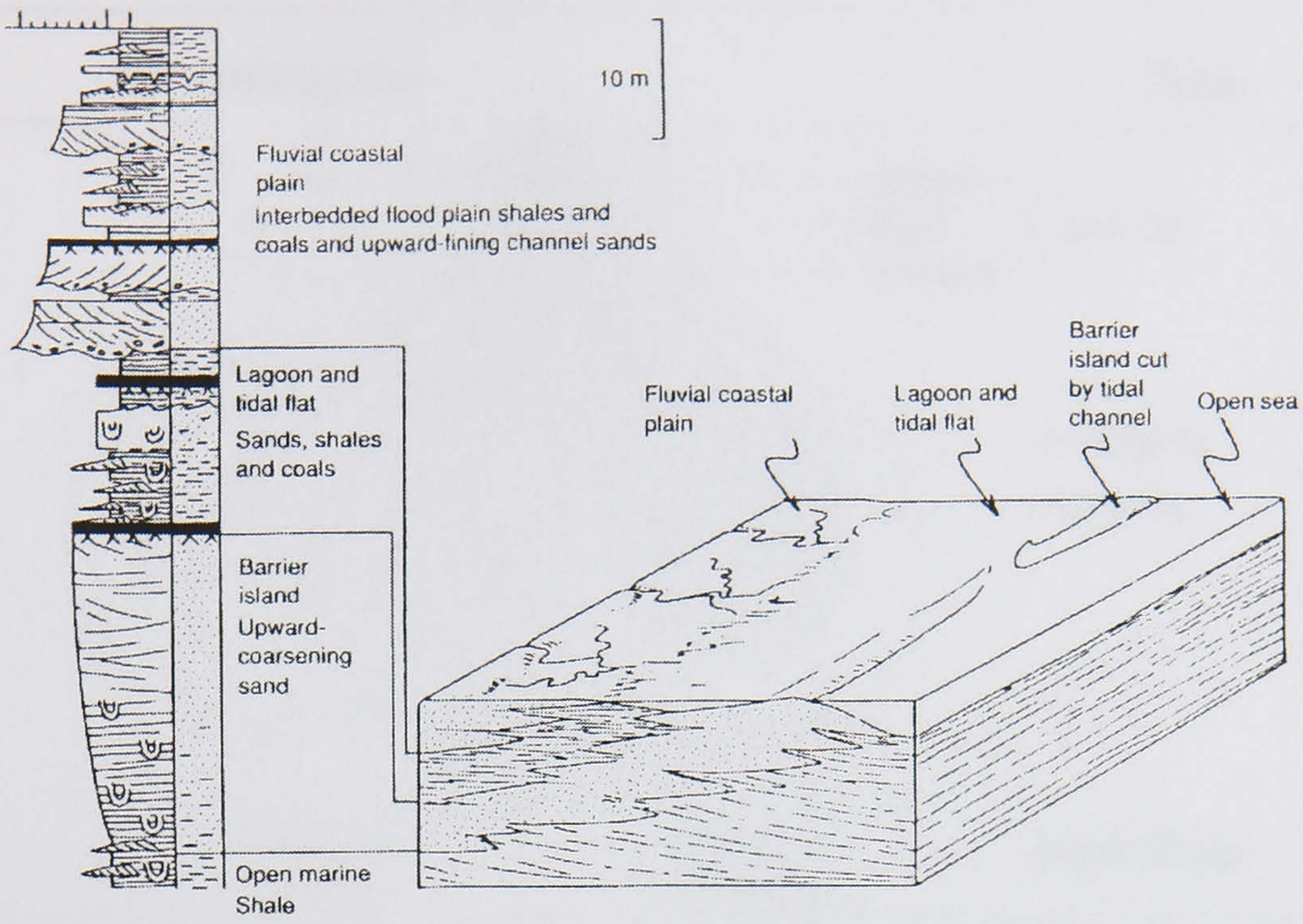


Figure 2.10: Environments, facies and sedimentary sequence in the barrier island environment [108].

the point bars consists of sands and gravels, the vertical accretion of sediments may give rise to a reservoir including high permeability lenses of the isolated point bars.

2.1.2 Heterogeneous Linear Reservoirs

Braided and meandering stream environments, barrier island environments, and turbidite sediments in shallow marine environments may be modelled well as linear systems. In each environment, the sediment shows various internal heterogeneities just like those explained in the previous section.

In the braided stream systems, the vertical and lateral stackings of the low sinuosity river channels may develop the heterogeneous linear reservoirs as shown in Fig. 2.2. In the meandering river systems, the presence of the point bars and the levees alongside of the main channel can form the heterogeneous linear systems as shown in Fig. 2.7.

In barrier island environments where marine currents are strong enough to redistribute land-derived sediment, linear bars parallel to shorelines are formed [108] (Fig. 2.10). In fluvial coastal plains with gentle gradients, meandering rivers will develop, and fine-grained

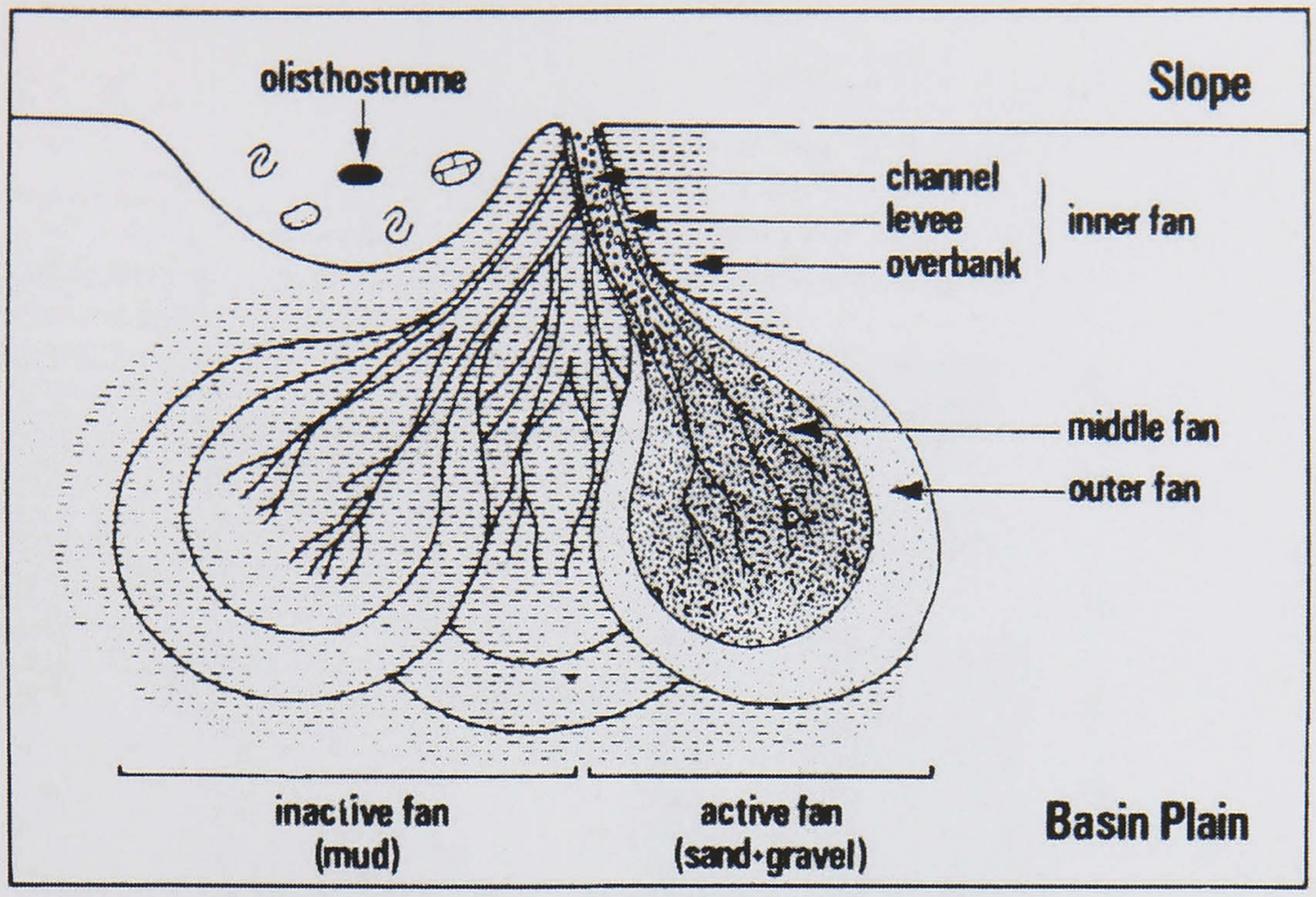


Figure 2.11: Areal distribution of fan elements [102].

sediments dominate over upward-fining channel sand sequences. The coastal plain passes transitionally seaward into lagoons and tidal flats which are generally fine-grained. Barrier islands also pass seawards, which causes to deposit upward-coarsening grain-size profiles. The depositions of sediments in the coastal plains, lagoons, tidal flats, and barrier islands will show lateral linear heterogeneities as shown Fig. 2.10.

Most turbidites are deposited in broad, cone shape submarine fans as shown in Fig. 2.11. Some may occur in the lower-reaches of submarine canyons and farther seaward in deep-sea channels [12]. The structure of the submarine fans is analogous to that of alluvial fans, and the turbidites show similar linear geometries to river channels. Individual turbidite beds range in thickness from a few millimetres to several meters. However, the successions of them form beds up to kilometres in thickness [113]. The facies model of turbidites in Fig. 2.12 indicates the possible occurrence of vertical heterogeneities in terms of permeability and porosity due to the contrast of the grain size. The lateral stacking can also occur in addition to the vertical stacking as shown in Fig. 2.13. It will show linear deposits with internal heterogeneities.

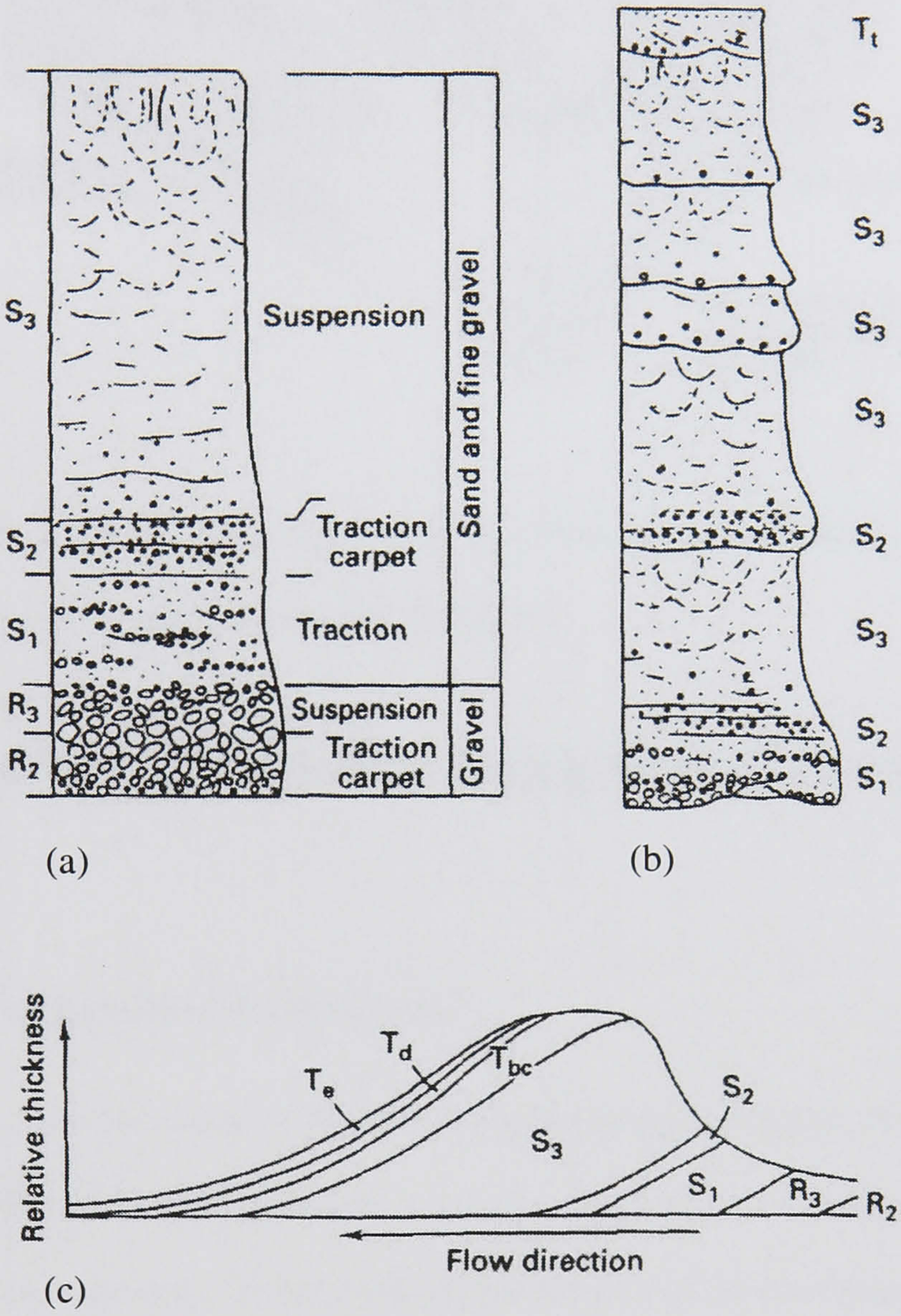


Figure 2.12: Traction and suspension deposits of high-density coarse-grained turbidity currents (a) higher sandy parts of the bed (b) deposits of a surging high-density flow (c) sequence along the length of a flow [74].

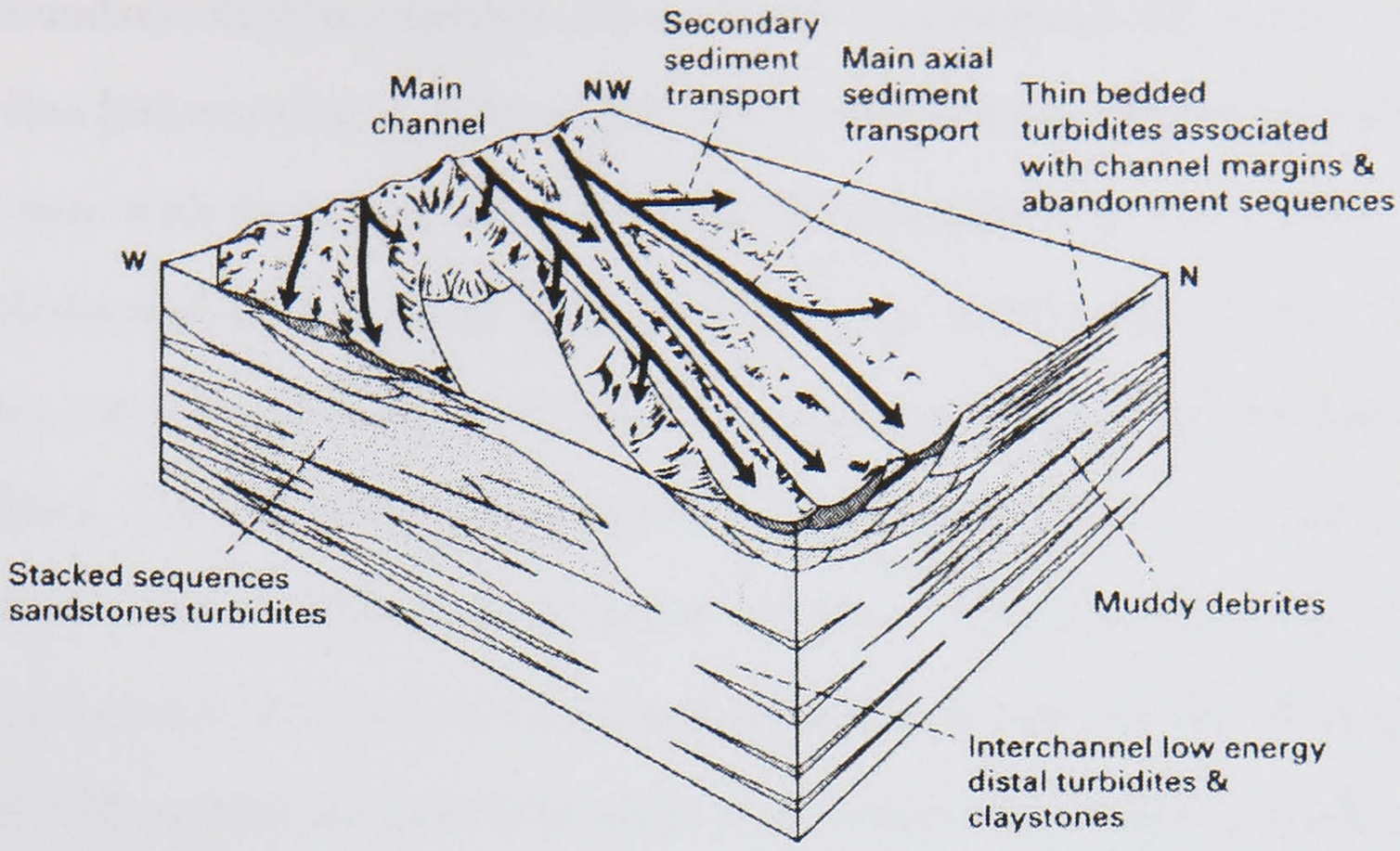


Figure 2.13: Depositional model for the Palaeocene Forties submarine-fan system in the North Sea [122].

2.2 Background of Well Test Analysis in Heterogeneous Reservoirs

2.2.1 Semi-Permeable Wall Model

Layered reservoirs are the most common heterogeneous systems observed in petroleum geology. Since 1960, over 50 papers on pressure analysis of layered reservoirs have been published [41]. The behaviour of layered reservoirs is well documented in Ref. [98].

The analysis of layered reservoirs with formation crossflow was first presented in the petroleum literature by Russel and Prats [103], and Katz and Tek [56]. Russel and Prats obtained an analytical solution using Hankel and Laplace transforms, while Katz and Tek obtained it using separation of variables and Fourier transform both for a constant-pressure production condition. Niko [84] developed an analytical solution for two-layer crossflow reservoirs with skin and wellbore storage using a constant-rate production condition. The problem was solved in Laplace domain rigorously based on the work by Jacquard [55]. Although these solutions clarify the behaviour of the layered reservoirs with crossflow, they are quite complicated to apply to general n layer reservoirs. Such complexities come

from the boundary condition between layers which is mathematically quite intractable.

Later, Gao [38] and Gao and Deans [39, 40] presented an analytical solution for multi-layer reservoirs with formation crossflow. The "semi-permeable wall model" which was originally developed by Polubarinova-Konica [92] was used in their paper. This model assumes the vertical resistance to flow is concentrated on the "semi-permeable walls" between layers. The model avoids the need of boundary conditions between layers and reduces the dimensions of three-dimensional diffusivity flow equation. The introduction of this model enables us to obtain analytical solutions for any number of layers easily.

Bourdet [13] applied the semi-permeable wall model to two-layer crossflow reservoirs and presented type-curves considering both skin and wellbore storage effects. Ehlig-Economides and Joseph [33] derived an analytical solution for general n layer reservoirs using the semi-permeable wall model. These papers imply the possibility of computerised automatic pressure history matching for layered reservoirs with crossflow.

It is also possible to apply the semi-permeable wall model to multi-layered composite reservoirs. Gao [38] presented a two-layer model where the upper layer is homogeneous, but in the lower layer, the horizontal and vertical permeabilities change after a certain distance from a well. Hatzignatiou et al. [53] also presented a two-layer reservoir model where each layer consists of an inner and an outer region separated by a mobility discontinuity. Anbarci et al. [7] presented a method to detect the location of flood fronts via pressure transient analysis using the semi-permeable wall model. Their two-layer model consists of three zones in each layer in order to represent the transition of the flood fronts in different layers. Gomes and Ambastha [41] extended the work by Anbarci et al. [7], and presented an analytical solution for multi-layered composite reservoirs. Their general solution can model many situations (e.g., infinite, closed, and constant-pressure outer boundaries, top and bottom constant-pressure boundaries, and partially penetrating wells). In addition, the distance to the outer boundary of each layer can be different.

These solutions using the semi-permeable wall model seem to work well. However, Larsen [67] drew attention to the use of the model without checking the effect of layer refinement. It was found that improved accuracy can be obtained by the layer refinement

using more layers in the mathematical model than that in the reservoir model. The accuracy depends on the contrasts of layer properties and boundary conditions at the wellbore. The semi-permeable wall model without the layer refinement is accurate if the contrast of layer properties is small, and layer pressure changes linearly in vertical direction. If the contrast becomes large, the double-porosity theory based on the assumption of parabolic pressure distribution [44] can yield a more accurate solution.

Larsen [70] applied the method to derive an analytical expression for a slanted well in a layered reservoir with or without crossflow. This will be reviewed later.

2.2.2 Strip-Wellbore Model

The strip-wellbore model was first used by Goode and Thambynayagam [42] to obtain an analytical solution of a horizontal well for transient pressure analysis. Although numerous papers have been published on the method to obtain synthetic pressure response of a horizontal well [20, 24, 88, 100, 85, 62], all of them use the line source approximation of a wellbore and instantaneous Green's function technique developed by Gringarten and Ramey [45]. However, the application of the line source well to heterogeneous reservoir cases is relatively complicated, and is only valid after the flow around the well becomes radial. It cannot capture the early-time linear-flow which may be observed when acidising treatments are applied to a well [126].

The strip-wellbore model is based on the finding of effective well radius of a fracture by Prats [94]. It was found the effect of the fracture can be represented by the effective well radius which is a quarter of the total fracture length for high-capacity fractures. Goode and Thambynayagam [42] employed the fact to replace the strip source by the effective radius. However, the relation was modified by considering the effect of anisotropy. Suzuki and Nanba [116] adopted the model for a horizontal well solution in a layered reservoir with crossflow.

The problem of the strip-wellbore model is the determination of equivalent pressure point which produces the same pressure as that for a constant wellbore pressure condition. This is because the strip-wellbore model assumes a constant production rate con-

dition along the wellbore (although the same thing can be said for the line source well approximation.). To cope with this problem, Goode and Thambynayagam [42] used the equivalent point measured at 86.6% from the edge of the well suggested by Gringarten et al. [49]. Muskat [82] used the 87.5% point for partially penetrating well cases, and Daviau et al. [24] and Rosa and Carvalho [100] used 85% and 84% points for horizontal well cases, respectively. Gringarten and Ramey [47] present a graphical method to determine the equivalent pressure point for a partially penetrating well in a semi-infinite reservoir. Abbaszadeh and Hegeman [2] provided a correlation of the equivalent pressure point for a vertical well in a semi-infinite reservoir with a constant-pressure boundary.

In general, the position of the equivalent pressure point varies with time. However, it is assumed that the point calculated from the stabilised pressure distribution at late times can be applied for all times [2]. In addition, the position of the equivalent pressure point changes according to the well and reservoir arrangements. Thus, the equivalent pressure point should be found for each problem.

To avoid these problems, another approach was presented first by Streltsova [114], who reported that for restricted flow entry cases, averaging pressure along the open interval yields a good approximation to the actual solution. Later, Wilkinson and Hammond [123] suggested that such a method becomes exact in the limit of vanishingly small wellbore radius. Kuchuk et al. [62] summarised the merits of using the pressure averaging method as follows. First, the method always accounts for the geometry of the problem (the well and the reservoir), while, the equivalent pressure point method does not. Second, the method is exact at all times in the limit of small wellbore radius, while, the equivalent pressure point is not. Third, the equivalent pressure point is calculated to give the final pseudo-steady pressure correctly. However, the pseudo-steady state is often not achieved in actual tests. From those reasons, the pressure averaging method seems more appealing than the equivalent pressure point method.

Although Goode and Thambynayagam [42] considered the effect of anisotropy for the effective wellbore radius, the effect of the elliptical flow around the wellbore has not been taken into account for their model. Kuchuk and Brigham [58] presented a complete solu-

tion for a well in homogeneous anisotropic systems. To solve the problem, a homogeneous and anisotropic reservoir was transformed to an equivalent homogeneous and isotropic reservoir by changing a scale along each axis in the diffusivity equation as presented by Earlougher [29]. Later, Kuchuk et al. [62] applied this insight to horizontal well cases, yielding an exact early-time radial flow solution in anisotropic systems. The effect can be incorporated in the strip-wellbore model without any difficulty [30].

2.2.3 Pressure Analysis in Lens Reservoirs and Other Related Reservoirs

Lens Reservoirs

Despite the possibility of reservoirs where lenses are embedded, only a few papers on pressure analysis for the lens reservoirs have been appeared.

Ayestaran et al. [8] investigated the effect of shale lenses embedded within a reservoir by numerical simulation. In their paper, the extent of the shale or siltstone barrier intersected by a well was estimated from transient well testing. Hatzignatiou et al. [53] examined pressure behaviour in multi-composite reservoirs. In their radial two-layer model, each layer consists of an inner and an outer region separated by a mobility discontinuity. To eliminate boundary conditions between layers, the semi-permeable wall model was adopted for their analytical solution. Three characteristic flow regimes (an early-time period when the system behaves as a two-layer reservoir without crossflow which consists of the inner region of the system, a transition period when the crossflow becomes fully developed, and a late-time period when the system acts as a homogeneous system with properties equal to the average properties of the outer region of the system) were found. Their results are useful since their model becomes the same as the lens reservoir case if the same properties are assigned to a inner zone and two outer zones. Mesmari [79] and Corbett et al. [22] studied the effect of high permeability lenses (instead of shale barriers in Ref. [8]) on pressure response using numerical simulation. It was found that, for a single lens case, the response is equivalent to that of a homogeneous reservoir having the same reservoir matrix permeability, but with a negative skin at late times (if there is no damage skin).

The absolute value of the negative skin becomes large as the permeability contrast or the size of the lens increases. However, the characteristic pressure behaviour (e.g., flow regimes, influential variables on the response, etc.) has not been fully investigated.

As discussed in Chapter 1, the lens reservoir will show the similar behaviour to layered reservoirs with crossflow, composite reservoirs, and horizontally fractured reservoirs. Although numerous papers on these topics have been published, selected papers are reviewed for the discussion in Chapter 3.

Layered Reservoirs With Crossflow

Russel and Prats [103, 104], and Katz and Tek [56] examined pressure behaviour of two-layer crossflow reservoirs. It was found that the system performance is the same as that of the reservoir without crossflow at very early times. At late times, it becomes the same as that of an equivalent single-layer reservoir with an arithmetic total permeability-thickness product and an arithmetic total porosity-compressibility-thickness product. Prijambodo et al. [96] used numerical simulation (the finite difference method) to show pressure response of a well producing at constant rate in two-layer reservoirs. It was found that the interlayer crossflow becomes dominant during a transitional period between the early and late time periods where the slope of the semi-log straight line changes. It was also found the effect of skin is substantial, and the pressure response changes considerably if the skin of each layer is different. Bourdet [13] provided pressure and pressure derivative type-curves including the effects of skin and wellbore storage, which show three distinct flow regimes. Bourdet [13] and Gringarten [44] mentioned that a two-layer crossflow reservoir with high permeability contrast between layers exhibits the same behaviour as that of a double-porosity reservoir. Kuchuk et al. [59] presented a multi-layer testing technique to determine layer permeabilities and skin factors uniquely by a number of flow tests which measure the wellbore pressure and flow rate at the top of each layer. Ehlig-Economides and Joseph [33] and Park and Horne [90] derived early-time and late-time limiting analytical forms of pressure and flow rate behaviour for general n layer reservoirs. Parameter estimation methods which can be automatically done using a computer were also pre-

sented. Larsen [69] discussed practical problems and possibilities of combined analyses of production logging tests and pressure transient data in layered reservoirs. It was found that contrasts in layer skin factors severely affect pressure and flow rate responses in reservoirs with or without crossflow and any PLT profile can be matched with any not too extreme permeability profile by adjustment of skin values alone. However, considering PLT data in addition to the transient pressure data reduces the non-uniqueness of the response significantly in many cases.

Radial Composite Reservoirs

Pressure analysis of a radial composite reservoir were discussed by Eggenschwiler et al. [31] and Satman et al. [107]. It was found that two semi-log straight lines of pressure, and a transition period between them can be observed for that systems. The slope of the first semi-log straight line reflects the mobility of the inner region, and the slope of the second line reflects that of the outer region. It was also found if the mobility contrast between the two regions is high, the pore volume of the inner region can be determined from the data taken immediately after the first semi-log straight line on a Cartesian plot. Olarewaju and Lee [86] presented an analytical solution for a well produced at either constant bottom-hole pressure or constant rate. The effect of phase redistribution at the wellbore was considered in their model. Chen et al. [18] reported an analytical solution for a three-region radial composite reservoir in order to apply it to gas reservoirs with edge-water drive. Rosa and Horne [101] presented an analytical solution for a well eccentrically located in a circular sub-region within a reservoir. The pressure response shows the similar behaviour to typical radial composite reservoirs except that the well eccentricity acts as a skin factor.

Horizontally Fractured Reservoirs

Transient pressure behaviour of a vertical well with a finite thickness, uniform-flux, infinite-conductivity horizontal fracture was examined by Gringarten and Ramey [46]. In the early-time period, the flow is of a storage type and a unit-slope straight line can be obtained on log-log pressure versus time plots. Then, vertical linear flow into the fracture

face develops and a half-slope straight line on the log-log plot can be observed. After a transitional period, pseudo-radial flow which is characterised by a straight line on semi-log pressure versus time plots is observed. Hartsock and Warren [52] investigated the effect of a finite-conductivity horizontal fracture on well productivity. An apparent skin which can be obtained by analysing the late-time semi-log pressure straight line was used for the analysis. It was found the apparent skin is given as a function of three dimensionless parameters related to fracture conductivity, reservoir thickness, and the radius of the fracture. Recently, Valkó and Economides [118] developed an semi-analytical solution of pressure transient behaviour for a vertical well intersecting a finite-conductivity horizontal fracture. It was found that at early times, the behaviour varies depending on the mutual effect of wellbore radius, reservoir thickness, and fracture conductivity. At late times, the semi-log straight line which reflects the reservoir permeability-thickness product can be obtained.

2.2.4 Pressure Analysis in Linear Composite Reservoirs

Pressure analysis of a vertical well in elongated linear homogeneous reservoirs is well discussed by Ehlig-Economides and Economides [32]. The line source solution for a radial flow model and the image well technique were used to obtain an analytical solution for the systems. It was found that drawdown and buildup behaviours are identical to those of the conventional radial flow systems at early times. However, at late times after a transition period, linear flow characterised by a half-slope straight line on log-log pressure versus time plots develops. For a heterogeneous reservoir case, Bourgeois et al. [15] developed an analytical solution of a vertical well in reservoirs where a main linear channel is bounded laterally with finite or infinite width levees using successive Laplace and Fourier transforms. It was found that pressure derivative behaves as a homogeneous reservoir at the beginning. Then, it shows an upward trend when the poor mobility levee zone starts contributing to the flow. It was also found that the slope of a plot of pressure versus square-time in the late-time region varies according to an averaged permeability obtained by weighting zone-permeabilities with the width of each zone. Based on mate-

rial balance considerations, Stewart and Whaballa [112] developed a method to calculate pressure behaviour of a vertical well in a compartmentalised reservoir which is divided into several regions by partially communicating discontinuities. An analytical solution for the transient pressure response was derived using the concepts of boundary pressure time delay and desuperposition. The pressure behaviour of a stacked linear channel model in which the upper channel supports the pressure for the lower channel through partially communicating boundary was demonstrated in their paper.

Pressure analysis of a vertical well in a laterally composite reservoir was investigated by Bixel et al. [11]. In their paper, an analytical solution of a vertical well located near a linear discontinuity was developed. Permeability, porosity, compressibility, and viscosity can have different values on each side of the discontinuity. It was found that the early-time behaviour of the drawdown curve does not reflect the discontinuity. Thus, it is possible to calculate permeability and skin value from that period. If mobility contrast is large, long shut-in times may be required to extrapolated to a correct static reservoir pressure.

Yaxley [124] presented an analytical model which describes the effect of a partially communicating fault on pressure behaviour. The partially communicating fault was modelled as a vertical, semi-permeable barrier through which the fluid leakage rate is proportional to the instantaneous pressure difference across the barrier. The model allows unequal thickness on opposite sides of the barrier. From type curves generated by his solution, it is possible to estimate formation transmissibility and the transmissibility of the barrier itself. Ambastha et al. [6] extended Yaxley's model, and developed an analytical solution of a composite strip reservoir with a partially communicating fault using the image well technique. Their model is quite flexible and a well position is arbitrary within the strip reservoir. It was found that analysis of interference tests in a composite reservoir is not straightforward since the response depends on the property contrasts and the location of the observation well. Kuchuk and Habashy [64] developed an analytical solution for calculating pressure response of a vertical well in laterally composite reservoirs. Their solution is based on the principles of reflection and transmission, and allows any number of composite regions to be modelled.

For a horizontal well in homogeneous and anisotropic linear reservoirs, Goode and Thambynayagam [42] developed an analytical solution using successive Laplace and Fourier transforms. Four flow regimes (early-time radial flow, intermediate-time linear flow, late-intermediate-time radial flow, and late-time linear flow) were defined, and simplified equations for each flow regime were derived. These equations enable us to calculate reservoir properties by special plots designed for each flow regime. Kuchuk and Habashy [63] developed an analytical solution for a horizontal well in infinitely large layered reservoirs using the same reflection and transmission method as in Ref. [64]. It was reported that if the system has only a few layers, it may exhibit the transient characteristic features of the system. However, as the number of the layers increases, the behaviour of the system may not show any characteristic feature. A field case of a reservoir which has 19 layers was demonstrated. However, the application of their method to laterally as well as vertically heterogeneous reservoirs seems to be very difficult.

All papers reviewed above cannot deal with the general heterogeneous linear reservoir in which a well may penetrate several different permeability regions. However, Larsen's paper [70] gives us a good clue to how to solve the problems. In his paper, the dimensions of the governing diffusivity equation was reduced by introducing the semi-permeable wall model, and a slanted well in an infinitely large layered reservoir was modelled as composite fractures. The wellbore pressure can be obtained by considering correction term representing the difference of pressure between a slanted well segment and a uniform-flux fracture. A horizontal well case is included as a special case by considering a slanted well with limited flow entry. However, obtaining a correlation term for a horizontal well penetrating different heterogeneous regions is quite difficult. In this thesis, the use of the correlation term is avoided and the strip-wellbore is directly incorporated into the heterogeneous linear reservoir model described in Chapter 4.

2.3 Summary

In the first part of this chapter, the geological backgrounds of the lens reservoirs and the heterogeneous linear reservoirs were discussed through the literature survey. In the second part, the literature useful for the pressure analysis of the lens reservoirs and the heterogeneous linear reservoirs was reviewed. The insights obtained through this literature survey were summarised as follows.

1. Lens reservoirs can be observed in braided and meandering stream environments. In the braided stream environments, the lens may consist of a mid-channel bar of coarse sediment. In the meandering stream environments, it may consists of a point bar of coarse sediment isolated by the cut-off of a meander loop. Since permeability increases as the grain size of sediments becomes larger, the vertical accretion of sediments can give rise to a high permeability lens embedded in a reservoir matrix.

2. Heterogeneous linear reservoirs can be observed in braided and meandering stream environments, barrier island environments, and shallow marine turbidite environments. In the braided stream environments, vertical and lateral stackings of low sinuosity river channels with internal heterogeneities may develop the heterogeneous linear reservoirs. In the meandering stream environments, the presence of the point bars and the levees alongside of a main channel may give rise to the heterogeneous linear reservoirs. Linear structures parallel to shorelines can be formed in the barrier island environments, and those of the turbidite sediments similar to the river channels may develop in the shallow marine turbidite environments. The internal heterogeneities and the vertical / lateral stackings of those structures give rise to the heterogeneous linear reservoirs.

3. It has been reported that the pressure behaviour of the lens reservoir is equivalent to a homogeneous reservoir with the same reservoir matrix permeability and a pseudo-skin at late times. However, the characteristic pressure behaviour (e.g., flow regimes, influential variables on the response, etc.) has not been fully investigated before.

4. The analytical solution adopting the semi-permeable wall model presented by Gomes and Ambastha [41] can be directly used to generate the synthetic pressure response of the lens reservoirs. If a permeability contrast between the lens and the reservoir matrix is large, the layer refinement may be necessary to obtain an accurate pressure response.

5. It can be said that the lens reservoirs are general cases of layered reservoirs, composite reservoirs, and horizontally fractured reservoirs. Papers on those reservoirs should be reviewed since the knowledge on those reservoirs may be useful in understanding the pressure response of the lens reservoirs.

6. Although numerous papers on the linearly heterogeneous reservoirs (e.g., reservoirs with linear discontinuities, layered reservoirs, etc.) have been published, no papers on the pressure response of a vertical or horizontal well in the general heterogeneous linear reservoirs where the well may penetrate several permeability regions have been appeared. The semi-permeable wall model, which is widely used in analytical solutions of crossflow reservoirs, is useful to obtain an analytical solution for the heterogeneous linear reservoirs since the model reduces the dimensions of the three-dimensional diffusivity equation. The strip-wellbore model combined with the pressure averaging method (which is superior to the equivalent pressure point method in several points) is effective to implement the wellbore-reservoir connection for the heterogeneous linear reservoir model.

CHAPTER 3

HIGH PERMEABILITY LENS INTERSECTED BY A VERTICAL WELL

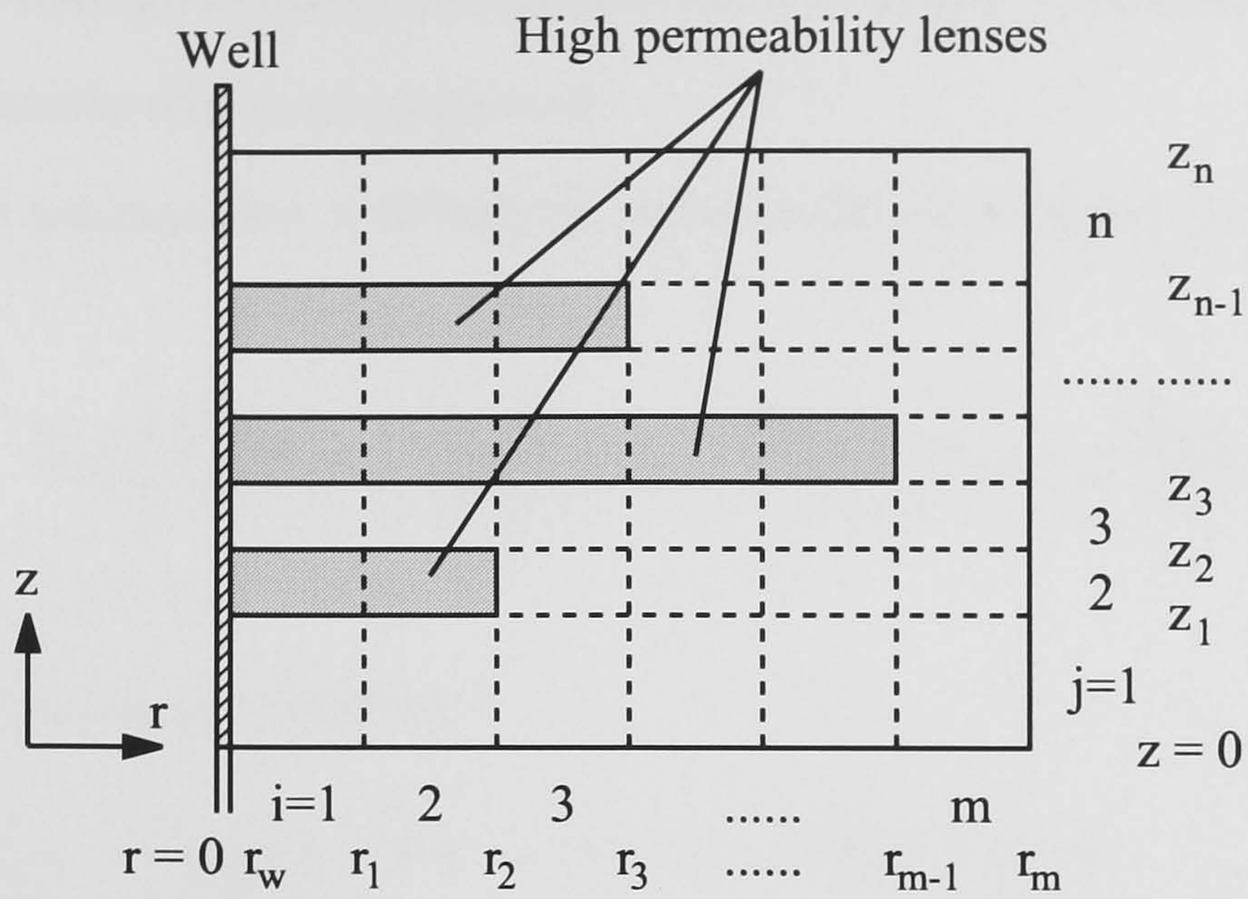


Figure 3.1: Multi-layered composite reservoir model.

In this chapter, the transient behaviour of reservoirs including a high permeability lens is discussed by examining pressure derivative curves and the fractional flow rates from the lens. The analytical model of a vertical well in multi-layered composite reservoirs developed by Gomes and Ambastha [41] is presented first to ensure completeness of this thesis. However, a different set of dimensionless variables based on the reservoir matrix properties has been used for the formulation presented here. In addition, a field case is analysed using the multi-layered composite model. Finally, cases where several lenses are embedded in a reservoir are discussed.

3.1 Model Description

3.1.1 Governing Equation

The reservoir model examined here is basically two dimensional. A schematic reservoir model in a cylindrical coordinate system is shown in Fig. 3.1. The reservoir is divided into m by n regions. Since reservoir properties can be specified in each region independently, high permeability lenses located at the centre of the reservoir are easily incorporated into the model. A slightly compressible, single phase fluid of constant viscosity is produced at

a constant rate from the wellbore. Initial pressure is constant in each part of the reservoir. Capillary and gravity effects are neglected.

Under these assumptions, a diffusivity equation for each region (i, j) is expressed in Darcy units as

$$k_{r,i,j}\left(\frac{\partial^2 p_{i,j}}{\partial r^2}+\frac{1}{r}\frac{\partial p_{i,j}}{\partial r}\right)+k_{z,i,j}\frac{\partial^2 p_{i,j}}{\partial z^2}=\phi_{i,j}\mu c_{t,i,j}\frac{\partial p_{i,j}}{\partial t}, \tag{3.1}$$

for $1 \leq i \leq m, 1 \leq j \leq n$.

The initial condition is given by

$$\lim_{t\to 0} p_{i,j} = p_{in} \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq n. \tag{3.2}$$

If the reservoir is confined by no-flow top and bottom boundaries, the boundary conditions are expressed as

$$\left.\frac{\partial p_{i,n}}{\partial z}\right|_{z=z_n} = 0, \tag{3.3}$$

$$\left.\frac{\partial p_{i,1}}{\partial z}\right|_{z=0} = 0, \tag{3.4}$$

for $1 \leq i \leq m$. If the top boundary is a constant pressure type (such as gas cap drive reservoirs), Eq. 3.3 is expressed as

$$p_{i,n}\big|_{z=z_n} = p_{in} \qquad \text{for } 1 \leq i \leq m. \tag{3.5}$$

Similarly, if the lower boundary is a constant type (such as bottom water drive reservoirs), Eq. 3.4 is replaced by

$$p_{i,1}\big|_{z=0} = p_{in} \qquad \text{for } 1 \leq i \leq m. \tag{3.6}$$

Wellbore boundary conditions for fully penetrating well may be written as

$$p_{wf} = p_{1,j}\big|_{r=r_w} - s_j r\left.\frac{\partial p_{1,j}}{\partial r}\right|_{r=r_w} \qquad \text{for } 1 \leq j \leq n, \tag{3.7}$$

$$qB = -C\frac{dp_{wf}}{dt} + 2\pi r_w \sum_{j=1}^{n}\left.\frac{k_{r,1,j}h_j}{\mu}\frac{\partial p_{1,j}}{\partial r}\right|_{r=r_w}, \tag{3.8}$$

where the second term of the right hand side in Eq. 3.8 represents the sum of the layer flow rate. The individual layer flow rate q_j is given by

$$-q_j = -\left.\frac{2\pi r_w k_{r,1,j}h_j}{\mu}\frac{\partial p_{1,j}}{\partial r}\right|_{r=r_w} \qquad \text{for } 1 \leq j \leq n. \tag{3.9}$$

If the well penetrates the reservoir partially, layers not open to flow should be eliminated from Eqs. 3.7 and 3.8. Alternatively, the wellbore boundary condition given by

$$\left.\frac{\partial p_{1,j}}{\partial r}\right|_{r=r_w} = 0 \qquad \text{for the layer } j \text{ not open to the flow,} \tag{3.10}$$

should be used instead.

If an outer reservoir boundary is infinite, the condition is expressed as

$$\lim_{r\to\infty} p_{m,j} = p_{in} \qquad \text{for } 1 \le j \le n. \tag{3.11}$$

Similarly, no-flow outer boundary and constant pressure outer boundary conditions are given by

$$\left.\frac{\partial p_{m,j}}{\partial r}\right|_{r=r_{m,j}} = 0, \tag{3.12}$$

$$p_{m,j}\big|_{r=r_{m,j}} = p_{in}, \tag{3.13}$$

for $1 \le j \le n$, respectively, where $r_{m,j}$ is a external radius of the layer j.

Boundary conditions on pressure and flow rate continuities between the grid blocks (i,j) and $(i+1,j)$ are given by

$$p_{i,j}\big|_{r=r_i} = p_{i+1,j}\big|_{r=r_i}, \tag{3.14}$$

$$\left.\frac{\partial p_{i,j}}{\partial r}\right|_{r=r_i} = M_{i,j}\left.\frac{\partial p_{i+1,j}}{\partial r}\right|_{r=r_i}, \tag{3.15}$$

for $1 \le i \le m-1, 1 \le j \le n$ where $M_{i,j} = k_{r,i+1,j}/k_{r,i,j}$.

The last boundary conditions not mentioned yet are pressure and flow rate continuities between the grid blocks (i,j) and $(i,j+1)$. However, if the concept of semi-permeable wall model is introduced, such boundary conditions are no longer necessary as shown in the next section.

3.1.2 Semi-permeable Wall Model

The semi-permeable wall model was introduced to the petroleum industry by Gao [38] and Gao and Deans [39] for generating synthetic pressure response of layered reservoirs with crossflow. The model uses the vertical equilibrium concept which assumes no vertical

pressure gradient exists in each layer. That means vertical permeabilities are infinite within each layer and finite only at the walls between layers.

To represent pressure for each layer, the average pressure given by Eq. 3.16 is employed in this model.

$$\langle p_{i,j} \rangle = \int_{z_{j-1}}^{z_j} p_{i,j}\, dz \Big/ \delta z_j, \tag{3.16}$$

$$\delta z_j = z_j - z_{j-1}, \tag{3.17}$$

for $1 \leq i \leq m, 1 \leq j \leq n$.

The integral transform of Eq. 3.1 on z over the interval $[z_{j-1}, z_j]$ can be expressed using the average pressure as

$$k_{r,i,j}\delta z_j \left(\frac{\partial^2 \langle p_{i,j} \rangle}{\partial r^2} + \frac{1}{r}\frac{\partial \langle p_{i,j} \rangle}{\partial r} \right) + k_{z,i,j}\frac{\partial p_{i,j}}{\partial z}\bigg|_{z=z_j} - k_{z,i,j}\frac{\partial p_{i,j}}{\partial z}\bigg|_{z=z_{j-1}} = \phi_{i,j}\mu c_{t,i,j}\delta z_j \frac{\partial \langle p_{i,j} \rangle}{\partial t}, \tag{3.18}$$

for $1 \leq i \leq m, 1 \leq j \leq n$. Taking the finite difference approximation of the second and third terms on the left hand side yields

$$k_{r,i,j}\delta z_j \left(\frac{\partial^2 \langle p_{i,j} \rangle}{\partial r^2} + \frac{1}{r}\frac{\partial \langle p_{i,j} \rangle}{\partial r} \right) + \lambda_{i,j}\{\langle p_{i,j+1} \rangle - \langle p_{i,j} \rangle\} + \lambda_{i,j-1}\{\langle p_{i,j-1} \rangle - \langle p_{i,j} \rangle\} = \phi_{i,j}\mu c_{t,i,j}\delta z_j \frac{\partial \langle p_{i,j} \rangle}{\partial t}, \tag{3.19}$$

for $1 \leq i \leq m, 1 \leq j \leq n$ where $\lambda_{i,j}$ is a parameter related to the crossflow between the grid blocks (i, j) and $(i, j+1)$ which is given by

$$\lambda_{i,j} = \frac{2}{\frac{\delta z_j}{k_{z,i,j}} + \frac{\delta z_{j+1}}{k_{z,i,j+1}}}. \tag{3.20}$$

Because of the introduction of the average pressure of Eq. 3.16, the boundary conditions of Eqs. 3.3 and 3.4 should be modified as

$$\lambda_{i,n} = 0, \tag{3.21}$$

$$\lambda_{i,0} = 0, \tag{3.22}$$

for $1 \leq i \leq m$. If either of the top or bottom boundary is the constant pressure type, the condition of Eq. 3.21 or Eq. 3.22 should be replaced by

$$\lambda_{i,n} = \frac{2k_{z,i,n}}{\delta z_n}, \tag{3.23}$$

$$\langle p_{i,n+1} \rangle = p_{in}, \tag{3.24}$$

or

$$\lambda_{i,0} = \frac{2k_{z,i,1}}{\delta z_1}, \tag{3.25}$$

$$\langle p_{i,0} \rangle = p_{in}. \tag{3.26}$$

In the rest of this chapter, delimiters "⟨" and "⟩" are omitted for simplicity.

3.1.3 Dimensionless Variables

The analytical technique used here is the same as that discussed by Gomes and Ambastha [41]. However, a different set of dimensionless variables based on the reservoir properties are chosen as follows.

Dimensionless pressure and dimensionless time are defined in Darcy units as

$$p_D = \frac{2\pi k_{r,m} h_m}{qB\mu}(p_{in} - p), \tag{3.27}$$

$$t_D = \frac{k_{r,m} t}{\phi_m \mu c_{t,m} {r_w}^2}, \tag{3.28}$$

where the subscript m indicates the matrix properties. Dimensionless radius and dimensionless height are defined by

$$r_D = \frac{r}{r_w}, \tag{3.29}$$

$$z_D = \frac{z}{r_w}. \tag{3.30}$$

By using these dimensionless variables, the governing equation of Eq. 3.19 may be expressed as

$$\kappa_{i,j}\left(\frac{\partial^2 p_{D,i,j}}{\partial {r_D}^2} + \frac{1}{r_D}\frac{\partial p_{D,i,j}}{\partial r_D}\right) + \tilde{\lambda}_{i,j}(p_{D,i,j+1} - p_{D,i,j}) + \tilde{\lambda}_{i,j-1}(p_{D,i,j-1} - p_{D,i,j}) = \omega_{i,j}\frac{\partial p_{D,i,j}}{\partial t}, \tag{3.31}$$

for $1 \leq i \leq m, 1 \leq j \leq n$ where

$$\kappa_{i,j} = \frac{k_{r,i,j}\delta z_j}{k_{r,m}h_m}, \tag{3.32}$$

$$\omega_{i,j} = \frac{\phi_{i,j}c_{t,i,j}\delta z_j}{\phi_m c_{t,m}h_m}, \tag{3.33}$$

$$\tilde{\lambda}_{i,j} = \frac{{r_w}^2}{k_{r,m}h_m}\lambda_{i,j}. \tag{3.34}$$

The initial condition of Eq. 3.2 is given in the dimensionless form by

$$\lim_{t_D \to 0} p_{D,i,j} = 0 \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq n. \tag{3.35}$$

The wellbore boundary conditions of Eqs. 3.7 and 3.8 become

$$p_{D,wf} = p_{D,1,j}\big|_{r_D=1} - s_j \left.\frac{\partial p_{D,1,j}}{\partial r_D}\right|_{r_D=1} \qquad \text{for } 1 \leq j \leq n, \tag{3.36}$$

$$1 = C_D \frac{dp_{D,wf}}{dt_D} - \sum_{j=1}^{n} \kappa_{1,j} \left.\frac{\partial p_{D,1,j}}{\partial r_D}\right|_{r_D=1} \qquad \text{for } 1 \leq j \leq n, \tag{3.37}$$

where dimensionless wellbore storage constant C_D is given by

$$C_D = \frac{C}{2\pi\phi_m c_{t,m}h_m {r_w}^2}. \tag{3.38}$$

The layer flow rate of Eq. 3.9 is given in dimensionless units by

$$q_{D,j} = \frac{q_j}{q} = -\kappa_{1,j} \left.\frac{\partial p_{D,1,j}}{\partial r_D}\right|_{r_D=1}. \tag{3.39}$$

For partially penetrating wells, layers not open to flow should be eliminated using

$$\left.\frac{\partial p_{D,1,j}}{\partial r_D}\right|_{r_D=1} = 0 \qquad \text{for the layer } j \text{ not open to flow.} \tag{3.40}$$

The outer boundary conditions from Eq. 3.11 to Eq. 3.13 become

$$\lim_{r_D \to \infty} p_{D,m,j} = 0 \qquad \text{for the infinite outer boundary,} \tag{3.41}$$

$$\left.\frac{\partial p_{D,m,j}}{\partial r_D}\right|_{r_D=\frac{r_{m,j}}{r_w}} = 0 \qquad \text{for the no-flow outer boundary,} \tag{3.42}$$

$$p_{D,m,j}\big|_{r_D=\frac{r_{m,j}}{r_w}} = 0 \qquad \text{for the constant pressure boundary,} \tag{3.43}$$

for $1 \leq j \leq n$. Finally, the interblock boundary conditions of Eqs. 3.14 and 3.15 are expressed as

$$p_{D,i,j}\big|_{r_D=\frac{r_i}{r_w}} = p_{D,i+1,j}\big|_{r_D=\frac{r_i}{r_w}}, \tag{3.44}$$

$$\left.\frac{\partial p_{D,i,j}}{\partial r_D}\right|_{r_D=\frac{r_i}{r_w}} = M_{i,j} \left.\frac{\partial p_{D,i+1,j}}{\partial r_D}\right|_{r_D=\frac{r_i}{r_w}}, \tag{3.45}$$

for $1 \leq i \leq m-1, 1 \leq j \leq n$.

3.1.4 Application of the Laplace Transform

Applying the Laplace transform on t_D to Eq. 3.31 and using the initial condition given by Eq. 3.35 yield

$$\kappa_{i,j}\left(\frac{\partial^2 \overline{p}_{D,i,j}}{\partial {r_D}^2}+\frac{1}{r_D}\frac{\partial \overline{p}_{D,i,j}}{\partial r_D}\right)+\tilde{\lambda}_{i,j}(\overline{p}_{D,i,j+1}-\overline{p}_{D,i,j}) +\lambda_{i,j-1}(\overline{p}_{D,i,j-1}-\overline{p}_{D,i,j})=\omega_{i,j}\overline{p}_{D,i,j}l, \tag{3.46}$$

for $1 \le i \le m, 1 \le j \le n$ where $\overline{p}_{D,i,j}$ is pressure in the Laplace domain expressed as

$$\overline{p}_{D,i,j}=\int_0^\infty e^{-lt_D}p_{D,i,j}\,dt_D. \tag{3.47}$$

and l is the Laplace variable.

The Laplace transform of the wellbore boundary conditions of Eqs. 3.36 and 3.37 become

$$\overline{p}_{D,wf} = \overline{p}_{D,1,j}\big|_{r_D=1}-s_j\left.\frac{\partial \overline{p}_{D,1,j}}{\partial r_D}\right|_{r_D=1} \qquad \text{for } 1 \le j \le n, \tag{3.48}$$

$$\frac{1}{l} = C_D\overline{p}_{D,wf}l-\sum_{j=1}^{n}\kappa_{1,j}\left.\frac{\partial \overline{p}_{D,1,j}}{\partial r_D}\right|_{r_D=1}. \tag{3.49}$$

The Laplace transform of the layer flow rate of Eq. 3.39 is given by

$$\overline{q}_{D,j}=-\kappa_{1,j}\left.\frac{\partial \overline{p}_{D,1,j}}{\partial r_D}\right|_{r_D=1} \qquad \text{for } 1 \le j \le n. \tag{3.50}$$

For partially penetrating wells, Eq. 3.40 becomes

$$\left.\frac{\partial \overline{p}_{D,1,j}}{\partial r_D}\right|_{r_D=1}=0 \qquad \text{for the layer } j \text{ not open to flow.} \tag{3.51}$$

The Laplace transform of the outer boundary conditions from Eq. 3.41 to Eq. 3.43 yields

$$\lim_{r_D\to\infty}\overline{p}_{D,m,j} = 0 \qquad \text{for the infinite outer boundary,} \tag{3.52}$$

$$\left.\frac{\partial \overline{p}_{D,m,j}}{\partial r_D}\right|_{r_D=\frac{r_{m,j}}{r_w}} = 0 \qquad \text{for the no-flow outer boundary,} \tag{3.53}$$

$$\overline{p}_{D,m,j}\big|_{r_D=\frac{r_{m,j}}{r_w}} = 0 \qquad \text{for the constant pressure boundary,} \tag{3.54}$$

for $1 \le j \le n$. The interblock boundary conditions of Eqs. 3.44 and 3.45 are given by

$$\overline{p}_{D,i,j}\big|_{r_D=\frac{r_i}{r_w}} = \overline{p}_{D,i+1,j}\big|_{r_D=\frac{r_i}{r_w}}, \tag{3.55}$$

$$\left.\frac{\partial \overline{p}_{D,i,j}}{\partial r_D}\right|_{r_D=\frac{r_i}{r_w}} = M_{i,j}\left.\frac{\partial \overline{p}_{D,i+1,j}}{\partial r_D}\right|_{r_D=\frac{r_i}{r_w}}, \tag{3.56}$$

for $1 \le i \le m-1, 1 \le j \le n$.

3.1.5 Wellbore Storage Effect

In general, the dimensionless pressure drawdown at the wellbore is given by the following convolution integral [60], i.e.,

$$p_{wf,D}(t_D) = \int_0^{t_D} q_{sf,D}(\tau) \frac{\partial p_{sf,D}(t_D - \tau)}{\partial \tau} d\tau, \tag{3.57}$$

where $p_{wf,D}$ is the dimensionless pressure drawdown with the wellbore storage effect (and with or without the skin damage), $p_{sf,D}$ is the dimensionless sandface pressure drawdown (with or without the skin damage). $q_{sf,D}$ is the fractional sandface rate expressed as

$$q_{sf,D} = \frac{q_{sf}}{q} = 1 - C_D \frac{dp_{wf,D}}{dt_D}, \tag{3.58}$$

where q is the surface flow rate.

In the Laplace domain, Eqs. 3.57 and 3.58 are given by

$$\overline{p}_{wf,D} = \overline{p}_{sf,D} l\overline{q}_{sf,D}, \tag{3.59}$$

$$\overline{q}_{sf,D} = \frac{1}{l} - lC_D\overline{p}_{wf,D}. \tag{3.60}$$

where l is the Laplace variable. Substituting Eq. 3.60 into Eq. 3.59 yields

$$\overline{p}_{wf,D} = \frac{\overline{p}_{sf,D}}{1 + l^2 C_D \overline{p}_{sf,D}}. \tag{3.61}$$

Eq. 3.61 indicates that the wellbore storage may be taken into account after the Laplace pressure without the wellbore storage has been obtained. In that case, the term related to the wellbore storage in Eq. 3.49 can be eliminated, and Eq. 3.49 can be modified as

$$\frac{1}{l} = -\sum_{j=1}^{n} \kappa_{1,j} \left. \frac{\partial \overline{p}_{D,1,j}}{\partial r_D} \right|_{r_D=1}. \tag{3.62}$$

It is also possible to consider the effect of the wellbore storage on the flow rate after the flow rate without the wellbore storage is obtained. From Eq. 3.60, the flow rate with the wellbore storage can be expressed using the flow rate without the wellbore storage $\overline{q}_{sf,D,C_D=0}$ as

$$\overline{q}_{sf,D} = (1 - l^2 C_D \overline{p}_{wf,D}) \overline{q}_{sf,D,C_D=0}. \tag{3.63}$$

For the layered reservoir case where the wellbore storage effect exists, the ratio of the layer production rate to the total rate becomes

$$\begin{aligned}\frac{\overline{q}_{sf,D,j}}{\sum_{j=1}^{n}\overline{q}_{sf,D,j}} &= \frac{(1-l^2C_D\overline{p}_{wf,D})\overline{q}_{sf,D,j,C_D=0}}{(1-l^2C_D\overline{p}_{wf,D})\sum_{j=1}^{n}\overline{q}_{sf,D,j,C_D=0}} \\ &= \overline{q}_{sf,D,j,C_D=0}. \end{aligned} \tag{3.64}$$

for $1 \leq j \leq n$. The equation above indicates the ratio of the layer production rate to the total production rate is equal to the layer production rate without the wellbore storage effect [33, 90].

3.1.6 Solution in the Laplace Domain

It is well-known that the Bessel equation,

$$x^2\frac{\partial^2 y}{\partial x^2} + x\frac{\partial y}{\partial x} - x^2 y = 0, \tag{3.65}$$

has a general solution,

$$y = AI_0(x) + BK_0(x), \tag{3.66}$$

where A and B are constants determined by boundary conditions, $I_0(x)$ is a modified Bessel function of the first kind and order zero, and $K_0(x)$ is that of the second kind and order zero [78].

By analogy, the solution of the governing equation of Eq. 3.46 has a general solution expressed as

$$\overline{p}_{D,i,j} = A_{i,j}I_0(\sigma_i r_D) + B_{i,j}K_0(\sigma_i r_D), \tag{3.67}$$

where $A_{i,j}$ and $B_{i,j}$ are constants associated with the grid block (i,j) and determined by the boundary conditions.

Substituting Eq. 3.67 into Eq. 3.46 and rearranging it following the procedure in Ref. [67] yield

$$\begin{aligned}&-\frac{\tilde{\lambda}_{i,j-1}}{\sqrt{\kappa_{i,j-1}\kappa_{i,j}}}(\sqrt{\kappa_{i,j-1}}\overline{p}_{D,i,j-1}) + \frac{\omega_{i,j}l + \tilde{\lambda}_{i,j-1} + \tilde{\lambda}_{i,j}}{\kappa_{i,j}}(\sqrt{\kappa_{i,j}}\overline{p}_{D,i,j}) \\ &\quad -\frac{\tilde{\lambda}_{i,j}}{\sqrt{\kappa_{i,j}\kappa_{i,j+1}}}(\sqrt{\kappa_{i,j+1}}\overline{p}_{D,i,j+1}) = {\sigma_i}^2(\sqrt{\kappa_{i,j}}\overline{p}_{D,i,j}),\end{aligned} \tag{3.68}$$

for $1 \leq i \leq m$, $1 \leq j \leq n$. Eq. 3.68 constitutes a system of equations for each layer i, and is treated as an eigenvalue problem where ${\sigma_i}^2$ are eigenvalues and $\{\sqrt{\kappa_{i,j}}\overline{p}_{D,i,j}\}$ are eigenvectors for the system.

The system of equations of Eq. 3.68 has a non-trivial solution if, and only if,

$$\det|\mathbf{G}_i - {\sigma_i}^2\mathbf{I}| = 0, \tag{3.69}$$

where $\mathbf{I}$ is an identity matrix, and $\mathbf{G}_i$ is a $n \times n$ matrix given by

$$\mathbf{G}_i = \{g^i_{j,j'}\} = \begin{cases} -\frac{\tilde{\lambda}_{i,j-1}}{\sqrt{\kappa_{i,j-1}\kappa_{i,j}}} & \text{for } j' = j-1, j \geq 2, \\ \frac{\omega_{i,j}l + \tilde{\lambda}_{i,j-1} + \tilde{\lambda}_{i,j}}{\kappa_{i,j}} & \text{for } j' = j, \\ -\frac{\tilde{\lambda}_{i,j}}{\sqrt{\kappa_{i,j}\kappa_{i,j+1}}} & \text{for } j' = j+1, j \leq n-1, \\ 0 & \text{otherwise,} \end{cases} \tag{3.70}$$

for $1 \leq i \leq m$ [33]. Eq. 3.69 is an nth degree polynomial in ${\sigma_i}^2$, and the k roots of the polynomial are the eigenvalues expressed as $(\sigma_i^k)^2$.

Since Eq. 3.46 is linear, the general solution is given by the linear superposition of Eq. 3.67 for each k, i.e.,

$$\overline{p}_{D,i,j} = \sum_{k=1}^{n} \{A^k_{i,j} I_0(\sigma_i^k r_D) + B^k_{i,j} K_0(\sigma_i^k r_D)\}, \tag{3.71}$$

for $1 \leq i \leq m, 1 \leq j \leq n$.

In Eq. 3.71, constants $A^k_{i,j}$ and $B^k_{i,j}$ are not independent of each other. Substituting Eq. 3.71 into Eq. 3.68 yields

$$\mathbf{G}^k_i \mathbf{A}^k_i = \mathbf{0}, \tag{3.72}$$

$$\mathbf{G}^k_i \mathbf{B}^k_i = \mathbf{0}, \tag{3.73}$$

for $1 \leq i \leq m$ where $\mathbf{G}^k_i$ is a $n \times n$ matrix given by

$$\mathbf{G}^k_i = \{g^{i,k}_{j,j'}\} = \begin{cases} -\frac{\tilde{\lambda}_{i,j-1}}{\sqrt{\kappa_{i,j-1}\kappa_{i,j}}} & \text{for } j' = j-1, j \geq 2, \\ \frac{\omega_{i,j}l + \tilde{\lambda}_{i,j-1} + \tilde{\lambda}_{i,j}}{\kappa_{i,j}} - (\sigma_i^k)^2 & \text{for } j' = j, \\ -\frac{\tilde{\lambda}_{i,j}}{\sqrt{\kappa_{i,j}\kappa_{i,j+1}}} & \text{for } j' = j+1, j \leq n-1, \\ 0 & \text{otherwise,} \end{cases} \tag{3.74}$$

and $\mathbf{A}_i^k, \mathbf{B}_i^k$ are the eigenvectors expressed as

$$\mathbf{A}_i^k = \begin{pmatrix} \sqrt{\kappa_{i,1}} A_{i,1}^k \\ \sqrt{\kappa_{i,2}} A_{i,2}^k \\ \vdots \\ \sqrt{\kappa_{i,n}} A_{i,n}^k \end{pmatrix}, \tag{3.75}$$

$$\mathbf{B}_i^k = \begin{pmatrix} \sqrt{\kappa_{i,1}} B_{i,1}^k \\ \sqrt{\kappa_{i,2}} B_{i,2}^k \\ \vdots \\ \sqrt{\kappa_{i,n}} B_{i,n}^k \end{pmatrix}. \tag{3.76}$$

Using the relationship of Eq. 3.72, the elements $A_{i,j}^k (j \geq 2)$ can be expressed by $A_{i,1}^k$ as

$$\begin{aligned} A_{i,2}^k &= -\frac{1}{g_{1,2}^{i,k}\sqrt{\kappa_{i,2}}}(g_{1,1}^{i,k}\sqrt{\kappa_{i,1}}A_{i,1}^k) = E_{i,2}^k A_{i,1}^k, \\ A_{i,3}^k &= -\frac{1}{g_{2,3}^{i,k}\sqrt{\kappa_{i,3}}}(g_{2,1}^{i,k}\sqrt{\kappa_{i,1}}A_{i,1}^k + g_{2,2}^{i,k}\sqrt{\kappa_{i,2}}A_{i,2}^k) \\ &= E_{i,3}^k A_{i,1}^k, \\ &\cdots\cdots\cdots\cdots \\ A_{i,n}^k &= -\frac{1}{g_{n-1,n}^{i,k}\sqrt{\kappa_{i,n}}}(g_{n-1,n-2}^{i,k}\sqrt{\kappa_{i,n-2}}A_{i,n-2}^k + g_{n-1,n-1}^{i,k}\sqrt{\kappa_{i,n-1}}A_{i,n-1}^k) \\ &= E_{i,n}^k A_{i,1}^k. \end{aligned} \tag{3.77}$$

Thus, replacing $A_{i,1}^k$ with A_i^k yields

$$A_{i,j}^k = E_{i,j}^k A_i^k. \tag{3.78}$$

Similarly, constants $B_{i,j}^k$ are given from Eq. 3.73 by

$$B_{i,j}^k = E_{i,j}^k B_i^k. \tag{3.79}$$

The constants $\mathbf{A}_i^k$ and $\mathbf{B}_i^k$ are eigenvectors of Eq. 3.68. Clearly, any multiple of an

eigenvector is also an eigenvector. Thus,

$$\begin{pmatrix} {A_{i,1}^k}' \\ {A_{i,2}^k}' \\ \vdots \\ {A_{i,n}^k}' \end{pmatrix} = \frac{1}{A_i^k} \begin{pmatrix} \sqrt{\kappa_{i,1}} A_{i,1}^k \\ \sqrt{\kappa_{i,2}} A_{i,2}^k \\ \vdots \\ \sqrt{\kappa_{i,n}} A_{i,n}^k \end{pmatrix}, \tag{3.80}$$

is also an eigenvector of Eq. 3.68. Rewriting the above equation yields

$$\begin{pmatrix} A_{i,1}^k \\ A_{i,2}^k \\ \vdots \\ A_{i,n}^k \end{pmatrix} = A_i^k \begin{pmatrix} \frac{{A_{i,1}^k}'}{\sqrt{\kappa_{i,1}}} \\ \frac{{A_{i,2}^k}'}{\sqrt{\kappa_{i,2}}} \\ \vdots \\ \frac{{A_{i,n}^k}'}{\sqrt{\kappa_{i,n}}} \end{pmatrix}. \tag{3.81}$$

Thus, $A_{i,j}^k$ can be expressed as

$$A_{i,j}^k = {E_{i,j}^k}' A_i^k, \tag{3.82}$$

which have the same form as Eq. 3.78. A similar expression can be obtained for $B_{i,j}^k$ as

$$B_{i,j}^k = {E_{i,j}^k}' B_i^k. \tag{3.83}$$

Therefore, if the eigenvectors are calculated by a numerical algorithm together with the eigenvalues, it is possible to replace Eqs. 3.78 and 3.79 with Eqs. 3.82 and 3.83.

Using Eqs. 3.78 and 3.79 (or Eqs. 3.82 and 3.83), the general solution of Eq. 3.71 can be expressed as

$$\overline{p}_{D,i,j} = \sum_{k=1}^{n} E_{i,j}^k \{A_i^k I_0(\sigma_i^k r_D) + B_i^k K_0(\sigma_i^k r_D)\}, \tag{3.84}$$

for $1 \leq i \leq m, 1 \leq j \leq n$. Substituting Eq. 3.84 into the wellbore boundary conditions of Eqs. 3.48 and 3.62 yields

$$\begin{aligned} \overline{p}_{D,wf} &= \sum_{k=1}^{n} E_{1,j}^k [\{A_1^k I_0(\sigma_1^k) + B_1^k K_0(\sigma_1^k)\} \\ &\quad -s_j \sigma_1^k \{A_1^k I_1(\sigma_1^k) - B_1^k K_1(\sigma_1^k)\}] \qquad \text{for } 1 \leq j \leq n, \qquad (3.85) \\ \frac{1}{l} &= -\sum_{j=1}^{n} \kappa_{1,j} \sum_{k=1}^{n} \sigma_1^k E_{1,j}^k \{A_1^k I_1(\sigma_1^k) - B_1^k K_1(\sigma_1^k)\}. \qquad (3.86) \end{aligned}$$

For partially penetrating wells, layers not open to flow should be eliminated. That condition is given by substituting Eq. 3.84 into Eq. 3.51 as

$$\sum_{k=1}^{n} \sigma_1^k E_{1,j}^k \{A_1^k I_1(\sigma_1^k) - B_1^k K_1(\sigma_1^k)\} = 0, \tag{3.87}$$

for the layer j not open to flow.

The infinite outer boundary condition of Eq. 3.52 becomes

$$\lim_{r_D \to \infty} \sum_{k=1}^{n} E_{m,j}^k \{A_m^k I_0(\sigma_m^k r_D) + B_m^k K_0(\sigma_m^k r_D)\} = 0, \tag{3.88}$$

for $1 \leq j \leq n$. However, Eq. 3.88 is reduced to

$$A_m^k = 0, \tag{3.89}$$

for $1 \leq k \leq n$ since

$$\lim_{x \to \infty} I_0(x) = \infty,$$

and

$$\lim_{x \to \infty} K_0(x) = 0.$$

The no-flow outer boundary condition of Eq. 3.53 is expressed as

$$\sum_{k=1}^{n} \sigma_m^k E_{m,j}^k \{A_m^k I_1(\sigma_m^k r_{D,m}) - B_m^k K_1(\sigma_m^k r_{D,m})\} = 0, \tag{3.90}$$

for $1 \leq j \leq n$ where $r_{D,m} = r_m / r_w$. Similarly, the constant pressure outer boundary condition of Eq. 3.54 is given by

$$\sum_{k=1}^{n} E_{m,j}^k \{A_m^k I_0(\sigma_m^k r_{D,m}) + B_m^k K_0(\sigma_m^k r_{D,m})\} = 0, \tag{3.91}$$

for $1 \leq j \leq n$.

The interblock pressure continuity boundary condition of Eq. 3.55 and the interblock rate continuity boundary condition of Eq. 3.56 are given by

$$\begin{aligned} &\sum_{k=1}^{n} E_{i,j}^k \{A_i^k I_0(\sigma_i^k r_{D,i}) + B_i^k K_0(\sigma_i^k r_{D,i})\} \\ &\qquad = \sum_{k=1}^{n} E_{i+1,j}^k \{A_{i+1}^k I_0(\sigma_{i+1}^k r_{D,i}) + B_{i+1}^k K_0(\sigma_{i+1}^k r_{D,i})\}, \end{aligned} \tag{3.92}$$

$$\sum_{k=1}^{n} \sigma_i^k E_{i,j}^k \{A_i^k I_1(\sigma_i^k r_{D,i}) - B_i^k K_1(\sigma_i^k r_{D,i})\}$$
$$= M_{i,j} \sum_{k=1}^{n} \sigma_{i+1}^k E_{i+1,j}^k \{A_{i+1}^k I_1(\sigma_{i+1}^k r_{D,i}) - B_{i+1}^k K_1(\sigma_{i+1}^k r_{D,i})\}, \qquad (3.93)$$

for $1 \leq i \leq m-1$, $1 \leq j \leq n$ where $r_{D,i} = r_i/r_w$.

Here, the number of unknown parameters (A_i^k and B_i^k) is $2mn$, while the number of equations (Eqs. 3.85 to 3.93) is also $2mn$. Hence, the system of equations is solvable. Once the system of equations is solved, the pressure at the wellbore with the damage skin may be expressed from Eq. 3.85 as

$$\begin{aligned} \overline{p}_{D,wf} &= \sum_{k=1}^{n} E_{1,1}^k [\{A_1^k I_0(\sigma_1^k) + B_1^k K_0(\sigma_1^k)\} \\ &\quad -s_1 \sigma_1^k \{A_1^k I_1(\sigma_1^k) - B_1^k K_1(\sigma_1^k)\}]. \end{aligned} \qquad (3.94)$$

The fractional flow rate can be also obtained from Eq. 3.50 by

$$\overline{q}_{D,j} = -\kappa_{1,j} \sum_{k=1}^{n} \sigma_1^k E_{1,j}^k \{A_1^k I_1(\sigma_1^k) - B_1^k K_1(\sigma_1^k)\} \qquad \text{for } 1 \leq j \leq n. \qquad (3.95)$$

The effect of the wellbore storage can be included for the pressure and the fractional flow rate using Eqs. 3.61 and 3.63.

The solution in Laplace domain can be converted to the real domain using the Stehfest algorithm which is shown in the next section.

3.1.7 Stehfest Algorithm

The Stehfest algorithm [111] is widely used for the numerical inversion of Laplace transformed pressure in well test application. In this method, the inversion of the function $\overline{f}$ in the Laplace domain is performed numerically using the following relationship,

$$f(t) = \frac{\ln 2}{t} \sum_{i=1}^{N} V_i \overline{f}\left(\frac{\ln 2}{t} i\right), \qquad (3.96)$$

where N is an even integer, and

$$V_i = (-1)^{\frac{N}{2}+i} \sum_{k=[\frac{i+1}{2}]}^{\min(i,\frac{N}{2})} \frac{k^{\frac{N}{2}}(2k)!}{(\frac{N}{2}-k)!k!(k-1)!(i-k)!(2k-i)!}. \qquad (3.97)$$

The coefficients V_i only depend on N and if the calculation is repeated, it is enough to evaluate V_i only once.

In well test analysis, 6 or 8 is often used as a optimum value for N. Theoretically, results become more accurate as N becomes larger. In practice, however, the accuracy becomes worse if N is too large, since the value of V_i becomes too large to be dealt with in computers and rounding errors will occur.

In addition, even though the integer N is increased, it does not change the density of observation points but increases the span of the sampling interval [89]. The sampling interval of the function $\overline{f}(l)$ is $\ln 2/t \leq l \leq N \ln 2/t$, thus larger N does not increase the density of points. This means the larger N does not contribute to capture rapid change of the function $\overline{f}(l)$ within the interval. However, N contributes to the accuracy of the calculation by capturing global features of the function. At present, it can be said that the best way to obtain the optimum N is to choose it by trial and error.

3.2 Results and Discussions

3.2.1 Model Validation and Layer Refinement

When crossflow between layers is significant, layer refinement is necessary as discussed by Larsen [67, 68]. This is due to the pseudo-steady state approximation which assumes linear pressure change in the z direction within each layer.

Results of the analytical solution for a lens reservoir were compared with a result of finely gridded numerical simulation in Fig. 3.2. A numerical simulation model with radial geometry was developed. The size of the simulation grid is 62 $\times$ 1 $\times$ 90 (r, θ, and z), which was considered to be enough by examining sensitivity to the grid size and initial timesteps.

Two-layer and three-layer models for the analytical solution are shown in Fig. 3.3. We consider an upper half region of the reservoir since the system is symmetric on the middle plane which divides the reservoir into the upper and lower parts.

The effect of the layer refinement can be observed in pressure derivative curves at early

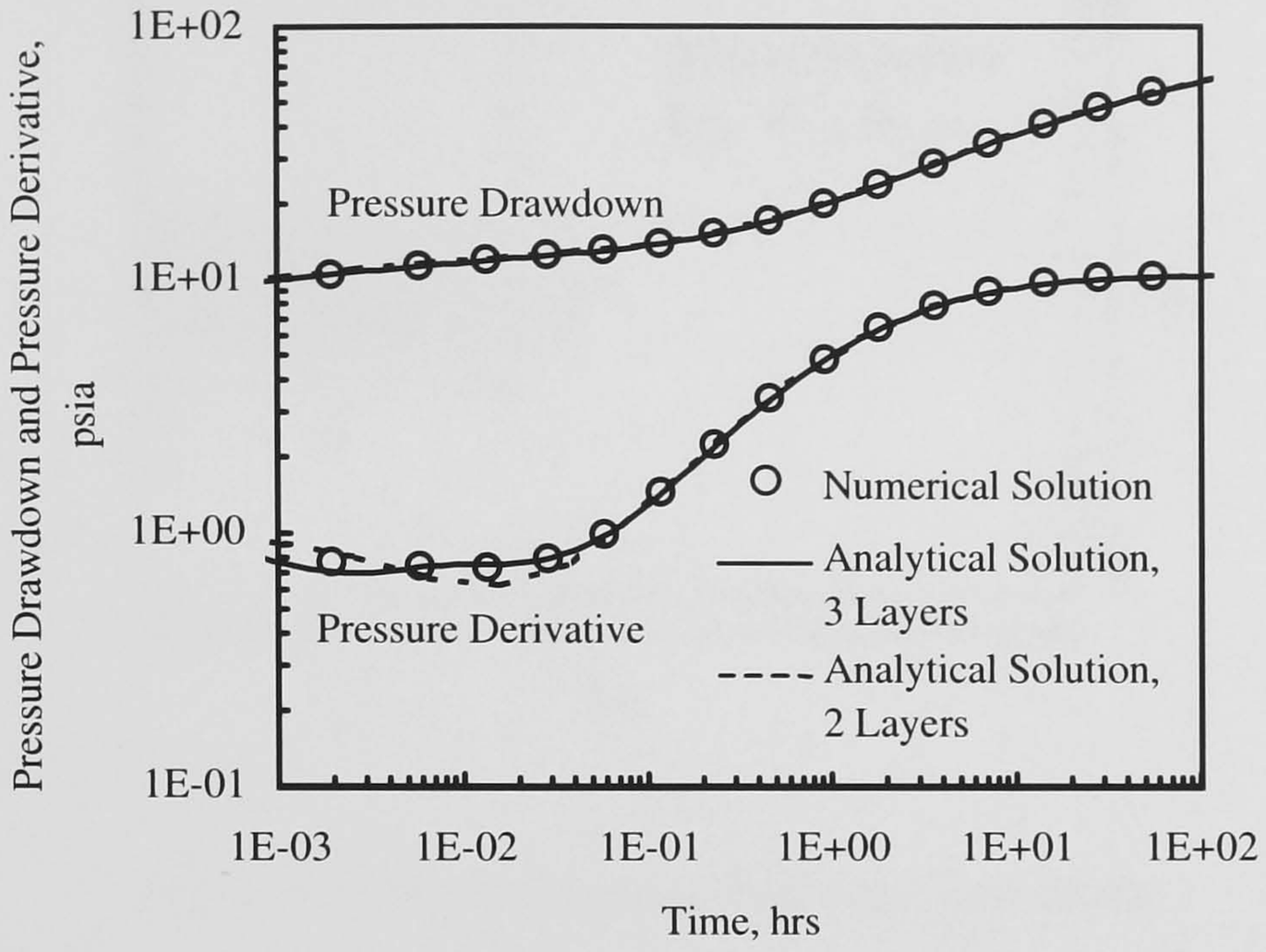


Figure 3.2: Comparison of the analytical solution with the numerical simulation.

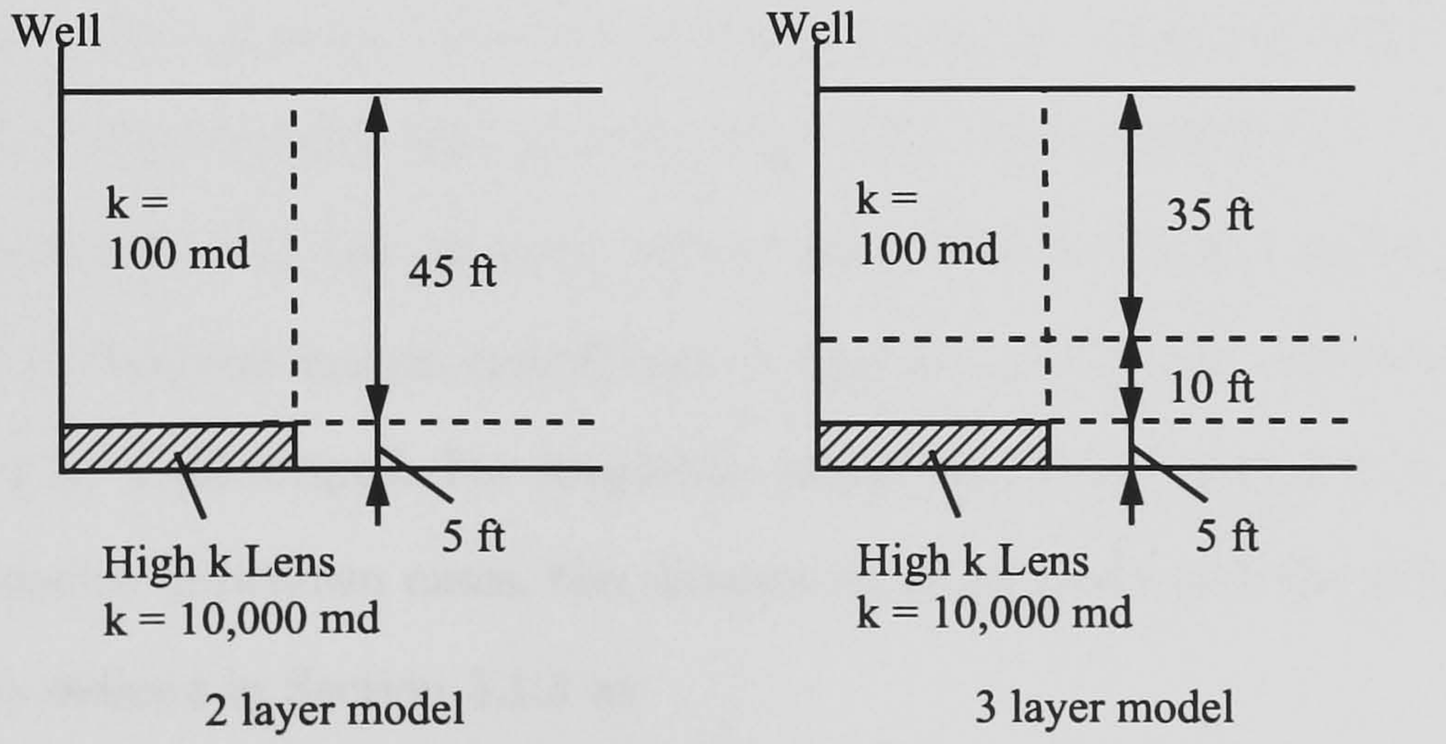


Figure 3.3: Models with and without layer refinement.

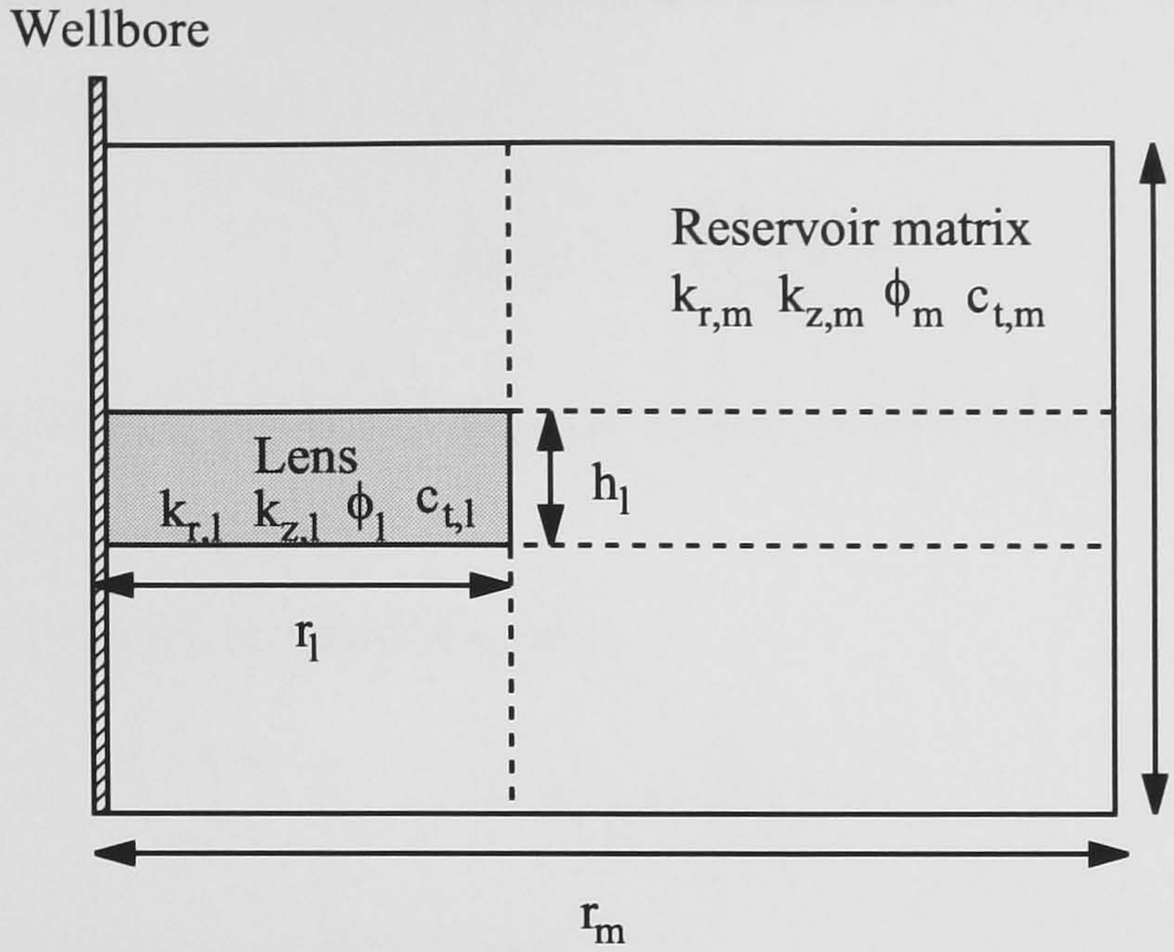


Figure 3.4: Schematic single lens reservoir model.

times. A better match is obtained for the three-layer model since it can yield the non-linear vertical pressure change more accurately. As in numerical simulation, it is necessary to check sensitivity by refining or amalgamating layers in order to find an effective layering of the reservoir.

3.2.2 Dimensional Analysis for Reservoirs Including Only One Lens

In the rest of this chapter, reservoirs containing only one high permeability lens located in the centre are discussed mainly (Fig. 3.4). The fluid is produced at a constant rate from the well centred in the reservoir and the lens. However, fractional rates from the lens and from the reservoir matrix can change during the production. The lens properties are represented by a subscript l. For simplicity, an infinite outer boundary is assumed.

For pressure drawdown cases, the dimensionless pressure and the dimensionless time are already defined in Section 3.1.3 as

$$p_{D,DD} = \frac{2\pi k_{r,m} h_m}{qB\mu}(p_{in} - p), \tag{3.98}$$

$$t_D = \frac{k_{r,m} t}{\phi_m \mu c_{t,m} {r_w}^2}, \tag{3.99}$$

where the subscript DD indicates the pressure drawdown cases. Dimensionless pressure derivative can be written as

$$p_{D,DD}{}' = \frac{dp_{D,DD}}{d\ln t_D}. \tag{3.100}$$

Dimensional analysis shows the dimensionless pressure is dominated by 9 dimensionless variables in addition to t_D, i.e.,

dimensionless horizontal permeability ratio

$$k_{h,D} = \frac{k_l}{k_m}, \tag{3.101}$$

dimensionless vertical permeability ratio

$$k_{v,D} = \frac{k_{z,l}}{k_{z,m}}, \tag{3.102}$$

dimensionless vertical to horizontal permeability ratio

$$k_{vh,D} = \frac{{r_w}^2}{{h_m}^2}\frac{k_{z,m}}{k_m}, \tag{3.103}$$

dimensionless thickness ratio

$$h_D = \frac{h_l}{h_m}, \tag{3.104}$$

dimensionless lens radius

$$r_D = \frac{r_l}{r_w}, \tag{3.105}$$

dimensionless porosity-compressibility ratio

$$(\phi c_t)_D = \frac{\phi_l c_{t,l}}{\phi_m c_{t,m}}, \tag{3.106}$$

skin factors

$$s_l, s_m, \tag{3.107}$$

and dimensionless wellbore storage constant

$$C_D = \frac{C}{2\pi\phi_m c_{t,m} h_m {r_w}^2}. \tag{3.108}$$

In order to simplify the following discussion, it is assumed that the dimensionless porosity-compressibility ratio $(\phi c_t)_D$ is 1, and there are no skins ($s_l = s_m = 0$) and no wellbore storage effect ($C_D = 0$) unless otherwise mentioned. Thus, we can write $p_{D,DD}$ as

$$p_{D,DD} = p_{D,DD}(t_D, k_{h,D}, k_{v,D}, k_{vh,D}, h_D, r_D). \tag{3.109}$$

Pressure buildup responses are also available by the superposition of the pressure drawdown responses in time as

$$p_{D,BU}(\Delta t_D) = p_{D,DD}(\Delta t_D) + p_{D,DD}(t_{p,D}) - p_{D,DD}(t_{p,D} + \Delta t_D), \tag{3.110}$$

where the dimensionless pressure buildup is defined as

$$p_{D,BU} = \frac{2\pi k_m h_m}{qB\mu}(p_{ws,t=\Delta t} - p_{wf,t=t_p}). \tag{3.111}$$

The pressure derivative can be written based on the Horner method as

$$p_{D,BU}{}' = -\frac{dp_{D,BU}}{d\ln\left(\frac{t_{p,D}+\Delta t_D}{\Delta t_D}\right)}. \tag{3.112}$$

3.2.3 Flow Periods for Lens Reservoirs

Lens reservoirs and layered reservoirs show common features in early-time period. Fig. 3.5 shows typical pressure responses for the lens reservoir case and the three-layer reservoir case. For the lens reservoir case, the external radius of the high permeability zone is smaller than that of the reservoir matrix (Fig. 3.4), while it is the same for the layered reservoir case.

In the layered reservoir case, it is well-known that there are three flow periods [13, 96], i.e.,

(1) Early-time period whose response is identical to that of a reservoir without crossflow between layers (commingled system)

(2) Transition period between the periods (1) and (3) characterised by the downward concavity of a pressure derivative curve in which the crossflow becomes significant.

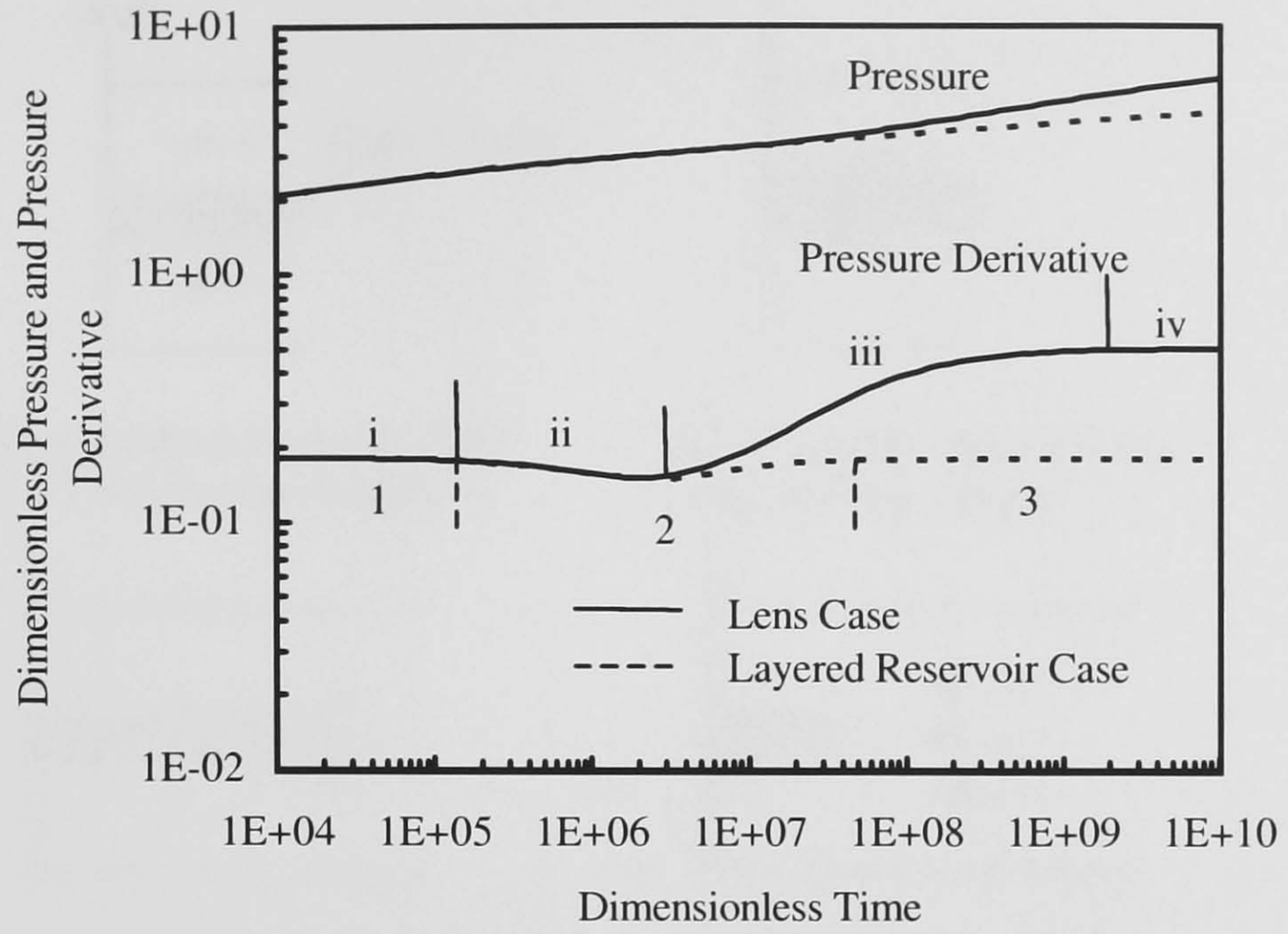


Figure 3.5: Comparison of the typical lens reservoir case with the layered reservoir case.

In two-layer systems with crossflow, if the vertical to horizontal permeability ratios of each layer j ($j = 1, 2$) are smaller than 1 (i.e., $k_{v,j}/k_{h,j} < 1$), this flow period can be seen only if $k_{h,1}/k_{h,2} \geq 5$ (Prijambodo et al. [96]).

(3) Late-time period whose response is identical to an equivalent homogeneous system which has a total permeability-thickness product and a total porosity-compressibility-thickness product.

In the lens reservoir case, four flow periods are observed as shown in Fig. 3.6.

(i) Early-time period which is the same as for the layered reservoir case

(ii) Transition period which is the same as for the layered reservoir case

(iii) Convergence flow period in which the flow converges into the lens. This period begins when pressure disturbance reaches the edge of the lens.

(iv) Pseudo-radial flow period in which the response is identical to a homogeneous reservoir with the matrix permeability and a negative pseudo-skin (the pseudo-skin may

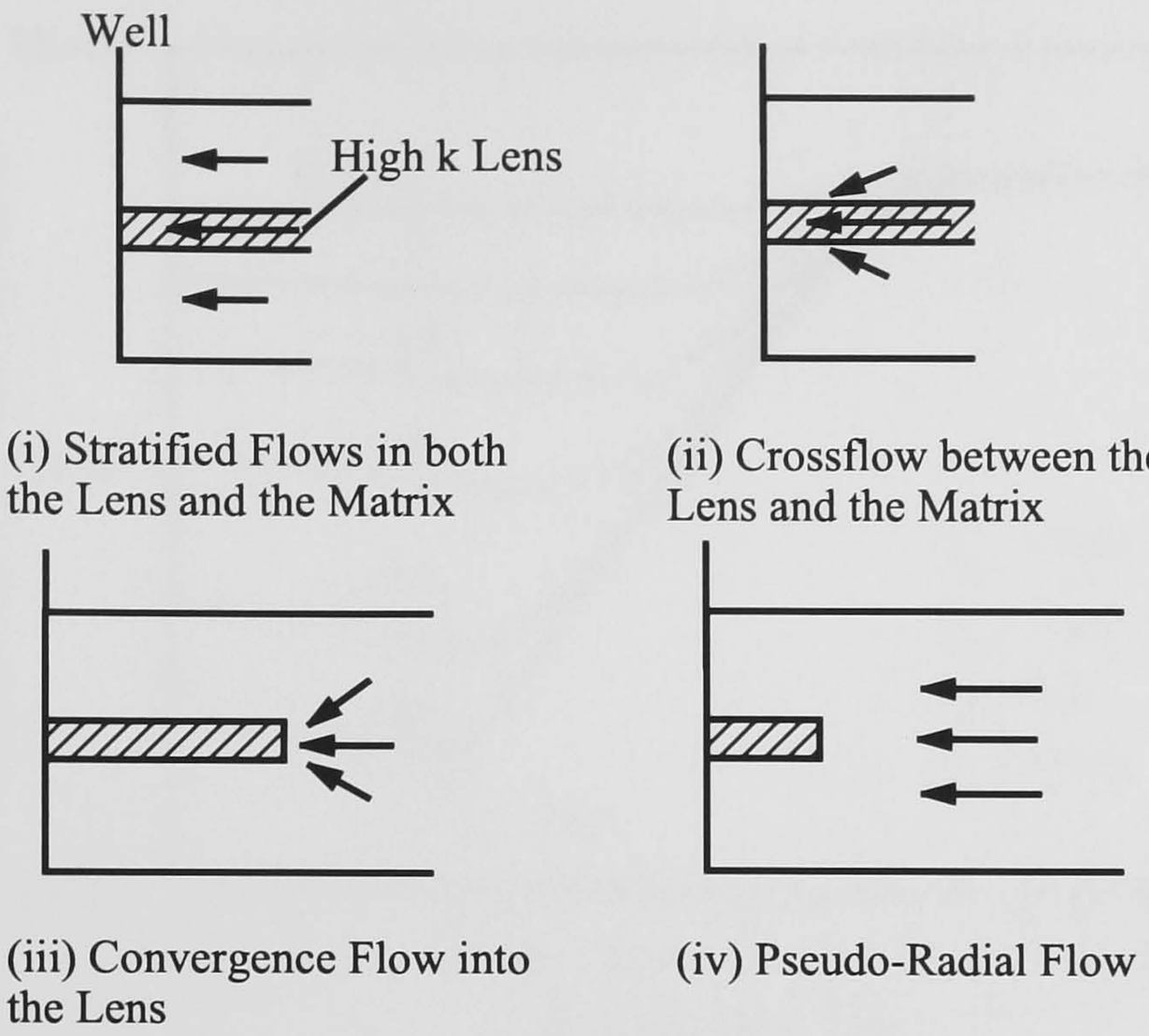


Figure 3.6: Schematic illustration of flow periods for the lens reservoir.

be positive when damage skin exists). The dimensionless pressure derivative becomes 0.5 in this flow period.

The time at which the convergence flow period (iii) begins depends on the external radius of the lens. If it is small, the period can begin during the flow period (1). If it is large, the period can begin during the flow period (3).

3.2.4 Sensitivities to Various Dimensionless Variables

In this section, sensitivities to the various dimensionless variables derived in Section 3.2.2 are examined. Such information is important to understand the pressure response of the lens reservoirs. It is also useful when a pressure analysis is performed by an interactive pressure - pressure derivative type-curve matching.

Dimensionless Horizontal Permeability Ratio

Fig. 3.7 shows the sensitivity to the dimensionless horizontal permeability ratio, $k_{h,D}$. In the flow period (i), the level of pressure derivative becomes low as the ratio increases. The

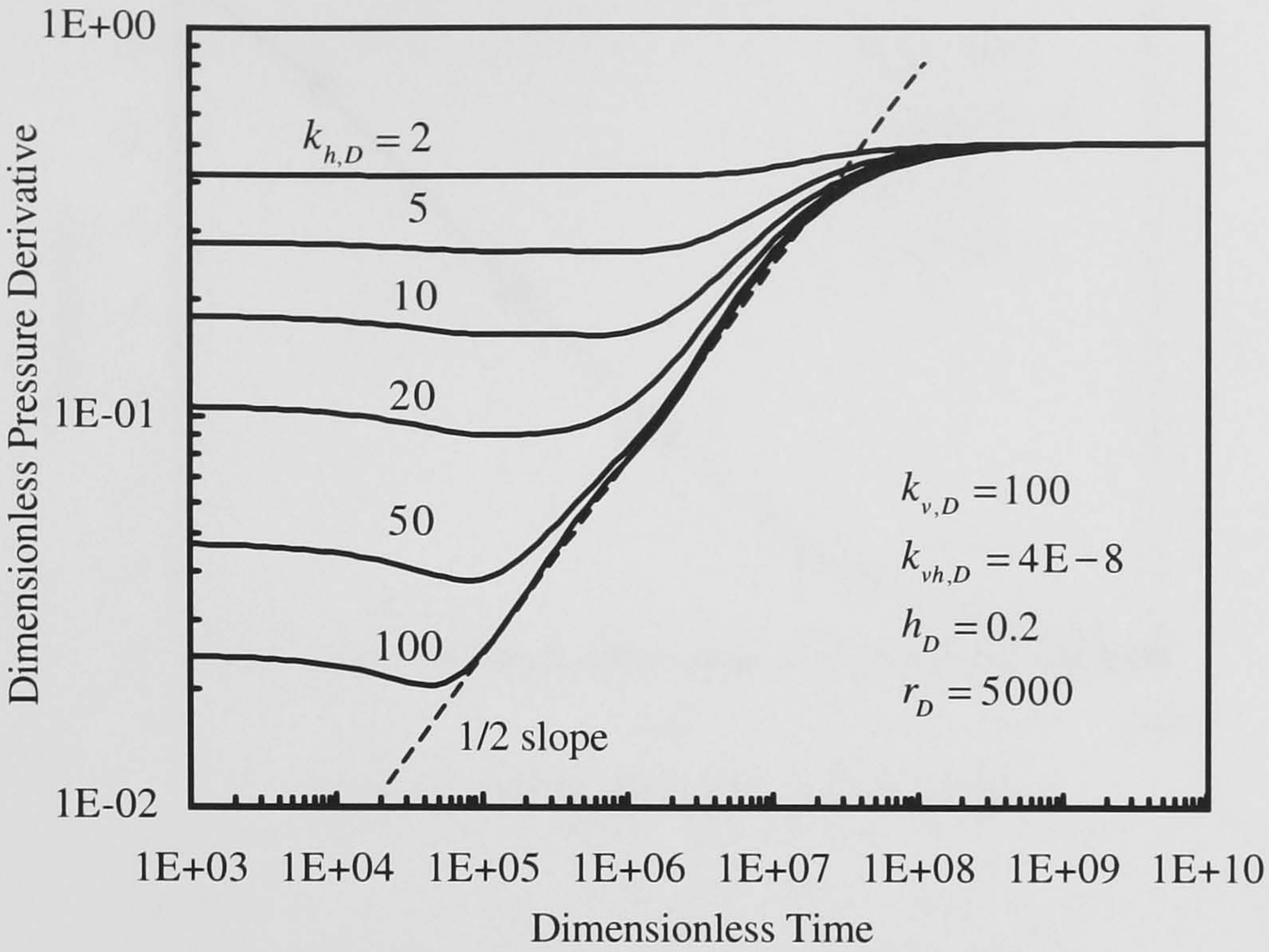


Figure 3.7: Effect of varying $k_{h,D}$ on pressure derivative.

level can be predicted by

$$p_{D,(i)}{}' = \frac{0.5}{1 + h_D(k_{h,D} - 1)}, \tag{3.113}$$

which is the same as the early-time period (1) for the layered reservoirs. The pressure derivative curve in the convergence flow period (iii) shows a half-slope line if the permeability ratio is high enough. In such case, most fluid is produced from the lens. In addition, if the dimensionless thickness ratio is much smaller than 1 ($h_D \ll 1$), we can expect the linear flow (represented by the half slope derivative curve) into the surface of the lens which is the same as an infinite conductivity horizontally fractured reservoir case [46].

Fig. 3.8 shows the effect of $k_{h,D}$ on the pseudo-skin obtained from the data in the pseudo-radial flow period assuming the reservoir is a homogeneous reservoir with the reservoir matrix permeability. It can be observed that the pseudo-skin decreases as $k_{h,D}$ increases. The same observation was reported in Ref. [79].

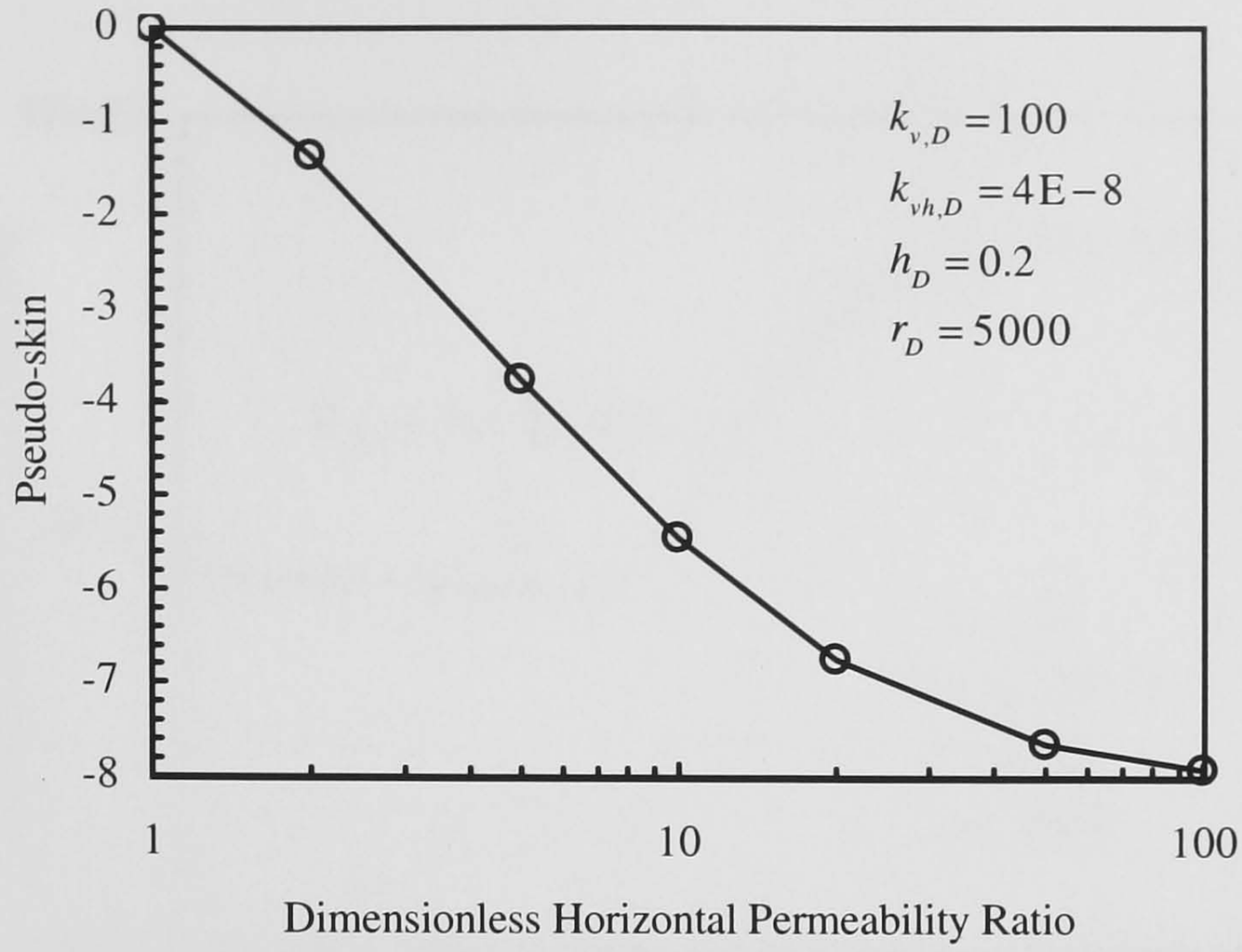


Figure 3.8: Effect of varying $k_{h,D}$ on pseudo-skin.

Dimensionless Vertical Permeability Ratio

The effect of the dimensionless vertical permeability ratio, $k_{v,D}$, is shown in Fig. 3.9. The effect on the pressure derivative cannot be observed in this case. The effect on the pseudo-skin obtained from the pseudo-radial flow period is also negligible since the pseudo-skin becomes the same (-7.2) in each case.

Dimensionless Vertical to Horizontal Permeability Ratio

The effect of dimensionless vertical to horizontal permeability ratio, $k_{vh,D}$, can be recognised as shown in Fig. 3.10. As the ratio decreases, the beginning of the transition flow period (ii) is delayed since the crossflow is more restricted. If it is small enough, a hump can be seen in the convergence flow period (iii), which indicates a temporary inefficient flow of fluid. Since the flow in the vertical direction is limited, the convergence flow into the lens is impeded. The imbalance between the flow of fluid into the lens and the production from the lens causes large pressure loss during the period. After enough lens inflow from the vertical direction is achieved, the pseudo-radial flow (iv) is observed in each case. However, since the differentiation of pressure data for calculating pressure derivatives am-

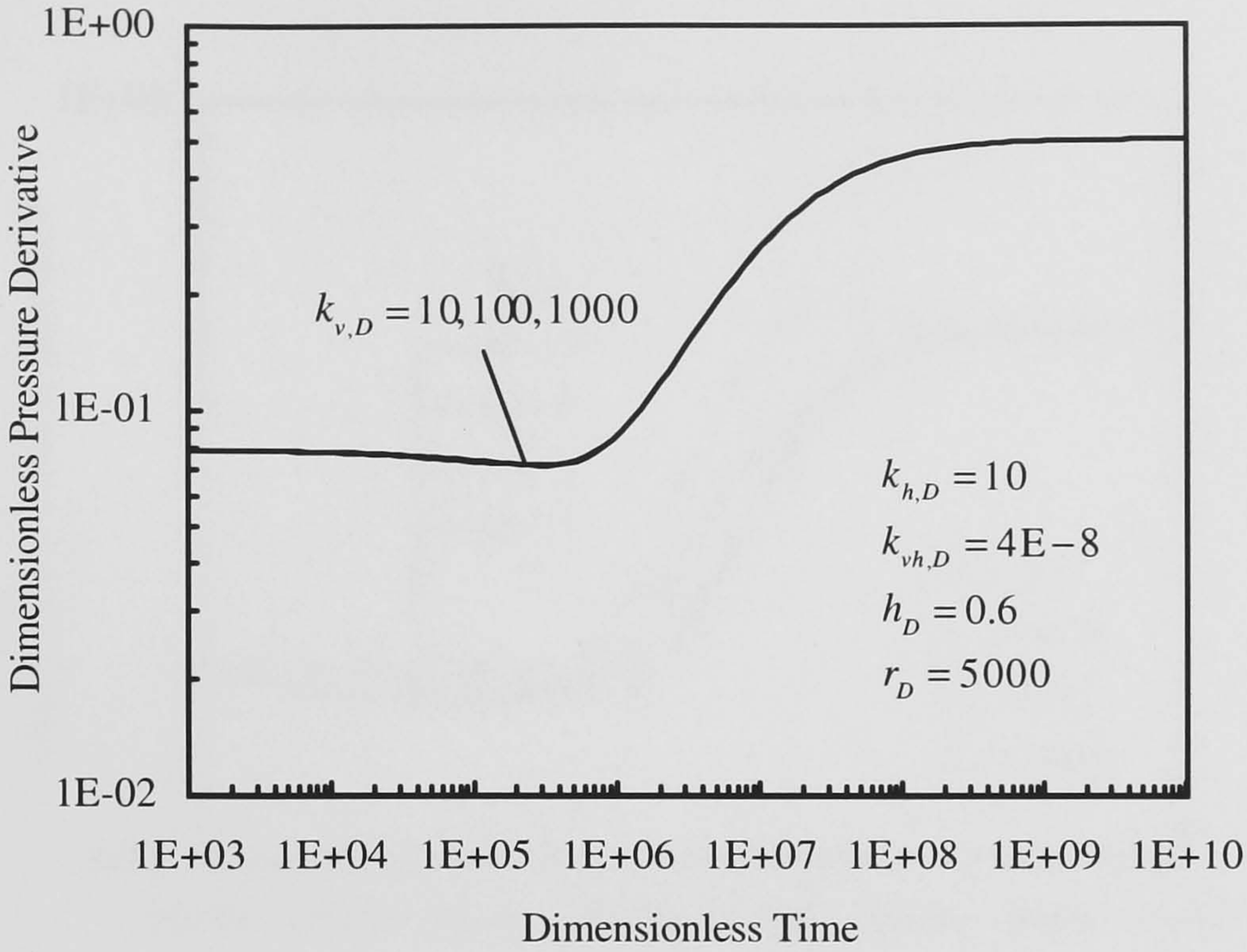


Figure 3.9: Effect of varying $k_{v,D}$ on pressure derivative.

plifies noise due to gauge resolution, electronic circuitry, vibrations, etc. [14], it is very difficult to see this small effect in actual welltest analyses even though the resolution of pressure gauges is under 0.01 psi nowadays.

The effect of $k_{vh,D}$ on the pseudo-skin is shown in Fig. 3.11. It is shown that if $k_{vh,D}$ is large enough, the pseudo-skin becomes constant. However, the absolute value of the pseudo-skin decreases slightly as $k_{vh,D}$ decreases. This is due to the imbalance between the inflow into the lens and the production from the lens observed in the pressure derivative response.

Dimensionless Thickness Ratio

The effect of the dimensionless thickness ratio, h_D, is similar to that of the dimensionless horizontal permeability ratio as shown in Fig. 3.12. In the early-time period (i), the level of the pressure derivative can be predicted by Eq. 3.113. A radial composite reservoir case ($h_D = 1$) is also shown in the same figure. The downward concavity in the transition period

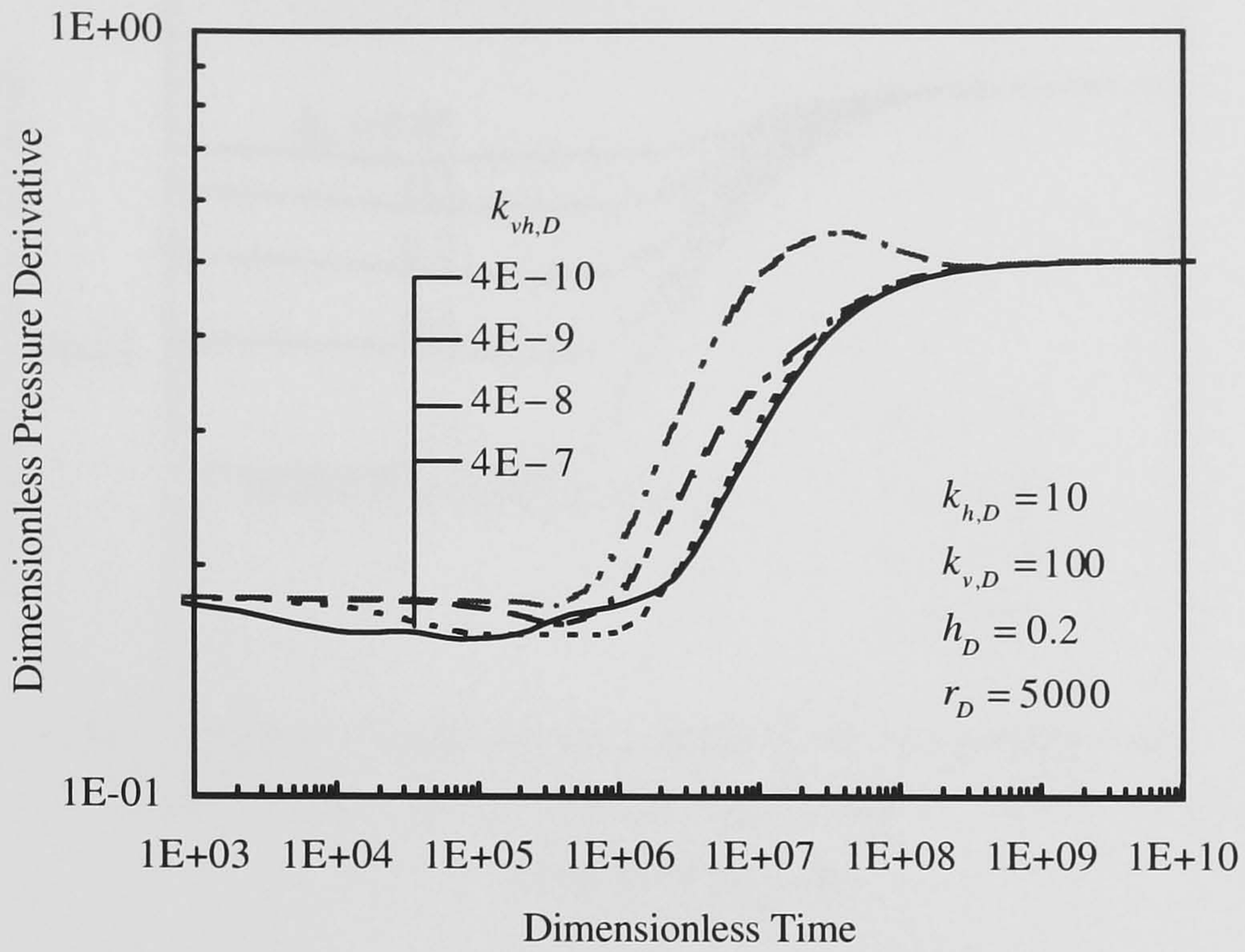


Figure 3.10: Effect of varying $k_{vh,D}$ on pressure derivative.

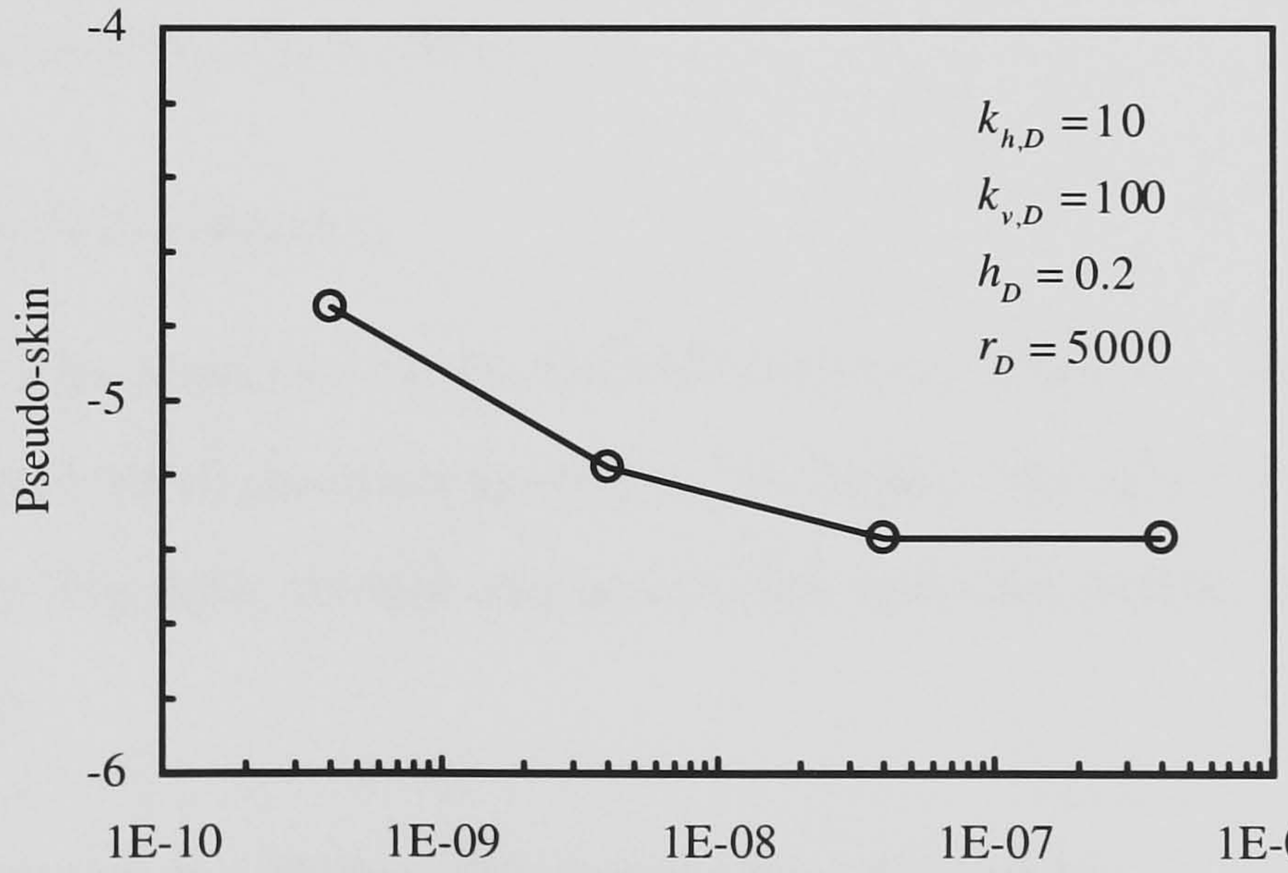


Figure 3.11: Effect of varying $k_{vh,D}$ on pseudo-skin.

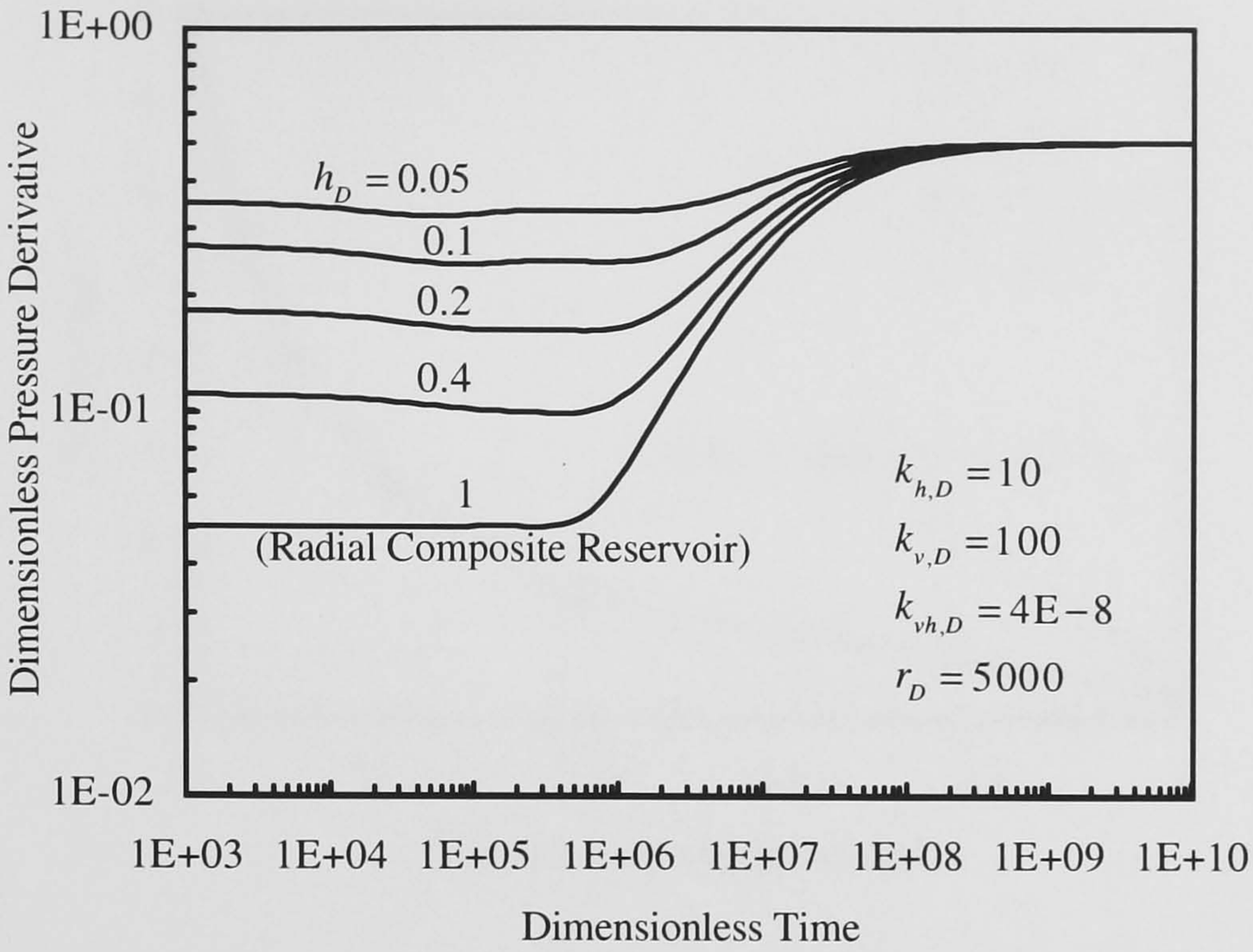


Figure 3.12: Effect of varying h_D on pressure derivative.

(ii) diminishes as the ratio increases since the effect of the crossflow becomes smaller.

The effect of h_D on the pseudo-skin is shown in Fig. 3.13. It can be observed that the pseudo-skin decreases as h_D increases.

Dimensionless Lens Radius

Fig. 3.14 shows the sensitivity to the dimensionless lens radius, r_D. The lens radius only affects the time at which the convergence flow (iii) begins. The effect of r_D on the pseudo-skin is shown in Fig. 3.15. At first, the pseudo-skin decreases rapidly, and then gradually as r_D increases.

3.2.5 Effects of the Other Dimensionless Variables

For simplicity, it has been assumed that the lens is positioned in the centre of the reservoir matrix, the dimensionless porosity-compressibility ratio, $(\phi c_t)_D$, is 1, and there are no skins ($s_l = s_m = 0$) and no wellbore storage effect ($C_D = 0$). This section shows sensitivities to

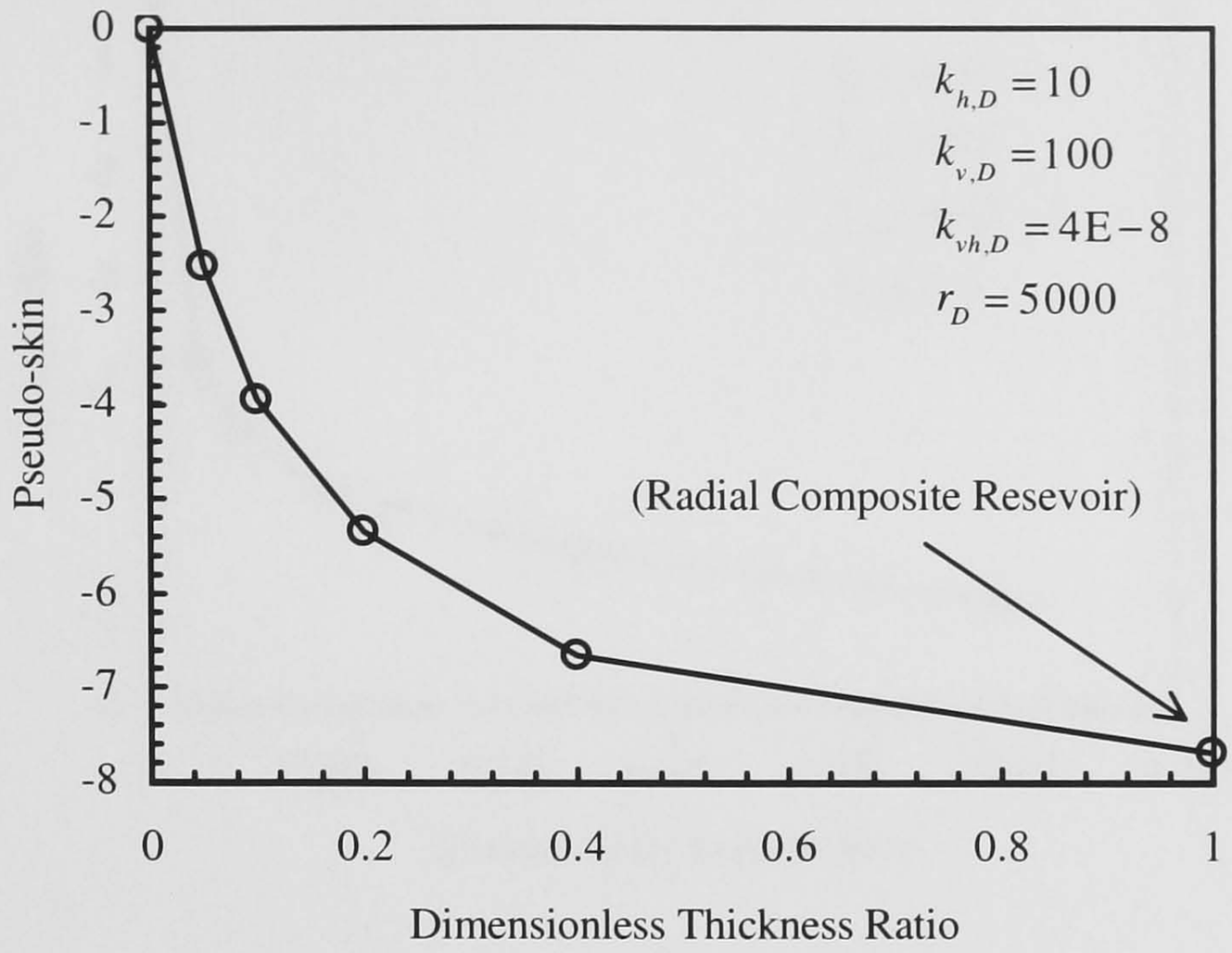


Figure 3.13: Effect of varying h_D on pseudo-skin.

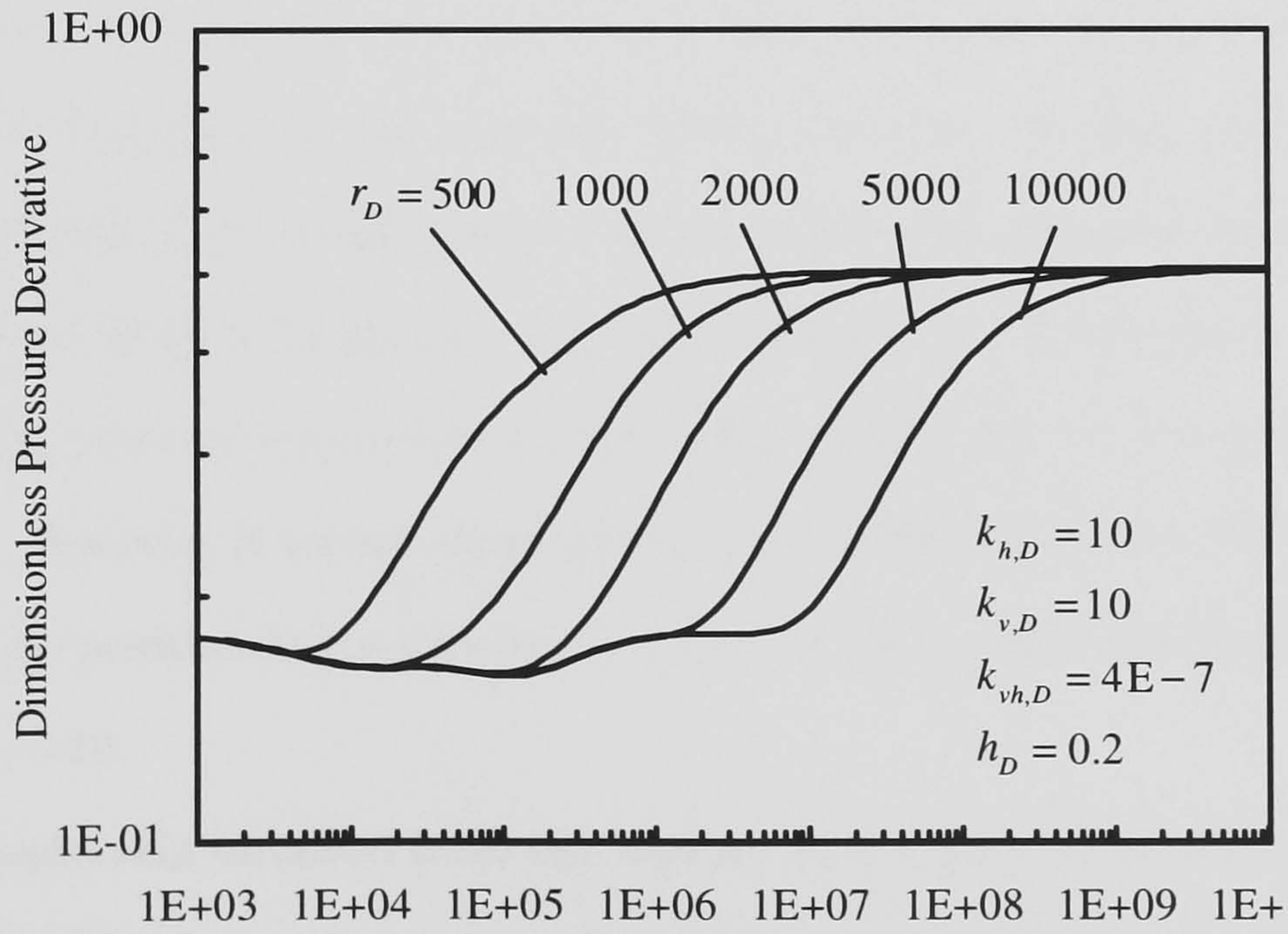


Figure 3.14: Effect of varying r_D on pressure derivative.

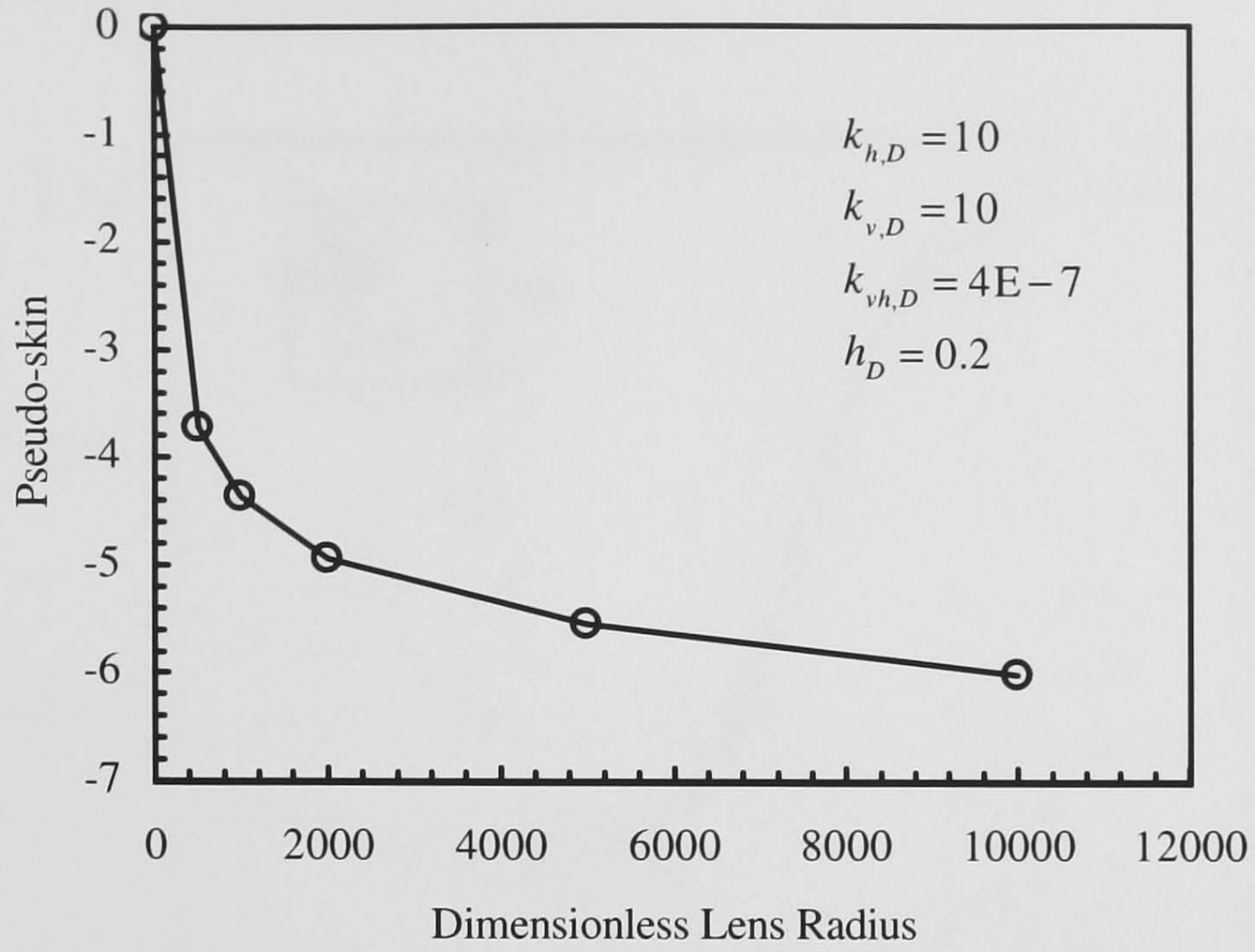


Figure 3.15: Effect of varying r_D on pseudo-skin.

those parameters on transient pressure response.

Dimensionless Distance of the Lens from the Reservoir Top

Fig. 3.16 shows the effect of the vertical position of the lens, d/h_m, where d is a distance from the top boundary of the reservoir to the centre of the lens, and h_m is a reservoir thickness. It affects the transition period (ii) and the convergence flow period (iii) since the crossflow is affected by the vertical position of the lens. As the lens is off-centred in the reservoir, the pressure derivative increases during those periods, causing a extra pressure drawdown. However, it seems impossible to observe this effect in actual welltest analyses since the lens position has a very minor effect on the pressure derivative curve as in the case of Fig. 3.10.

The pseudo-skin obtained from the analysis in the pseudo-radial flow period is shown in Fig. 3.17. The figure shows that the pseudo-skin becomes constant when the lens is located away from the top (or bottom) of the reservoir. However, if the lens is located at the top of the reservoir, the pseudo-skin becomes slightly small. This is because the convergence flow into the lens cannot be fully developed due to the presence of the reservoir

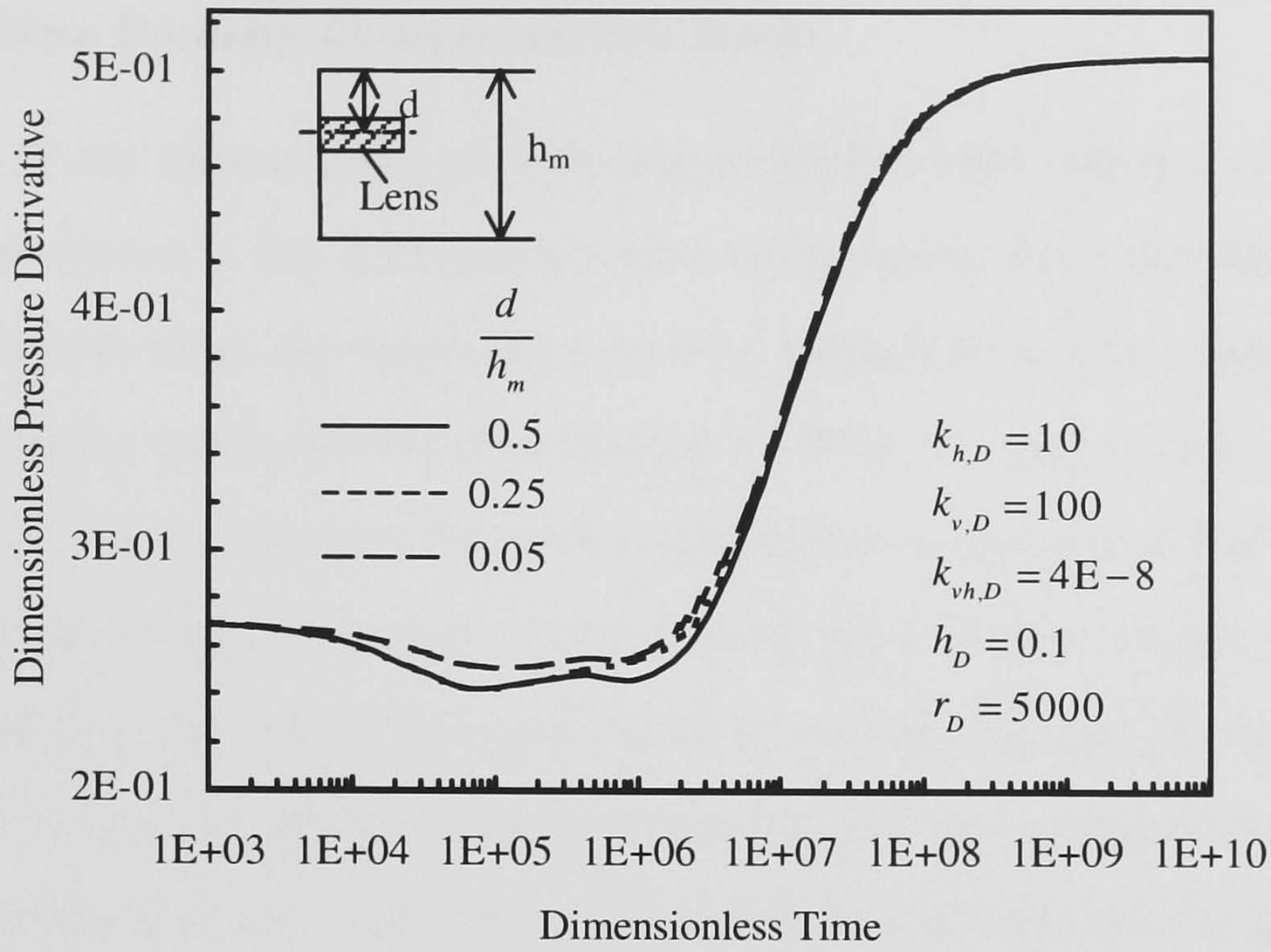


Figure 3.16: Effect of the vertical position of the lens.

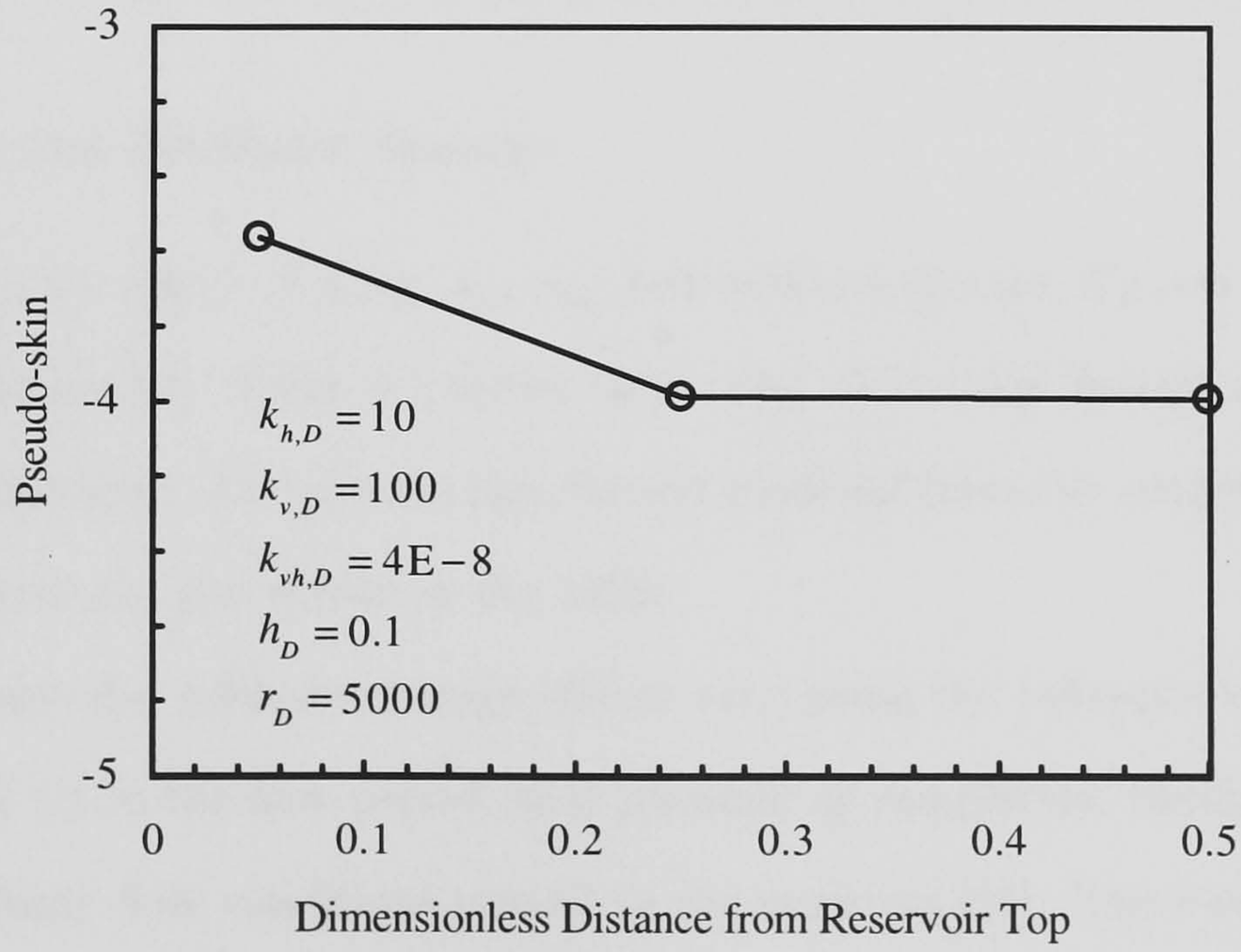


Figure 3.17: Effect of the vertical position of the lens on pseudo-skin.

top (or bottom) boundary.

Dimensionless Porosity-Compressibility Ratio

The effects of the dimensionless porosity-compressibility ratio, $(\phi c_t)_D$, on the pressure derivative are shown in Fig. 3.18 together with the fractional flow rate from the lens. As the ratio becomes large, the downward concavity, which indicates the significance of the crossflow from the matrix into the lens, diminishes. When the ratio is large, a small hump can be seen in the convergence flow period (iii) similar to the case of Fig. 3.10. In this case, the fractional flow of the lens is large at early times and the crossflow is negligible since enough fluid still exists in the lens. However, once the pressure disturbance reaches the edge of the lens, the intense convergence flow into the lens is required due to the large fractional flow at the early times. That causes the imbalance between the inflow into the lens and the production from the lens similar to the case of Fig. 3.10, which develops the hump in the period. However, since the effect is also very small in this case, it is very difficult to observe the effect in actual field examples.

The effect of $(\phi c_t)_D$ on the pseudo-skin is negligible. The pseudo-skin obtained from the analysis of the pseudo-radial flow period becomes constant (-5.4) in each case.

Skin Factors and Wellbore Storage

Fig. 3.19 shows the effects of skins, s_l , s_m, and wellbore storage, C_D, on the pressure and the pressure derivative. Table 3.1 shows each value of the skin factors and the wellbore storage examined here. The pseudo-skin factors obtained from the analysis of the pseudo-radial flow period are also shown in the table.

The skin and the wellbore storage effects may mask the influence of the lens (from the flow period (i) to the flow period (iii)) partially or completely. Similar situations can occur if non-Darcy flow conditions prevail in the reservoir [16]. The velocity of the fluid in the lens can be large if the permeability contrast between the reservoir matrix and the lens becomes large, which can give rise to the non-Darcy flow in the lens especially in gas wells. If the lens effect is masked completely, we will observe a similar response to

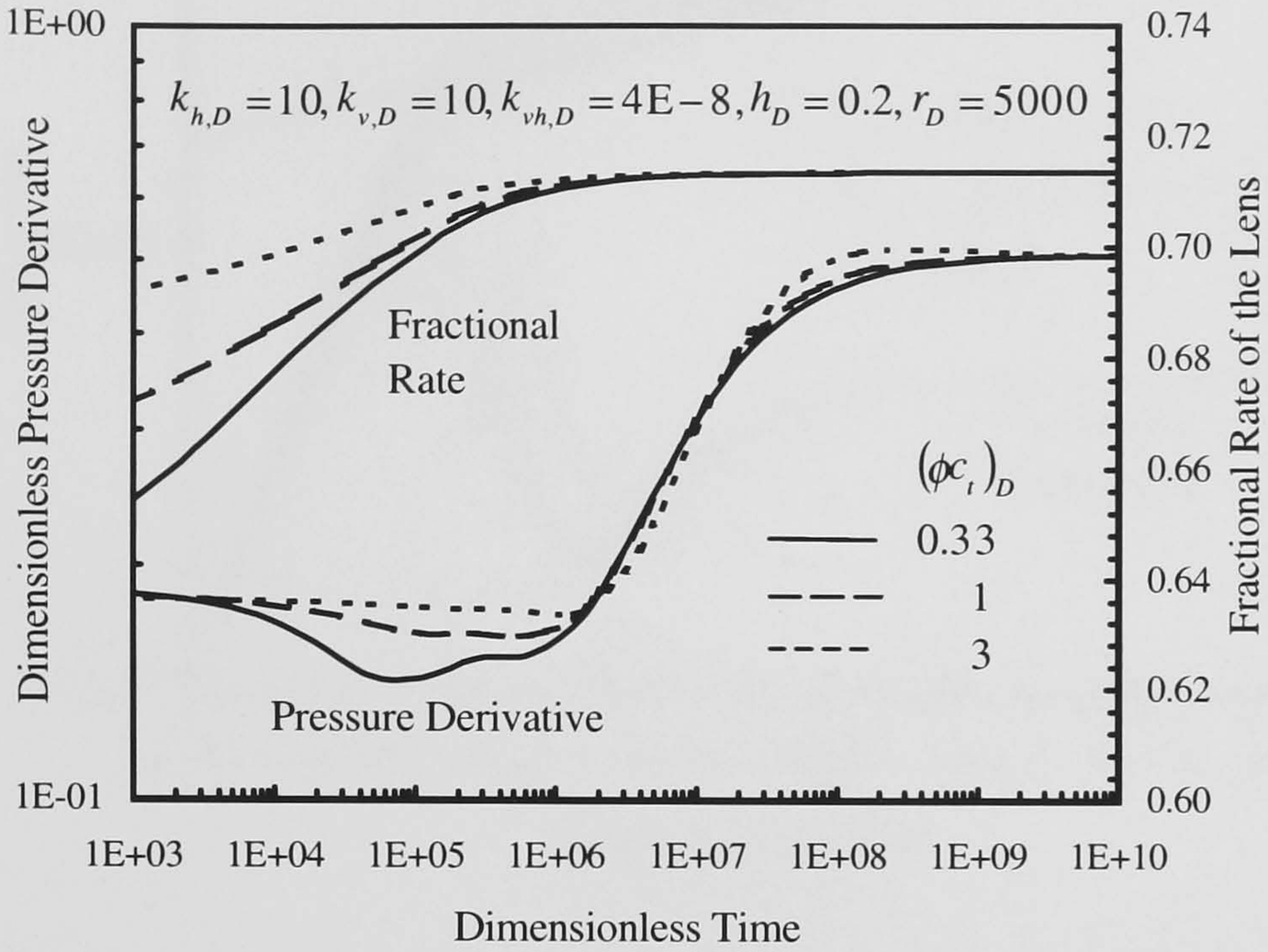


Figure 3.18: Effect of varying $(\phi c_t)_D$ on pressure derivative and fractional rate of the lens.

Table 3.1: Wellbore storage and skin factors for cases in Fig. 3.19.

Case	C_D	s_l	s_m	Pseudo-skin
1	0	0	0	-4.9
2	10^4	0	0	-4.9
3	10^4	10	0	-3.2
4	10^4	0	10	-4.5
5 [a]	10^4	10	10	-1.3

[a]This case is not shown in Fig. 3.19.

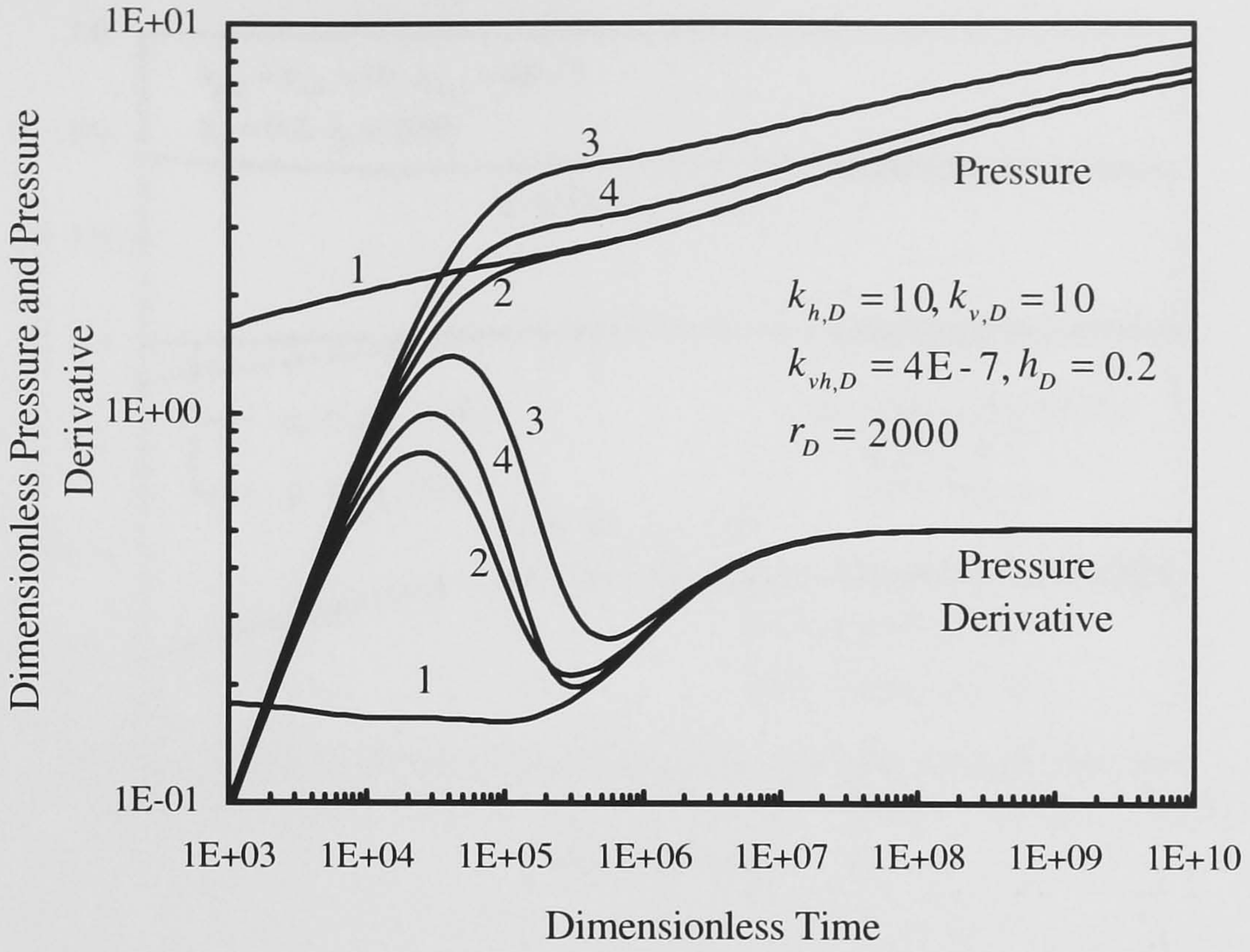


Figure 3.19: Effect of skin factors and wellbore storage.

a homogeneous reservoir with a reservoir matrix permeability and a constant skin. This causes difficulties when we analyse data from this type of reservoirs.

In Table 3.1, the decrease of the absolute value of the pseudo-skin can be recognised when the mechanical damage exists in the lens region and/or the reservoir matrix region. The damage of the lens region is more influential than that of the reservoir matrix region on the pseudo-skin (Cases 3 and 4). Even though both regions are damaged, a negative skin can be still obtained (Case 5). If the effect of the lens at early times is masked completely, and the reservoir is misinterpreted as a homogeneous reservoir (although we can identify the high permeability lens from core and wireline logs), we may misjudge that the well is not damaged due to the negative pseudo-skin in spite of the high skin factors.

3.2.6 Fractional Flow Rate Transition

Obtaining the flow rate profile along the wellbore is important to limit the non-uniqueness of the pressure matching. A combined analysis of pressure and flow rate yields a more convincing model than that without it. In this section, fractional flow rate from the lens

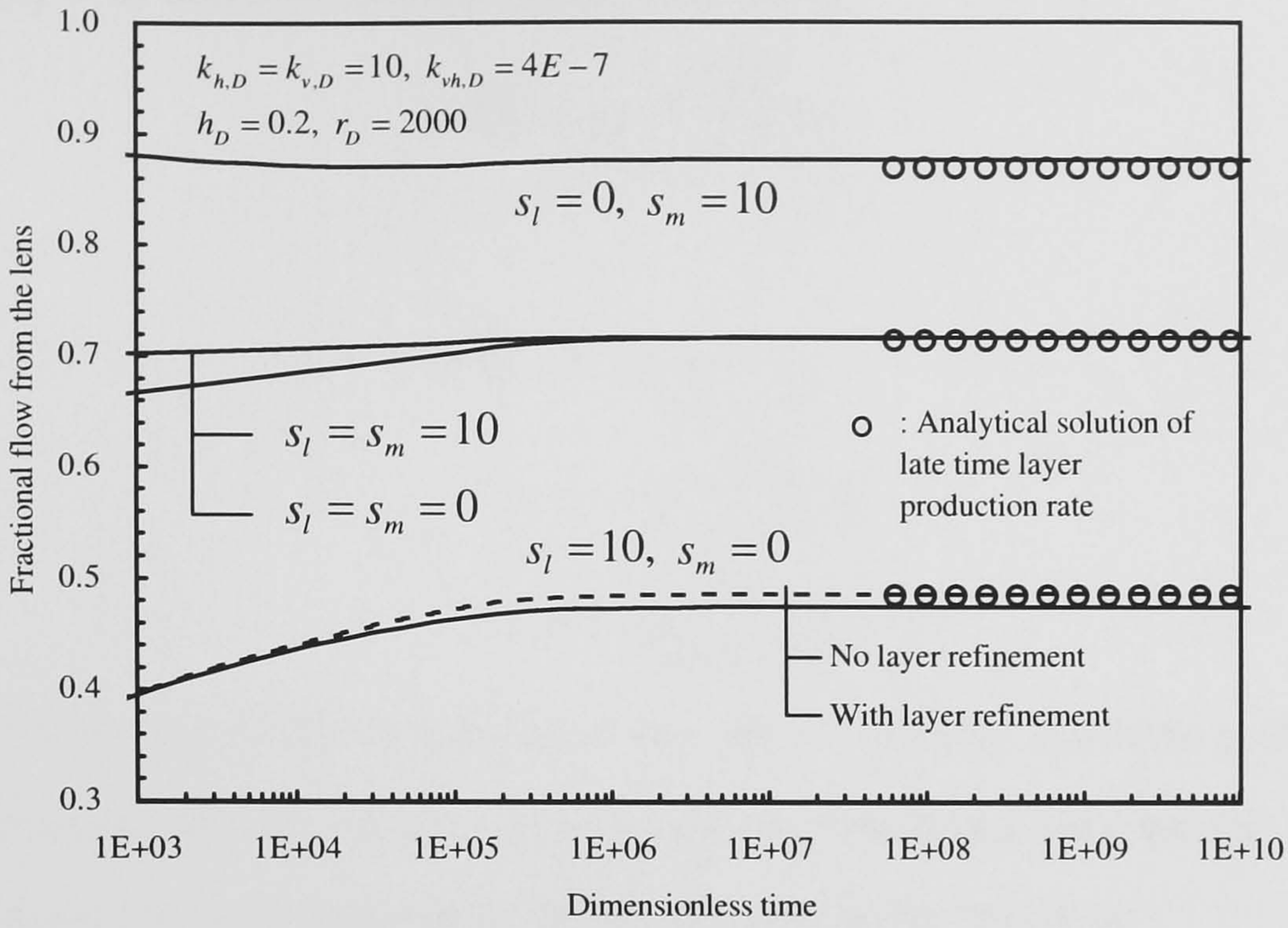


Figure 3.20: Transition of fractional flow rate from the lens.

is examined by referring to the flow rate behaviour of a layered reservoir with crossflow. The wellbore storage effect is not considered in this section ($C_D = 0$).

Fig. 3.20 shows the fractional flow rate transition from the lens when the lens or the matrix portion of the well is damaged or not damaged. It is observed that if the skin values for the lens and the matrix are different, the change in the fractional flow rate can be significant. If the skin values are the same, the late time rate becomes identical to that of the non-damaged well case.

When the range of investigation is smaller than the radius of the lens, the behaviour of the lens reservoir should be the same as that of the layered reservoir with crossflow. Hence, the theory on the crossflow reservoir can be applicable for that period.

Ehlig-Economides and Joseph [33], and Park and Horne [90] presented early-time and late-time analytical expressions of layer flow rate for a layered reservoir with crossflow. It can be shown that if some of the layers have non-zero skin values, the rate of the layer with non-zero skin ($s_j \neq 0$) becomes zero at very early times. The rate of the layer with

zero skin ($s_j = 0$) becomes constant and it is given by

$$q_{jD}^{ET} = \frac{\sqrt{\omega_j \kappa_j}}{\sum_{s_k=0} \sqrt{\omega_k \kappa_k}}, \tag{3.114}$$

where

$$q_{jD}^{ET} = \frac{q_j}{\sum_{s_k=0} q_k}, \tag{3.115}$$

$$\kappa_j = \frac{k_j h_j}{\sum_{s_k=0} k_k h_k}, \tag{3.116}$$

$$\omega_j = \frac{\phi_j h_j}{\sum_{s_k=0} \phi_k h_k}. \tag{3.117}$$

As a special case, if the skin factor is zero for every layer, the layer production rate becomes constant at early times. In this case, the fractional rate for the layer j in a n-layered reservoir can be written in the similar way to Eq. 3.114 as

$$q_{jD}^{ET} = \frac{\sqrt{\omega_j \kappa_j}}{\sum_{k=1}^{n} \sqrt{\omega_k \kappa_k}}, \tag{3.118}$$

where

$$q_{jD}^{ET} = \frac{q_j}{\sum_{k=1}^{n} q_k}, \tag{3.119}$$

$$\kappa_j = \frac{k_j h_j}{\sum_{k=1}^{n} k_k h_k}, \tag{3.120}$$

$$\omega_j = \frac{\phi_j h_j}{\sum_{k=1}^{n} \phi_k h_k}. \tag{3.121}$$

If the skin factor is non-zero for every layer, the layer production rate becomes also constant at early times. In this case, the fractional rate for the layer j is given by

$$q_{jD}^{ET} = \frac{\kappa_j / s_j}{\sum_{k=1}^{n} \kappa_k / s_k}. \tag{3.122}$$

Late-time rate becomes constant regardless of the skin of each layer. For a two-layer reservoir, the fractional flow rates are expressed as

$$q_{1D}^{LT} = \kappa_1 \left\{ 1 + \frac{\kappa_2 (s_2 - s_1)}{K_1^0(\sigma) + \kappa_1 s_2 + \kappa_2 s_1} \right\}, \tag{3.123}$$

$$q_{2D}^{LT} = \kappa_2 \left\{ 1 + \frac{\kappa_1 (s_1 - s_2)}{K_1^0(\sigma) + \kappa_1 s_2 + \kappa_2 s_1} \right\}, \tag{3.124}$$

where

$$K_1^0(\sigma) = \frac{K_0(\sigma)}{\sigma K_1(\sigma)}, \tag{3.125}$$

$$\sigma = \sqrt{\frac{\lambda}{\kappa_1} + \frac{\lambda}{\kappa_2}}, \tag{3.126}$$

$$\lambda = \frac{2{r_w}^2}{(k_1 h_1 + k_2 h_2)(\frac{h_1}{k_{v,1}} + \frac{h_2}{k_{v,2}})}, \tag{3.127}$$

and $K_0(\sigma)$ and $K_1(\sigma)$ are the modified Bessel functions of the second kind and order zero and first, respectively.

In Fig. 3.20, the early-time constant-rate behaviour as described above cannot be observed. However, at early times, the fractional rate of the case where both of the skins are 10 is different from that of the no damage case, which is justified by the difference between Eq. 3.118 and Eq. 3.122. When the skin of the lens region is 10 and that of the matrix region is zero, the fractional rate from the lens increases as the time becomes large at early times. When the skin of the lens region is 0, and that of the matrix region is 10, the rate decreases as the time becomes large. These observations are supported by Ref. [90] since the rate of the layer with non-zero skin becomes zero at early times. At late times, it can be observed that the fractional flow rate from the lens shows good match with that calculated from Eq. 3.123 in each case. Small differences are due to the effect of the layer refinement discussed in Section 3.2.1 since Eq. 3.123 and Eq. 3.124 were derived by only considering two layers. If the layer refinement is not taken into account in the lens reservoir model, the result (the dotted line) exactly matches with the result of Eq. 3.123 (although this is not accurate) as shown in Fig. 3.20.

However, if the radius of the lens is small, it is impossible to predict the late-time fractional rate from Eqs. 3.123 and 3.124. The effect of the lens radius on the flow rate transition is shown in Fig. 3.21. In this case, there are no skins for both the lens and the matrix regions. It is observed that, at late times, the fractional flow rate becomes small as the dimensionless lens radius decreases. However, the fractional flow rate still becomes constant at late times. The rate history (the increase of the rate followed by its decrease, and final stabilisation) is somewhat similar to that of a commingled reservoir [73]. In commingled reservoirs, the fractional flow rate from the layer j becomes equal to the ratio of the pore volumes described by Eq. 3.121 after boundary effects are felt and semi-steady

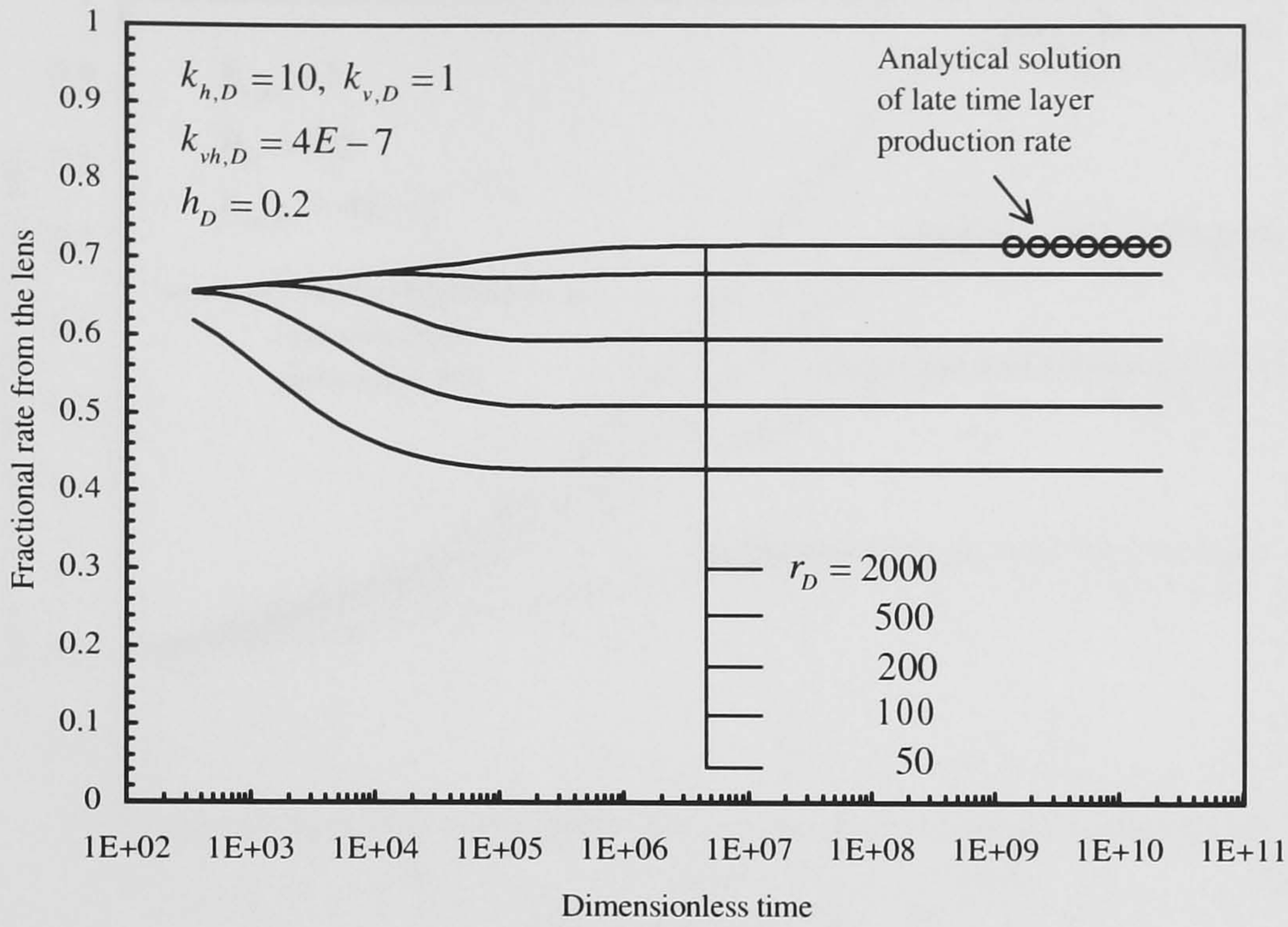


Figure 3.21: Effect of the lens radius on fractional flow rate from the lens.

state is reached [77]. However, no simple equation to predict the fractional flow rate from the lens for the lens reservoir case has been obtained yet.

In Fig. 3.22, the stabilised late-time fractional rate was plotted as a function of the dimensionless lens radius and the dimensionless horizontal permeability ratio. If the lens radius is large enough, the rate reaches that for a crossflow reservoir case (the thick line). However, when the horizontal permeability ratio is large, a larger lens radius is required to reach the same level as the crossflow reservoir. On the other hand, if the lens radius becomes small, the rate approaches the dimensionless thickness ratio as expected.

As shown in this section, the late-time flow rate includes the information on the lens radius. Hence, it is important to obtain the flow rate data as well as the pressure transient data for determining the lens radius.

3.2.7 Effects of the Shape of the Lens and the Well Position

So far, it has been assumed that the shape of the lens is circular, and the well intersects the lens at its centre. However, since the lens shape and the well position will not always

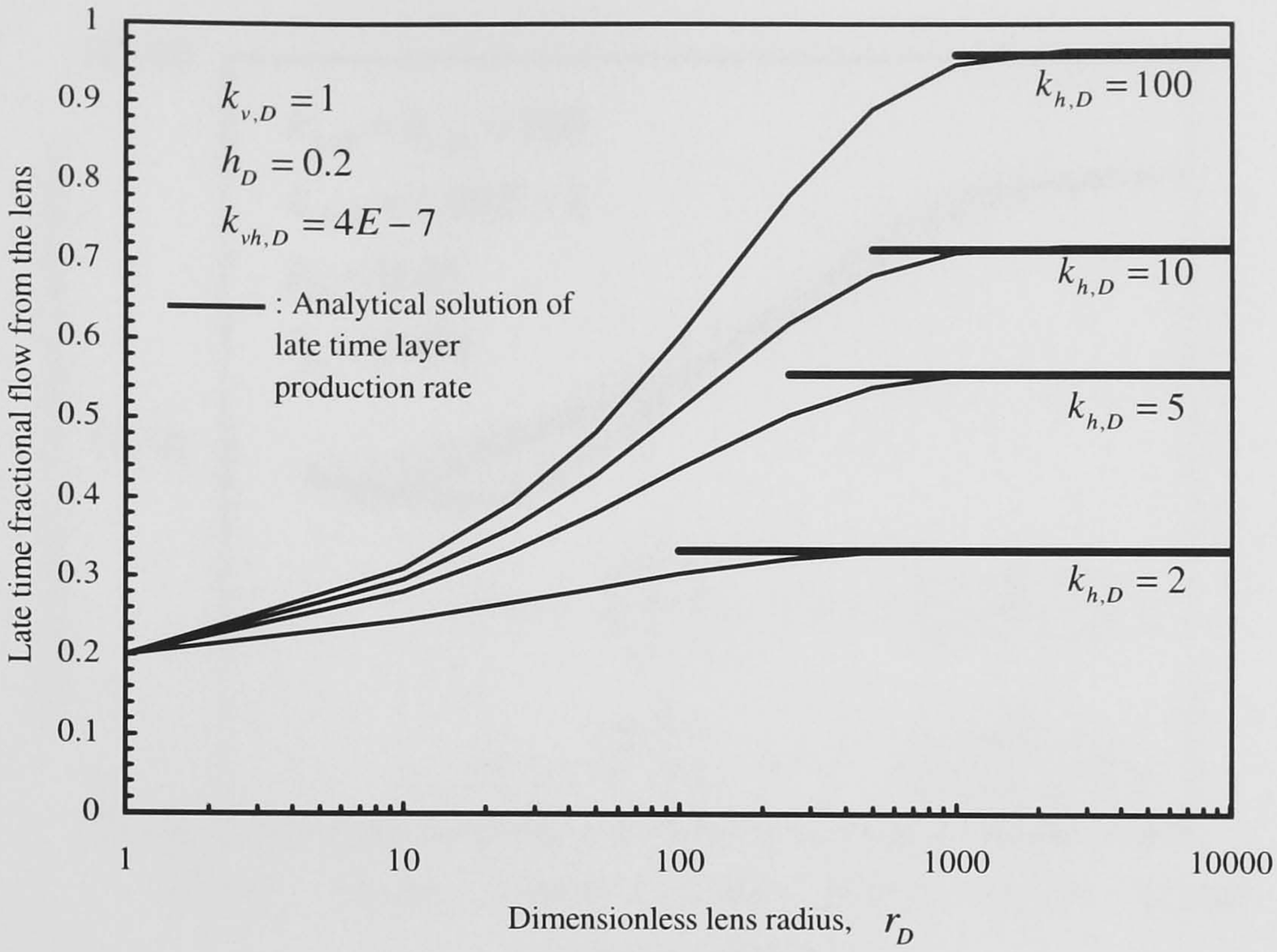


Figure 3.22: Effect of the lens radius on late-time fractional flow rate from the lens.

be in such ideal conditions, the effect of this asymmetry on the pressure response should be examined. The effects of those factors are shown in Fig. 3.23.

A numerical simulation model with Cartesian geometry was used to generate these type-curves. The size of the simulation grid is 69 $\times$ 69 $\times$ 28 (x, y, and z), which was considered to be enough by examining sensitivity to the grid size and initial timesteps.

The same dimensionless variables as those for the circular lens reservoir were employed for these square and rectangular lens cases. However, in Eq. 3.105, the lens radius r_l was replaced by equivalent lens radius, $\tilde{r}_l$, which is the radius of a circle with the same area as that of the non-circular lens, i.e.,

$$\tilde{r}_D = \frac{\tilde{r}_l}{r_w}, \tag{3.128}$$

$$\tilde{r}_l = \sqrt{\frac{(\text{Area of the non-circular lens})}{\pi}}. \tag{3.129}$$

The equivalent lens radius is the same in each case in the figure. This means that the pore volume of the lens is identical in each case, since the lens thickness is constant. The

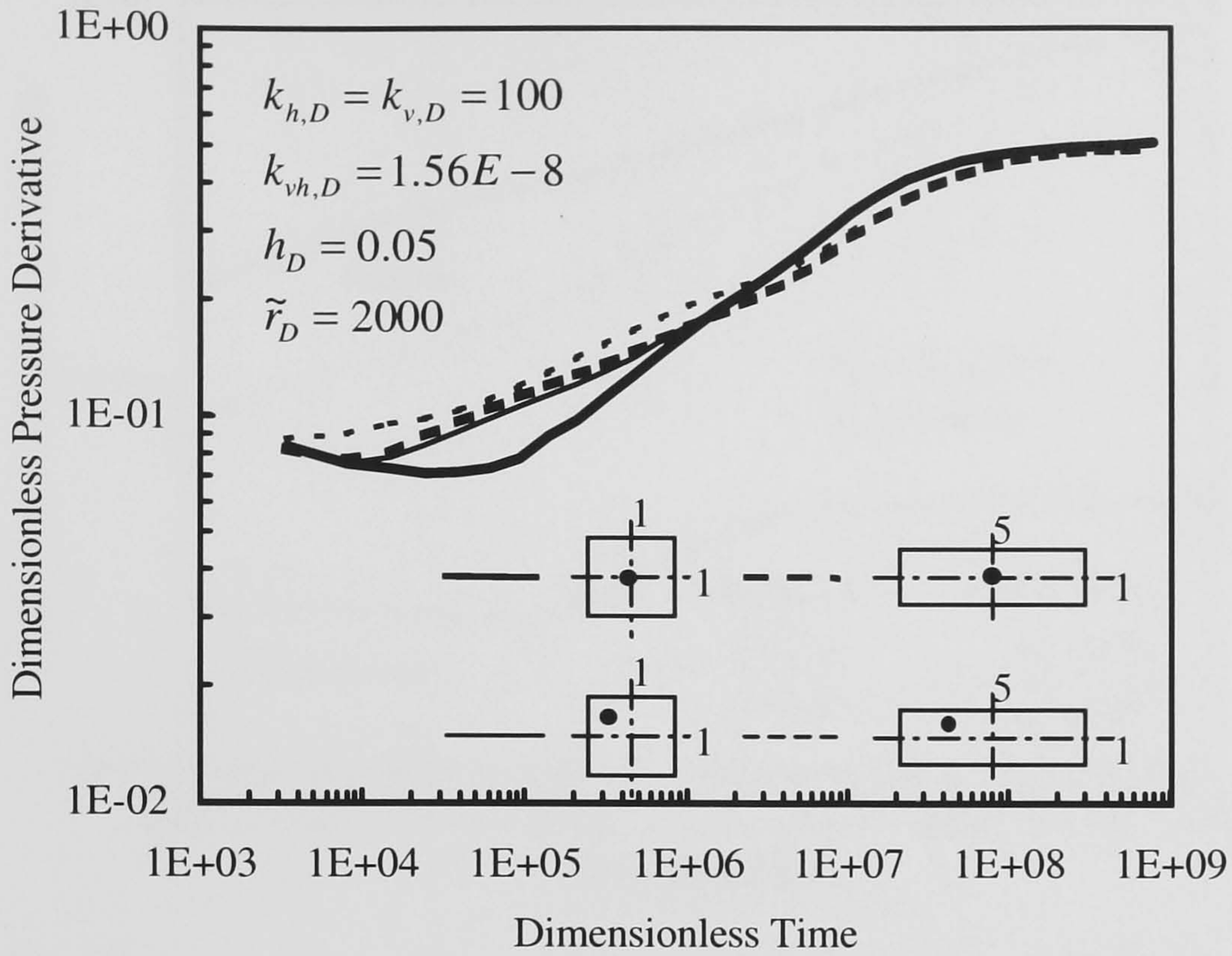


Figure 3.23: Effect of the lens shape and the well position.

aspect ratios for the rectangular cases are set to 1:5.

In the square lens case with the well in the centre, the behaviour is indistinguishable from that of the circular lens case (not shown here). However, if the well position becomes close to one of the corners of the lens, the onset of the convergence flow into the lens becomes earlier because the pressure disturbance reaches the lens boundary earlier. A similar behaviour can be observed in the rectangular lens case. At late times, all the curves become identical, showing the similar pseudo-radial flow period to that observed for the circular lens cases. If the pseudo-skin is analysed in that late time period, the same skin factors of -6.5 can be obtained in all cases.

The pressure behaviours of a long thin lens case and a short thick lens case are shown in Fig. 3.24. The lens (pore) volume is the same in both two cases. The lens radius and the lens thickness have been changed so as to be able to compare the behaviour between the two cases (assuming the volume of the wellbore is negligible). The pseudo-skins calculated from the data in the pseudo-radial flow period are -3.3 for the long thin lens case and -

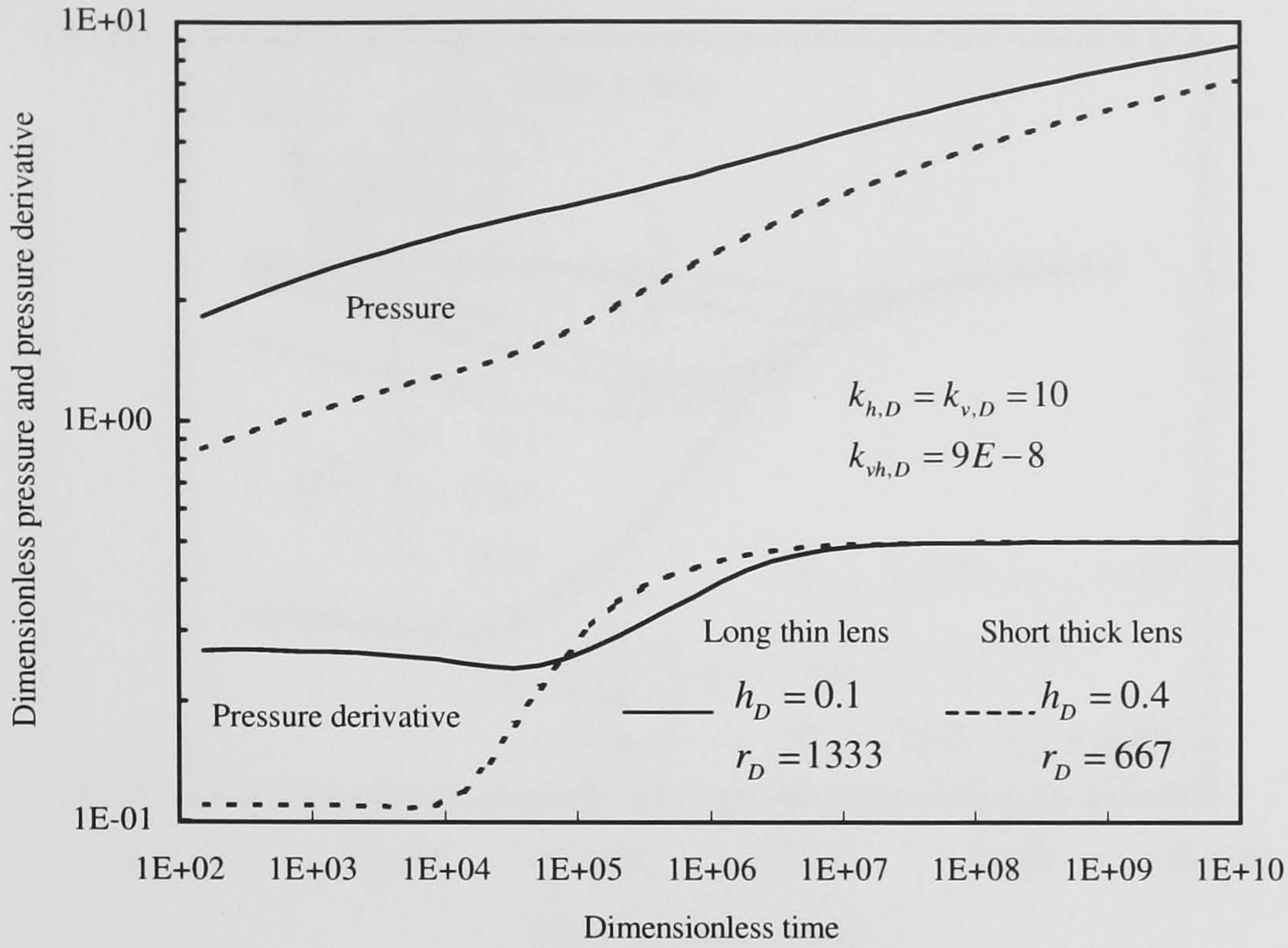


Figure 3.24: Comparison of pressure and pressure derivative responses between two cases which have the same lens volume.

4.8 for the short thick lens case. Valkó and Economides [118] reported in their study on transient behaviour of finite-conductivity horizontal fractures that for a given fracture volume, the increase of fracture radius at the expense of reducing fracture width always deteriorates the performance of the fractured well (except for the case when the wellbore radius is already commensurable to the fracture radius).

From these examples, it is expected that the pseudo-skin of a lens with the aspect ratio up to 1:5 becomes almost the same regardless of its shape and the well position, if the thickness and the equivalent radius of the lens are the same. In addition, the divergence in Fig. 3.23 at early times is small to be recognised in actual field tests. Hence, it can be said that the lens elongated in one direction with the aspect ratio up to 1:5 and an arbitrary well position can be approximated by a circular lens penetrated by a well in the centre with the same equivalent lens radius. However, even if the pore volume of the lens is the same, but the thickness and the equivalent lens radius are changed, the pseudo-skin becomes no longer the same. The performance of a short thick lens is better than that

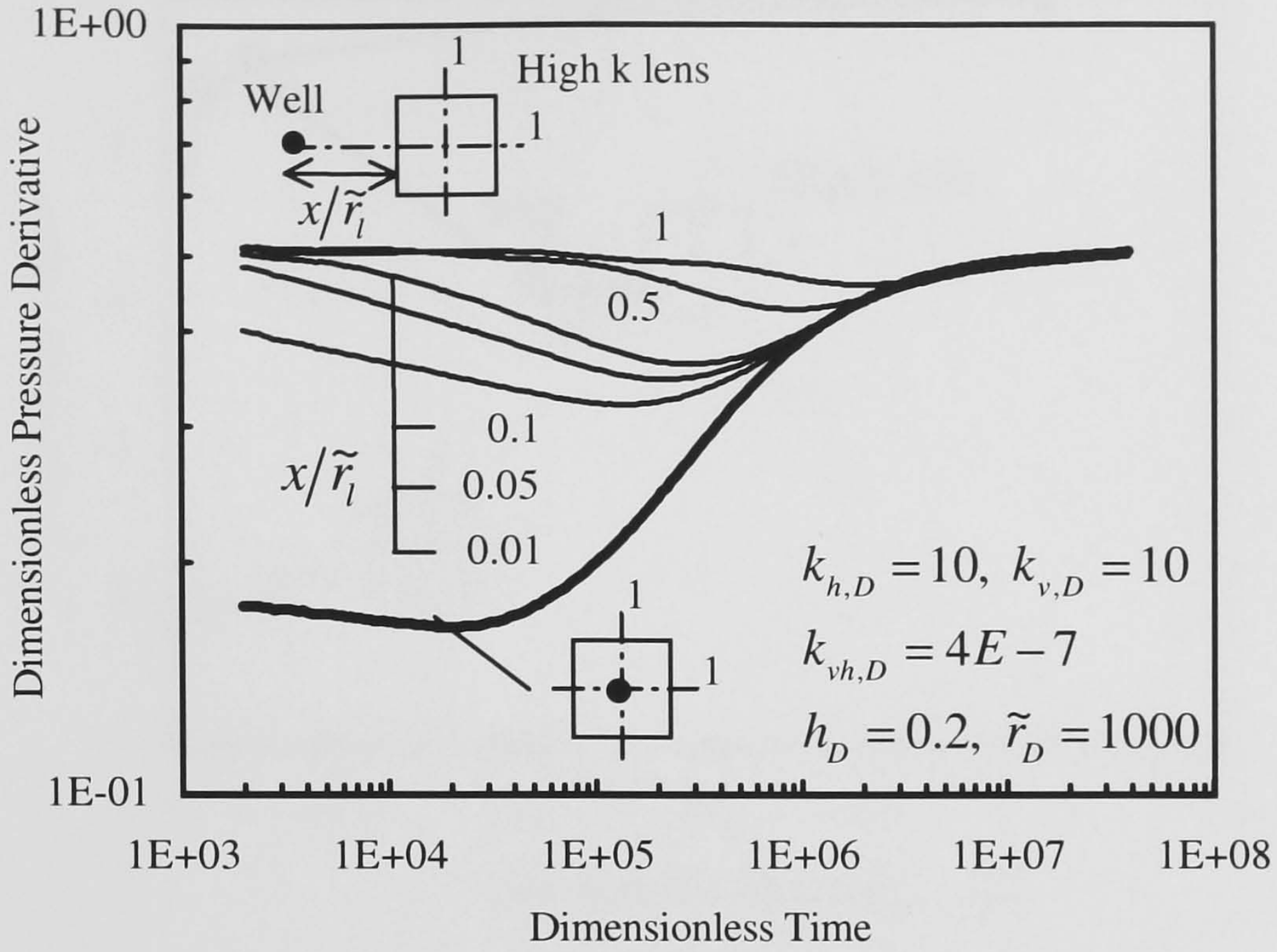


Figure 3.25: Effect of varying distance of the lens from the wellbore on pressure derivative.

of a long thin lens for a given lens volume in terms of the pseudo-skin calculated in the pseudo-radial flow period.

3.2.8 Effects of a Distant Lens from the Wellbore

Even though the lens is not penetrated by the well, it affects pressure response according to the size, the rock properties, or the position of the lens. Fig. 3.25 shows the effect of the position of the high permeability lens located away from the wellbore. The square-lens numerical simulation model in the previous section was used to generate those type-curves. The distance from the centre of the wellbore to the nearest lens boundary x was expressed in dimensionless form as $x/\tilde{r}_l$ where $\tilde{r}_l$ is the equivalent lens radius defined by Eq. 3.129. The skin and the wellbore storage effects are ignored. The case where the well penetrates the lens at the centre was also plotted as a reference.

A downward concavity can be seen at early times even though the lens is not penetrated by the wellbore, and the size of the concavity becomes small as the distance to the lens

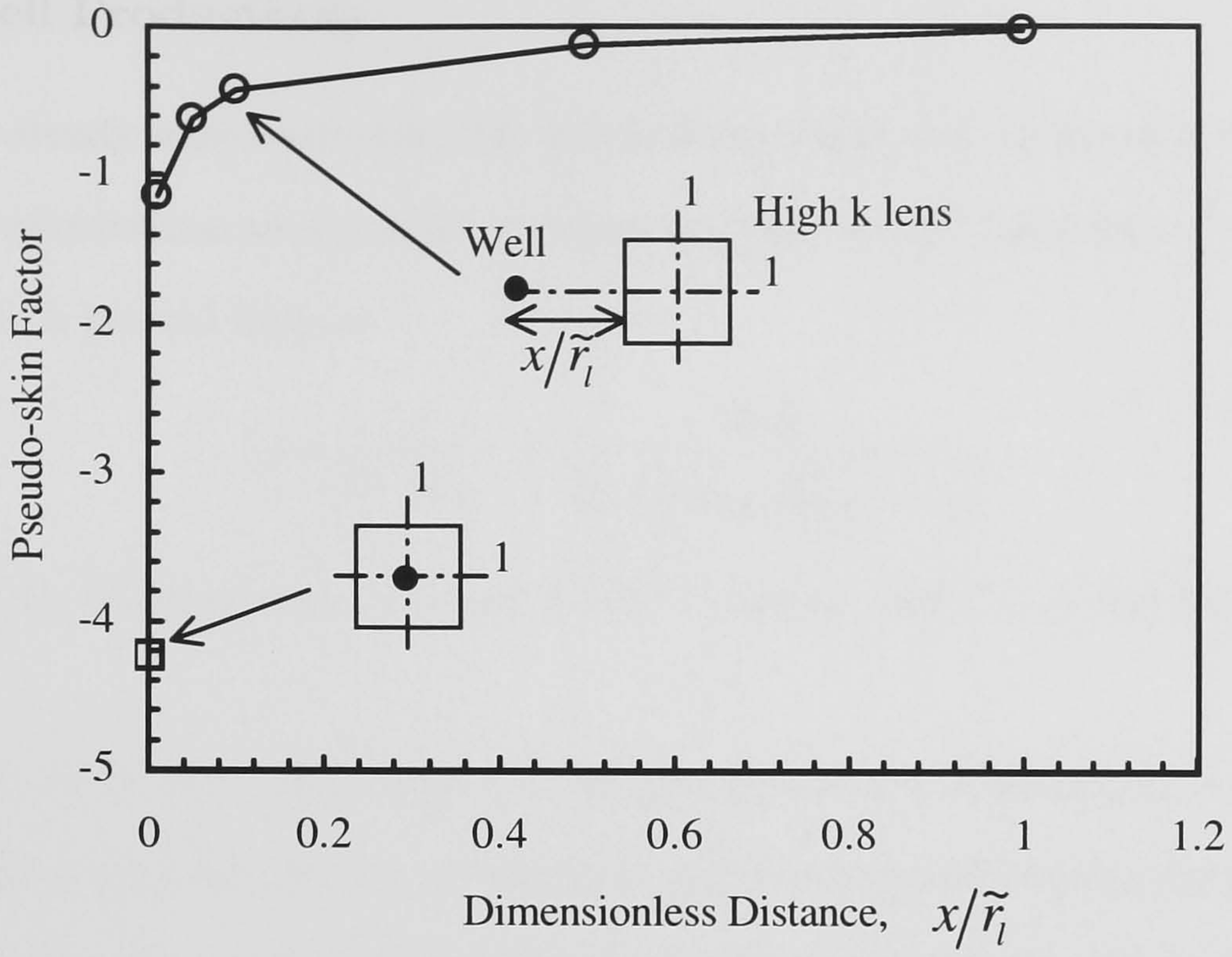


Figure 3.26: Effect of varying distance of the lens from the wellbore on pseudo-skin factor.

increases. The behaviour is quite similar to that of a double-porosity reservoir, hence, some geological information is necessary to choose an appropriate reservoir model.

At late times, all the derivative curves become identical and show a 0.5 straight line, which corresponds to the pseudo-radial flow period. The pseudo-skin calculated by analysing data in that period is shown in Fig. 3.26. The pseudo-skin approaches zero as the distance to the lens increases. In this case, if the distance is approximately the same as the equivalent lens radius (i.e., $x/\tilde{r}_l \approx 1$), the pseudo-skin becomes almost zero. When the dimensionless distance is 0.01, the pseudo-skin becomes -1.1, while, if the well penetrates the lens in the centre, it becomes -4.2. This is 3.8 times larger than the non-penetrated lens case. Since the well position has a negligible effect on the pseudo-skin as discussed in the previous section, it is critical that the well penetrates part of the lens for achieving improved well productivity. The method to calculate the well productivity for the lens reservoir is discussed in the next section.

3.2.9 Well Productivity

Under semi-steady state conditions, the productivity index is an important factor to compare well productivities among wells or before and after a well stimulation. The index can be written in a general form as

$$J = \frac{q}{\overline{p} - p_{wf}} = \frac{2\pi kh}{B\mu\left(\frac{1}{2}\ln\frac{4A}{e^{\gamma}C_A r_w{}^2} + s\right)}, \tag{3.130}$$

where A is the drainage area, γ is the Euler's constant, and C_A is the Dietz shape factor [23].

As mentioned before, the response of the lens reservoir is equivalent to a homogeneous reservoir with a reservoir matrix permeability and a pseudo-skin during the pseudo-radial flow period (iv). If the flow period (iv) is observed, the pseudo-skin can be analysed by regarding the reservoir as a equivalent homogeneous reservoir with a reservoir matrix permeability. The pseudo-skin can include the effect of the damage skin. Therefore, it can be applied to Eq. 3.130 directly in order to calculate the well productivity of the lens reservoirs which may have various drainage shapes.

3.2.10 Field Example

A pressure buildup test (DST3) for the well 30/6-19 was performed in fluvial reservoir environment of the North Sea. Core and plug permeability data of this well and the geological interpretation are shown in Figs. 3.27 and 3.28. Details of this test and accompanying core images are fully described in Ref. [125] in a larger study of well test interpretation in the Middle Jurassic Ness Formation reservoir.

In a large scale, the sandstone body shown in Fig. 3.27 is considered to be elongated in one direction. Fig. 3.28 is the cross-section perpendicular to that direction. Inside the sandstone body, three zones can be observed: a thick medium permeability sandstone zone (the permeability is around 100 md), a thin high permeability sandstone zone (1,000 md), and a thin low permeability interval (< 1 md) between the two. Fluid and reservoir parameters for a buildup curve matching are shown in Fig. 3.29 and Table 3.2. It was assumed that the shape of the high permeability zone was radial. The zone was considered

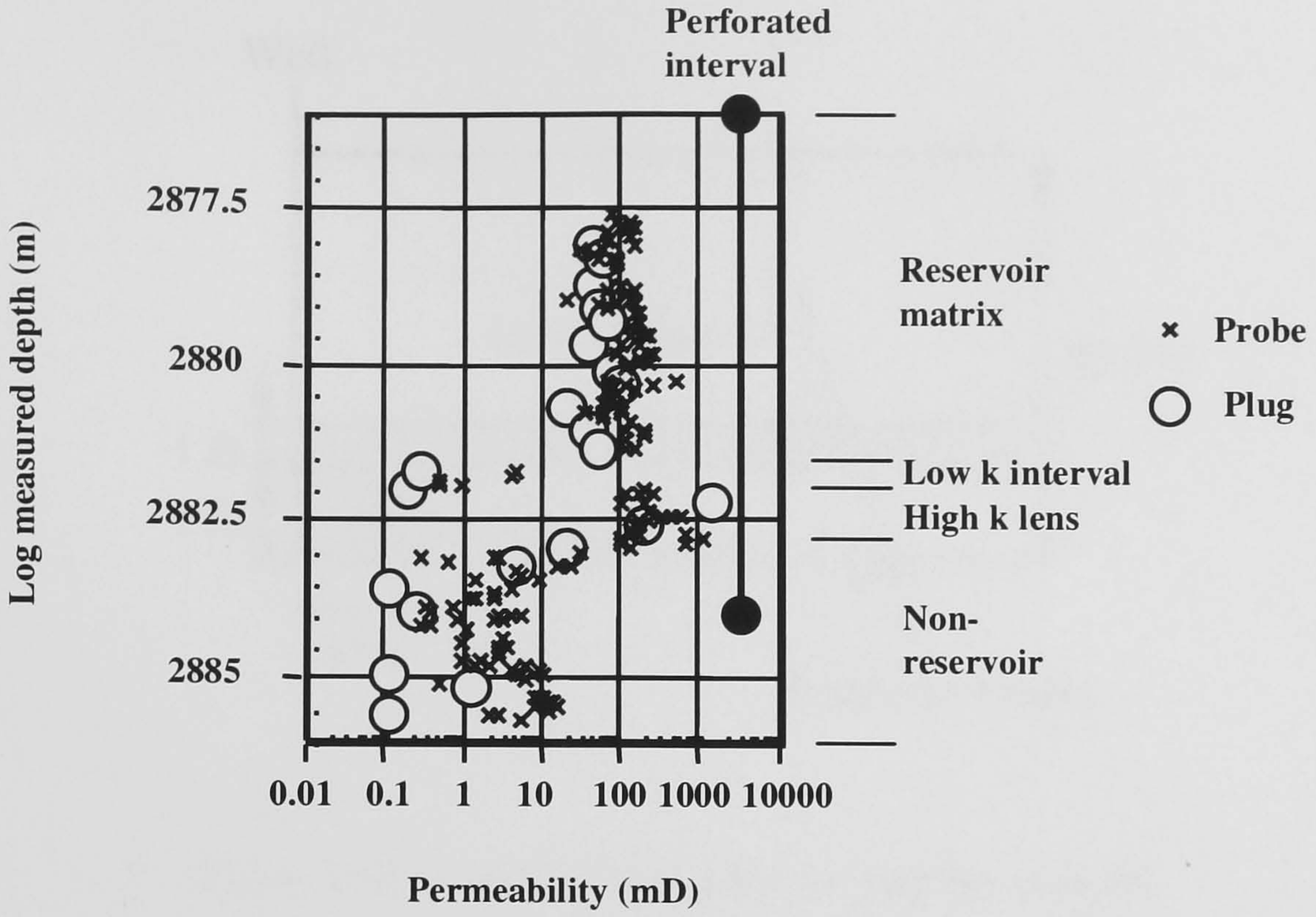


Figure 3.27: Probe and plug permeability data of well 30/6-19. The probe data confirm that the high and low plugs in the interval 2881.6-2882.8' are representative of the formation properties.

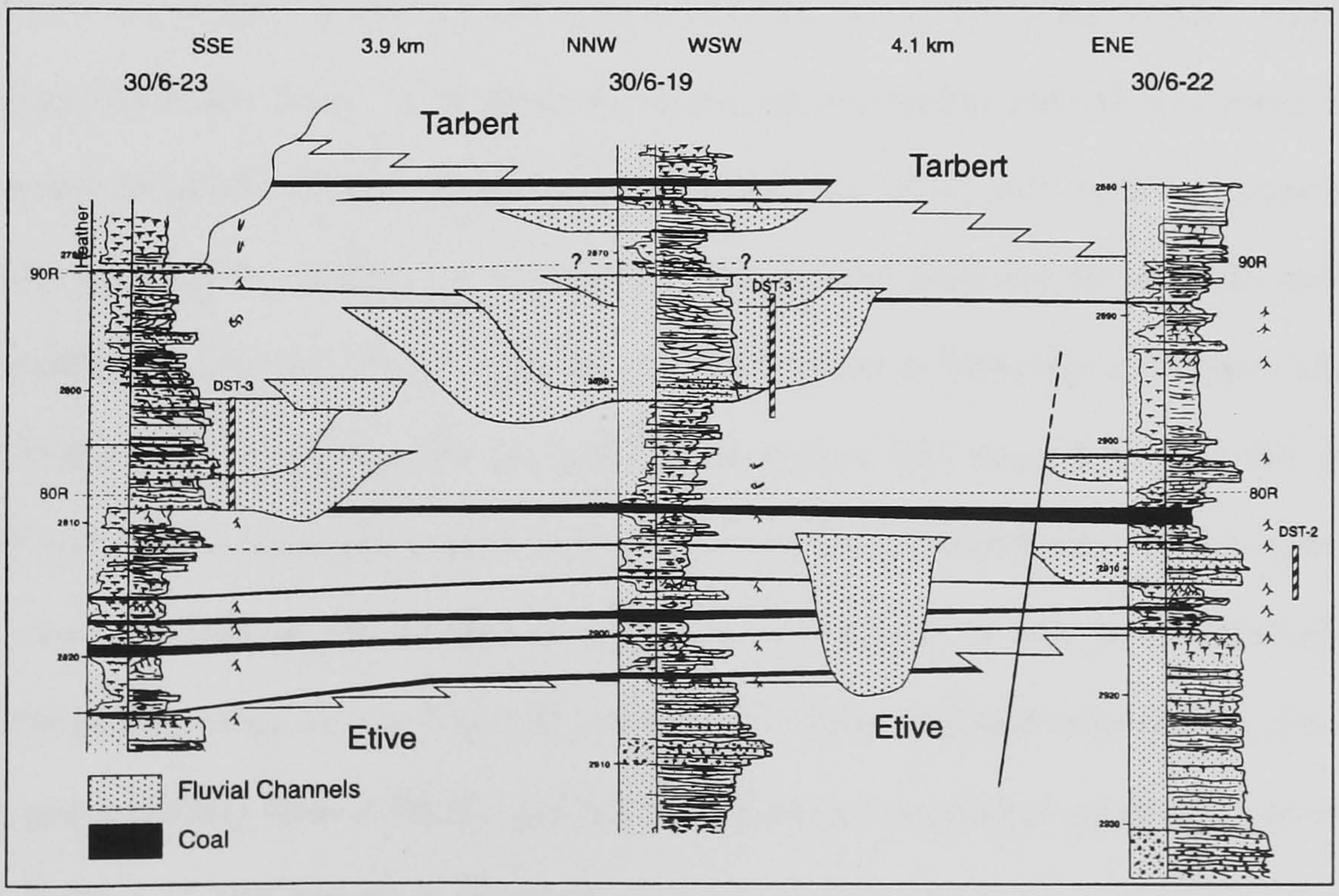


Figure 3.28: Geological interpretation of well 30/6-19 (DST3) [125].

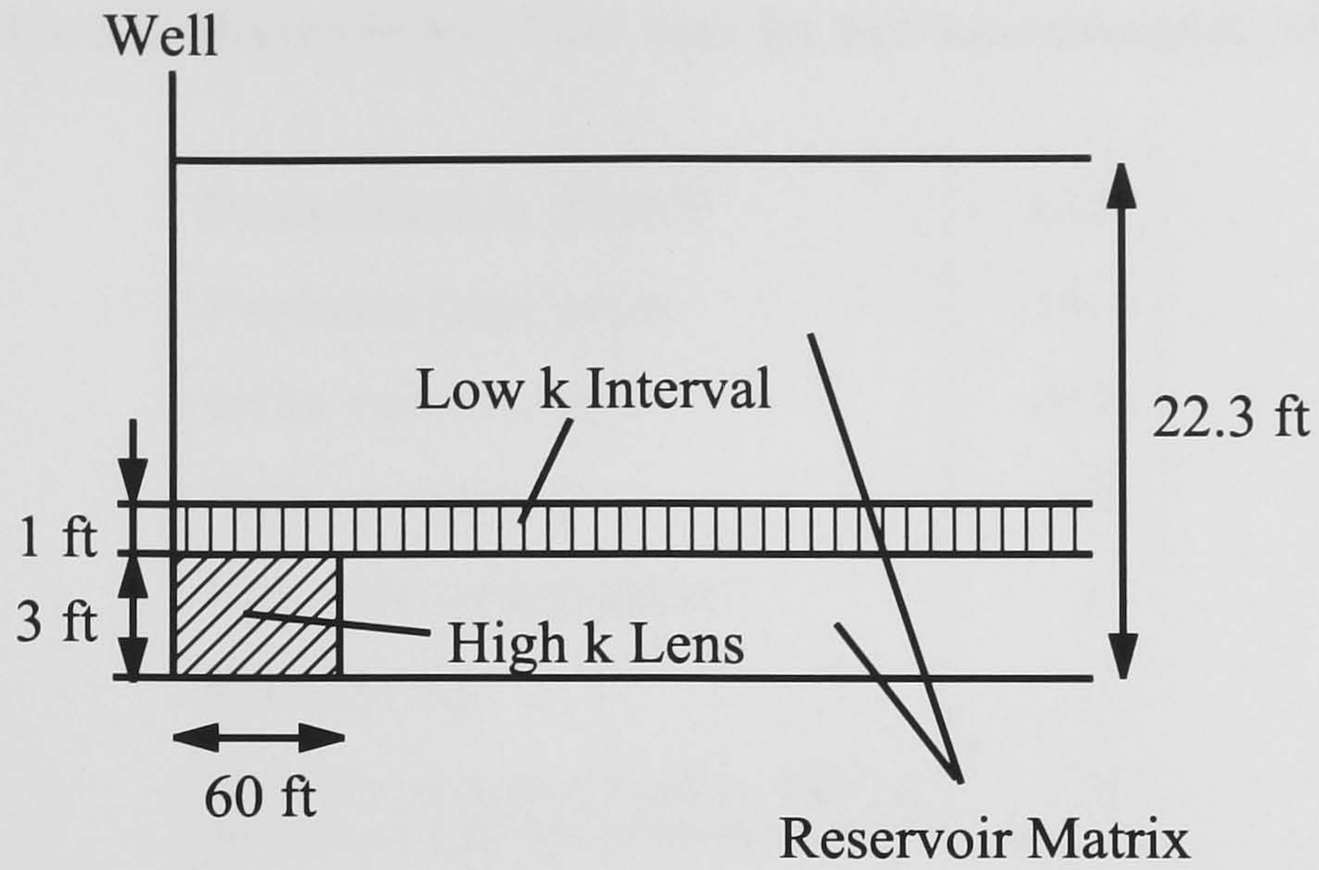


Figure 3.29: Reservoir model for the buildup analysis.

to be surrounded by the medium permeability matrix as a lens. A buildup pressure and a pressure derivative were calculated based on drawdown responses using Eqs. 3.110 and 3.112.

In Fig. 3.30, the field data was plotted together with the analytical solution with the infinite outer boundary. Input model permeabilities are in good agreement with the core and plug permeability data. The analysis of the pseudo-radial period (iv) using the radial homogeneous reservoir model showed the permeability of 97 md, which is approximately the thickness weighted arithmetic averaged permeability between the matrix and the thin low permeability interval. If the high permeability zone is laterally extensive, the permeability obtained by analysing the pseudo-radial period (iv) should be 215 md (thickness weighted arithmetic averaged permeability from the reservoir matrix, the low permeability interval, and the high permeability sand zone). Therefore, it can be concluded that the high permeability zone is not laterally extensive. This analysis also shows the radius of the high permeability lens is 60 ft, and the skin factors associated with each zone was 5.0.

The decline of the pressure derivative curve at late times was considered to be the effect of a layer thickening of the channel or a layer thickening due to the amalgamation of channels toward the edge of the reservoir matrix. The thickening and/or the amalgamation

Table 3.2: Reservoir and fluid data for buildup curve matching.

Production rate, STB/D	3,145
Producing time, hours	30.0
Initial pressure, psi	4,600
Wellbore radius, ft	0.36
Formation volume factor	1.4
Viscosity, cp	0.6
Wellbore storage constant, bbl/psi	0.0
Reservoir matrix	
Thickness, ft	22.3
Horizontal permeability, md	95
Vertical permeability, md	63
Porosity	0.2
Total compressibility, 1/psi	1.18E-5
Skin	5.0
High permeability lens	
Thickness, ft	3.0
External radius, ft	60
Horizontal permeability, md	1,020
Vertical permeability, md	680
Porosity	0.2
Total compressibility, 1/psi	1.18E-5
Skin	5.0
Low permeability interval (not perforated)	
Thickness, ft	1.0
Horizontal permeability, md	0.2
Vertical permeability, md	0.13
Porosity	0.2
Total compressibility, 1/psi	1.18E-5
Skin	5.0

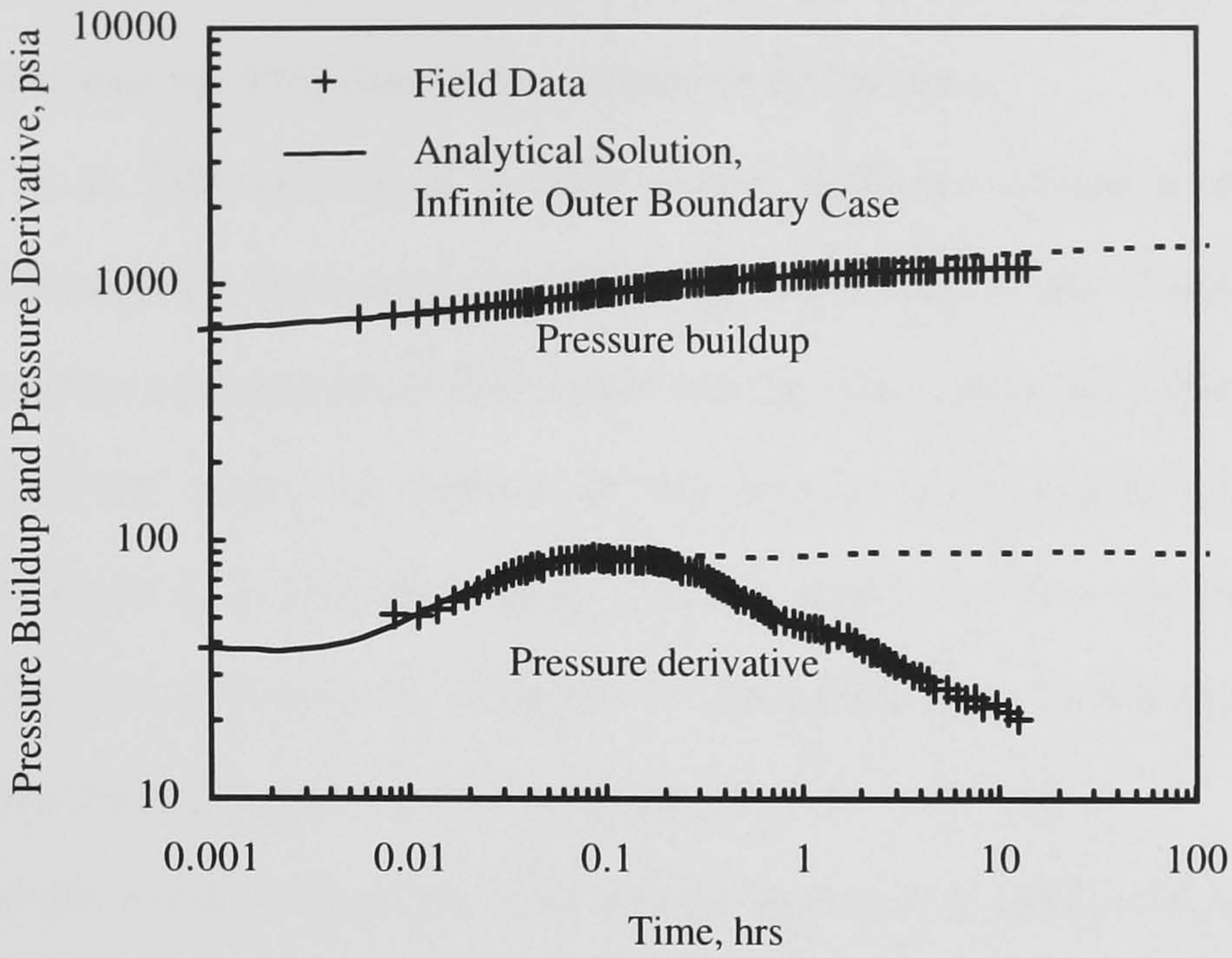


Figure 3.30: Results of the buildup curve matching for well 30/6-19 (DST3). The region not modelled correctly due to the layer thickening is shown by dotted lines.

of fluvial channels can often be observed and it seems the likely solution to the expanding permeability-thickness product kh away from the well. Other geological or geophysical data of this field [105] further constrain the interpretation. However, it is beyond the scope of this study.

The analysis of pressure data in the pseudo-radial flow period (iv) using the homogeneous reservoir model showed the pseudo-skin of 0.2. Thus, it is estimated that the presence of the lens reduces the skin by 4.8 ($= 5.0 - 0.2$).

Eq. 3.130 cannot be used in this example since the thickness of the reservoir matrix is considered not to be constant. Furthermore, any information to determine the drainage shape and its area is not available. However, in order to show the effect of the lens on the productivity, it was assumed the thickness was the constant of 22.3 ft (the thickness at the wellbore), the drainage area was 3.0×10^7 sq ft, the shape of the reservoir matrix was rectangular with $1 : 5$ aspect ratio, and the well was in the centre of the drainage area ($C_A = 2.3606$). In this case, the productivity is calculated to be 1.22 STB/D/psi

for the skin of 5.0, while 1.82 STB/D/psi for the skin of 0.2. Therefore, we can see the productivity increases by 49% due to the presence of the lens.

As Kuchuk et al. [59] mentioned in their paper, wellbore crossflow from the wellbore into the high permeability lens may occur during the period of the buildup testing. This may distort pressure and pressure derivative curves and cause an unreliable parameter estimation. However, since the volume of the lens is much smaller than that of the reservoir matrix adjacent to the lens region, it can be expected that the wellbore crossflow ends shortly after the well shut-in. Further confirmation may be obtained by downhole flowmeter measurements using the method described in their paper.

In this pressure buildup example, the computing time is around 8 seconds using an 150 MHz Pentium PC which is approximately 50 times faster than the fine-grid numerical simulation. Since the computation time is small enough, it is recommended that automatic pressure-pressure derivative type-curve matching algorithm should be implemented in the computer program developed here.

3.2.11 Multiple-Lens Case

Pressure Response of Multiple-Lens Reservoirs

The presence of several lenses makes pressure response very complicated. Such situations can occur due to the vertical and lateral stackings of sand bodies (as shown in Fig. 2.6) containing high permeability lenses. An example of the pressure response of the multiple-lens reservoirs is shown in Fig. 3.31. Fig. 3.32 and Table 3.3 show the schematic two-lens reservoir model and reservoir and fluid parameters used to generate the synthetic pressure response. The skin factors are zero in both the lens and the reservoir matrix regions. The wellbore storage effect is not considered here ($C_D = 0$).

In this case, it is possible to detect three radial flow periods from the pressure derivative curve. Each radial flow period is closely related to the radius of investigation. During the first radial flow period, the radius of investigation doesn't reach the external radius of the smaller lens. Thus, the pressure behaviour is completely the same as that for a 5-layer reservoir with crossflow (two high permeability layers plus three low permeability layers).

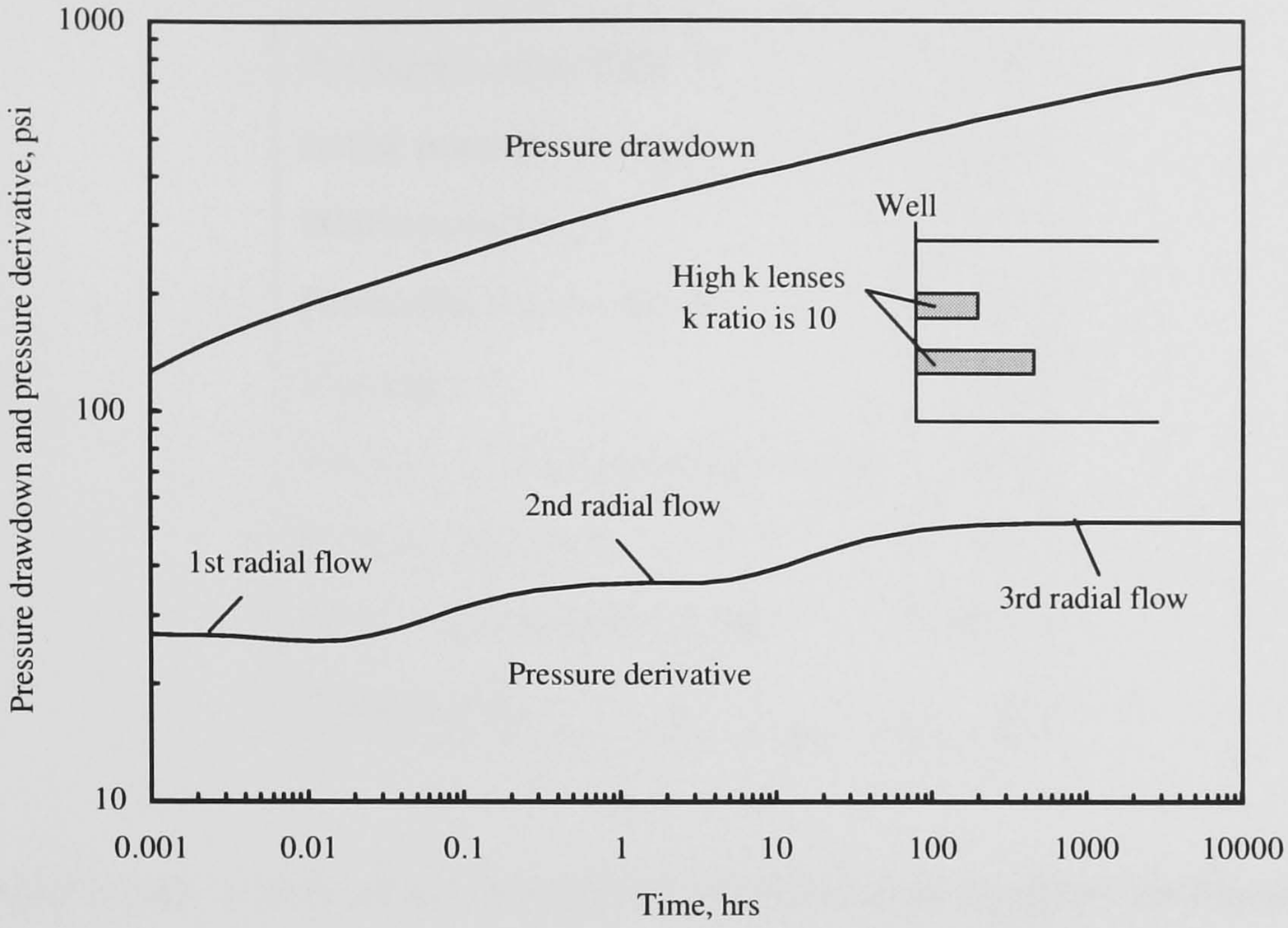


Figure 3.31: Pressure behaviour of a two-lens reservoir.

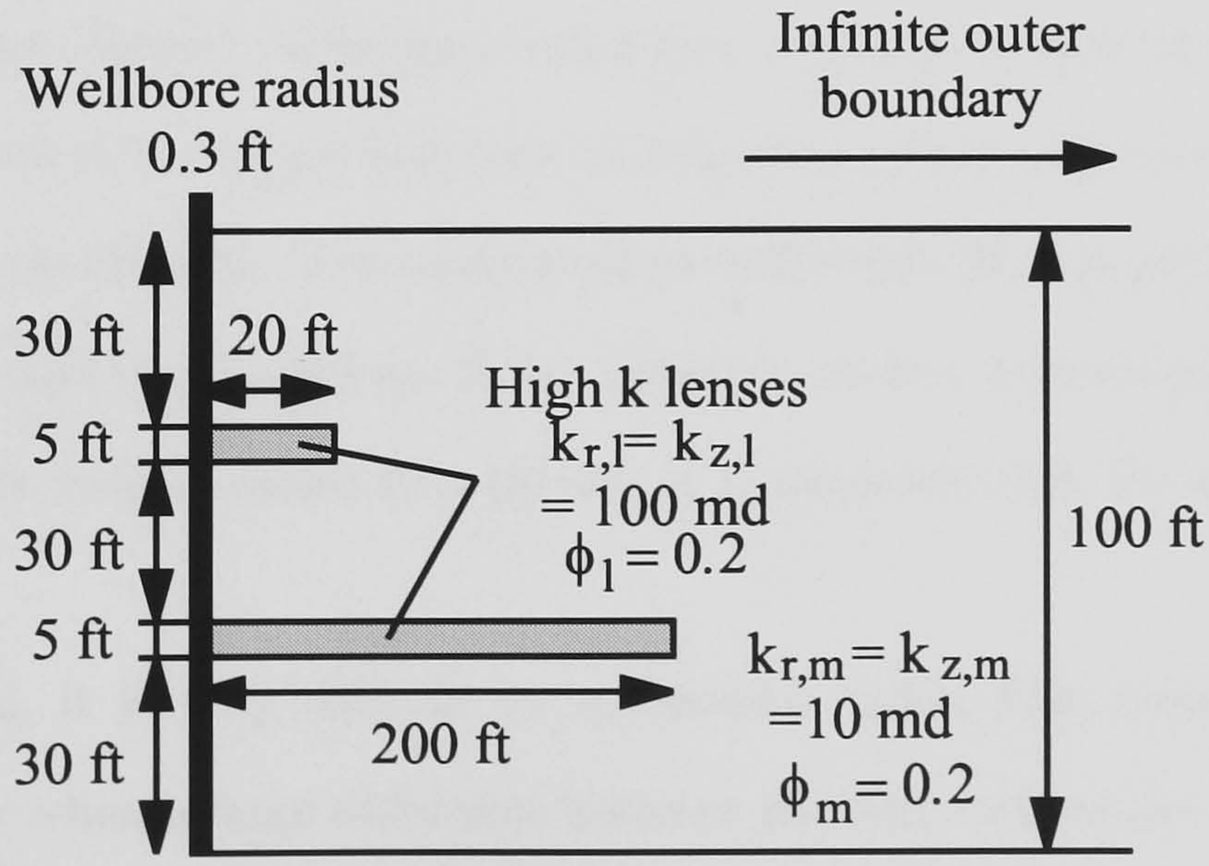


Figure 3.32: Two-lens reservoir model.

Table 3.3: Reservoir and fluid data for the two-lens case.

Production rate, STB/D	500
Initial reservoir pressure, psi	3,000
Wellbore radius, ft	0.3
Formation volume factor	1.5
Viscosity, cp	1.0
Wellbore storage constant, bbl/psi	0.0
Porosity, fraction	0.2
Total compressibility, 1/psi	1E-5
Damage skin	0

If the pressure data of this period is analysed, the thickness weighted arithmetic averaged permeability of five regions can be obtained. The second radial flow corresponds to the behaviour of a 3-layer reservoir with crossflow (one high permeability layer plus two low permeability layers). This occurs since the radius of investigation is between the external radius of the small lens and that of the large lens. The pressure analysis of this period shows the averaged permeability of three layers which consist of the large radius lens and the two matrix "layers". The final radial flow period is similar to the pseudo-radial flow period observed to the single lens case. It behaves as if the reservoir were an homogeneous system with pseudo-skin. The analysis of pressure data in this period yields the reservoir matrix permeability. Reversely, if the reservoir matrix permeability is obtained by the analysis of the pseudo-radial flow period, it is expected that the lenses are not laterally extensive.

In general, it is very difficult to see several radial flow periods since they can be observed only when a large difference between the lens radii exists. Thus, it is impossible to analyse the data by simple plots, such as semi-log pressure versus time plots. For that case, the type-curve matching of pressure and pressure derivative is necessary to derive information from such pressure data. Hence, the implementation of the automatic type-curve matching in the computer program [3] is highly recommended. However, the first

radial flow may be observable at very early times if wellbore storage effect is negligible. Also, the final pseudo-radial flow period may be observed if the production period is long enough. Such data can be used to derive the averaged permeability or the reservoir matrix permeability.

Equivalent Single-Lens Approximation

Mesmari [79] mentioned that the late-time response of the multiple-lens reservoir is equivalent to the single-lens reservoir where the lens radius is the arithmetic average of the multiple-lens radii when the lenses have the same rock properties (e.g., permeabilities, porosities, rock compressibilities). However, the effect of the thickness of each lens was not considered in his method. This is not adequate since this leads to a conclusion that a very thin lens affects the pseudo-skin to the same degree as a very thick lens. Here, the following average radii are considered for the single-lens approximation of the multiple-lens reservoirs, i.e.,

- Arithmetic averaged lens radius (Ref. [79]),

$$\hat{r}_l = \frac{\sum_{i=1}^{n} r_{l,i}}{n}, \tag{3.131}$$

- Thickness weighted arithmetic averaged lens radius,

$$\hat{r}_l = \frac{\sum_{i=1}^{n} h_i r_{l,i}}{\sum_{i=1}^{n} h_i}, \tag{3.132}$$

- Volumetric averaged lens radius,

$$\hat{r}_l = \sqrt{\frac{\sum_{i=1}^{n} h_i {r_{l,i}}^2}{\sum_{i=1}^{n} h_i}}, \tag{3.133}$$

where n is the number of the lenses embedded in the reservoir matrix.

The single lens was fixed in the centre of the reservoir since the reservoir top or bottom boundary affects the pressure response as shown in Figs. 3.16 and 3.17 (although the effect is small).

Figs. 3.34 and 3.35 show the pressure responses of two-lens reservoirs where the thicknesses of lenses are different as shown in Fig. 3.33. The responses of equivalent

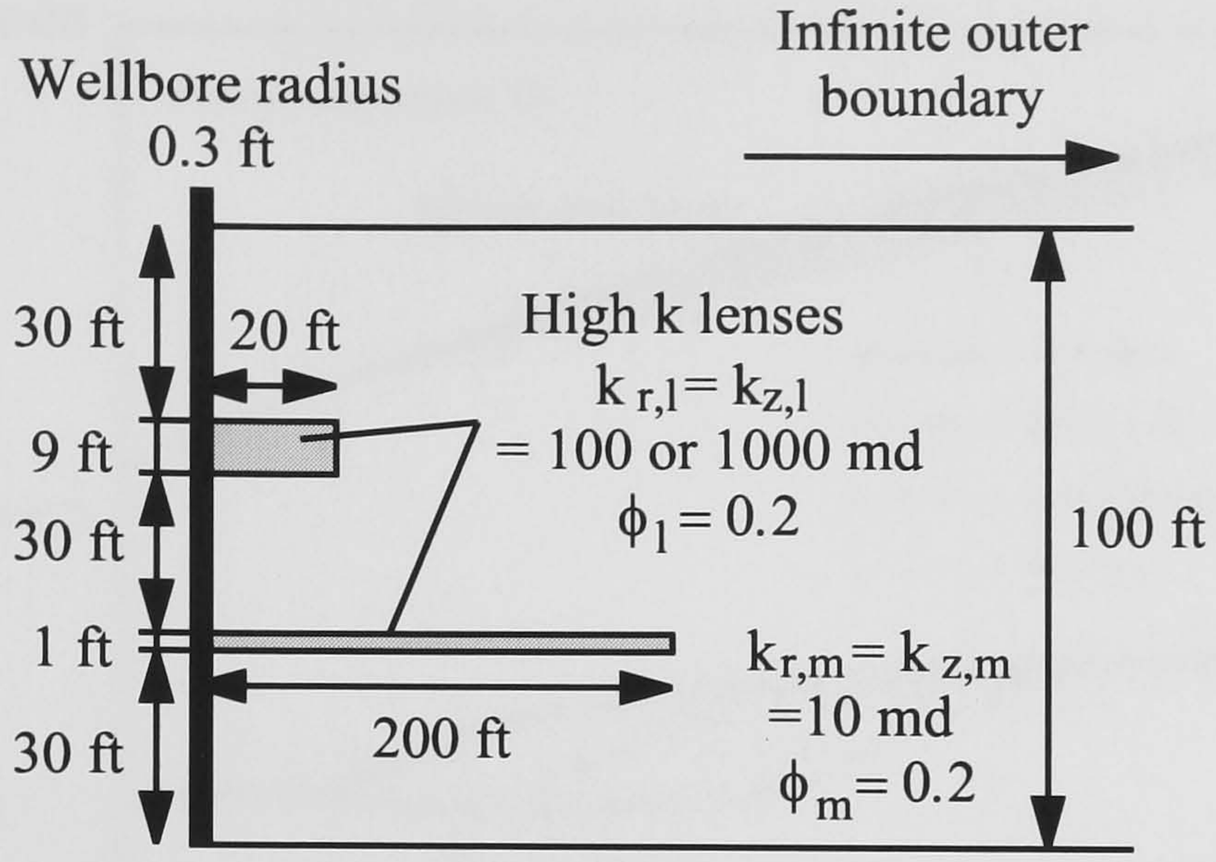


Figure 3.33: Two-lens reservoir model in which the lens thicknesses are different.

single-lens reservoirs where the equivalent single-lens radii are calculated by Eqs. 3.131 to 3.133 are also plotted in the figures. The permeability ratios between the lens and the reservoir matrix are 10 in Fig. 3.34 and 100 in Fig. 3.35, respectively. The other fluid and reservoir parameters are the same as in Table 3.3.

It can be observed that the pressure responses of the single-lens cases are identical to the response of the two-lens case at very early times. This is because the fluid flow is essentially horizontal and the pressure disturbance has not reached the external boundary of the smaller lens (its radius is 20 ft) yet. At middle times, the pressure and the pressure derivative curves show the large diversity of responses. At late times, the pressure response of the two-lens reservoir cannot be approximated by the equivalent single-lens reservoir where the equivalent radius is calculated by the simple arithmetic average given by Eq. 3.131, especially the permeability contrast is high as shown in Fig. 3.35.

Table 3.4 shows the averaged radii and the pseudo-skin factors obtained by the analysis in the pseudo-radial flow period. From this table, it can be observed that the thickness weighted arithmetic averaged radius expressed as Eq. 3.132 gives a better approximation than the other two approximation methods.

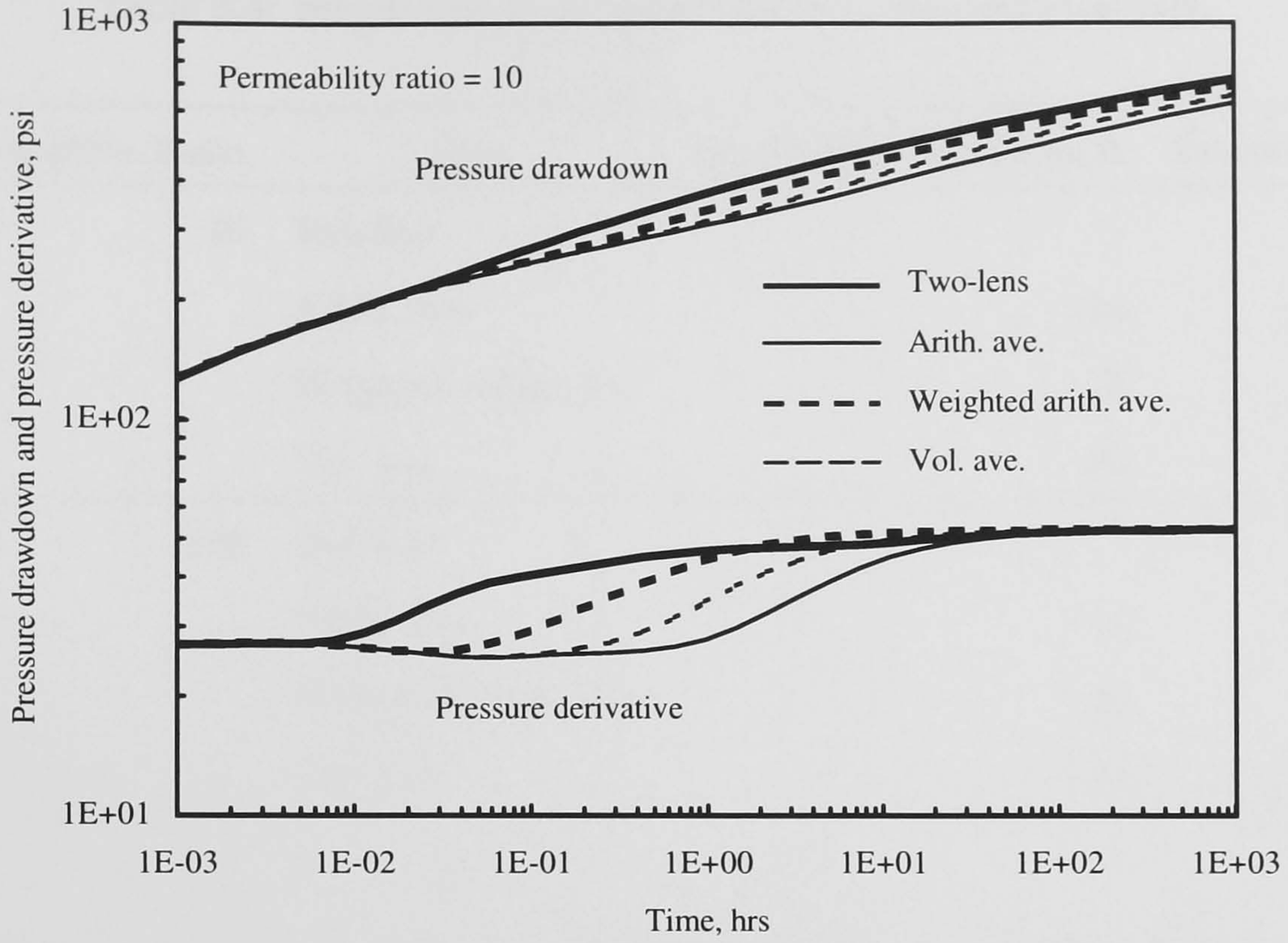


Figure 3.34: Single-lens approximations of a two-lens reservoir. Permeability ratio between the lens and the reservoir matrix is 10.

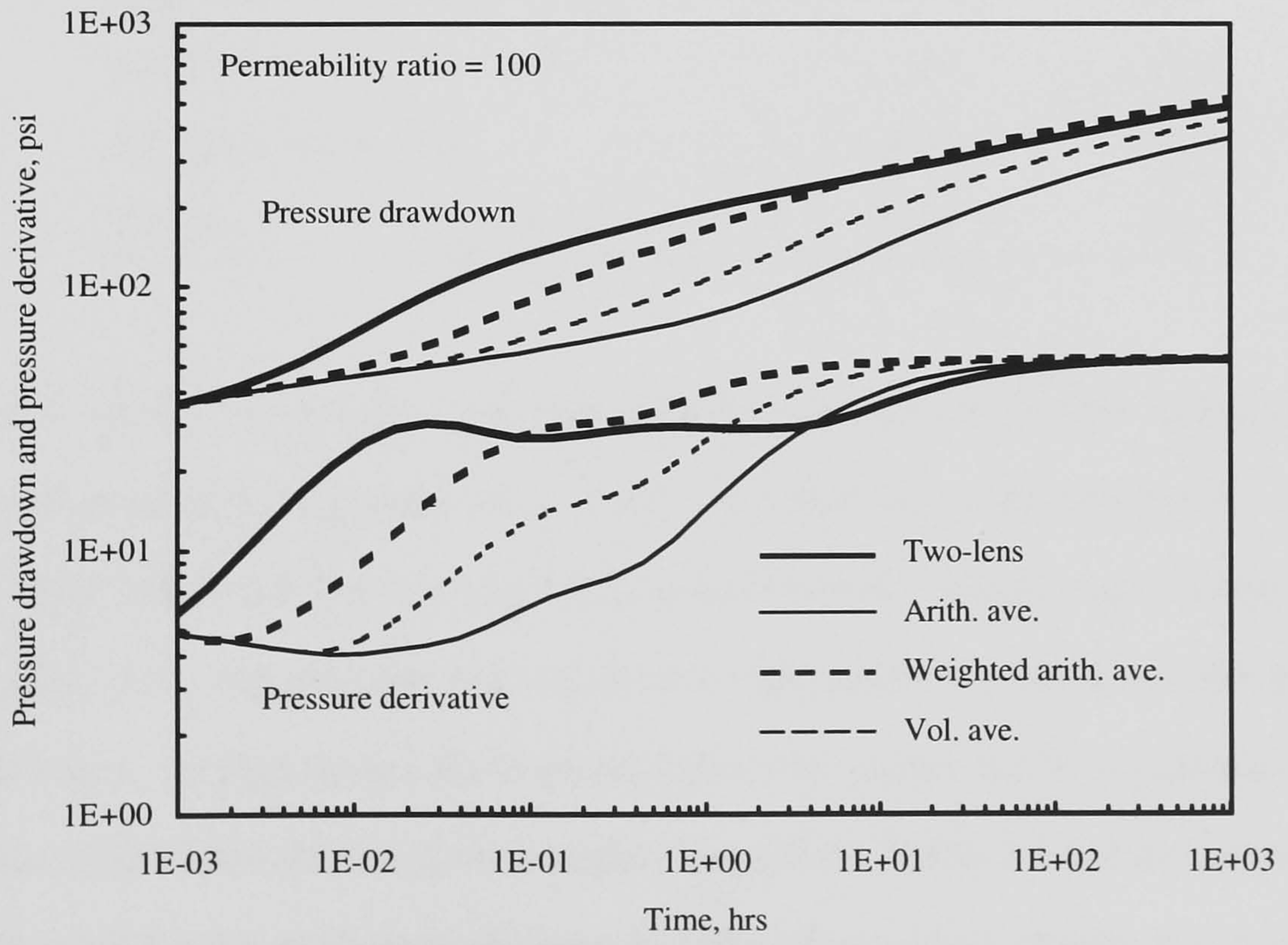


Figure 3.35: Single-lens approximations of a two-lens reservoir. Permeability ratio between the lens and the reservoir matrix is 100.

Table 3.4: Single-lens approximations of a two-lens reservoir.

Permeability Ratio	Case	Eq. Single-lens Radius, ft	Pseudo-skin
10	Two-lens	—	-1.9
	Arith. Ave.	110	-2.7
	Weighted Arith. Ave.	38	-2.1
	Vol. Ave.	66	-2.5
100	Two-lens	—	-4.1
	Arith. Ave.	110	-5.3
	Weighted Arith. Ave.	38	-3.9
	Vol. Ave.	66	-4.6

Table 3.5: Comparison of pseudo-skin factors between the four-lens reservoir and the equivalent single-lens reservoirs.

Case	Eq. Single-lens Radius, ft	Pseudo-skin
Four-lens	—	-2.9
Arith. Ave.	80	-3.4
Weighted Arith. Ave.	46.5	-2.9
Vol. Ave.	72.9	-3.2

Fig. 3.37 shows the pressure response of a four-lens reservoir case together with the responses of single-lens approximation cases. The four-lens reservoir model is shown in Fig. 3.36. The other reservoir and the fluid parameters are the same as in Table 3.3. As we can see in Fig. 3.37, the pressure and the pressure derivative responses become identical at very early times. At late times, the pressure derivative curves become identical. However, the pressure drawdown curves show a slight divergence. Table 3.5 shows the pseudo-skin factors obtained by the analysis in the pseudo-radial flow period. Again, it can be observed that the single-lens reservoir model with the thickness weighted arithmetic averaged radius of Eq. 3.132 gives a better approximation than the other two.

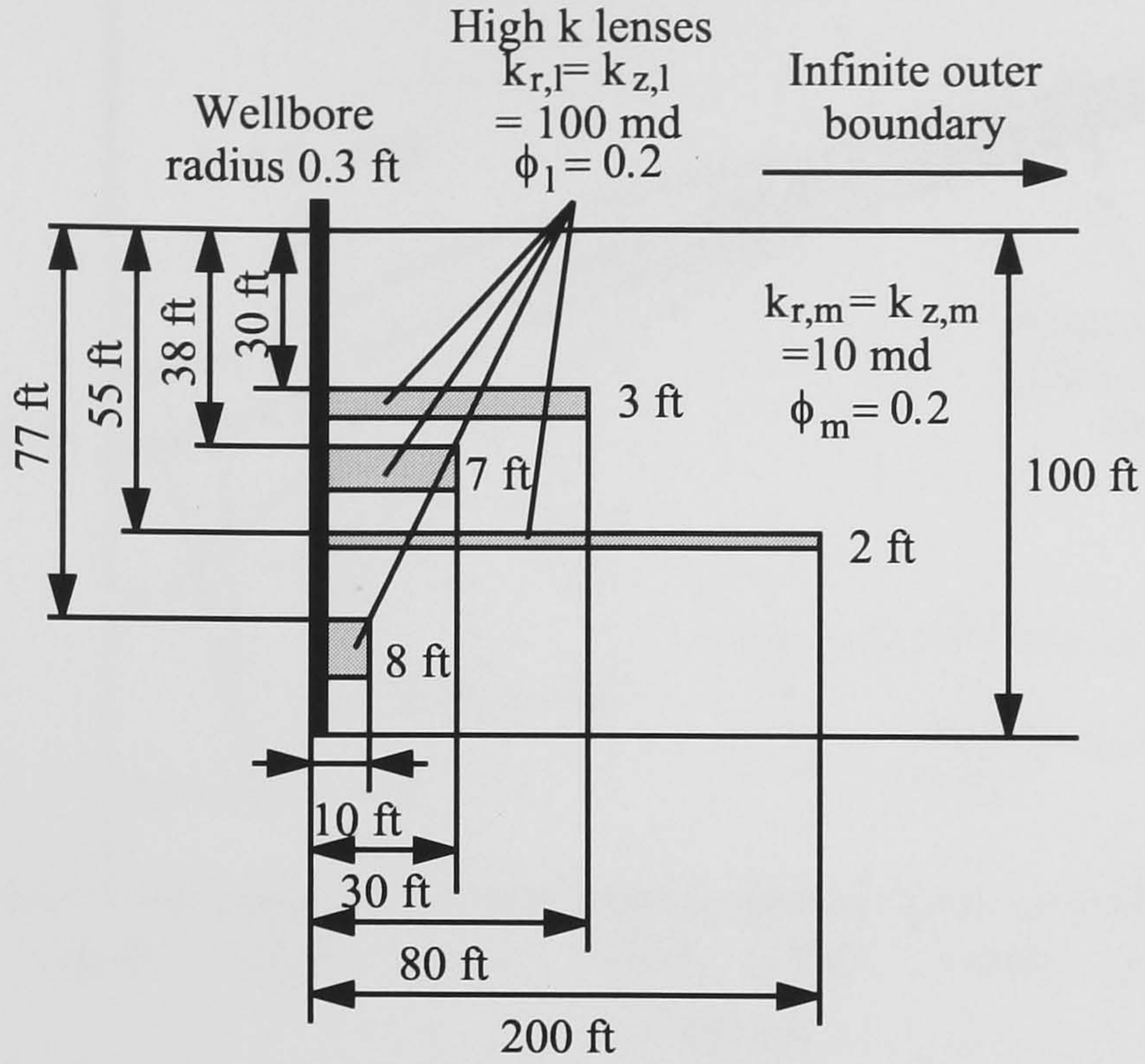


Figure 3.36: Four-lens reservoir model.

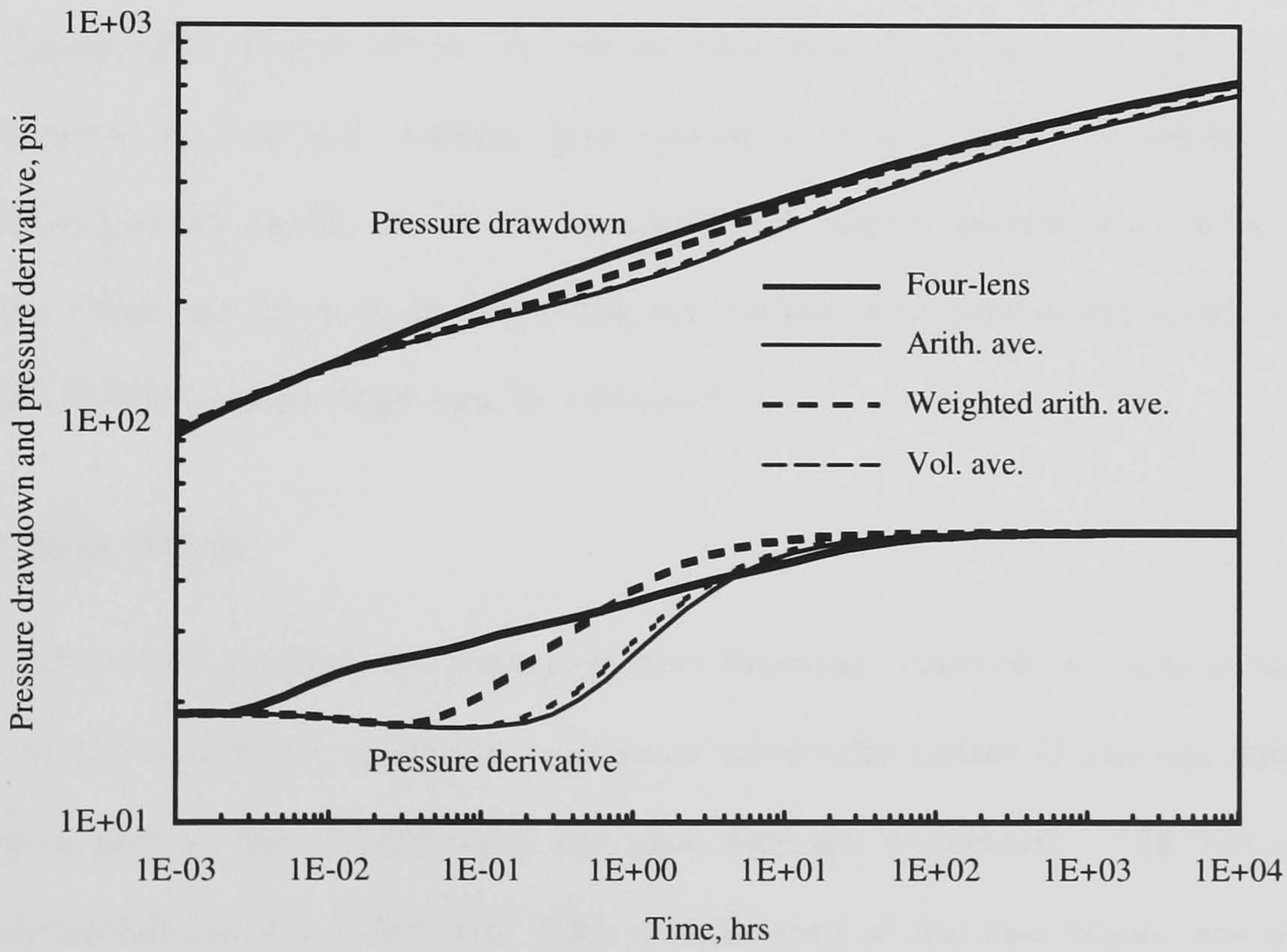


Figure 3.37: Pressure response of a four-lens reservoir and the single-lens approximations.

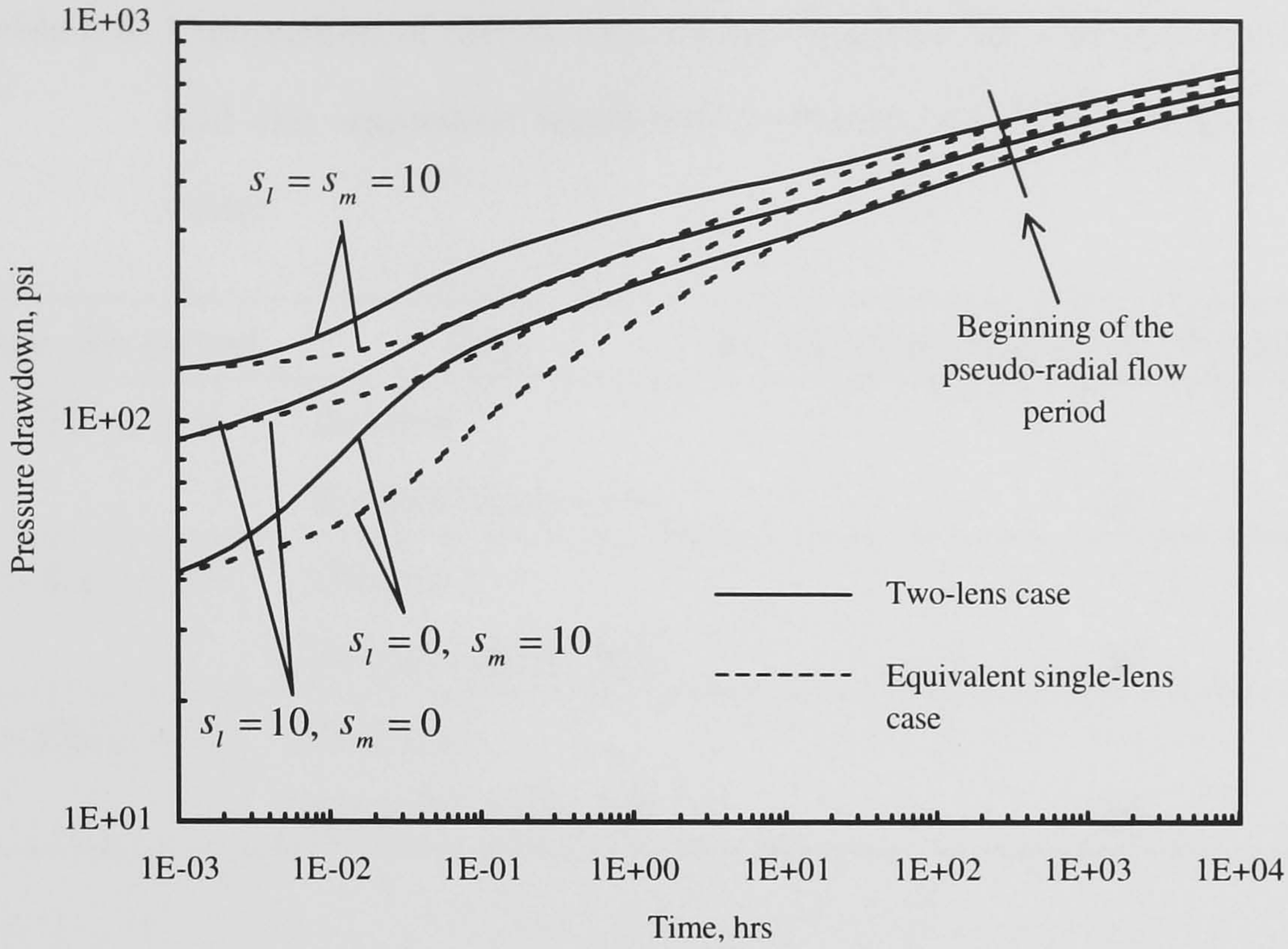


Figure 3.38: Single-lens approximations of a two-lens reservoir with damage skin.

From those facts shown above, it can be said that if the pseudo-radial flow period can be observed in a actual multiple-lens reservoir, it is possible to obtain an average lens radius expressed as Eq. 3.132 by matching the actual pseudo-skin with the model pseudo-skin. This can be done by adjusting the radius of an equivalent single lens until a good match of the pseudo-skins can be obtained.

Damage Skin Effect

Fig. 3.38 shows the pressure responses of the two-lens reservoir as shown in Fig. 3.33 and those of the equivalent single-lens reservoir where the radius of the equivalent single-lens is calculated by Eq. 3.132 when the skin damage is present. The horizontal and vertical permeabilities are 1,000 md. The skin factors of the two lenses are assumed to be identical (s_l), and it is directly used for the skin factor of the equivalent single lens. The skin factor of the reservoir matrix is constant (s_m). The pseudo-skin factors of the two-lens reservoir and the equivalent single-lens reservoir calculated in the same way as

Table 3.6: Comparison of pseudo-skin factors between the two-lens reservoir and the equivalent single-lens reservoir when the damage skin exists.

Damage skin factors	Case	Eq. Single-lens Radius, ft	Pseudo-skin
$s_l = 10, s_m = 0$	Two-lens	—	-3.4
	Weighted Arith. Ave.	38	-3.3
$s_l = 0, s_m = 10$	Two-lens	—	-3.9
	Weighted Arith. Ave.	38	-3.8
$s_l = 10, s_m = 10$	Two-lens	—	-2.8
	Weighted Arith. Ave.	38	-2.9

the previous examples are shown in Table 3.6.

It can be observed that even though the damage skin exists, the equivalent single-lens approximation of Eq. 3.132 gives a good estimate of the pseudo-skin if the same damage skin factors of the lenses and the reservoir matrix are applied for the equivalent single lens and the reservoir matrix regions. Thus, if the damage skin factors of the lenses and the reservoir matrix can be estimated independently of the analysis in the pseudo-radial flow period (e.g., from the very early time period), it is possible to estimate the average lens radius given by Eq. 3.132.

Other Uncertain Factors

However, the following points are always kept in mind when this approximation is applied to actual field examples.

1. In this section, it is assumed that the shape of the lenses are circular, and the well penetrates the lenses at the centre of them. If such conditions are not satisfied, it may affect the pseudo-skin and the estimation of the average lens radius. However, as shown in Fig. 3.23, the pressure response of the elongated lens with aspect ratio up to 1:5 and an arbitrary well position within the lens can be approximated by that

of the circular lens with the well at the centre and the same equivalent lens radius as defined by Eq. 3.129 since the pseudo-skin factors become almost identical. Hence, the single-lens approximation examined in this section is applicable to the multiple lenses with the aspect ratio up to 1:5.

2. The effect of the lenses which exist near the wellbore, but are not penetrated by the well has been ignored. The presence of the lenses near the well is likely to occur, especially in multiple lens cases. Although the effect diminishes as the distance of the lens from the well increases, the effect can be still recognised as shown in Fig. 3.26. This affects the pseudo-skin and the estimation of the average lens radius. (The effect of the lenses which are not penetrated by the wellbore may be examined by numerical simulation. The reservoir including high permeability lenses can be generated by geostatistical techniques, such as boolean algorithm [28].)

Thus, it is safe to say that the single-lens approximation method given by Eq. 3.132 gives an order of the average lens radius. The reliability of the estimation may be confirmed by geological information, such as depositional environments, outcrops, etc.

CHAPTER 4

HORIZONTAL WELL IN DISCONTINUOUS PERMEABILITY RESERVOIRS

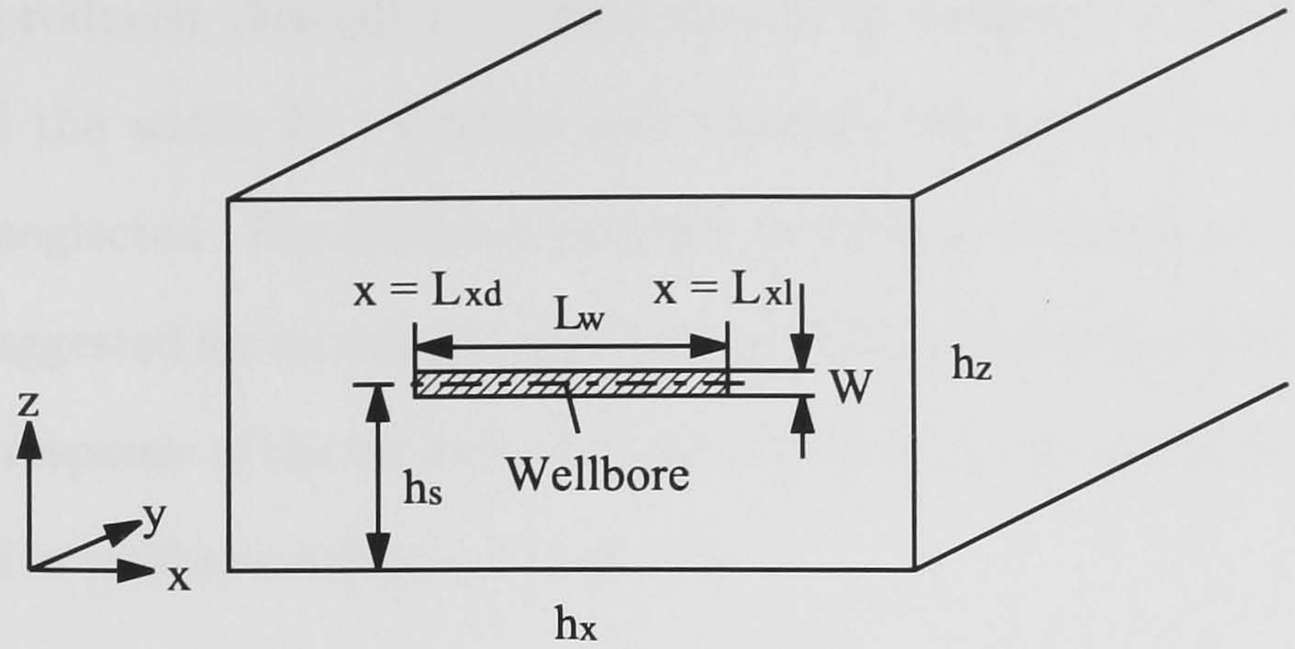


Figure 4.1: Strip wellbore model.

In this chapter, pressure response and fractional flow rate transition of a horizontal well penetrating two zones in a linearly elongated reservoir are discussed. As in the previous chapter, the contrast of the permeabilities in the different regions complicates the response and makes it difficult to analyse the response using conventional methods. First, pressure response of a horizontal well in a homogeneous reservoir is briefly reviewed for the discussion of heterogeneous reservoir cases. The strip wellbore model developed by Goode and Thambynayagam [42] is referred for this purpose. Then, the heterogeneous linear reservoir model developed in this study is explained. Finally, the pressure and the fractional flow rate transitions for the two-zone linear reservoirs are examined by sensitivity study to various reservoir parameters.

4.1 Horizontal Well in Homogeneous Reservoirs

4.1.1 Homogeneous Reservoir Model With a Strip Wellbore

Consider a homogeneous reservoir containing a horizontal well at the distance h_s from the bottom of the reservoir to the well centre (Fig. 4.1). The reservoir is bounded in the x and z directions, and the width and the thickness of the reservoir are h_x and h_z, respectively. However, the reservoir is infinitely elongated in the y direction. The anisotropic permeabilities in the x, y, and z directions are denoted by k_x, k_y, and k_z, respectively. A slightly compressible fluid (compressibility c_t is constant) of constant

viscosity μ is produced through a uniform-flux strip wellbore with the length L_w (= $L_{xl} - L_{xd}$) and the width W. Gravity and capillary effects and the effect of wellbore hydraulics are neglected. The reservoir pressure is initially constant at p_i.

Prats [94] suggested for an infinite-conductivity fully-penetrating vertical fracture case, the production response of the fracture with the width W is equivalent to that of a vertical well with effective wellbore radius $r_w{}'$ given by

$$r_w{}' = \frac{W}{4}. \tag{4.1}$$

Goode and Thambynayagam [42] applied this insight for their strip wellbore model, and obtained a similar equation for a horizontal well by considering the effect of anisotropy. They suggested that the effective wellbore radius or the width of the strip wellbore can be obtained by multiplying the fourth root of the permeability ratio between the y and z directions, i.e.,

$$r_w{}' = \frac{W}{4} = r_w \sqrt[4]{\frac{k_y}{k_z}}. \tag{4.2}$$

where r_w is the actual wellbore radius of a horizontal well.

An uniform-flux condition along the wellbore is assumed in the strip wellbore model due to the difficulty of introducing an infinite-conductivity condition. However, this is correct only at very early times. Goode and Thambynayagam [42] used the equivalent pressure point method to solve this difficulty.

In the equivalent pressure point method, the point on the wellbore strip which gives the same pressure drawdown as that in the infinite-conductivity solution has to be found (Fig. 4.2). In Fig. 4.2, Δp_{uf} is a pressure drawdown for the uniform-flux case, and Δp_{up} is a pressure drawdown for the uniform-pressure case, and h/W is a normalised distance from the edge of the wellbore strip. The equivalent pressure point is chosen to yield a final pseudo-steady pressure drop correctly (In general, the location of the equivalent point will vary with time [2].). However, it is also exact at very early times since the true solution has the uniform flux and the uniform pressure distributions. As mentioned in Section 2.2.2, Goode and Thambynayagam [42] used the 86.6% point for their strip wellbore model of a

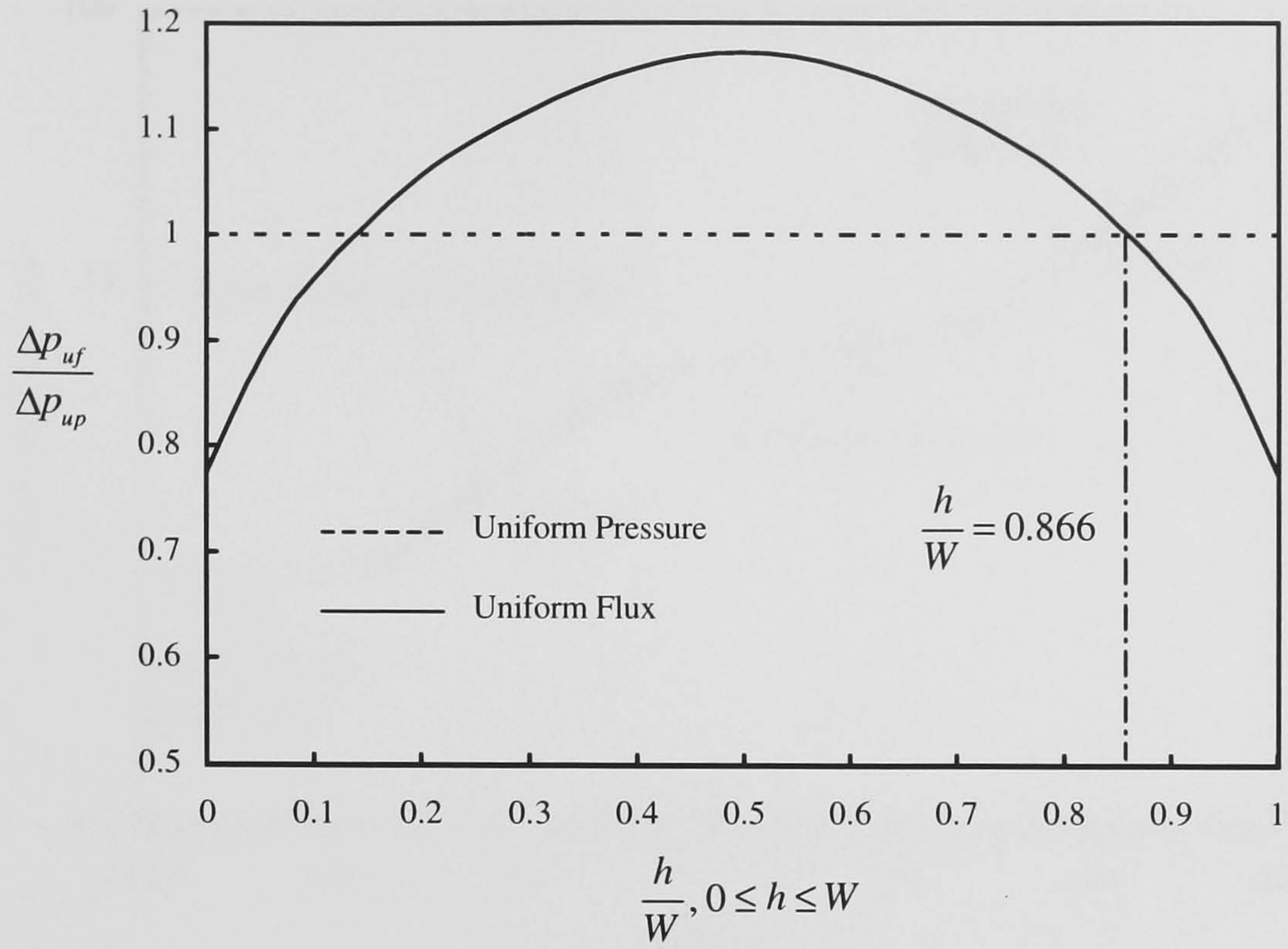


Figure 4.2: Equivalent pressure point [42].

horizontal well. It can be expressed mathematically as

$$z = h_s + 0.366W, \tag{4.3}$$

where z is the position of the equivalent pressure point on the strip wellbore measured from the bottom of the reservoir. A complete procedure to find a correct equivalent pressure point can be found in Ref. [100].

4.1.2 Flow Regimes for Homogeneous Reservoirs

Goode and Thambynayagam [42] presented an analytical solution for the homogeneous and anisotropic reservoirs described in Section 4.1.1 using the successive Fourier and Laplace transforms. They also presented simplified solutions for typical flow regimes observed in well test analysis of horizontal wells. Figs. 4.3 and 4.4 show a schematic pressure derivative response and streamlines for each flow period.

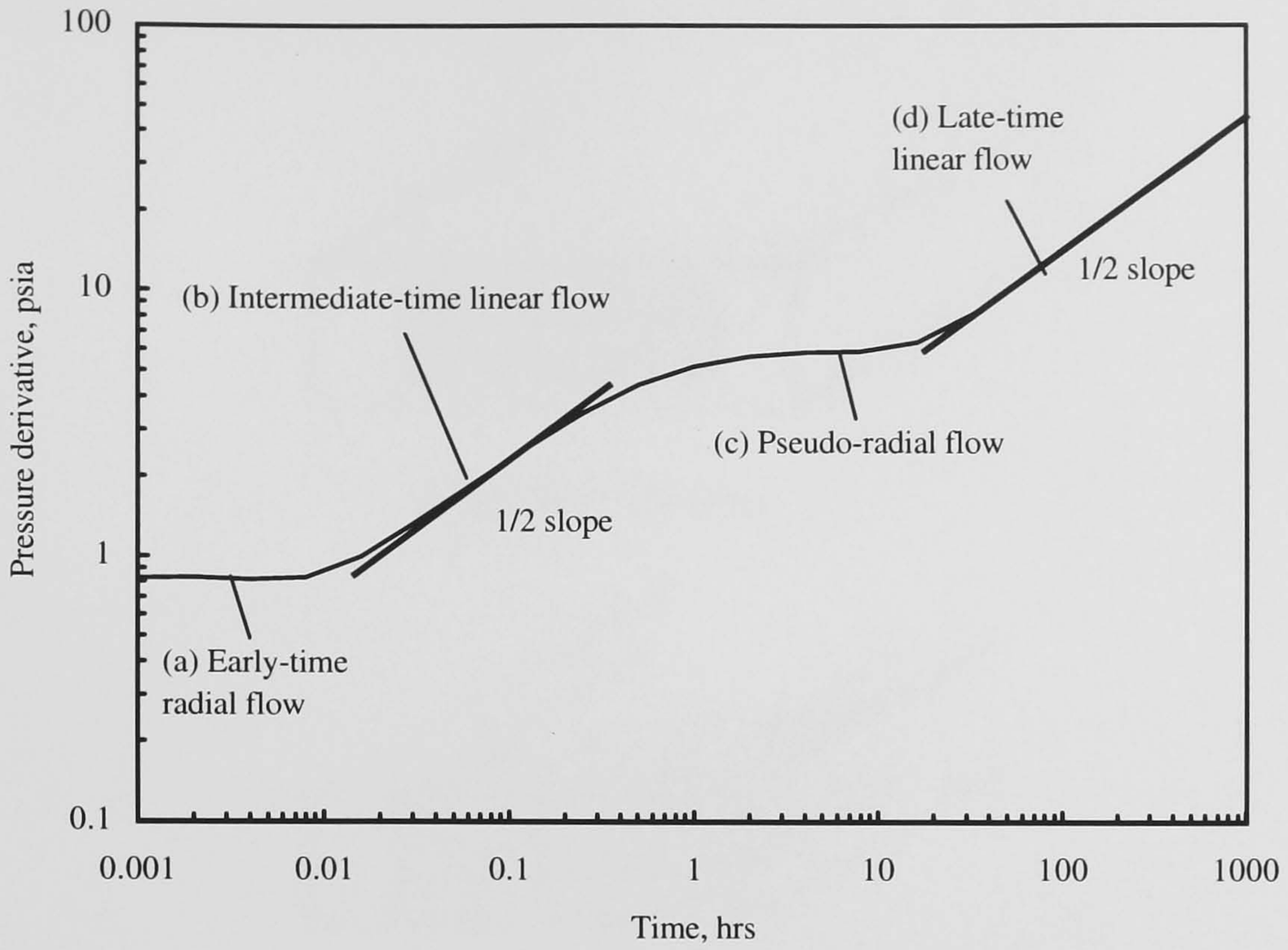


Figure 4.3: Pressure derivative curve for a typical homogeneous case.

Early-time Radial Flow Regime

The first radial flow regime continues until the effect of the nearest top or bottom boundary is felt, or the pressure response is affected by the well tips. This flow regime is similar to the radial flow period of a fully penetrating vertical well. If the wellbore storage effect is present, this flow regime may not be observed. An approximated solution for the early-time period is given by

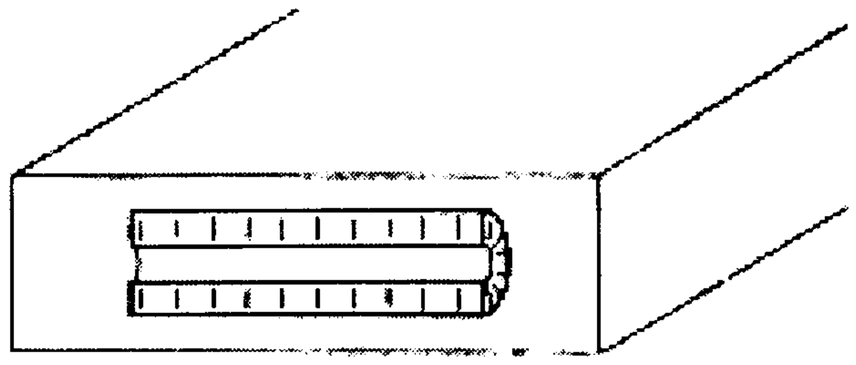
$$p_i - p_{wf} = \frac{162.6q\mu B}{\sqrt{k_y k_z} L_w}\left[\log\left(\frac{\sqrt{k_y k_z}\,t}{\phi\mu c_t {r_w}^2}\right) - 3.227 + 0.8686 S_m\right], \tag{4.4}$$

in field units.

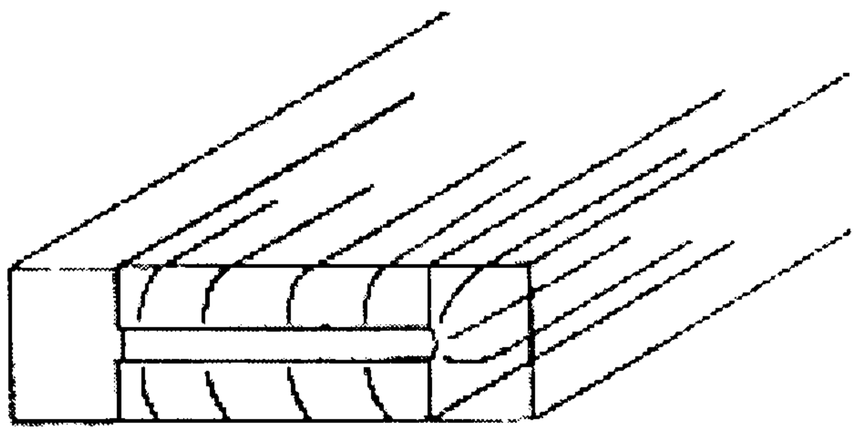
The derivative of Eq. 4.4 becomes a straight line on a logarithmic time plot as shown in Fig. 4.3, and its value m_{1r} is given by

$$m_{1r} = \frac{d(p_i - p_{wf})}{d\ln t} = -0.4343\frac{dp_{wf}}{d\log t} = \frac{70.62q\mu B}{\sqrt{k_y k_z} L_w}. \tag{4.5}$$

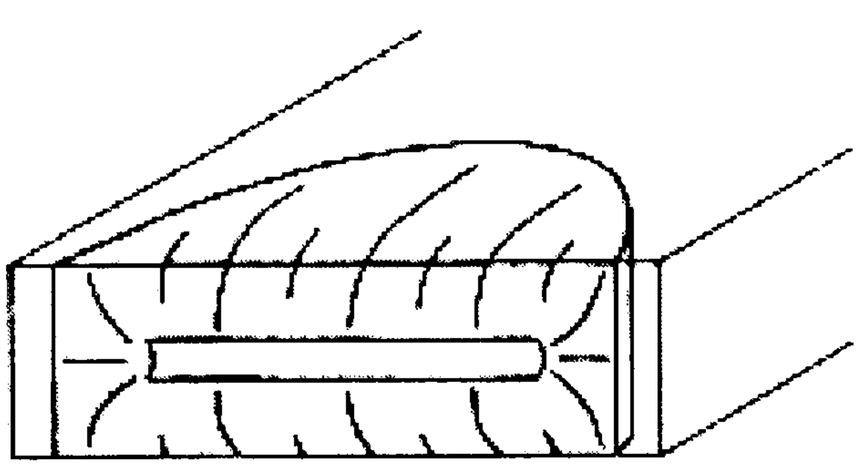
The wellbore length L_w applied for Eq. 4.5 is not easy to know since a design length and the length actually producing fluids are often different. However, if it is known (e.g.,



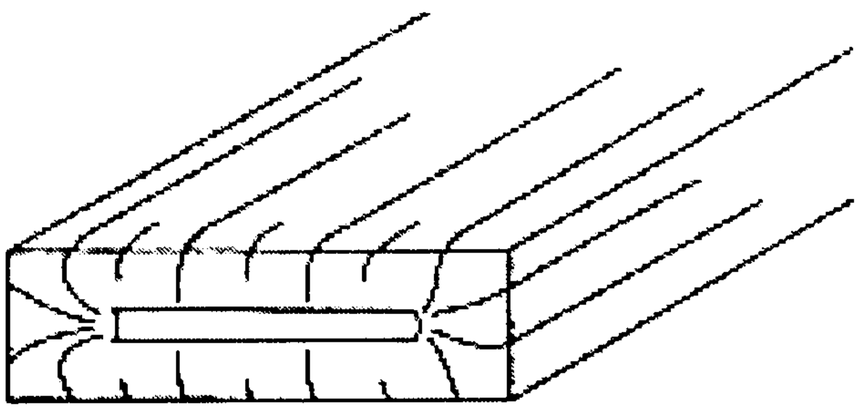
(a) Early-time radial flow



(b) Intermediate-time linear flow



(c) Pseudo-radial flow



(d) Late-time linear flow

Figure 4.4: Possible flow regimes for a linearly elongated homogeneous reservoir [42].

from a production logging survey), an average permeability $\sqrt{k_y k_z}$ can be calculated from Eq. 4.5 as

$$\sqrt{k_y k_z} = \frac{70.62 q \mu B}{m_{1r} L_w}. \tag{4.6}$$

From Eq. 4.4, the damage skin S_m is given by

$$S_m = 1.151 \left[\frac{p_i - p_{wf,t=1}}{m_{1r}} - \log \left(\frac{\sqrt{k_y k_z}}{\phi \mu c_t {r_w}^2} \right) + 3.227 \right], \tag{4.7}$$

where $p_{wf,t=1}$ is the pressure at $t = 1$ hour.

Goode and Thambynayagam [42] also provided the time at which the early-time radial flow ends due to the top or bottom boundary as

$$t_{erf1} = \frac{190.0 {r_w}^{-0.095} \phi \mu c_t}{k_z} \min\{{h_s}^{2.095}, (h_z - h_s)^{2.095}\}, \tag{4.8}$$

while Kuchuk et al. [62] presented it using a different analytical technique as

$$t_{erf1} = \frac{\phi \mu c_t}{0.0002637 \pi k_z} \min\{{h_s}^2, (h_z - h_s)^2\}. \tag{4.9}$$

From these equations, k_z is roughly estimated assuming the other parameters are known. Some authors presented similar expressions [85, 88], however, all of these should be used only qualitatively.

Intermediate-time Linear Flow Regime

When the horizontal well is long enough compared to the reservoir thickness, a linear flow regime may develop after the pressure disturbance reaches the top and bottom boundaries. An approximated solution for this flow regime is expressed as

$$p_i - p_{wf} = \frac{8.128 q B}{L_w h_z} \sqrt{\frac{\mu t}{k_y \phi c_t}} + \frac{141.2 q \mu B_o}{L_w \sqrt{k_y k_z}} (S_z + S_m), \tag{4.10}$$

where S_z is a pseudo-skin due to the partial penetration in the vertical direction, given by

$$S_z = \frac{0.07958 h_z}{{r_w}'} [\Psi(\eta_1) + \Psi(\eta_2) - \Psi(\eta_3) - \Psi(\eta_4)], \tag{4.11}$$

where

$$\eta_1 = \frac{0.52\pi {r_w}'}{h_z}, \tag{4.12}$$

$$\eta_2 = \frac{\pi}{h_z}(2h_s + 3.48{r_w}'), \tag{4.13}$$

$$\eta_3 = -\frac{3.48\pi {r_w}'}{h_z}, \tag{4.14}$$

$$\eta_4 = \frac{\pi}{h_z}(2h_s - 0.52{r_w}'). \tag{4.15}$$

The effective wellbore radius ${r_w}'$ is defined by Eq. 4.2 and $\Psi(\eta)$ is the Spence function expressed as

$$\Psi(\eta) = \sum_{m=1}^{\infty} \frac{\sin(m\eta)}{m^2} = -\int_0^{\eta} \ln\left[2\sin\left(\frac{u}{2}\right)\right] du. \tag{4.16}$$

The pressure derivative curve of Eq. 4.10 on a log-log plot becomes a half-slope line as shown in Fig. 4.3. On a pressure versus square-root time plot, the derivative curve becomes a straight line with its slope

$$m_{1l} = -\frac{dp_{wf}}{d\sqrt{t}} = \frac{8.128qB}{L_w h_z}\sqrt{\frac{\mu}{k_y \phi c_t}}. \tag{4.17}$$

From this flow regime, k_y can be calculated if L_w and h_z are known. The skin S_m can be calculated from

$$S_m = \frac{0.058}{h_z}\sqrt{\frac{k_z}{\phi\mu c_t}}\frac{p_i - p_{wf,t=0}}{m_{1l}} - S_z, \tag{4.18}$$

where $p_{wf,t=0}$ is the pressure at $t = 0$ obtained by extrapolating the straight line to this time. This flow regime may not appear when a gas cap or a bottom aquifer exists.

This flow regime ends at

$$t_{elf1} = \frac{20.8\phi\mu c_t {L_w}^2}{k_x}. \tag{4.19}$$

Pseudo-radial Flow Regime

This flow regime develops on the $x - y$ plane when the well length is sufficiently short as compared to the reservoir size. An approximated solution for this flow regime is given by

$$p_i - p_{wf} = \frac{162.6q\mu B}{\sqrt{k_x k_y} h_z}\left[\log\left(\frac{k_x t}{\phi\mu c_t {L_w}^2}\right) - 2.023\right] + \frac{141.2q\mu B_o}{L_w\sqrt{k_y k_z}}(S_z + S_m). \tag{4.20}$$

The derivative curve of Eq. 4.20 flattens on a logarithmic time plot as shown in Fig. 4.3, and its value m_{2r} is given by

$$m_{2r} = -0.4343\frac{dp_{wf}}{d\ln t} = \frac{70.62q\mu B}{\sqrt{k_x k_y}h_z}. \tag{4.21}$$

This equation provides an estimate of the average permeability $\sqrt{k_x k_y}$ if h_z is known a priori. The skin S_m is given by

$$S_m = \frac{1.151L_w}{h_z}\sqrt{\frac{k_z}{k_x}}\left[\frac{p_i - p_{wf,t=1}}{m_{2r}} - \log\left(\frac{k_x}{\phi\mu c_t {L_w}^2}\right) + 2.023\right] - S_z. \tag{4.22}$$

Goode and Thambynayagam [42] reported that the pseudo-radial flow regime would begin at

$$t_{brf2} = \frac{1230.0\phi\mu c_t {L_w}^2}{k_x}, \tag{4.23}$$

and end at

$$t_{erf2} = \frac{297.0(L_{xl} + L_{xd})^{2.095}\phi\mu c_t {L_w}^{-0.095}}{k_x}, \tag{4.24}$$

where L_{xl} and L_{xd} are the lateral distance from the left reservoir boundary to the right end and the left end of the wellbore strip, respectively (Fig. 4.1). Kuchuk et al. [62] reported that it would begin at

$$t_{brf2} = \frac{1.896 \times 10^4\phi\mu c_t {L_w}^2}{k_x}. \tag{4.25}$$

Late-time Linear Flow Regime

The second linear flow regime will develop after the pressure disturbance reaches the lateral boundaries of the reservoir. An approximated solution for this flow regime is expressed as

$$p_i - p_{wf} = \frac{8.128qB}{h_x h_z}\sqrt{\frac{\mu t}{k_y\phi c_t}} + \frac{141.2q\mu B}{L_w\sqrt{k_y k_z}}(S_x + S_z + S_m), \tag{4.26}$$

where S_x is a pseudo-skin due to the partial penetration in the horizontal direction, given by

$$S_x = \frac{0.6366{h_x}^2 L_w}{h_z\sqrt{\frac{k_y}{k_x}}}\sum_{n=1}^{\infty}\frac{{\Xi_n}^2}{n}, \tag{4.27}$$

where

$$\Xi_n = \frac{1}{nL_w}\left[\sin\left(\frac{n\pi L_{xl}}{h_x}\right) - \sin\left(\frac{n\pi L_{xd}}{h_x}\right)\right]. \tag{4.28}$$

A pressure drawdown versus square-root time plot shows a straight line with its slope

$$m_{2l} = -\frac{dp_{wf}}{d\sqrt{t}} = \frac{8.128qB}{h_x h_z}\sqrt{\frac{\mu}{k_y \phi c_t}}. \tag{4.29}$$

On a log-log pressure versus time plot, the pressure derivative curve of Eq. 4.26 becomes a half-slope line as shown in Fig. 4.3. If this flow regime can be observed and k_y is known a priori, it is possible to calculate the product $h_x h_z$. The skin S_m can be calculated from this flow regime as

$$S_m = \frac{0.058L_w}{h_x h_z}\sqrt{\frac{k_z}{\phi\mu c_t}}\left(\frac{p_i - p_{wf,t=0}}{m_{2l}}\right) - (S_x + S_z). \tag{4.30}$$

Other Flow Regimes

A hemi-radial flow regime may develop after the early-time radial flow regime if the horizontal well is very close to the top or bottom boundary of the reservoir. This behaviour is equivalent to that of a vertical well close to a fault. Thus, the pressure derivative for this flow regime m_{hr} becomes twice that of the early-time radial flow case (Eq. 4.5), i.e.,

$$m_{hr} = 2m_{1r} = \frac{141.2q\mu B}{\sqrt{k_y k_z}L_w}. \tag{4.31}$$

If the wellbore storage effect is present, and the early-time radial flow period is hidden by it, it is probable that the hemi-radial flow is interpreted as the early-time radial flow. In that case, the averaged permeability $\sqrt{k_y k_z}$ becomes half the true value [66].

When the well length is shorter than the reservoir thickness (although this is unusual for horizontal wells), a spherical flow regime may occur, and the well acts as a point source [66]. In this flow regime, the pressure derivative curve shows a straight line with the negative half slope on a log-log pressure versus time plot.

4.2 Description of the Heterogeneous Linear Reservoir Model

In the previous section, the published model of each flow period for a horizontal well in a homogeneous and anisotropic reservoir has been discussed. In this section, a heterogeneous

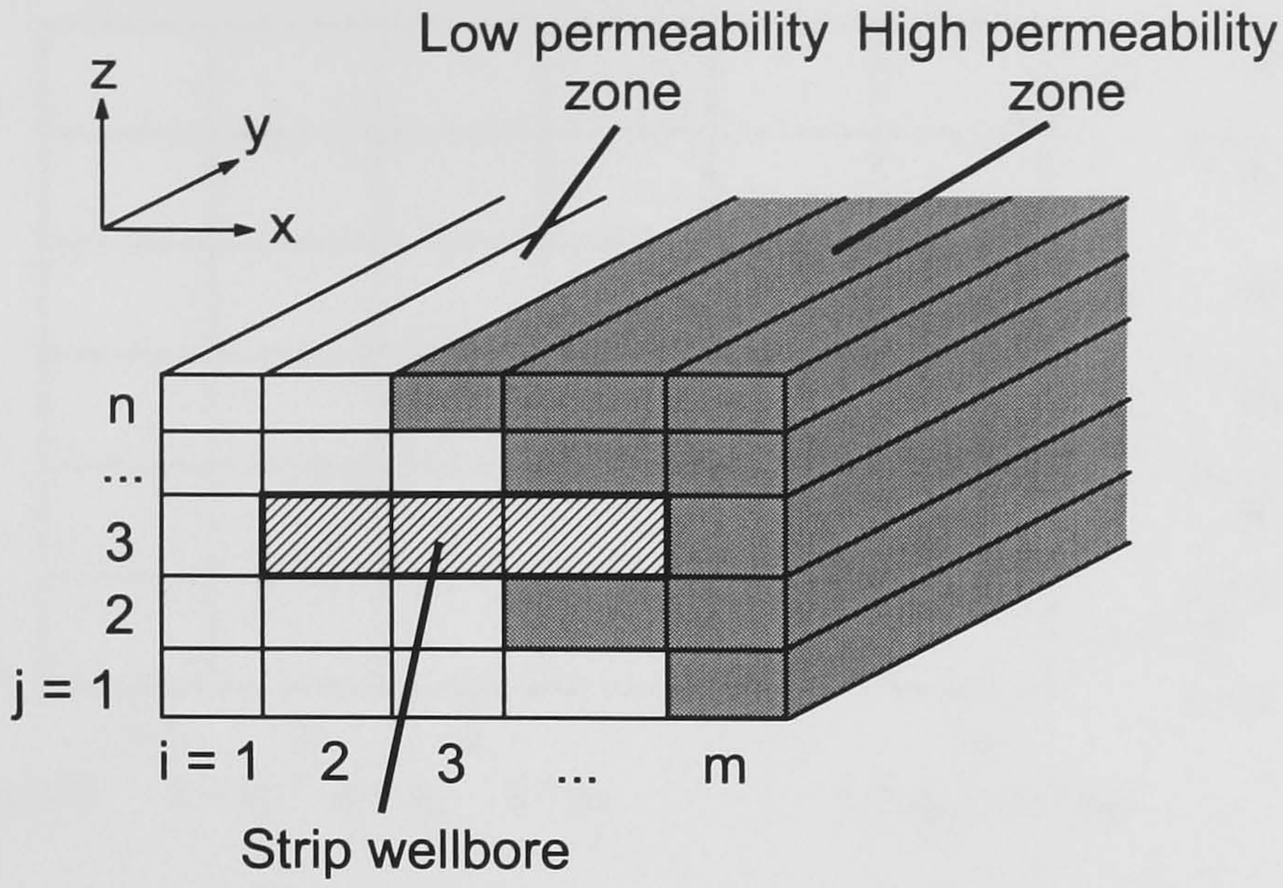


Figure 4.5: Heterogeneous reservoir model for horizontal wells.

and anisotropic reservoir model which can represent geological variation in the x and the z directions (See Fig. 4.5) is introduced.

4.2.1 Governing Equation

Reservoir models considered here may have several different permeability regions. Fig. 4.5 is an example of such multi-permeability linear reservoirs. In this case, the reservoir has two different (high and low) permeability regions. Both regions are partially penetrated by a horizontal well drilled along the x direction. The reservoir is bounded in the x and the z directions, however it is infinite in the y direction. Slightly compressible single phase fluid of constant viscosity is produced at a constant rate from the wellbore. Initial pressure is the same in each part of the reservoir. Capillary and gravity effects and the effect of the wellbore hydraulics are neglected. The reservoir is divided into $m \times n$ regions as shown in Fig. 4.5 according to the geometry of each permeability region. In the following discussion, a local coordinate system for the z direction as shown in Fig. 4.6 is used.

Under these assumptions, a diffusivity equation for each region can be obtained in Darcy units as

$$k_{x,i,j}\frac{\partial^2 p_{i,j}}{\partial x^2} + k_{y,i,j}\frac{\partial^2 p_{i,j}}{\partial y^2} + k_{z,i,j}\frac{\partial^2 p_{i,j}}{\partial z^2} = \phi_{i,j}\mu c_{t,i,j}\frac{\partial p_{i,j}}{\partial t}, \tag{4.32}$$

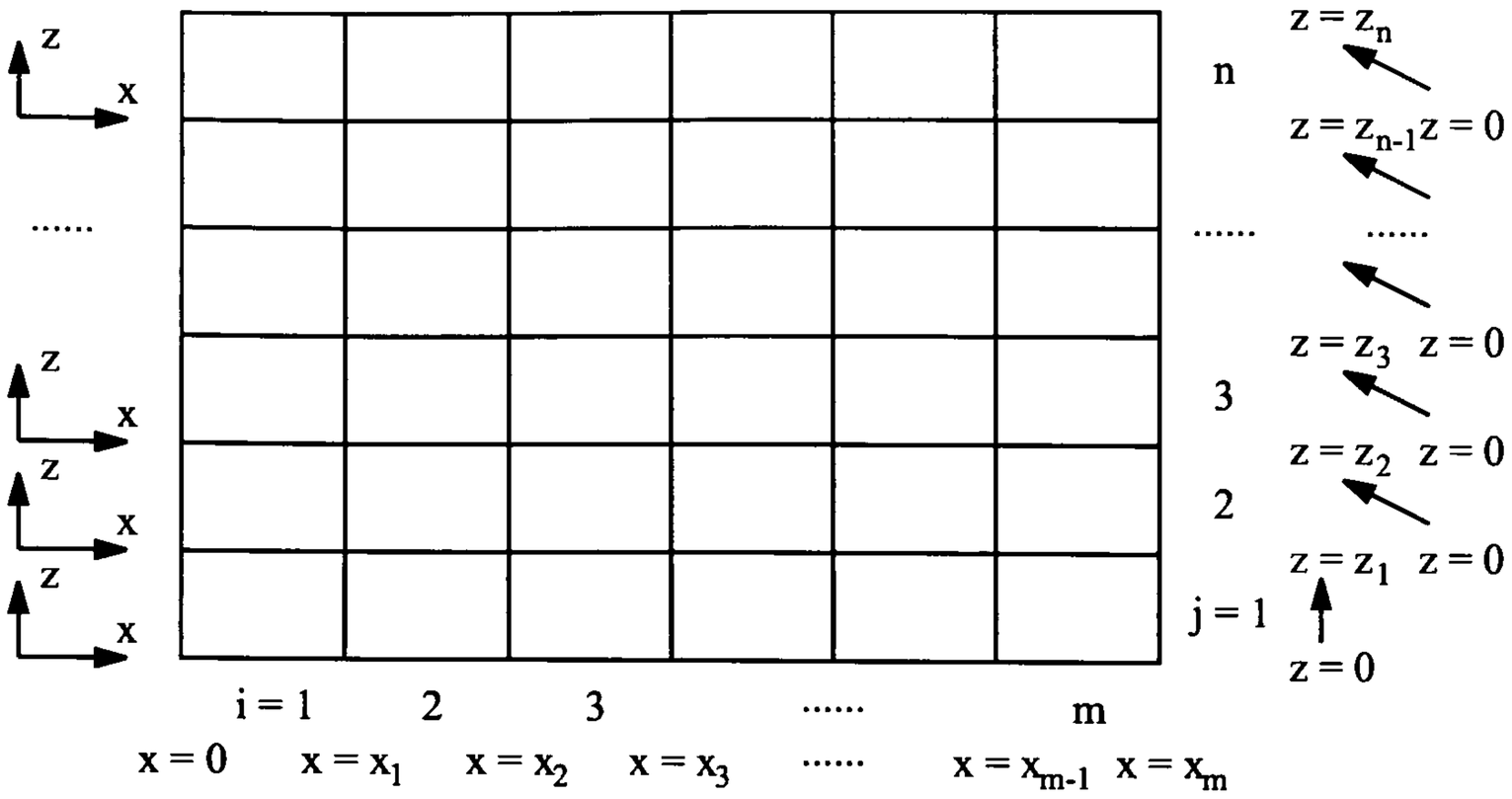


Figure 4.6: Local coordinate system for the z direction.

for the grid block (i, j) where $1 \leq i \leq m, 1 \leq j \leq n$.

The initial condition is given by

$$\lim_{t \to 0} p_{i,j} = p_{in} \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq n. \tag{4.33}$$

The no-flow boundary conditions for the top and the bottom layers become

$$\left. \frac{\partial p_{i,n}}{\partial z} \right|_{z=z_n} = 0, \tag{4.34}$$

$$\left. \frac{\partial p_{i,1}}{\partial z} \right|_{z=0} = 0, \tag{4.35}$$

for $1 \leq i \leq m$. If either the top or bottom boundary is a constant pressure type, Eq. 4.34 or Eq. 4.35 should be replaced by

$$p_{i,n}\big|_{z=z_n} = p_{in}, \tag{4.36}$$

or

$$p_{i,1}\big|_{z=0} = p_{in}, \tag{4.37}$$

for $1 \leq i \leq m$, respectively. The boundary condition in the y direction is given by

$$\lim_{y \to \infty} p_{i,j} = p_{in} \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq n. \tag{4.38}$$

Pressure and flow rate should be continuous in the z direction, which yields

$$p_{i,j}\big|_{z=z_j} = p_{i,j+1}\big|_{z=0}, \tag{4.39}$$

$$\left.\frac{\partial p_{i,j}}{\partial z}\right|_{z=z_j} = \zeta_{i,j}\left.\frac{\partial p_{i,j+1}}{\partial z}\right|_{z=0}, \tag{4.40}$$

for $1 \le i \le m, 1 \le j \le n-1$ where

$$\zeta_{i,j} = \frac{k_{z,i,j+1}}{k_{z,i,j}}. \tag{4.41}$$

The other boundary conditions are pressure and flow rate continuity boundary conditions in the x direction and boundary conditions at the wellbore. However, the former conditions are not introduced since the semi-permeable wall model is applied to Eq. 4.32 as shown in the next section. The latter will be fully explained in Section 4.2.3.

4.2.2 Application of the Semi-permeable Wall Model

The semi-permeable wall model has already been introduced in Section 3.1.2 for the vertical well case. For this horizontal well case, it is applied to the x direction of Eq. 4.32 to eliminate the pressure and the flow rate continuity boundary conditions in the x direction.

Integrating Eq. 4.32 over the interval $[x_{i-1}, x_i]$ yields

$$\begin{aligned} &k_{y,i,j}\delta x_i\frac{\partial^2\langle p_{i,j}\rangle}{\partial y^2} + k_{z,i,j}\delta x_i\frac{\partial^2\langle p_{i,j}\rangle}{\partial z^2} \\ &\quad + k_{x,i,j}\left.\frac{\partial p_{i,j}}{\partial x}\right|_{x=x_i} - k_{x,i,j}\left.\frac{\partial p_{i,j}}{\partial x}\right|_{x=x_{i-1}} = \phi_{i,j}\mu c_{t,i,j}\delta x_i\frac{\partial\langle p_{i,j}\rangle}{\partial t}, \end{aligned} \tag{4.42}$$

for $1 \le i \le m, 1 \le j \le n$ where

$$\langle p_{i,j}\rangle = \int_{x_{i-1}}^{x_i} p_{i,j}\,dx \Big/ \delta x_i, \tag{4.43}$$

$$\delta x_i = x_i - x_{i-1}. \tag{4.44}$$

Here, $\langle p_{i,j}\rangle$ is an averaged pressure over the interval $[x_{i-1}, x_i]$. Applying the finite difference method to the third and the fourth terms on the left hand side of Eq. 4.42 yields

$$\begin{aligned} &k_{y,i,j}\delta x_i\frac{\partial^2\langle p_{i,j}\rangle}{\partial y^2} + k_{z,i,j}\delta x_i\frac{\partial^2\langle p_{i,j}\rangle}{\partial z^2} \\ &\quad + \lambda_{i,j}\{\langle p_{i+1,j}\rangle - \langle p_{i,j}\rangle\} + \lambda_{i-1,j}\{\langle p_{i-1,j}\rangle - \langle p_{i,j}\rangle\} = \phi_{i,j}\mu c_{t,i,j}\delta x_i\frac{\partial\langle p_{i,j}\rangle}{\partial t}, \end{aligned} \tag{4.45}$$

for $1 \leq i \leq m, 1 \leq j \leq n$, where $\lambda_{i,j}$ is a parameter related to the crossflow between the grid blocks (i,j) and $(i+1,j)$, and is given by

$$\lambda_{i,j} = \frac{2}{\frac{\delta x_i}{k_{x,i,j}} + \frac{\delta x_{i+1}}{k_{x,i+1,j}}}. \tag{4.46}$$

If the left and the right reservoir boundaries of the reservoir are the no-flow types, they can be expressed as

$$\lambda_{0,j} = 0, \tag{4.47}$$

$$\lambda_{n,j} = 0, \tag{4.48}$$

for $1 \leq j \leq n$. If either the left or the right boundary is the constant pressure type boundary, Eq. 4.47 or Eq. 4.48 should be replaced by

$$\lambda_{0,j} = \frac{2k_{x,1,j}}{\delta x_1}, \tag{4.49}$$

$$\langle p_{0,j} \rangle = p_{in}, \tag{4.50}$$

or

$$\lambda_{n,j} = \frac{2k_{x,n,j}}{\delta x_n}, \tag{4.51}$$

$$\langle p_{n+1,j} \rangle = p_{in}, \tag{4.52}$$

for $1 \leq j \leq n$, respectively.

The introduction of the averaged pressure allows us to solve the two-dimensional problem of Eq. 4.45 more easily than to solve the original three-dimensional problem of Eq. 4.32 directly.

In the following discussion, delimiters $\langle$ and $\rangle$ are omitted for simplicity.

4.2.3 Strip Wellbore Model and Wellbore Boundary Conditions

In Section 4.1.1, the strip wellbore model developed by Goode and Thambynayagam [42] for homogeneous reservoirs was explained. Although the anisotropy of the reservoir was

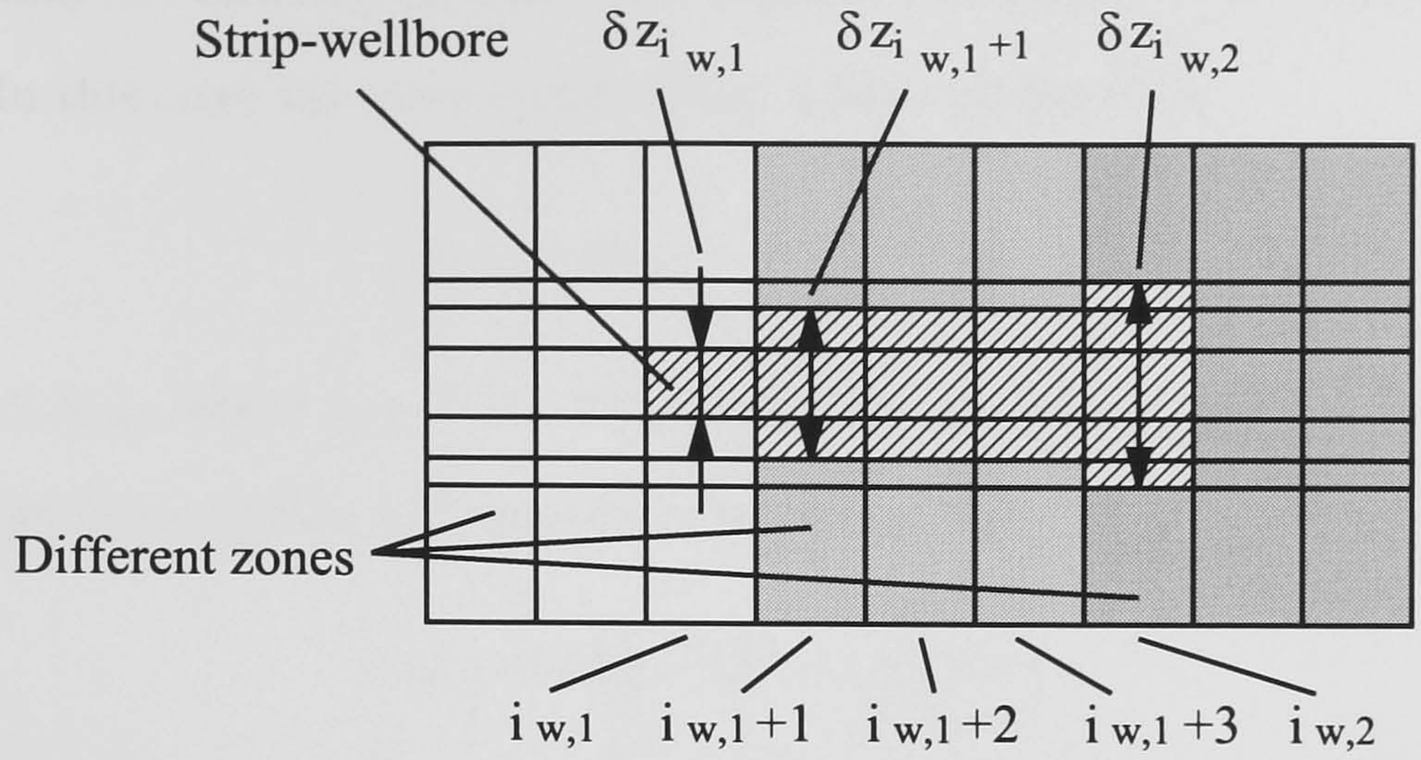


Figure 4.7: Strip wellbore model for heterogeneous reservoirs.

considered, several studies show if the reservoir is anisotropic in the y and z directions, the flow around the wellbore becomes elliptic [120, 58, 62]. In that case, the circular wellbore is transformed to an elliptical wellbore whose semi-axes are

$$a_w = r_w \sqrt[4]{\frac{k_y}{k_z}}, \tag{4.53}$$

$$b_w = r_w \sqrt[4]{\frac{k_z}{k_y}}, \tag{4.54}$$

in order to calculate correct pressure responses. When the flow near the well becomes pseudo-steady condition, the elliptical well with semi-axes a_w and b_w is equivalent to a circular well of radius $(a_w + b_w)/2$. If this effect is taken into account, the effective wellbore radius or the wellbore width of Eq. 4.2 can be modified by replacing r_w in Eq. 4.2 with $(a_w + b_w)/2$ as

$$r_w{}' = \frac{W}{4} = \frac{r_w}{2}\left(1 + \sqrt{\frac{k_z}{k_y}}\right). \tag{4.55}$$

Thus, the strip width is determined [30] by

$$W = 2r_w\left(1 + \sqrt{\frac{k_z}{k_y}}\right). \tag{4.56}$$

In this heterogeneous reservoir case, the wellbore is divided into several segments in order to deal with varying rock properties along the wellbore as shown in Fig. 4.7. The

strip width may be different in each well segment according to the permeability ratio $k_{z,i,j}/k_{y,i,j}$. In this case, the strip width of Eq. 4.56 is replaced by

$$\delta z_{w,i} = 2r_w\left(1+\sqrt{\frac{k_{z,i,j}}{k_{y,i,j}}}\right), \tag{4.57}$$

for $i_{w,1} \leq i \leq i_{w,2}$, where $z_{w,i}$ is the strip width for the well segment i. If damage skin exists, the expression of Eq. 4.57 may be extended as [30]

$$\delta z_{w,i} = 2r_w e^{-S_i}\left(1+\sqrt{\frac{k_{z,i,j}}{k_{y,i,j}}}\right), \tag{4.58}$$

where S_i is the damage skin factor for the element i. This expression allows us to model a horizontal well with negative skin using a finite wellbore radius condition. The early-time linear flow observed at very early times as reported by Streltsova [115] can be modelled unlike the line-source assumption of the wellbore.

Each wellbore strip $\delta z_{w,i}$ $(i_{w,1} \leq i \leq i_{w,2})$ may be divided vertically into several blocks according to the difference of the widths as shown in Fig. 4.7. Indices $j_{w,1}(i)$ and $j_{w,2}(i)$ are used to indicate the bottom and the top well blocks for the well segment i.

The wellbore boundary condition on flow rate becomes the summation of the flow rate of each grid blocks. It can be written mathematically as

$$\frac{qB}{2} = \lim_{y\to 0}\sum_{i=i_{w,1}}^{i_{w,2}}\sum_{j=j_{w,i}(i)}^{j_{w,2}(i)}\frac{k_{y,i,j}\delta x_i \delta z_j}{\mu}\frac{\partial p_{i,j}}{\partial y}, \tag{4.59}$$

where

$$\delta z_j = z_i - z_{i-1}. \tag{4.60}$$

For partially open wells (e.g., Ref. [43]), the well segments not open to flow should be eliminated from the above equation using

$$\lim_{y\to 0}\frac{\partial p_{i,j}}{\partial y} = 0, \tag{4.61}$$

for the well segment i not open to flow and $j_{w,1}(i) \leq j \leq j_{w,2}(i)$. The flow rate q_i of the well segment i can be obtained from

$$\frac{q_i B}{2} = \lim_{y\to 0}\sum_{j=j_{w,i}(i)}^{j_{w,2}(i)}\frac{k_{y,i,j}\delta x_i \delta z_j}{\mu}\frac{\partial p_{i,j}}{\partial y}, \tag{4.62}$$

for $i_{w,1} \leq i \leq i_{w,2}$.

In order to convert the above uniform-flux condition to the uniform-pressure condition, the pressure averaging method proposed by Wilkinson and Hammond [123], Kuchuk et al [62] was used in this study. In this method, the pressure obtained by the uniform-flux condition is averaged over the strip width. Since each wellbore strip $\delta z_{w,i}$ $(i_{w,1} \leq i \leq i_{w,2})$ is divided vertically into several regions, the averaged wellbore pressure for well segment i is given by

$$p_{wf,i} = \frac{1}{\sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \delta z_j} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \int_0^{\delta z_j} p_{wf,i,j}\, dz, \tag{4.63}$$

for $i_{w,1} \leq i \leq i_{w,2}$ where

$$p_{wf,i,j} = \lim_{y \to 0} p_{i,j}. \tag{4.64}$$

Using the averaged pressure, the wellbore boundary condition is given by

$$p_{wf} = p_{wf,i_{w,1}} = p_{wf,i_{w,1}+1} = \cdots = p_{wf,i_{w,2}}. \tag{4.65}$$

4.2.4 The Limit of Negative Skin Factor

In the previous section, the strip-wellbore model was introduced. The model can produce the early-time linear flow response observed at very early times, especially when the skin is negative (the effective wellbore radius is large).

However, there is a limit to the value of negative skin which can be substituted in Eq. 4.58. Suppose a stimulated well is located at the distance of h_s from the bottom of a homogeneous grid block with its height h_{wb} (Fig. 4.8). If the absolute value of the negative skin is extremely large, and a well is located very near to a top or bottom boundary of a reservoir or a layer, it is possible that the strip wellbore protrudes outside the layer within which the well was originally located. This will produce an incorrect pressure response since part of the wellbore produces fluid from the layer where the well doesn't exist. Thus, the following condition must be satisfied to avoid the protrusion of the strip wellbore, i.e.,

$$\frac{\delta z_{w,i}}{2} < \min\{h_s, h_{wb} - h_s\}. \tag{4.66}$$

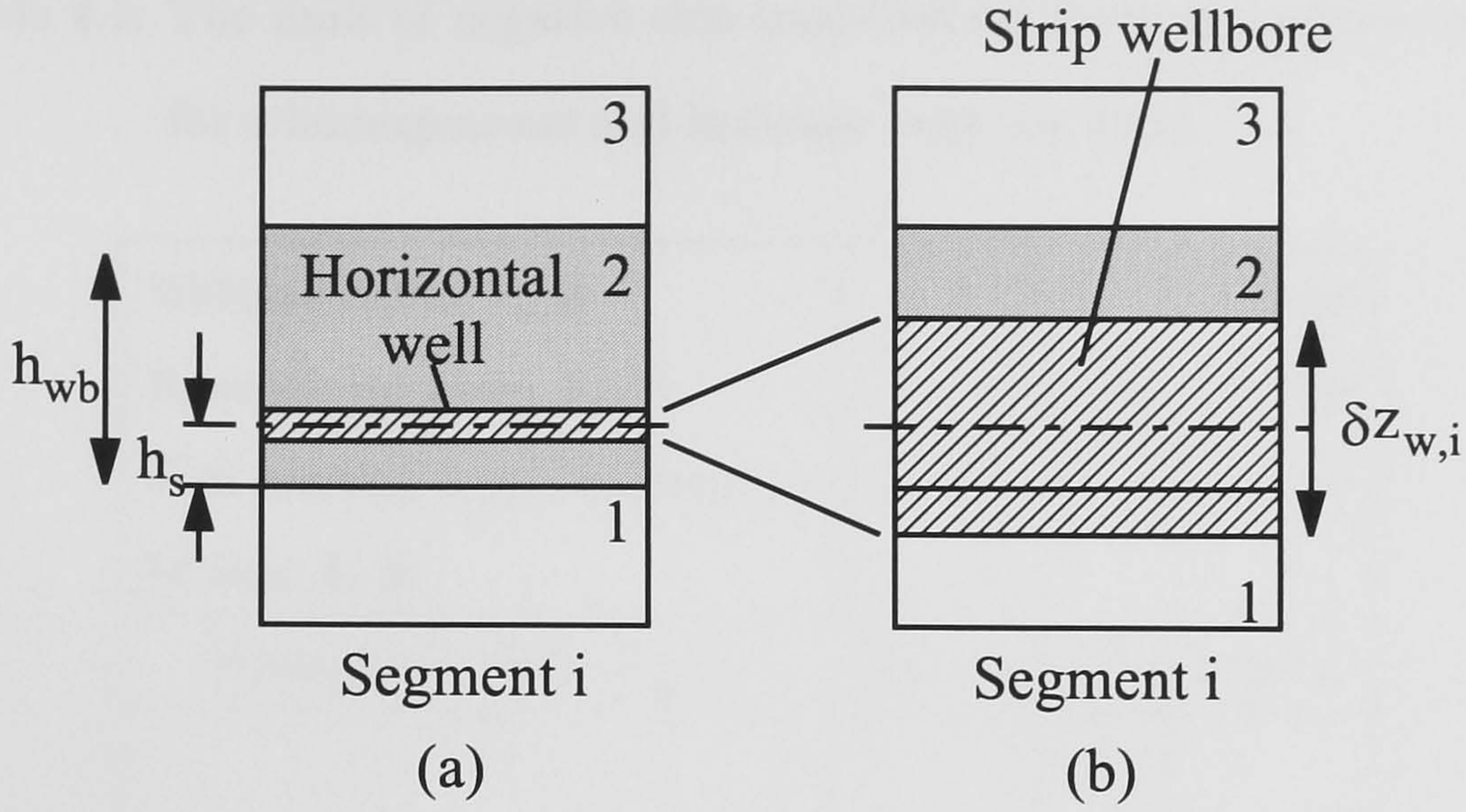


Figure 4.8: Improper strip-wellbore model in a three-layer system. (a) Original well position in Layer 2. (b) Strip-wellbore protrude from Layer 2 into Layer 1.

From Eq. 4.58, the above equation can be written as

$$S_i > \ln\left[\frac{r_w(1+\sqrt{\frac{k_{z,i,j}}{k_{y,i,j}}})}{\min\{h_s, h_{wb}-h_s\}}\right]. \tag{4.67}$$

Examples for homogeneous and isotropic reservoirs are shown in Table 4.1 together with the well and the reservoir parameters. When the negative skin is outside the range calculated by Eq. 4.67, the effect of the skin should be included after the well segment pressure without skin damage is obtained in Laplace domain.

If the zone of infinitesimal thickness skin exists around the well, the sandface pressure with damage skin for the well segment i can be expressed using the notation in Section 3.1.5 as

$$p_{sf,D,i} = p_{sf,D,i,S=0} + S_i, \tag{4.68}$$

where $p_{sf,D,i,S=0}$ is the sandface pressure without damage skin for the well segment i, and its Laplace transform is given by

$$\overline{p}_{sf,D,i} = \overline{p}_{sf,D,i,S=0} + \frac{S_i}{l}. \tag{4.69}$$

Table 4.1: The limit of negative skin modelled by the strip wellbore model for a homogeneous and isotropic reservoir case.

Wellbore radius, r_w ft	0.33
Reservoir thickness, h_{wb} ft	100
Well position from the reservoir bottom, h_s ft	50
Permeability Ratio, k_z/k_y	The range of skin factor
1	> -4.33
0.1	> -4.75
0.01	> -4.93
0.001	> -4.99

Substituting the above equation into Eq. 3.61 yields

$$\overline{p}_{wf,D,i} = \frac{l\overline{p}_{sf,D,i,S=0} + S_i}{l\{1 + lC_D(l\overline{p}_{sf,D,i,S=0} + S_i)\}}. \tag{4.70}$$

This equation indicates how to include the damage skin after the Laplace pressure without the damage skin is calculated. However, it is impossible to model the early-time linear flow with this method.

For the case of positive damage skin, the protrusion of the strip wellbore is unlikely to occur unless the wellbore is located extremely near to the reservoir or layer boundary. The limit of the wellbore model can be confirmed from Eq. 4.67 easily.

4.2.5 Dimensionless Variables and Equations

In this chapter, only pressure drawdown tests are considered. The response of pressure buildup tests may be generated by the superposition of the pressure drawdown responses using Eq. 3.110.

Dimensionless pressure for the pressure drawdown is defined as

$$p_{D,i,j} = \frac{2\pi\sqrt{\overline{k_y k_z}}h(p_{in} - p_{D,i,j})}{qB\mu}, \tag{4.71}$$

where h is the reservoir thickness given by

$$h = \sum_{j=1}^{n} \delta z_j, \tag{4.72}$$

and $\overline{\sqrt{k_y k_z}}$ is an average permeability expressed as

$$\overline{\sqrt{k_y k_z}} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\sqrt{k_{y,i,j}k_{z,i,j}}\delta x_i \delta z_j}{\sum_{i=1}^{m}\sum_{j=1}^{n}\delta x_i \delta z_j}. \tag{4.73}$$

Dimensionless time is expressed as

$$t_D = \frac{4\overline{\sqrt{k_y k_z}}t}{\overline{\phi c_t}\mu {L_w}^2}, \tag{4.74}$$

where L_w is the wellbore length given by

$$L_w = \sum_{i=i_{w,1}}^{i_{w,2}} \delta x_i, \tag{4.75}$$

and $\overline{\phi c_t}$ is an average porosity-compressibility product expressed as

$$\overline{\phi c_t} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\phi_{i,j}c_{t,i,j}\delta x_i \delta z_j}{\sum_{i=1}^{m}\sum_{j=1}^{n}\delta x_i \delta z_j}. \tag{4.76}$$

Dimensionless distances in each coordinate are defined by

$$x_D = \frac{2x}{L_w}, \tag{4.77}$$

$$y_D = \frac{2y}{L_w}, \tag{4.78}$$

$$z_D = \frac{2z}{L_w}. \tag{4.79}$$

Finally, dimensionless wellbore storage constant is given by

$$C_D = \frac{4C}{2\pi\overline{\phi c_t}\mu {L_w}^2}. \tag{4.80}$$

Using these dimensionless variables, Eq. 4.45 can be rewritten in dimensionless units as

$$\begin{aligned}\alpha_{i,j}\frac{\partial^2 p_{D,i,j}}{\partial {y_D}^2} + \beta_{i,j}\frac{\partial^2 p_{D,i,j}}{\partial {z_D}^2} + \tilde{\lambda}_{i,j}(p_{D,i+1,j} - p_{D,i,j}) \\ +\tilde{\lambda}_{i-1,j}(p_{D,i-1,j} - p_{D,i,j}) = \omega_{i,j}\frac{\partial p_{D,i,j}}{\partial t_D},\end{aligned} \tag{4.81}$$

for $1 \leq i \leq m, 1 \leq j \leq n$ where

$$\alpha_{i,j} = \frac{k_{y,i,j}\delta x_i}{\sqrt{k_y k_z}h}, \tag{4.82}$$

$$\beta_{i,j} = \frac{k_{z,i,j}\delta x_i}{\sqrt{k_y k_z}h}, \tag{4.83}$$

$$\omega_{i,j} = \frac{\phi_{i,j}c_{t,i,j}\delta x_i}{\overline{\phi c_t}h}, \tag{4.84}$$

$$\tilde{\lambda}_{i,j} = \frac{{L_w}^2}{4\sqrt{k_y k_z}h}\lambda_{i,j}. \tag{4.85}$$

The initial condition of Eq. 4.33 becomes

$$\lim_{t_D \to 0} p_{D,i,j} = 0 \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq n. \tag{4.86}$$

The no-flow top and bottom boundary conditions of Eqs. 4.34 and 4.35 are

$$\left.\frac{\partial p_{D,i,n}}{\partial z_D}\right|_{z_D=\frac{2z_n}{L_w}} = 0, \tag{4.87}$$

$$\left.\frac{\partial p_{D,i,1}}{\partial z_D}\right|_{z_D=0} = 0, \tag{4.88}$$

for $1 \leq i \leq m$. The constant pressure top and bottom boundary conditions of Eqs. 4.36, 4.37 can be expressed as

$$p_{D,i,n}\big|_{z_D=\frac{2z_n}{L_w}} = 0, \tag{4.89}$$

$$p_{D,i,1}\big|_{z_D=0} = 0, \tag{4.90}$$

for $1 \leq i \leq m$. The boundary condition of Eq. 4.38 is given by

$$\lim_{y_D \to \infty} p_{D,i,j} = 0 \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq n. \tag{4.91}$$

The pressure and the rate continuity boundary conditions between the neighbouring grid blocks (i,j) and $(i,j+1)$ of Eqs. 4.39 and 4.40 become

$$p_{D,i,j}\big|_{z_D=\frac{2z_j}{L_w}} = p_{D,i,j+1}\big|_{z_D=0}, \tag{4.92}$$

$$\left.\frac{\partial p_{D,i,j}}{\partial z_D}\right|_{z_D=\frac{2z_j}{L_w}} = \zeta_{i,j}\left.\frac{\partial p_{D,i,j+1}}{\partial z_D}\right|_{z_D=0}, \tag{4.93}$$

for $1 \leq i \leq m, 1 \leq j \leq n-1$. The wellbore boundary condition on the flow rate of Eq. 4.59 is expressed as

$$\lim_{y_D \to 0} \sum_{i=i_{w,1}}^{i_{w,2}} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \alpha_{i,j} \delta z_{D,j} \frac{\partial p_{D,i,j}}{\partial y_D} = -\pi. \tag{4.94}$$

For the partially open wells, Eq. 4.61 becomes

$$\lim_{y_D \to 0} \frac{\partial p_{D,i,j}}{\partial y_D} = 0, \tag{4.95}$$

for the well segment i not open to flow and $j_{w,1}(i) \leq j \leq j_{w,2}(i)$. The segment flow rate of Eq. 4.62 is expressed in the form of the fractional flow rate as

$$q_{D,i} = \frac{q_i}{q} = -\frac{1}{\pi} \lim_{y_D \to 0} \sum_{j=j_{w,i}(i)}^{j_{w,2}(i)} \alpha_{i,j} \delta z_{D,j} \frac{\partial p_{D,i,j}}{\partial y_D}, \tag{4.96}$$

for $i_{w,1} \leq i \leq i_{w,2}$. The wellbore condition on the pressure of Eq. 4.65 is given by

$$p_{D,wf} = p_{D,wf,i_{w,1}} = p_{D,wf,i_{w,1}+1} = \cdots = p_{D,wf,i_{w,2}}, \tag{4.97}$$

where

$$p_{D,wf,i} = \frac{1}{\sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \delta z_{D,j}} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \int_0^{\delta z_{D,j}} p_{D,wf,i,j} \, dz_D, \tag{4.98}$$

and

$$p_{D,wf,i,j} = \lim_{y_D \to 0} p_{D,i,j}. \tag{4.99}$$

4.2.6 Application of the Laplace Transform

Applying Laplace transform on t_D to Eq. 4.81 and using the initial condition given by Eq. 4.86 yield

$$\begin{aligned} &\alpha_{i,j} \frac{\partial^2 \overline{p}_{D,i,j}}{\partial {y_D}^2} + \beta_{i,j} \frac{\partial^2 \overline{p}_{D,i,j}}{\partial {z_D}^2} + \tilde{\lambda}_{i,j} (\overline{p}_{D,i+1,j} - \overline{p}_{D,i,j}) \\ &+ \tilde{\lambda}_{i-1,j} (\overline{p}_{D,i-1,j} - \overline{p}_{D,i,j}) = \omega_{i,j} \overline{p}_{D,i,j} l, \end{aligned} \tag{4.100}$$

where $\overline{p}_{D,i,j}$ is the Laplace transformed pressure given by

$$\overline{p}_{D,i,j} = \int_0^\infty e^{-lt_D} p_{D,i,j} \, dt_D, \tag{4.101}$$

and l is the Laplace variable.

The Laplace transforms of the no-flow top and the bottom boundary conditions of Eqs. 4.87 and 4.88 are given by

$$\left.\frac{\partial \overline{p}_{D,i,n}}{\partial z_D}\right|_{z_D=\frac{2z_n}{L_w}} = 0, \tag{4.102}$$

$$\left.\frac{\partial \overline{p}_{D,i,1}}{\partial z_D}\right|_{z_D=0} = 0, \tag{4.103}$$

for $1 \leq i \leq m$. The constant pressure top and bottom boundary conditions of Eqs. 4.89 and 4.90 become

$$\left.\overline{p}_{D,i,n}\right|_{z_D=\frac{2z_n}{L_w}} = 0, \tag{4.104}$$

$$\left.\overline{p}_{D,i,1}\right|_{z_D=0} = 0, \tag{4.105}$$

for $1 \leq i \leq m$. The outer boundary condition in the y direction of Eq. 4.91 becomes

$$\lim_{y_D \to \infty} \overline{p}_{D,i,j} = 0 \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq n. \tag{4.106}$$

The pressure and the rate continuity boundary conditions between the neighbouring grid blocks (i,j) and $(i,j+1)$ of Eqs. 4.92 and 4.93 are given by

$$\left.\overline{p}_{D,i,j}\right|_{z_D=\frac{2z_j}{L_w}} = \left.\overline{p}_{D,i,j+1}\right|_{z_D=0}, \tag{4.107}$$

$$\left.\frac{\partial \overline{p}_{D,i,j}}{\partial z_D}\right|_{z_D=\frac{2z_j}{L_w}} = \zeta_{i,j} \left.\frac{\partial \overline{p}_{D,i,j+1}}{\partial z_D}\right|_{z_D=0}, \tag{4.108}$$

for $1 \leq i \leq m, 1 \leq j \leq n-1$. The wellbore boundary condition on the flow rate given by Eq. 4.94 becomes

$$\sum_{i=i_{w,1}}^{i_{w,2}} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \alpha_{i,j} \delta z_{D,j} D_{i,j} = -\frac{\pi}{l}, \tag{4.109}$$

where

$$D_{i,j} = \lim_{y_D \to 0} \frac{\partial \overline{p}_{D,i,j}}{\partial y_D}. \tag{4.110}$$

For the partially open wells, Eq. 4.95 becomes

$$D_{i,j} = 0, \tag{4.111}$$

for the well segment i not open to flow and $j_{w,1}(i) \leq j \leq j_{w,2}(i)$. The fractional flow rate of the well segment i of Eq. 4.96 is given by

$$\overline{q}_{D,i} = \frac{\overline{q}_i}{q} = -\frac{1}{\pi} \sum_{j=j_{w,i}(i)}^{j_{w,2}(i)} \alpha_{i,j} \delta z_{D,j} D_{i,j}, \tag{4.112}$$

for $i_{w,1} \leq i \leq i_{w,2}$. The wellbore condition on pressure of Eq. 4.97 is expressed as

$$\overline{p}_{D,wf} = \overline{p}_{D,wf,i_{w,1}} = \overline{p}_{D,wf,i_{w,1}+1} = \cdots = \overline{p}_{D,wf,i_{w,2}}, \tag{4.113}$$

where

$$\overline{p}_{D,wf,i} = \frac{1}{\sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \delta z_{D,j}} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \int_0^{\delta z_{D,j}} \overline{p}_{D,wf,i,j}\, dz_D, \tag{4.114}$$

and

$$\overline{p}_{D,wf,i,j} = \lim_{y_D \to 0} \overline{p}_{D,i,j}. \tag{4.115}$$

4.2.7 Application of the Fourier Cosine Transform

Applying the Fourier cosine transform [17] to Eq. 4.100 and using the boundary conditions of Eqs. 4.106 yield

$$\begin{aligned} \beta_{i,j} \frac{\partial^2 \overline{\overline{p}}_{D,i,j}}{\partial {z_D}^2} - (\alpha_{i,j}\tau^2 + \tilde{\lambda}_{i,j} + \tilde{\lambda}_{i-1,j} + \omega_{i,j} l) \overline{\overline{p}}_{D,i,j} \\ + \tilde{\lambda}_{i,j} \overline{\overline{p}}_{D,i+1,j} + \tilde{\lambda}_{i-1,j} \overline{\overline{p}}_{D,i-1,j} = \alpha_{i,j} D_{i,j}, \end{aligned} \tag{4.116}$$

for $1 \leq i \leq m, 1 \leq j \leq n$ where $\overline{\overline{p}}_{D,i,j}$ is the Fourier transformed Laplace pressure given by

$$\overline{\overline{p}}_{D,i,j} = \int_0^\infty \overline{p}_{D,i,j} \cos(\tau y_D)\, dy_D, \tag{4.117}$$

and τ is the Fourier variable. If the grid block (i, j) is not the well block,

$$D_{i,j} = 0, \tag{4.118}$$

in Eq. 4.116.

The Fourier transform of the no-flow top and the bottom boundary conditions (Eqs. 4.102 and 4.103) are

$$\left.\frac{\partial \overline{\overline{p}}_{D,i,n}}{\partial z_D}\right|_{z_D=\frac{2z_n}{L_w}} = 0, \tag{4.119}$$

$$\left.\frac{\partial \overline{\overline{p}}_{D,i,1}}{\partial z_D}\right|_{z_D=0} = 0, \tag{4.120}$$

for $1 \leq i \leq m$. When either the top or the bottom reservoir boundary is the constant pressure type, the transformed expression of Eq. 4.104 or Eq. 4.105 is given by

$$\left.\overline{\overline{p}}_{D,i,n}\right|_{z_D=\frac{2z_n}{L_w}} = 0, \tag{4.121}$$

or

$$\left.\overline{\overline{p}}_{D,i,1}\right|_{z_D=0} = 0, \tag{4.122}$$

for $1 \leq i \leq m$.

The pressure and the rate continuity boundary conditions between the neighbouring grid blocks (i,j) and $(i,j+1)$ of Eqs. 4.107 and 4.108 are given by

$$\left.\overline{\overline{p}}_{D,i,j}\right|_{z_D=\frac{2z_j}{L_w}} = \left.\overline{\overline{p}}_{D,i,j+1}\right|_{z_D=0}, \tag{4.123}$$

$$\left.\frac{\partial \overline{\overline{p}}_{D,i,j}}{\partial z_D}\right|_{z_D=\frac{2z_j}{L_w}} = \zeta_{i,j}\left.\frac{\partial \overline{\overline{p}}_{D,i,j+1}}{\partial z_D}\right|_{z_D=0}, \tag{4.124}$$

for $1 \leq i \leq m, 1 \leq j \leq n-1$.

4.2.8 Solution in the Laplace Domain

Since Eq. 4.116 has the form of a second order linear differential equation, a general solution of the problem can be written as

$$\overline{\overline{p}}_{D,i,j} = A_{i,j}e^{\sigma_j z_D} + B_{i,j}e^{-\sigma_j z_D}, \tag{4.125}$$

where $A_{i,j}$ and $B_{i,j}$ are constants associated with the grid block (i,j), and they are determined by the boundary conditions. Substituting Eq. 4.125 into Eq. 4.116 and rearranging it following the procedure in Ref. [67] yield

$$-\frac{\tilde{\lambda}_{i-1,j}}{\sqrt{\beta_{i-1,j}\beta_{i,j}}}(\sqrt{\beta_{i-1,j}}\overline{\overline{p}}_{D,i-1,j}) + \frac{\alpha_{i,j}\tau^2+\tilde{\lambda}_{i-1,j}+\tilde{\lambda}_{i,j}+\omega_{i,j}l}{\beta_{i,j}}(\sqrt{\beta_{i,j}}\overline{\overline{p}}_{D,i,j}) - \frac{\tilde{\lambda}_{i,j}}{\sqrt{\beta_{i,j}\beta_{i+1,j}}}(\sqrt{\beta_{i+1,j}}\overline{\overline{p}}_{D,i+1,j}) = {\sigma_j}^2(\sqrt{\beta_{i,j}}\overline{\overline{p}}_{D,i,j}), \tag{4.126}$$

for $1 \leq i \leq m, 1 \leq j \leq n$. Eq. 4.126 constitutes a system of equation for each j, and is treated as an eigenvalue problem where ${\sigma_j}^2$ are eigenvalues and $\{\sqrt{\beta_{i,j}}\overline{\overline{p}}_{D,i,j}\}$ are eigenvectors for the system. The method to obtain eigenvalues and eigenvectors are similar to the vertical well case discussed in Chapter 3.

The system of equations of Eq. 4.126 has a non-trivial solution if, and only if

$$\det|\mathbf{G}_j - \sigma_j^2\mathbf{I}| = 0, \tag{4.127}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{G}_j$ is a $m \times m$ matrix given by

$$\mathbf{G}_j = \{g^j_{i,i'}\} = \begin{cases} -\frac{\tilde{\lambda}_{i-1,j}}{\sqrt{\beta_{i-1,j}\beta_{i,j}}} & \text{for } i' = i-1, i \geq 2, \\ \frac{\alpha_{i,j}\tau^2+\tilde{\lambda}_{i-1,j}+\tilde{\lambda}_{i,j}+\omega_{i,j}l}{\beta_{i,j}} & \text{for } i' = i, \\ -\frac{\tilde{\lambda}_{i,j}}{\sqrt{\beta_{i,j}\beta_{i+1,j}}} & \text{for } i' = i+1, i \leq m-1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.128}$$

for $1 \leq j \leq n$ [33]. Eq. 4.127 is an mth degree polynomial in ${\sigma_j}^2$, and the roots of the polynomial are the eigenvalues expressed as $(\sigma_j^k)^2$ where $1 \leq k \leq m$.

Since Eq. 4.116 is linear, its general solution is the linear superposition of Eq. 4.125 for each k, i.e.,

$$\overline{\overline{p}}_{D,i,j} = \sum_{k=1}^{m}(A^k_{i,j}e^{\sigma_j^k z_D} + B^k_{i,j}e^{-\sigma_j^k z_D}). \tag{4.129}$$

As in the case of Chapter 3, constants $A^k_{i,j}$ and $B^k_{i,j}$ are not independent of each other. Substituting Eq. 4.129 into Eq. 4.126 yields

$$\mathbf{G}^k_j\mathbf{A}^k_j = 0, \tag{4.130}$$

$$\mathbf{G}^k_j\mathbf{B}^k_j = 0, \tag{4.131}$$

where $\mathbf{G}_j^k$ is a $m \times m$ matrix given by

$$\mathbf{G}_j^k = \{g_{i,i'}^{j,k}\} = \begin{cases} -\frac{\tilde{\lambda}_{i-1,j}}{\sqrt{\beta_{i-1,j}\beta_{i,j}}} & \text{for } i' = i-1, i \geq 2, \\ \frac{\alpha_{i,j}\tau^2 + \tilde{\lambda}_{i-1,j} + \tilde{\lambda}_{i,j} + \omega_{i,j}l}{\beta_{i,j}} - (\sigma_j^k)^2 & \text{for } i' = i, \\ -\frac{\tilde{\lambda}_{i,j}}{\sqrt{\beta_{i,j}\beta_{i+1,j}}} & \text{for } i' = i+1, i \leq m-1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.132}$$

and

$$\mathbf{A}_j^k = \begin{pmatrix} \sqrt{\beta_{1,j}} A_{1,j}^k \\ \sqrt{\beta_{2,j}} A_{2,j}^k \\ \vdots \\ \sqrt{\beta_{m,j}} A_{m,j}^k \end{pmatrix}, \tag{4.133}$$

$$\mathbf{B}_j^k = \begin{pmatrix} \sqrt{\beta_{1,j}} B_{1,j}^k \\ \sqrt{\beta_{2,j}} B_{2,j}^k \\ \vdots \\ \sqrt{\beta_{m,j}} B_{m,j}^k \end{pmatrix}. \tag{4.134}$$

From Eq. 4.130, the following recursion formula is obtained, i.e.,

$$\begin{aligned} A_{2,j}^k &= -\frac{1}{g_{1,2}^{j,k}\sqrt{\beta_{2,j}}}(g_{1,1}^{j,k}\sqrt{\beta_{1,j}}A_{1,j}^k) = E_{2,j}^k A_{1,j}^k, \\ A_{3,j}^k &= -\frac{1}{g_{2,3}^{j,k}\sqrt{\beta_{3,j}}}(g_{2,1}^{j,k}\sqrt{\beta_{1,j}}A_{1,j}^k + g_{2,2}^{j,k}\sqrt{\beta_{2,j}}A_{2,j}^k) \\ &= E_{3,j}^k A_{1,j}^k, \\ &\cdots\cdots\cdots\cdots \\ A_{m,j}^k &= -\frac{1}{g_{m-1,m}^{j,k}\sqrt{\beta_{m,j}}}(g_{m-1,m-2}^{j,k}\sqrt{\beta_{m-2,j}}A_{m-1,j}^k + g_{m-1,m-1}^{j,k}\sqrt{\beta_{m-1,j}}A_{m-1,j}^k) \\ &= E_{m,j}^k A_{1,j}^k. \end{aligned} \tag{4.135}$$

Thus, replacing $A_{1,j}^k$ with A_j^k yields

$$A_{i,j}^k = E_{i,j}^k A_j^k. \tag{4.136}$$

Similarly, constants $B_{i,j}^k$ are given from Eq. 4.131 by

$$B_{i,j}^k = E_{i,j}^k B_j^k. \tag{4.137}$$

If eigenvectors of Eq. 4.126 is calculated by some effective algorithms together with the eigenvalues, $\{A^k_{i,j}\}$ can be written in the similar way to the Eq. 3.81 as

$$\begin{pmatrix} A^k_{1,j} \\ A^k_{2,j} \\ \vdots \\ A^k_{m,j} \end{pmatrix} = A^k_j \begin{pmatrix} \frac{{A^k_{1,j}}'}{\sqrt{\beta_{1,j}}} \\ \frac{{A^k_{2,j}}'}{\sqrt{\beta_{2,j}}} \\ \vdots \\ \frac{{A^k_{m,j}}'}{\sqrt{\beta_{m,j}}} \end{pmatrix}, \tag{4.138}$$

where $({A^k_{1,j}}', {A^k_{2,j}}', \ldots, {A^k_{m,j}}')^T$ is the eigenvector of Eq. 4.126, and A^k_j is a constant determined by the boundary conditions. The above equation can be expressed simply as

$$A^k_{i,j} = {E^k_{i,j}}' A^k_j. \tag{4.139}$$

Similarly, $\{B^k_{i,j}\}$ can be obtained from Eq. 4.131 as

$$B^k_{i,j} = {E^k_{i,j}}' B^k_j. \tag{4.140}$$

Using Eqs. 4.136 and 4.137 (or Eqs. 4.139 and 4.140), the general solution of Eq. 4.129 can be written as

$$\overline{\overline{p}}_{D,i,j} = \sum_{k=1}^{m} E^k_{i,j}(A^k_j e^{\sigma^k_j z_D} + B^k_j e^{-\sigma^k_j z_D}), \tag{4.141}$$

for $1 \le i \le m, 1 \le j \le n$.

Here, a particular solution of Eq. 4.116 is assumed to be $C_{i,j}$. Substituting $\overline{\overline{p}}_{D,i,j} = C_{i,j}$ into Eq. 4.116 yields

$$-\tilde{\lambda}_{i-1,j}C_{i-1,j} + (\alpha_{i,j}\tau^2 + \tilde{\lambda}_{i-1,j} + \tilde{\lambda}_{i,j} + \omega_{i,j}l)C_{i,j} - \tilde{\lambda}_{i,j}C_{i+1,j} = -\alpha_{i,j}D_{i,j}, \tag{4.142}$$

$$C_{0,j} = C_{m+1,j} = 0, \tag{4.143}$$

for

$$1 \le i \le m,$$

$$\min\{j_{w,1}(i) \mid i_{w,1} \le i \le i_{w,2}\} \le j \le \max\{j_{w,2}(i) \mid i_{w,1} \le i \le i_{w,2}\},$$

where

$$D_{i,j} = \begin{cases} \lim_{y_D \to 0} \frac{\partial \overline{p}_{D,i,j}}{\partial y_D} & \text{for } (i,j) \in \text{ well blocks}, \\ 0 & \text{for } (i,j) \notin \text{ well blocks}, \end{cases} \tag{4.144}$$

and

$$C_{i,j} = 0, \tag{4.145}$$

for

$$1 \leq i \leq m, \quad 1 \leq j < \min\{j_{w,1}(i) \mid i_{w,1} \leq i \leq i_{w,2}\},$$
$$n \geq j > \max\{j_{w,1}(i) \mid i_{w,1} \leq i \leq i_{w,2}\}.$$

Eq. 4.142 is a symmetric tridiagonal system of equations, and can be solved by a simple numerical algorithm for each layer j. Thus, the solution of the non-homogeneous system of equations (Eq. 4.116) is given by

$$\bar{\bar{p}}_{D,i,j} = \sum_{k=1}^{m} E_{i,j}^{k}(A_j^k e^{\sigma_j^k z_D} + B_j^k e^{-\sigma_j^k z_D}) + C_{i,j}. \tag{4.146}$$

Substituting Eq. 4.146 into the no-flow top and bottom boundary conditions of Eqs. 4.119 and 4.120 yields

$$\sum_{k=0}^{m} \sigma_n^k E_{i,n}^k (A_n^k e^{\sigma_n^k z_{D,n}} - B_n^k e^{-\sigma_n^k z_{D,n}}) = 0, \tag{4.147}$$

$$\sum_{k=0}^{m} \sigma_1^k E_{i,1}^k (A_1^k - B_1^k) = 0, \tag{4.148}$$

for $1 \leq i \leq m$ where $z_{D,n} = 2z_n/L_w$. If either the top or bottom boundary is the constant pressure type given by Eq. 4.121 or Eq. 4.122, Eq. 4.147 or Eq. 4.148 should be replaced by

$$\sum_{k=0}^{m} E_{i,n}^k (A_n^k e^{\sigma_n^k z_{D,n}} + B_n^k e^{-\sigma_n^k z_{D,n}}) + C_{i,n} = 0, \tag{4.149}$$

or

$$\sum_{k=0}^{m} E_{i,1}^k (A_1^k + B_1^k) + C_{i,1} = 0, \tag{4.150}$$

for $1 \leq i \leq m$. The interblock pressure and rate continuity boundary conditions of Eqs. 4.123 and 4.124 become

$$\sum_{k=0}^{m} E_{i,j}^k (A_j^k e^{\sigma_j^k z_{D,j}} + B_j^k e^{-\sigma_j^k z_{D,j}}) + C_{i,j}$$
$$= \sum_{k=0}^{m} E_{i,j+1}^k (A_{j+1}^k + B_{j+1}^k) + C_{i,j+1}, \tag{4.151}$$

$$\sum_{k=0}^{m} \sigma_j^k E_{i,j}^k (A_j^k e^{\sigma_j^k z_{D,j}} - B_j^k e^{-\sigma_j^k z_{D,j}})$$
$$= \zeta_{i,j} \sum_{k=0}^{m} \sigma_{j+1}^k E_{i,j+1}^k (A_{j+1}^k - B_{j+1}^k), \tag{4.152}$$

for $1 \le i \le m, 1 \le j \le n-1$.

In Eqs. 4.147 to 4.152, the number of unknown parameters (A_j^k and B_j^k) is $2mn$, and the number of equations is $2mn$. Thus, the system of equations is solvable.

Once the unknown parameters (A_j^k and B_j^k) are determined, the inverse Fourier cosine transform is applied to Eq. 4.146, i.e.,

$$\overline{p}_{D,i,j} = \frac{2}{\pi} \int_0^\infty \{\sum_{k=1}^{m} E_{i,j}^k (A_j^k e^{\sigma_j^k z_D} + B_j^k e^{-\sigma_j^k z_D}) + C_{i,j}\} \cos(\tau y_D)\, d\tau, \tag{4.153}$$

for $1 \le i \le m, 1 \le j \le n$. The wellbore pressure for the well grid block $\{(i,j) \mid i_{w,1} \le i \le i_{w,2}, j_{w,1}(i) \le j \le j_{w,2}(i)\}$ is obtained by

$$\overline{p}_{D,wf,i,j} = \lim_{y_D \to 0} \overline{p}_{D,i,j}$$
$$= \frac{2}{\pi} \int_0^\infty \{\sum_{k=1}^{m} E_{i,j}^k (A_j^k e^{\sigma_j^k z_D} + B_j^k e^{-\sigma_j^k z_D}) + C_{i,j}\}\, d\tau. \tag{4.154}$$

Using the pressure averaging method of Eq. 4.114, the wellbore pressure is obtained for each well segment i $(i_{w,1} \le i \le i_{w,2})$ as

$$\overline{p}_{D,wf,i} = \frac{1}{\sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \delta z_{D,j}} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \int_0^{\delta z_{D,j}} \overline{p}_{D,wf,i,j}\, dz_D$$
$$= \frac{2}{\pi \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \delta z_{D,j}} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \int_0^\infty \Big[\sum_{k=0}^{m} \frac{E_{i,j}^k}{\sigma_j^k} \{A_j^k (e^{\sigma_j^k \delta z_{D,j}} - 1)$$
$$- B_j^k (e^{-\sigma_j^k \delta z_{D,j}} - 1)\} + C_{i,j} \delta z_{D,j}\Big]\, d\tau. \tag{4.155}$$

The Laplace pressure gradients $D_{i,j}$ in Eq. 4.109 are also averaged over the blocks $j_{w,1}(i) \le j \le j_{w,2}(i)$ for the well segment i, i.e.,

$$D_i = \frac{1}{\sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \delta z_{D,j}} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \int_0^{\delta z_{D,j}} D_{i,j}\, dz_D$$
$$= \lim_{y_D \to 0} \frac{\partial \overline{p}_{D,wf,i}}{\partial y_D}. \tag{4.156}$$

Thus, Eq. 4.109 may be modified by using the averaged D_i as

$$\sum_{i=i_{w,1}}^{i_{w,2}} D_i \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \alpha_{i,j}\delta z_{D,j} = -\frac{\pi}{l}. \tag{4.157}$$

Eq. 4.113 yields a following non-linear system of equations, i.e.,

$$\begin{aligned} f_1(\mathbf{D}) &= \overline{p}_{D,wf,i_{w,1}+1} - \overline{p}_{D,wf,i_{w,1}} = 0, \\ f_2(\mathbf{D}) &= \overline{p}_{D,wf,i_{w,1}+2} - \overline{p}_{D,wf,i_{w,1}} = 0, \\ &\cdots\cdots\cdots \\ f_{i_{w,2}-i_{w,1}}(\mathbf{D}) &= \overline{p}_{D,wf,i_{w,2}} - \overline{p}_{D,wf,i_{w,1}} = 0, \end{aligned} \tag{4.158}$$

where $\mathbf{D} = (D_{i_{w,1}}, D_{i_{w,1}+1}, \ldots, D_{i_{w,2}})^T$.

Although $\mathbf{D}$ can be obtained from Eqs. 4.157 and 4.158 uniquely, they cannot be solved explicitly since they constitute a non-linear system of equations. However, they can be solved by some numerical iterative methods, such as the Newton-Raphson method [95].

Finally, the wellbore pressure of the horizontal well in the Laplace domain is given by

$$\overline{p}_{D,wf} = \overline{p}_{D,wf,i_{w,1}}. \tag{4.159}$$

The pressure in the real domain is calculated using the Stehfest algorithm shown in Section 3.1.7.

The fractional flow rate for the well segment i of Eq. 4.112 can be calculated in the Laplace domain using the averaged D_i as

$$\overline{q}_{D,i} = \frac{\overline{q}_i}{q} = -\frac{D_i}{\pi} \sum_{j=j_{w,1}(i)}^{j_{w,2}(i)} \alpha_{i,j}\delta z_{D,j}, \tag{4.160}$$

It can be also transformed to the real domain by the Stehfest algorithm.

The effect of the wellbore storage can be included for the pressure and the fractional flow rate using Eqs. 3.61 and 3.63.

4.3 Results and Discussions

4.3.1 Model Validation and Grid Refinement

The model developed in the previous section was validated by comparing with results of numerical simulations. The size of the simulation grid is 60 × 31 × 21 (x, y, and z). The

Table 4.2: Common reservoir, fluid, and well data for all cases.

Production rate, STB/D	1,000
Initial reservoir pressure, psi	3,000
Wellbore radius, ft	0.33
Formation volume factor	1.2
Viscosity, cp	0.8
Wellbore storage constant, bbl/psi	0.0
Porosity, fraction	0.2
Total compressibility, 1/psi	1E-5
Damage skin	0

numerical simulation model was checked by examining sensitivities to the grid size and initial timesteps. Common reservoir and fluid data used throughout this chapter (if not specified) are shown in Table 4.2.

Homogeneous and Isotropic Reservoir Case

The first example is a homogeneous and isotropic reservoir case. The reservoir and well data for this example are shown in Table 4.3. As discussed in Section 3.2.1, grid refinement is also necessary for this horizontal well case. Fig. 4.9 shows the effect of the grid refinement on the pressure response. The gridding of the reservoir model for each case is shown in Fig. 4.10. In order to calculate the pressure response effectively, only a quarter part of the whole reservoir is included in the model since the model is symmetric on both the $x-y$ and the $y-z$ planes.

In Case 1, humps of the pressure drawdown and pressure derivative curves can be observed at middle times. These humps do not appear in Case 2 and Case 3 where the grid refinement is used. The grid refinement of the well block (Case 3) has a minor effect on the pressure response, since Case 2 and Case 3 are almost identical. It is important to check the effect of the grid refinement to obtain reasonably accurate results.

The comparison between Case 3 and the numerical simulation is shown in Fig. 4.11.

Table 4.3: Reservoir and well data for the homogeneous reservoir case and the damage skin case.

Well length, ft	500
Reservoir thickness, ft	100
Distance of the well from the reservoir bottom, ft	50
Reservoir width, ft	2,500
Permeability in the x direction, md	100
Permeability in the y direction, md	100
Permeability in the z direction, md	100

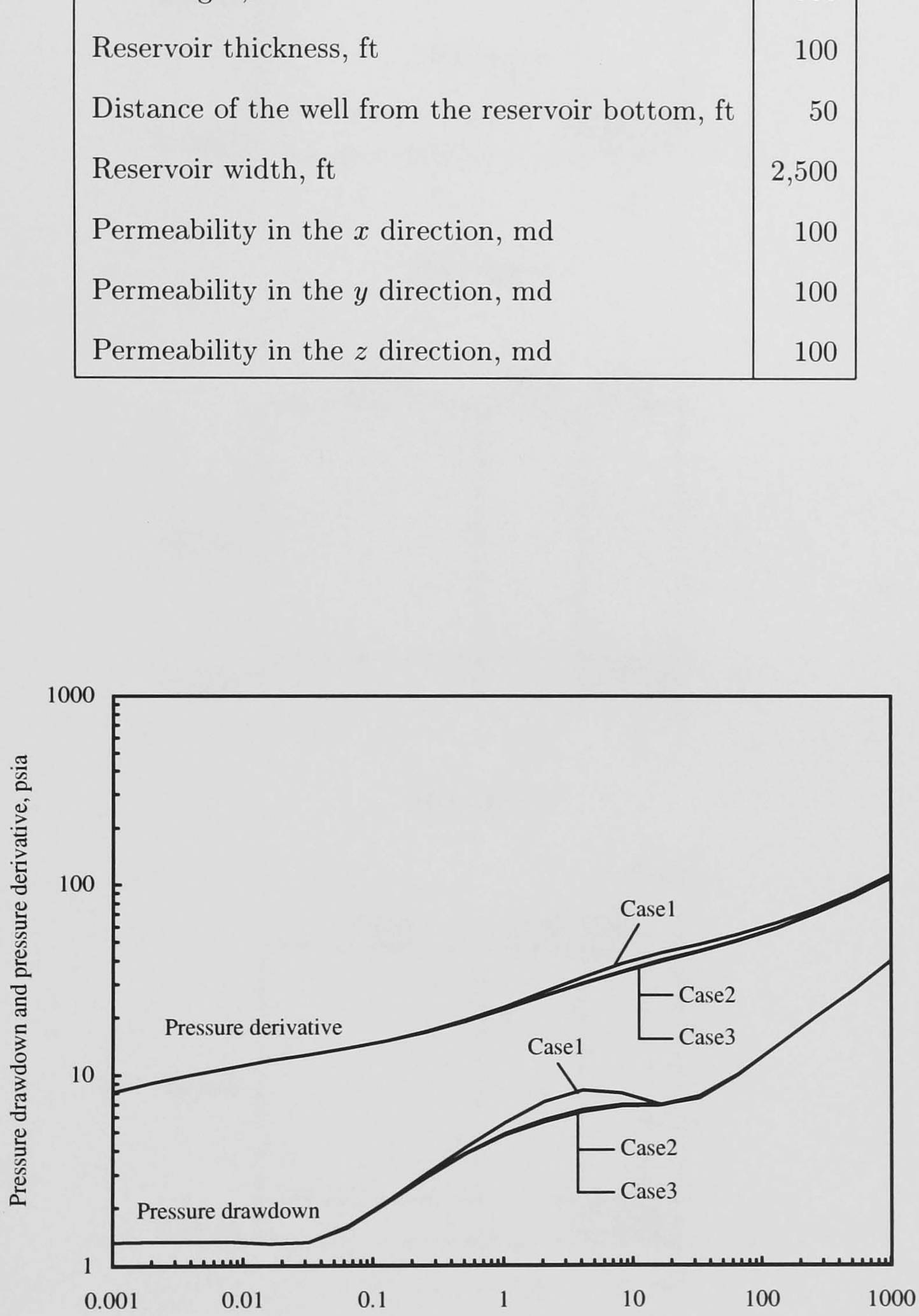
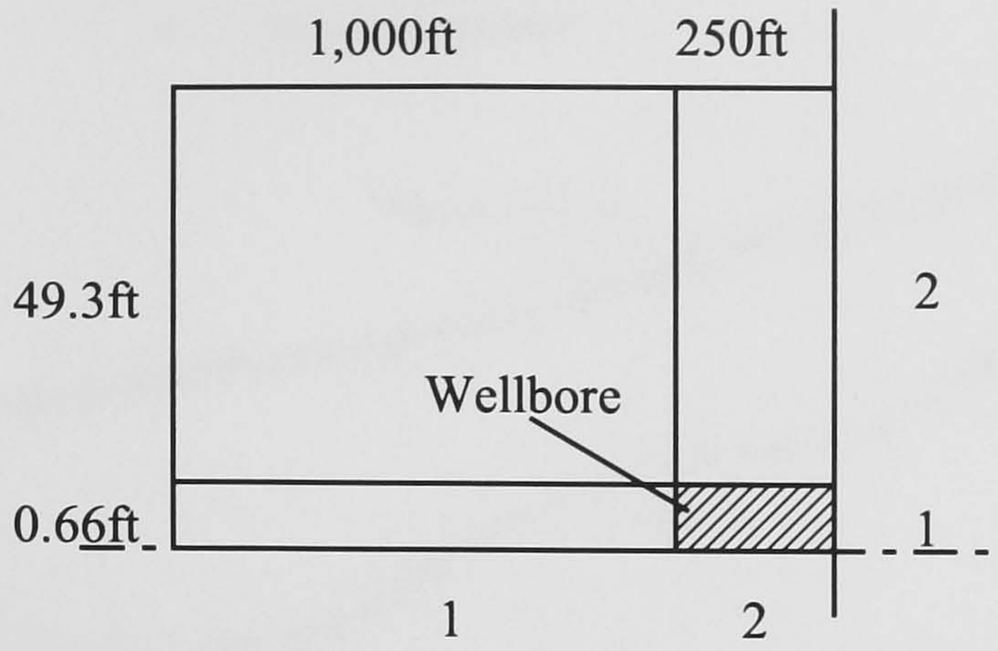
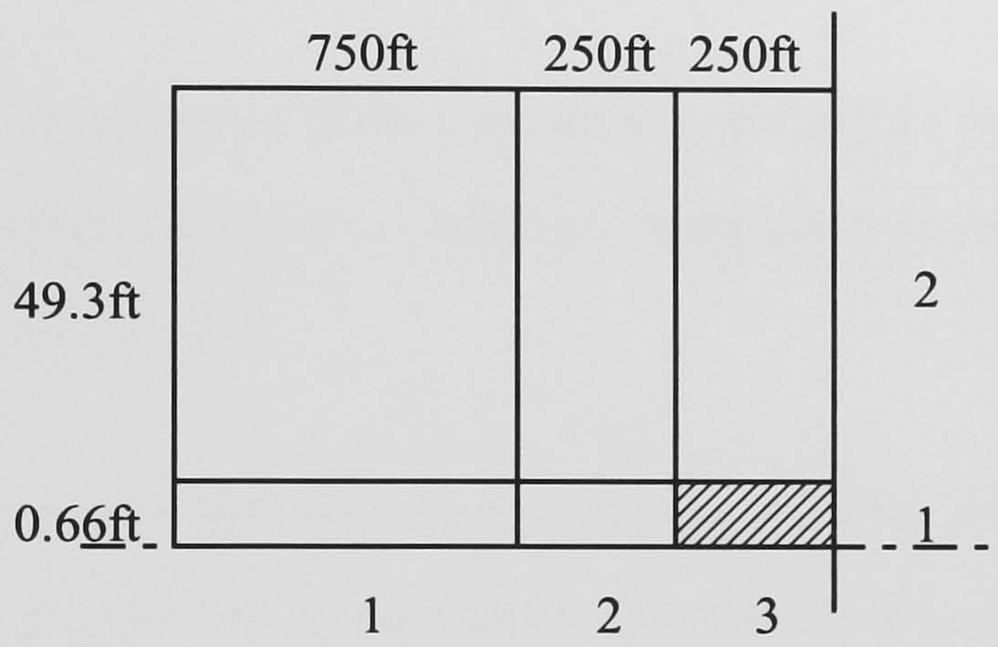


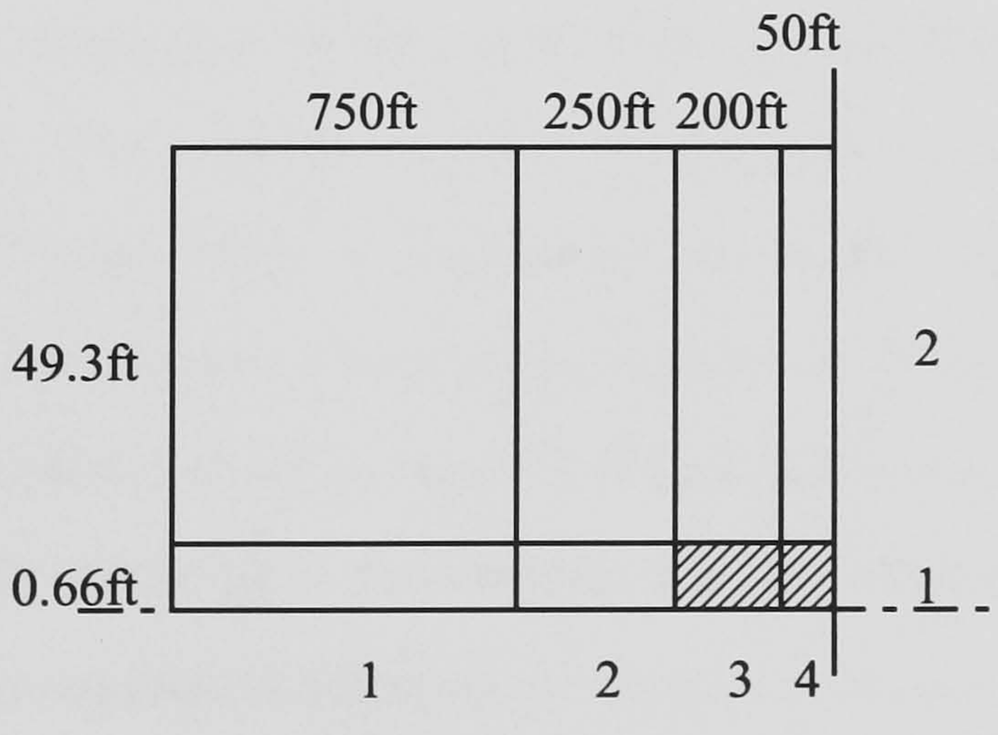
Figure 4.9: Effect of the grid refinement on the pressure response.



(a) Case 1



(b) Case 2



(c) Case 3

Figure 4.10: Model configurations for the grid refinement case.

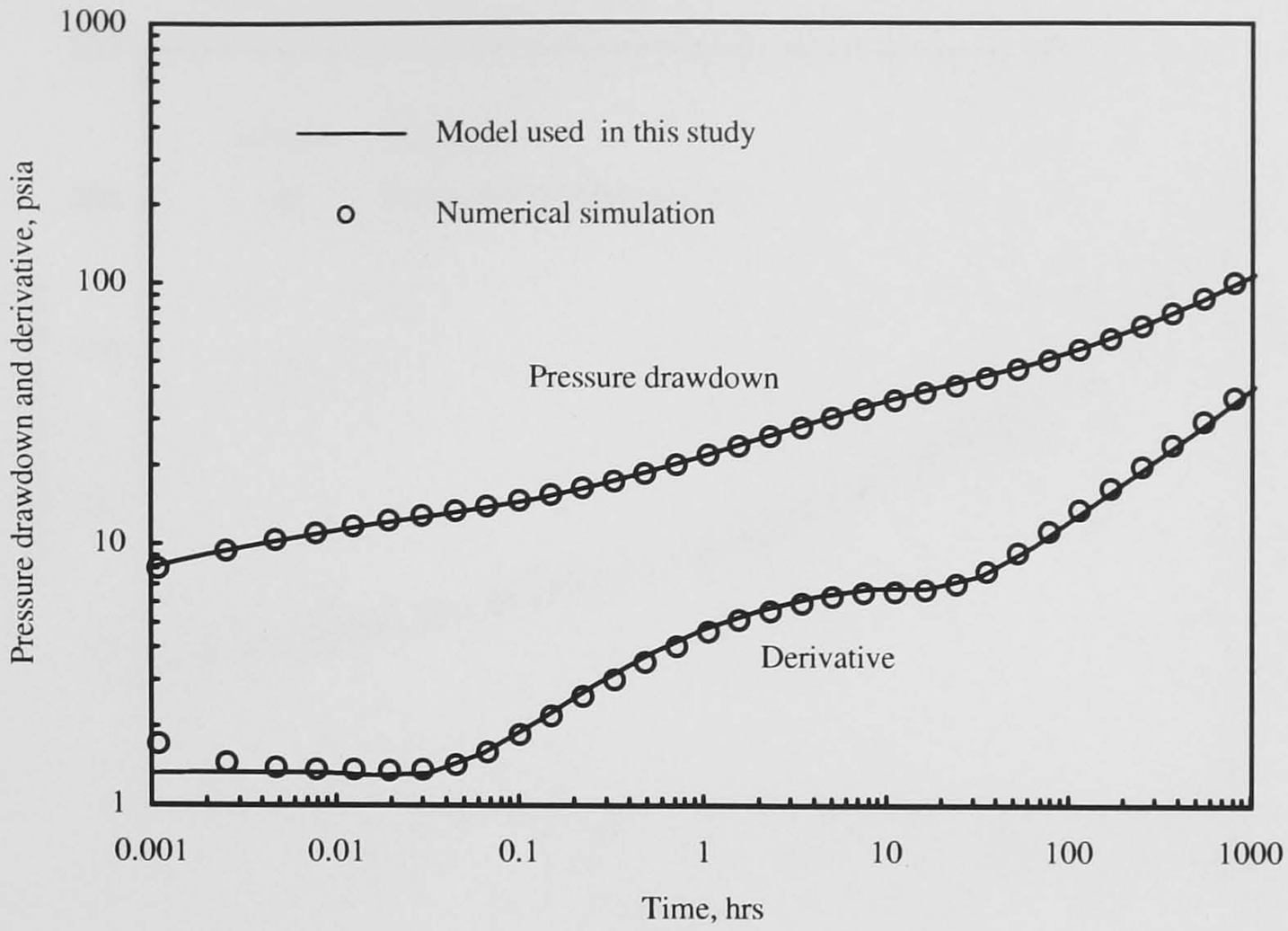


Figure 4.11: Comparison between the model used in this study and the numerical simulation. Isotropic homogeneous linearly elongated reservoir case.

The two results are in good agreement except very early times. The disagreement at the early times is due to the grid block size being too large in the numerical simulation to capture the pressure transient for such a short time.

Fig. 4.12 shows a comparison of this model with the published result by Goode and Thambynayagam [42]. The reservoir model is a homogeneous, isotropic, linear system with a horizontal well in the centre. A good match can be also observed in this example.

The computation time becomes large as the number of the grid blocks increases. In the homogeneous case shown above, it took 15, 40, and 230 seconds for Case 1, 2, and 3, respectively, using SGI Power Onyx Workstation with R10000 Processor. On the other hand, 245 seconds are required to calculate the pressure response in the numerical simulation for the same reservoir. Although the computation of Case 3 is not so fast as compared with the numerical simulation case, reservoir modelling is much easier. This is considerable benefit of the analytical model. The ease of the reservoir modelling becomes much important as faster computer become available.

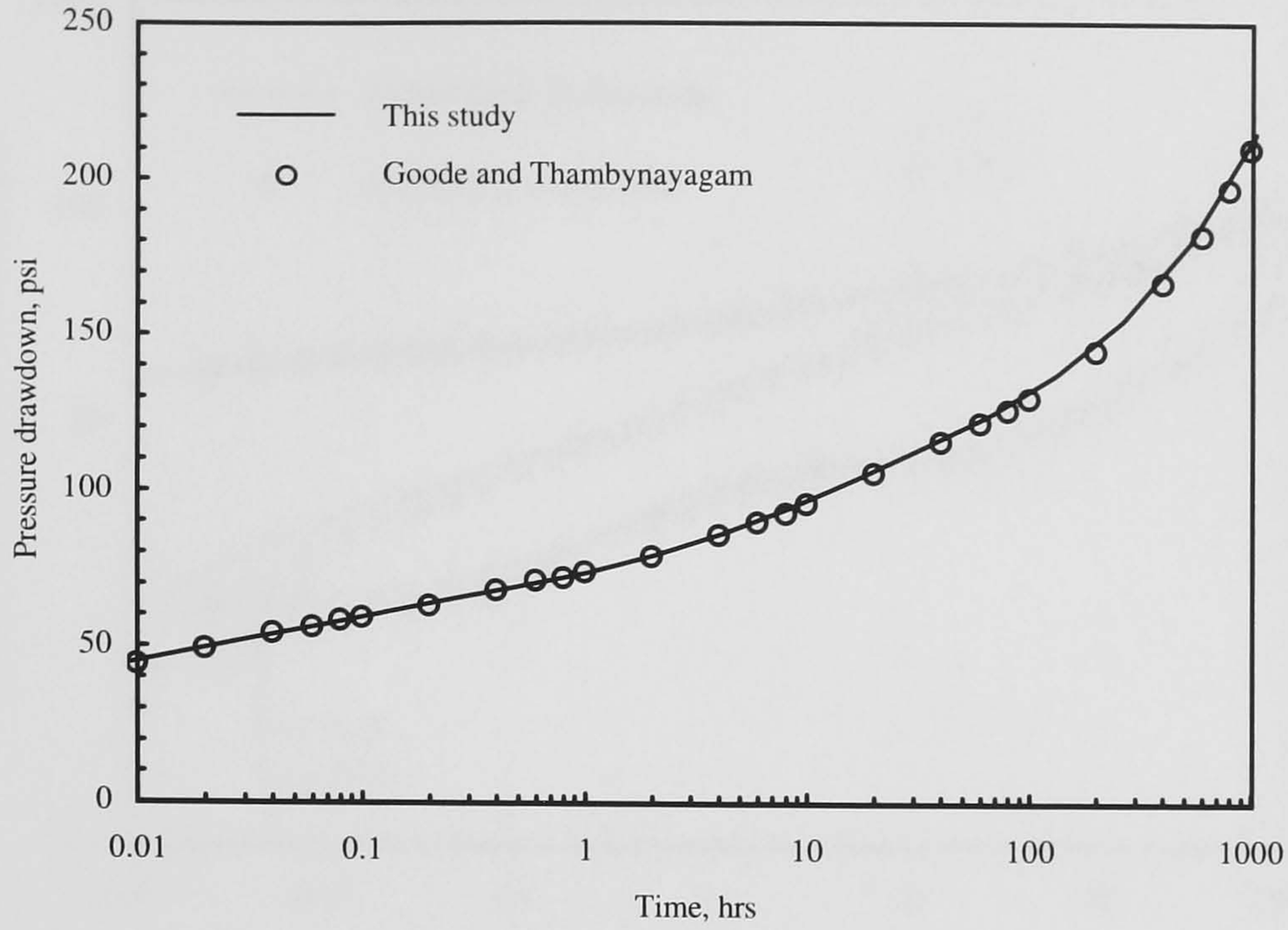


Figure 4.12: Comparison between the model used in this study and the published result by Goode and Thambynayagam [42].

The large part of the computation time is spent for the matrix calculation for the system of equations from Eq. 4.147 to Eq. 4.152 where the size of the matrix is determined by the number of the grid blocks. A multiple precision arithmetic program developed by Fukuda [37] was implemented in the computer program (especially it is often used in the matrix calculation part) to deal with overflow or underflow problem. As Fukuda admitted in his paper, his code is slow as compared with other software packages. The computation time will be much reduced by adopting a more efficient code.

Extension of the model by implementation of automatic type-curve match algorithms is relatively easy. This is another benefit of this model compared to the numerical simulation.

Damaged or Stimulated Well Case

The next example describes a damaged or stimulated well case. Basic parameters are the same as the previous example, however damage skin is set to -3 or 3. The results are shown in Fig. 4.13. Again, good agreements can be seen in each case except very early times. The reason of this disagreement is the same as the previous example. For the stimulation

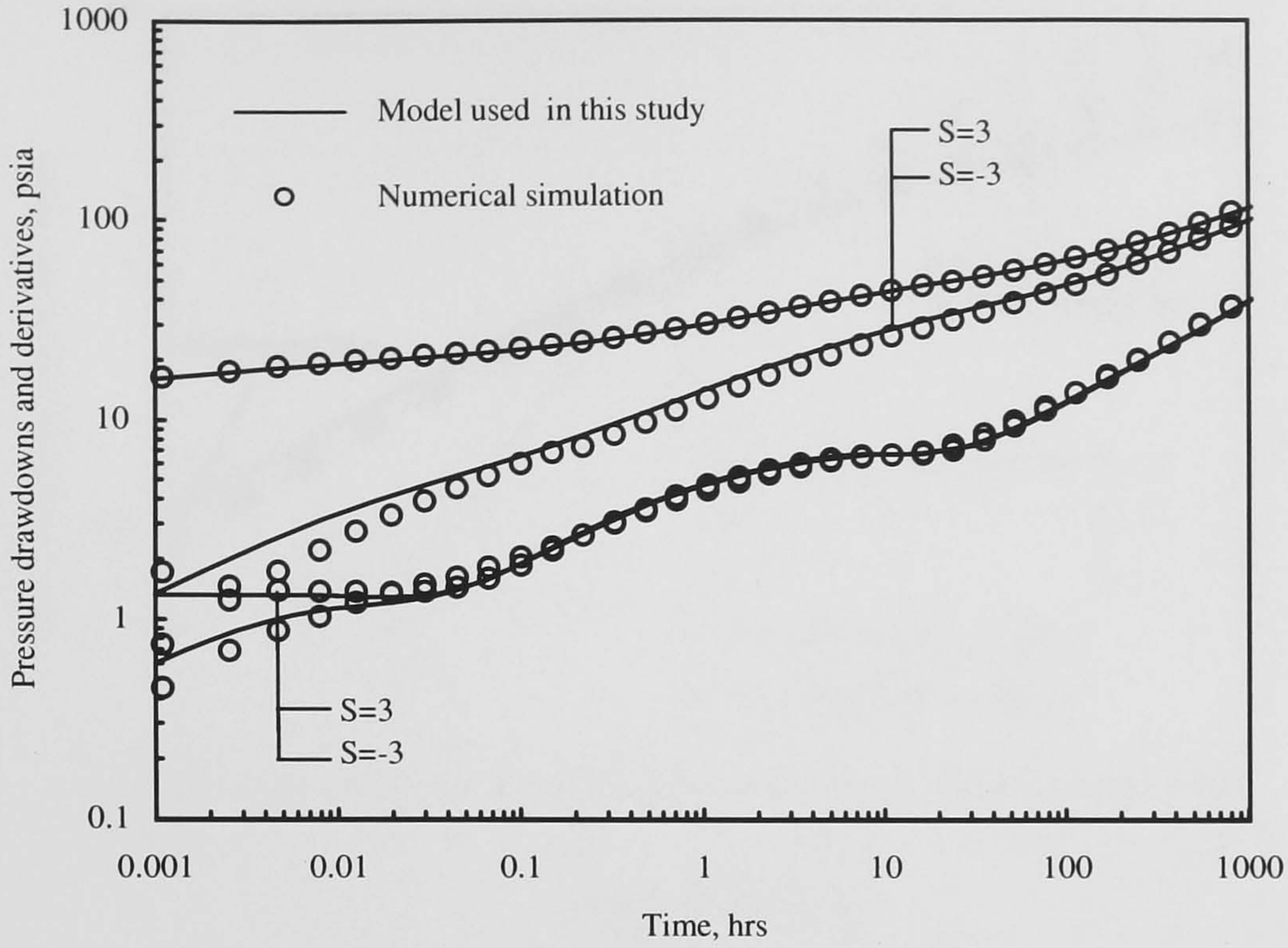


Figure 4.13: Comparison between the model used in this study and the numerical simulation. Isotropic homogeneous linearly elongated reservoir cases with damage skins -3 and 3.

case, the first straight line which shows the early-time radial flow period disappears. This is because the linear flow develops due to the large effective wellbore radius as mentioned by Streltsova [115].

Early-time Linear Flow

In the previous example, the deviation from the early-time radial flow curve was observed in the stimulated well case. Fig. 4.14 compares the results of stimulated horizontal well models with that of the finite radius solution or the line-source solution of a vertical well in a homogeneous and isotropic reservoir (see Ref. [81]). The pressure response of a horizontal well should show exactly the same behaviour as that of a vertical well at early times before the effect of a top or a bottom reservoir boundary, or the tip effect of the horizontal well is felt. Reservoir and fluid data are the same as in Tables. 4.2 and 4.3 except that the well has a varying negative skin factor.

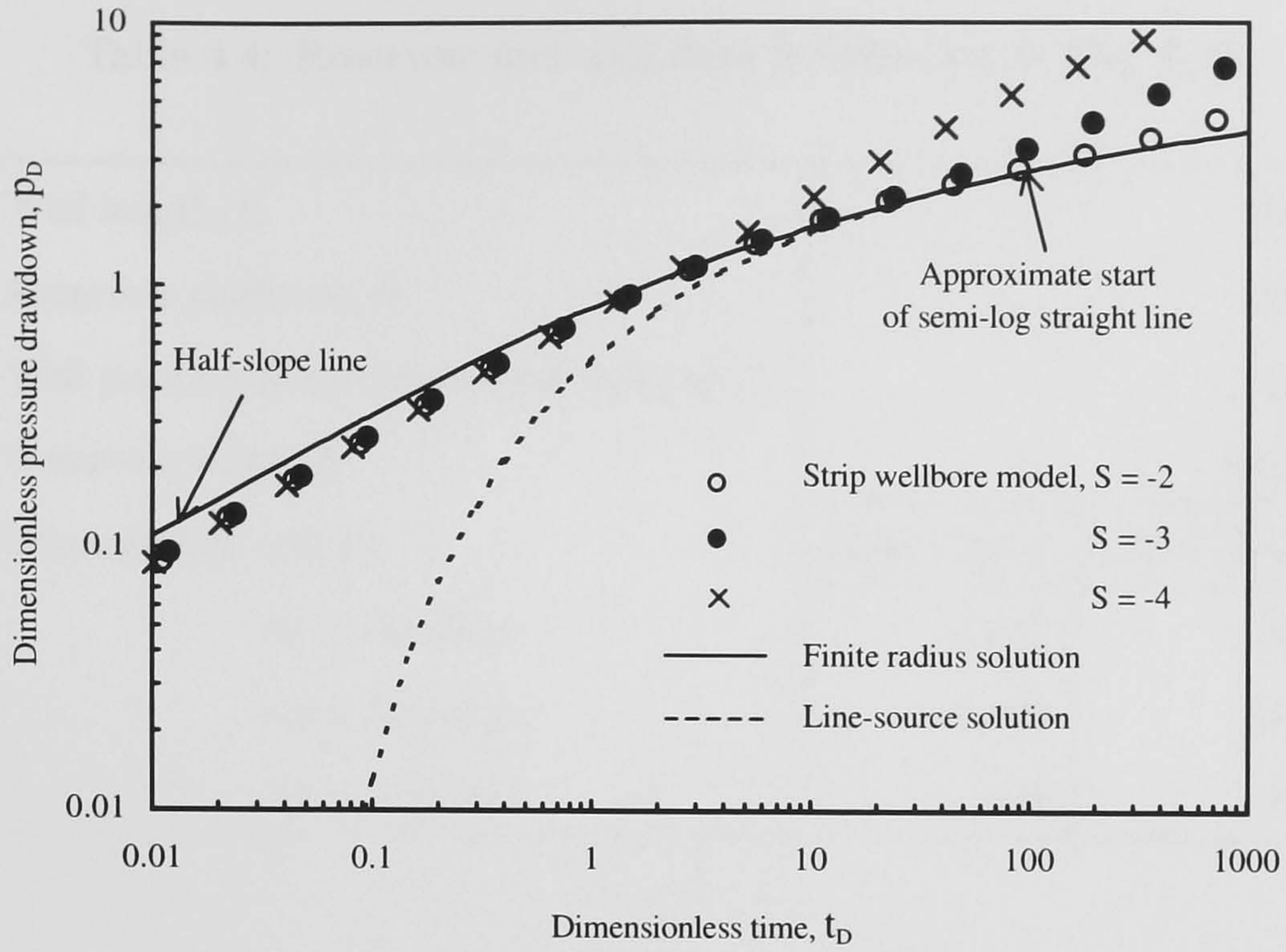


Figure 4.14: Early-time behaviour of a vertical well with finite wellbore radius or infinitesimal wellbore radius [81] and of a horizontal well with a varying negative skin factor.

For the vertical well case, dimensionless pressure drawdown,

$$p_D = \frac{2\pi kh(p_{in} - p)}{qB\mu}, \tag{4.161}$$

and dimensionless time,

$$t_D = \frac{kt}{\phi\mu c_t {r_w}^2}, \tag{4.162}$$

both in Darcy units were used to plot the figure, where h is the thickness of the reservoir, and r_w is the radius of the vertical well. For the horizontal well case, h is replaced by the wellbore length L_w, and r_w is replaced by the effective wellbore radius ${r_w}'$ expressed as

$${r_w}' = r_w e^{-S}, \tag{4.163}$$

where r_w is the wellbore radius of a horizontal well, and S is a skin factor.

It can be observed that the difference between the finite radius solution and the line-source solution becomes large as the dimensionless time decreases. At very early times, the finite radius solution shows a half-slope line as reported by Streltsova [115]. The results of

Table 4.4: Reservoir and well data for the case in Fig. 4.15.

Well length, ft		500
Reservoir thickness, ft		100
Well position from the reservoir bottom, ft		50
Reservoir width, ft		2,500
Permeability, md, in	High k Zone	Low k Zone
the x direction	1,000	100
the y direction	1,000	100
the z direction	50	20

the strip wellbore model compares well with the finite radius solution before the effect of the reservoir top and the bottom boundaries are felt. Thus, it is confirmed that the strip wellbore model can simulate the early-time linear flow, the early-time radial flow, and a transition flow between them.

The small difference at early times is because the area of the strip wellbore is $4/\pi$ times larger than the surface area of the radial wellbore. Thus, the pressure drop for the strip wellbore model is always lower than that for the exact finite radius solution at early times when the early-time linear flow develops.

In this case, if the skin factor is below -3, the straight line of the pressure derivative in the early-time radial flow period cannot be observed since the reservoir boundaries are felt before the semi-log straight line of pressure drawdown in the early-time radial flow period develops. This is confirmed by the pressure derivative curve in Fig. 4.13.

Heterogeneous and Anisotropic Reservoir Case

The last example is a heterogeneous and anisotropic reservoir case. Reservoir and fluid data are the same except the permeability arrangement as shown in Table 4.4. Since the permeability ratio k_z/k_y is different between the high and the low permeability zones, the wellbore strip widths are different between the two zones. The wellbore strip in the low permeability zone is divided into three as shown in Fig. 4.15. The comparison with

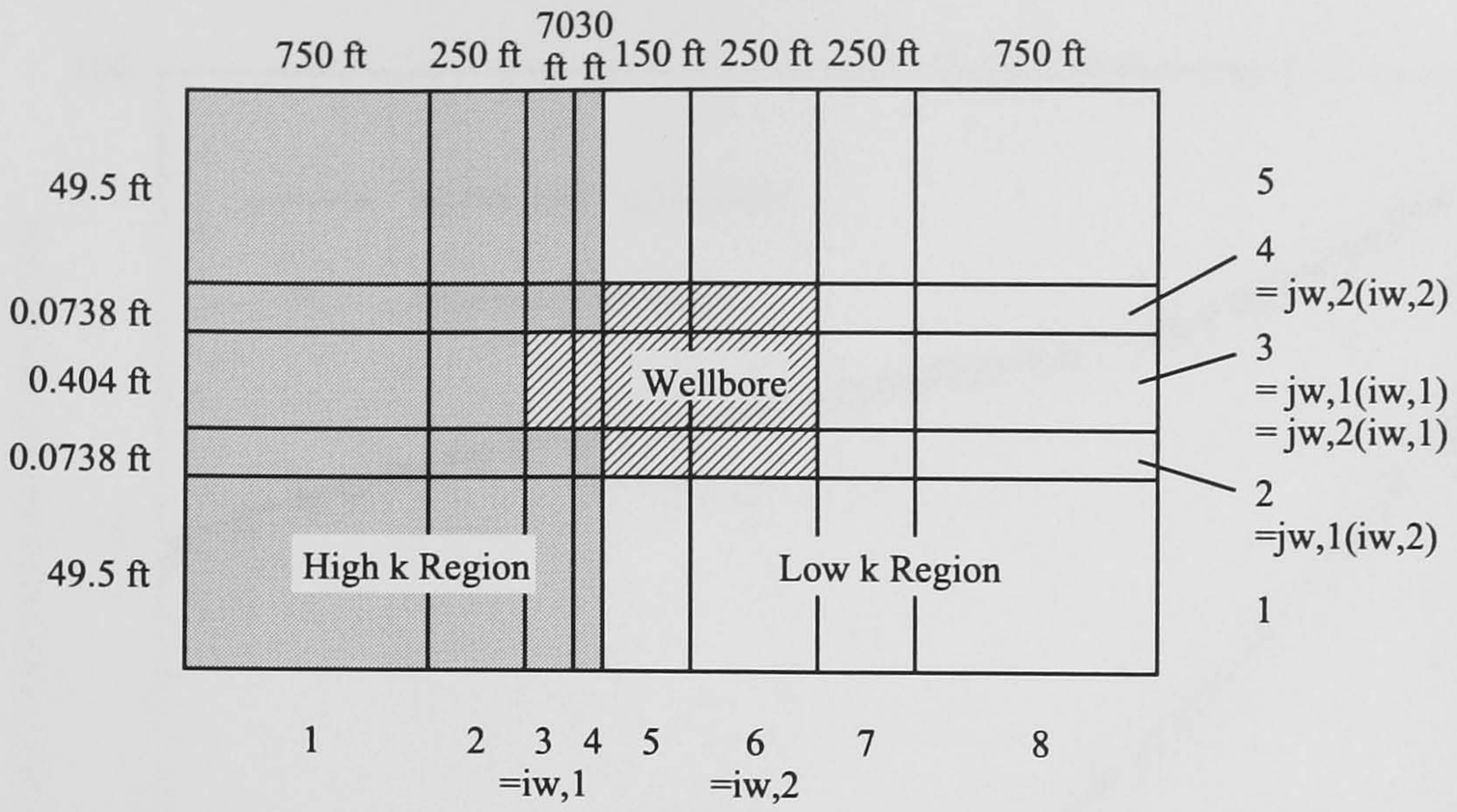


Figure 4.15: Schematic model for an anisotropic heterogeneous reservoir.

the numerical simulation is shown in Fig. 4.16. A good match can be seen again except very early times. The difference at the very early times is due to the numerical problem described before.

4.3.2 The Effect of Heterogeneities on Pressure Responses

In this section, the sensitivities to the various reservoir parameters for reservoirs with a high permeability region are examined. Although pressure response may be non-unique by itself, these example will provide useful information when well test analysis is performed by type-curve matching method.

The Size of the High Permeability Region Not Penetrated by the Wellbore

A schematic reservoir model for the first case is shown in Fig. 4.17. The reservoir and the well data are shown in Table 4.5. In this case, each zone is isotropic and isotropic permeability of the high permeability zone is 1,000 md, while that of the low permeability zone including the well is 100 md. The effects of the size of the high permeability region are shown in Fig. 4.18. A homogeneous reservoir case is also plotted as a reference.

As the high permeability zone becomes larger, the derivative curve at late times shifts downward due to the increase of the flow capacity of the reservoir. In the homogeneous

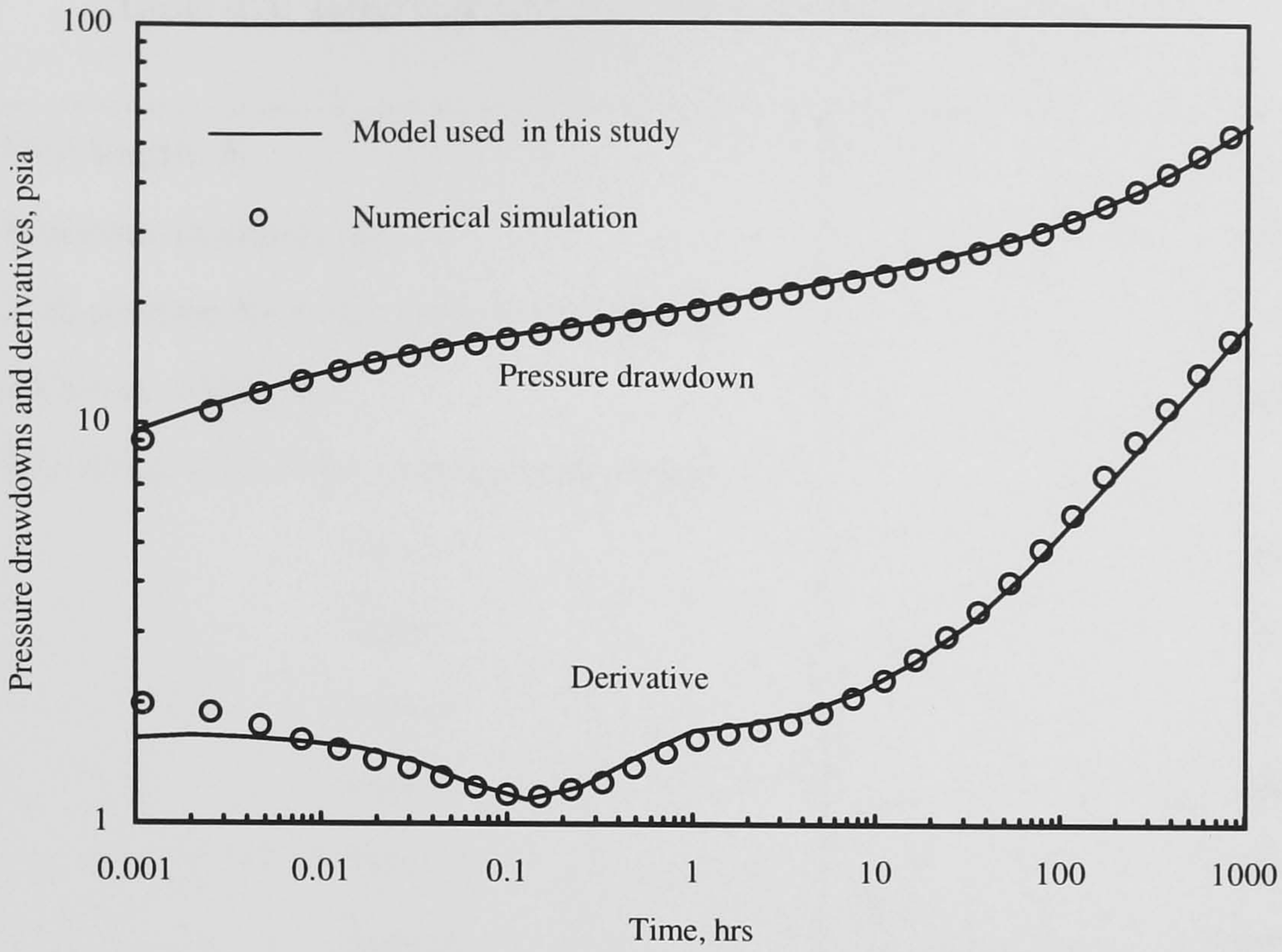


Figure 4.16: Comparison between the model used in this study and the numerical simulation. Anisotropic heterogeneous linear reservoir case.

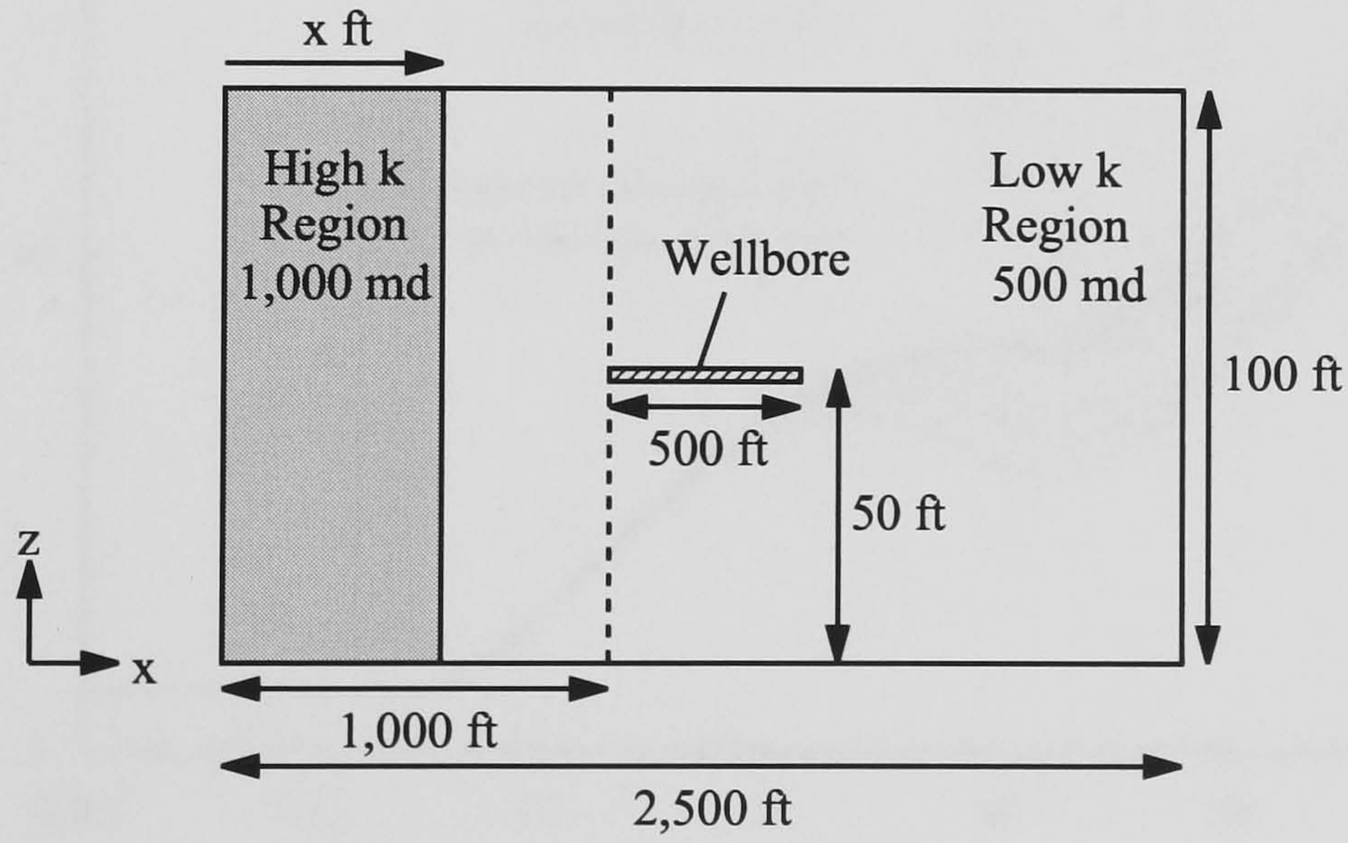


Figure 4.17: Schematic model for a heterogeneous reservoir with a high permeability region not penetrated by a well.

Table 4.5: Reservoir and well data for the case in Fig. 4.17.

Well length, ft	500	
Reservoir thickness, ft	100	
Well position from the reservoir bottom, ft	50	
Reservoir width, ft	2,500	
The width of the high permeability region, x ft		
Case 1	100	
Case 2	200	
Case 3	500	
Case 4	800	
Case 5	900	
Case 6	990	
Isotropic permeability, md	High k Zone	Low k Zone
	1,000	100

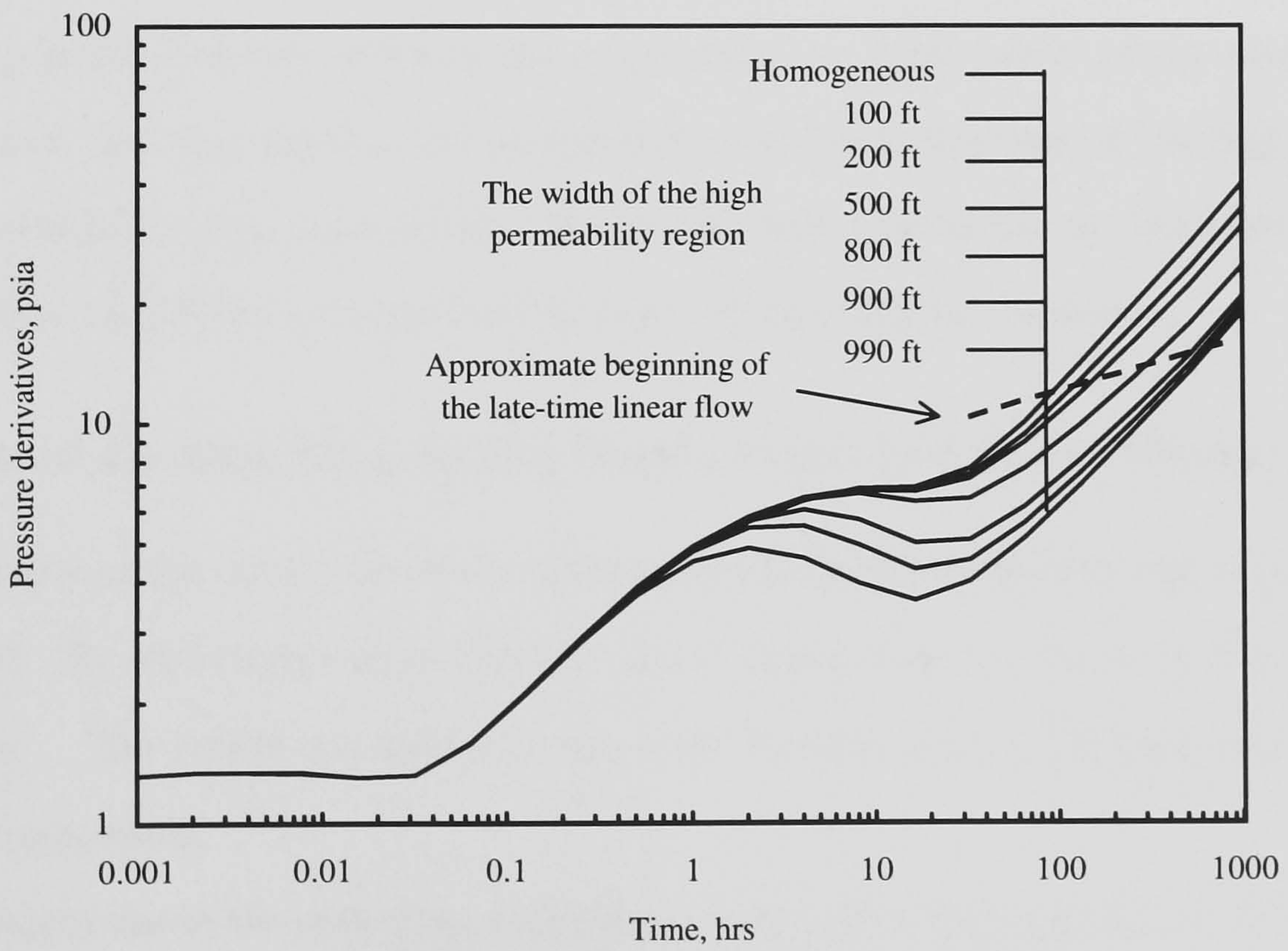


Figure 4.18: Effect of the size of the high permeability region not penetrated by the wellbore.

case, the four flow periods (the early-time radial flow, the intermediate-time linear flow, the pseudo-radial flow, and the late-time linear flow) may be observed as shown in Section 4.1.2. However, in this heterogeneous case, if the width of the high permeability zone is over 500 ft, the pseudo-radial flow period can be no longer observed. In the case where the width of the high permeability region is 990 ft, the diversion from the curve of the homogeneous reservoir case starts at the end of the intermediate-time linear flow period. If the permeability in the x direction is larger, the diversion will occur much earlier. Since the early-time radial flow can be still observed, the product $k_y k_z$ and the damage skin in the low permeability zone can be calculated by Eqs. 4.6 and 4.7.

At late times, half slope lines can be observed in all cases. If the permeability is calculated using the slope of the line on a pressure versus square-root time plot from Eq. 4.29, the averaged permeability weighted with the width of each zone $\overline{k_y}$ is obtained, i.e.,

$$\overline{k_y} = \frac{w_h k_{y,h} + (w_x - w_h) k_{y,l}}{w_x}, \tag{4.164}$$

where w_x is the reservoir width in the x direction, w_h is the width of the high permeability zone, and $k_{y,h}$ and $k_{y,l}$ are permeabilities in the y direction of the high and the low permeability zones, respectively. This is the same observation as that mentioned by Bourgeois et al. [15] for a vertical well in linear three-composite reservoirs.

The Size of the High Permeability Region Penetrated by the Wellbore

The second example shows the effects of the size of the high permeability region penetrated by a well. The schematic reservoir model and its parameters are shown in Fig. 4.19 and Table 4.6. The results are shown in Fig. 4.20, together with a homogeneous reservoir case for comparison.

A straight line in the early-time radial flow period and a half-slope line in the late-time linear flow period can be observed in each case. However, the intermediate-time linear flow and the pseudo-radial flow cannot be observed. It can be found that the averaged

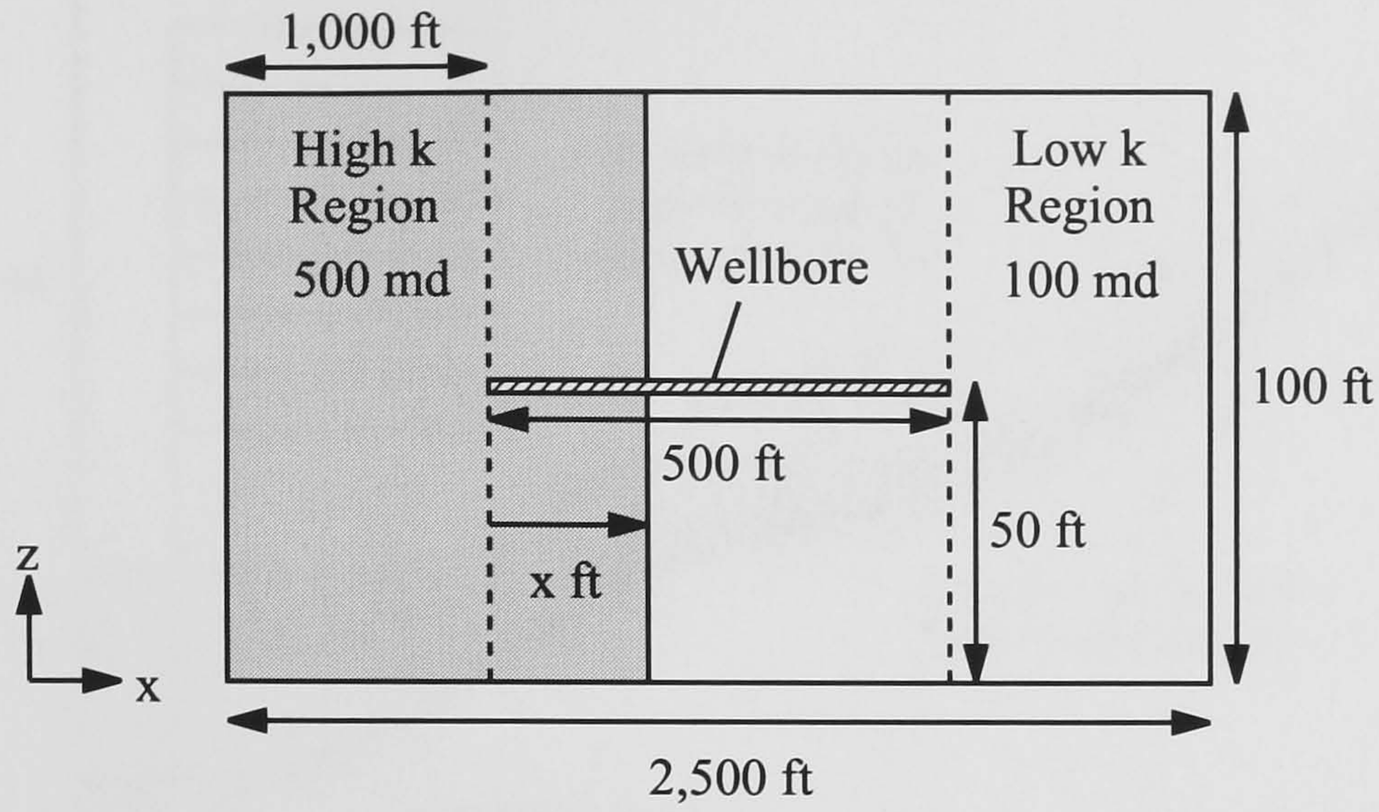


Figure 4.19: Schematic model for a heterogeneous reservoir with a high permeability region penetrated by a well.

Table 4.6: Reservoir and well data for the case in Fig. 4.19.

Well length, ft	500	
Reservoir thickness, ft	100	
Well position from the reservoir bottom, ft	50	
Reservoir width, ft	2,500	
The width of the high permeability region penetrated by the wellbore, x ft		
Case 1	5	
Case 2	50	
Case 3	100	
Case 4	250	
Case 5	400	
Case 6	450	
Case 7	495	
Isotropic permeability, md	High k Zone	Low k Zone
	500	100

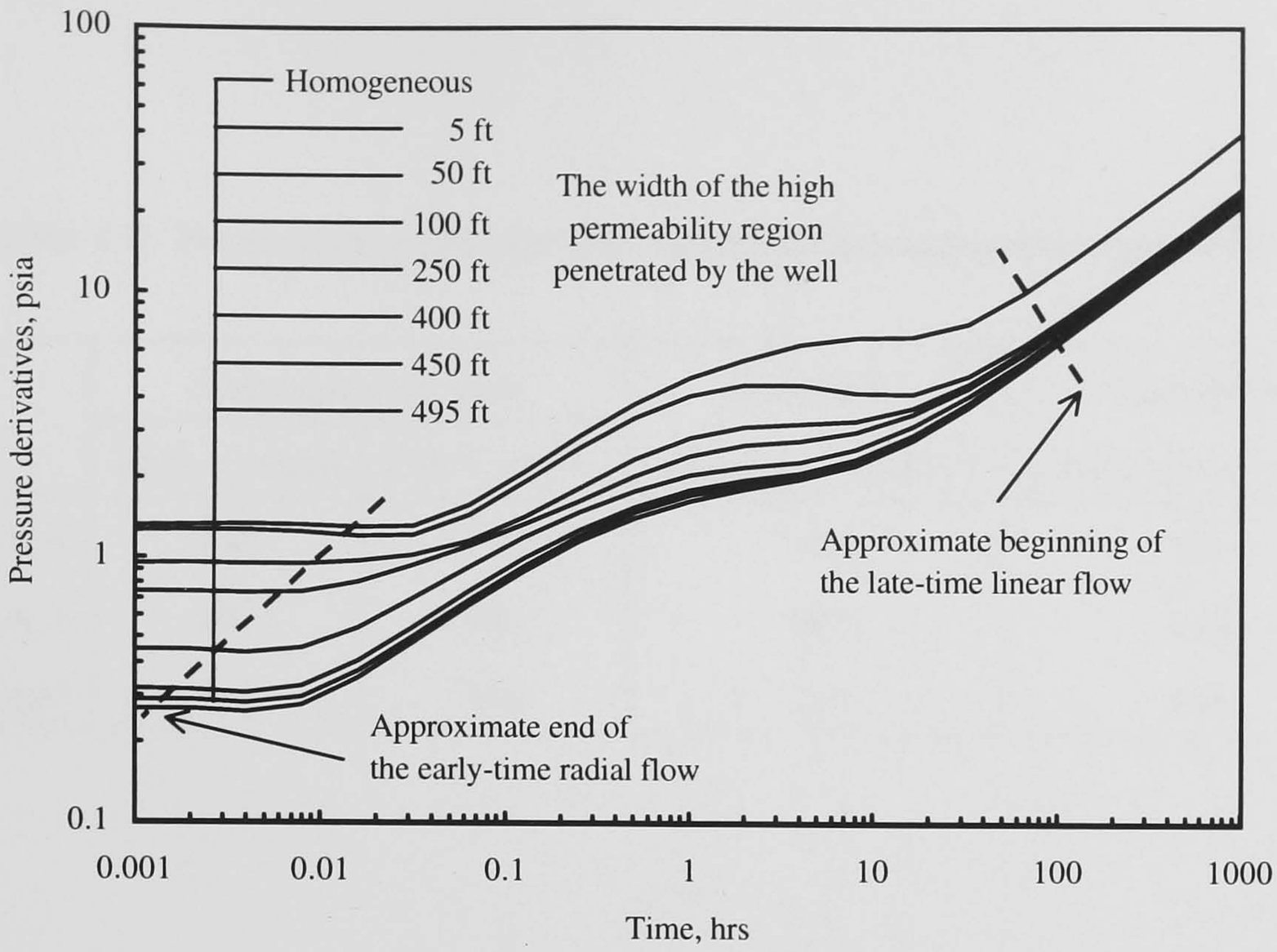


Figure 4.20: Effect of the size of the high permeability region penetrated by the wellbore.

permeability $\overline{\sqrt{k_y k_z}}_w$ expressed as

$$\overline{\sqrt{k_y k_z}}_w = \frac{l_h \sqrt{k_{y,h} k_{z,h}} + (L_w - l_h)\sqrt{k_{y,l} k_{z,l}}}{L_w} \tag{4.165}$$

can be obtained if the data in the early-time radial flow period is analysed from Eq. 4.6. In Eq. 4.165, l_h is the length of the well penetrating the high permeability zone, $k_{y,h}$ and $k_{z,h}$ are the directional permeabilities of the high permeability zone, and $k_{y,l}$ and $k_{z,l}$ are those of the low permeability zone, respectively.

Equivalent homogeneous reservoir responses using the averaged permeability for the case 5 in Table. 4.6 are shown in Fig. 4.21. Values of permeabilities are shown in Table 4.7. The averaged permeability calculated from Eq. 4.165 is 420 md, which is allocated to the permeabilities in the y and the z directions. The permeabilities in the x direction for the equivalent cases 1 and 2 are set to 100 md and 500 md.

In Fig. 4.21, it can be observed that both the pressure and the pressure derivative responses are identical in the early-time flow period, and k_x has nothing to do with the responses. This fact implies the flow is essentially vertical and the crossflow between the

Table 4.7: Permeability data for the equivalent homogeneous reservoir cases.

	Heterogeneous case		Equivalent homogeneous case 1	Equivalent homogeneous case 2
	High k region	Low k region		
k_x, md	500	100	100	500
k_y, md	500	100	420	420
k_z, md	500	100	420	420

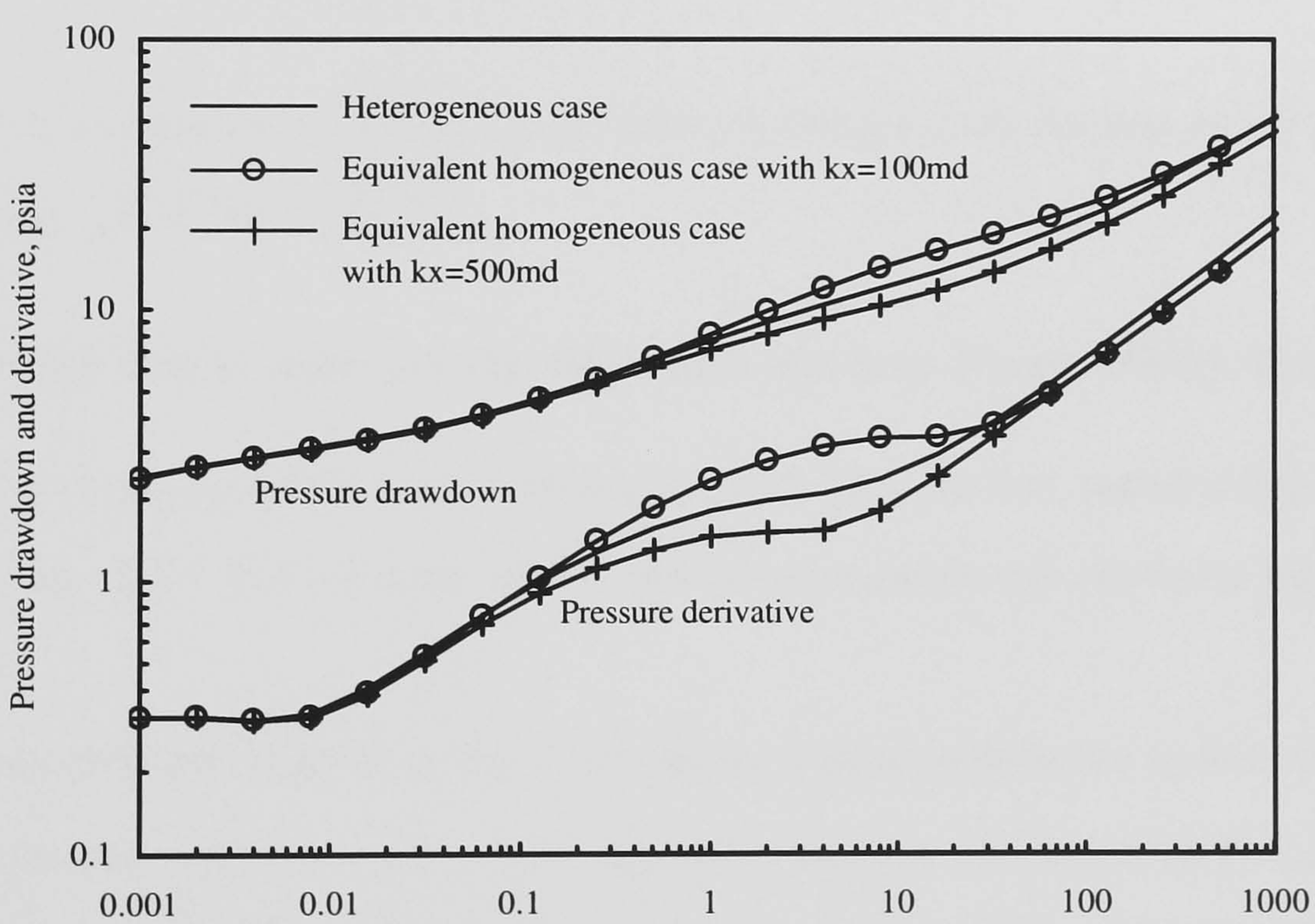


Figure 4.21: Equivalent homogeneous reservoir pressure responses.

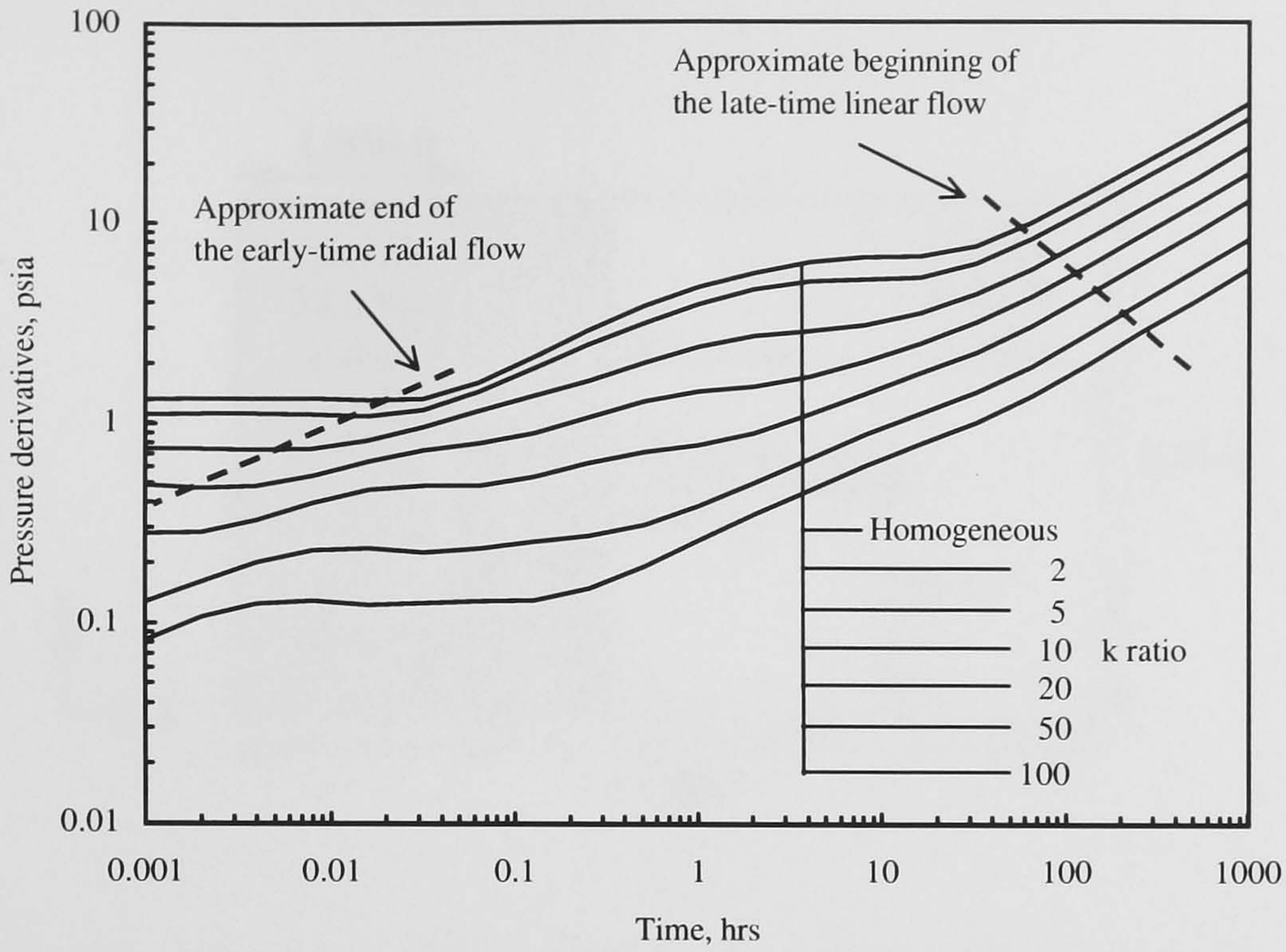


Figure 4.22: Effect of the permeability ratio between the high and the low permeability regions.

two zones is not dominant. Thus, it is possible to calculate the damage skin from Eq. 4.7 by replacing $\sqrt{k_y k_z}$ with $\overline{\sqrt{k_y k_z}}_w$.

Permeability Ratio Between the High and the Low Permeability Regions

The effects of permeability ratio between the high and the low permeability zones are shown in Fig. 4.22. The reservoir model and the parameters are shown in Fig. 4.23 and Table 4.8.

The averaged permeability of Eq. 4.165 can be estimated from the level of the pressure derivative in the early time flow period. From the curve in the late time flow period, the averaged permeability $\overline{k_y}$ of Eq. 4.164 can be obtained from Eq. 4.29. The intermediate time period between the early-time and the late-time periods shows a complicated response since crossflow from the lower permeability zone to the higher permeability zone occurs in addition to the intermediate-time linear flow or the pseudo-radial flow for each zone. When the permeability ratio becomes large, the early-time flow ends earlier and the straight line

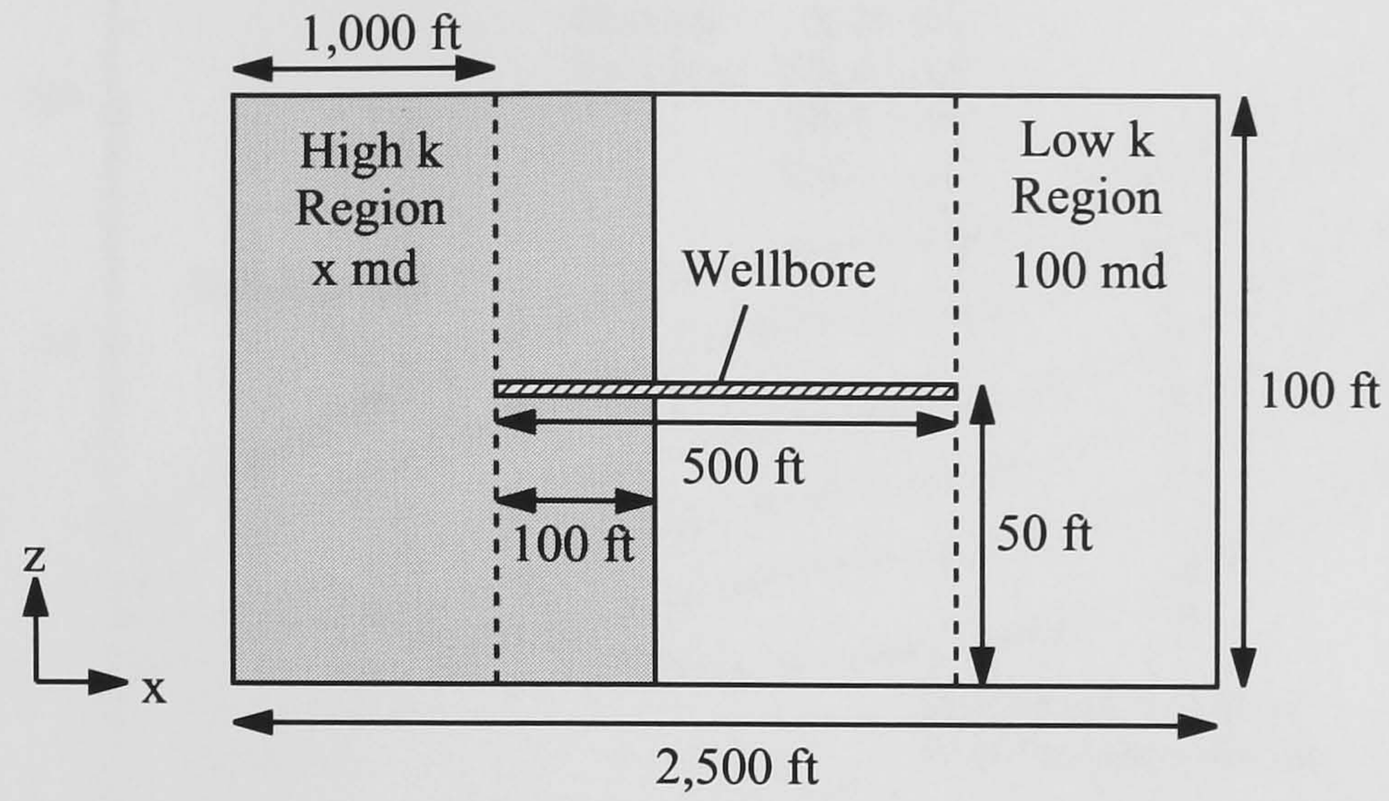


Figure 4.23: Schematic model with the varying permeability region penetrated by a well.

Table 4.8: Reservoir and well data for the case in Fig. 4.23.

Well length, ft			500
Reservoir thickness, ft			100
Well position from the reservoir bottom, ft			50
Reservoir width, ft			2,500
Isotropic permeability, md		High k Zone	Low k Zone
k ratio	2	200	100
k ratio	5	500	100
k ratio	10	1,000	100
k ratio	20	2,000	100
k ratio	50	5,000	100
k ratio	100	10,000	100

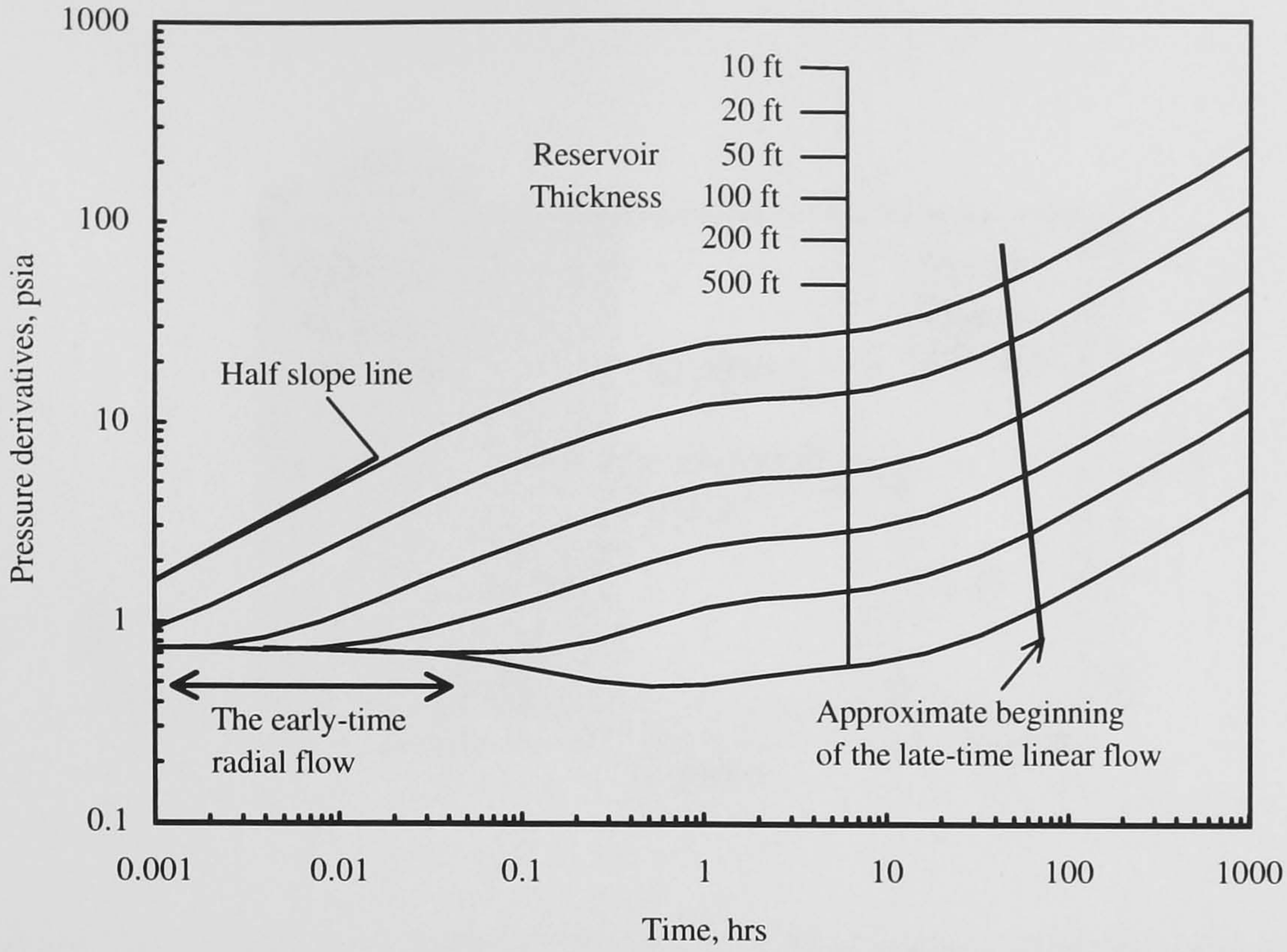


Figure 4.24: Effect of the reservoir thickness in the reservoir with a permeability contrast.

of the pressure derivative becomes undetectable. The straight line of the 100 permeability ratio case in the intermediate time period should not be confused as that in the early-time radial flow period, which may cause a wrong estimation of the averaged permeability $\overline{\sqrt{k_y k_z}}_w$.

Reservoir Thickness

The next example shows the effects of the reservoir thickness. In each case, the wellbore is located at the centre of the reservoir. The sensitivity of the reservoir thickness on the pressure responses are shown in Fig. 4.24. The reservoir model and parameters are shown in Fig. 4.25 and Table 4.9.

The duration of the early-time flow period becomes longer as the reservoir thickness increases since the time for the pressure disturbance to reach the top and the bottom no-flow boundaries becomes longer. For cases in which the reservoir thickness is small, the intermediate-time linear flow which is represented by a half slope straight line on the

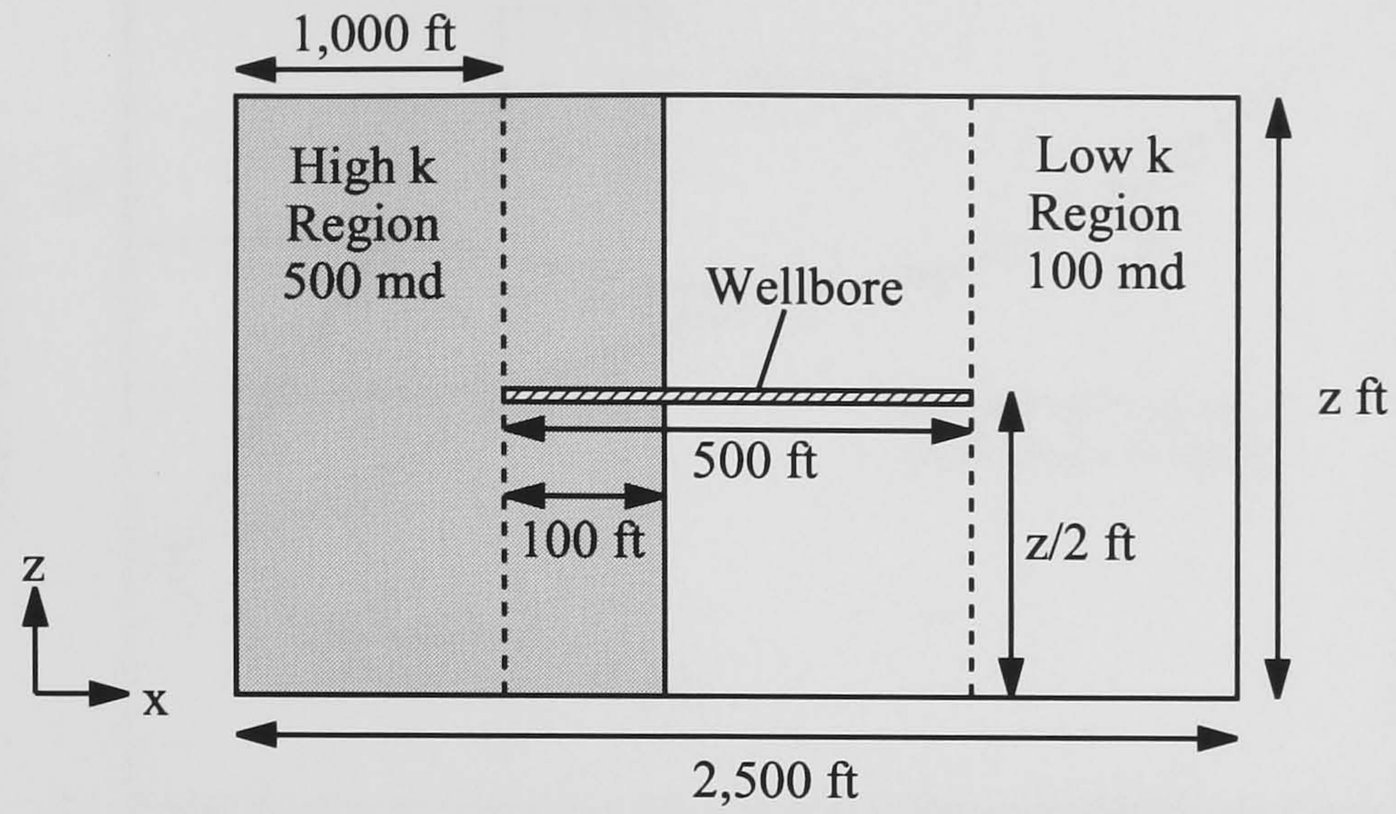


Figure 4.25: Schematic reservoir model with the varying reservoir thickness.

Table 4.9: Reservoir and well data for the case in Fig. 4.25.

Well length, ft	500	
Reservoir width, ft	2,500	
Reservoir thickness, z ft		
Case 1	10	
Case 2	20	
Case 3	50	
Case 4	100	
Case 5	200	
Case 6	500	
Well position from the reservoir bottom, ft	$z/2$	
Isotropic permeability, md	High k Zone	Low k Zone
	500	100

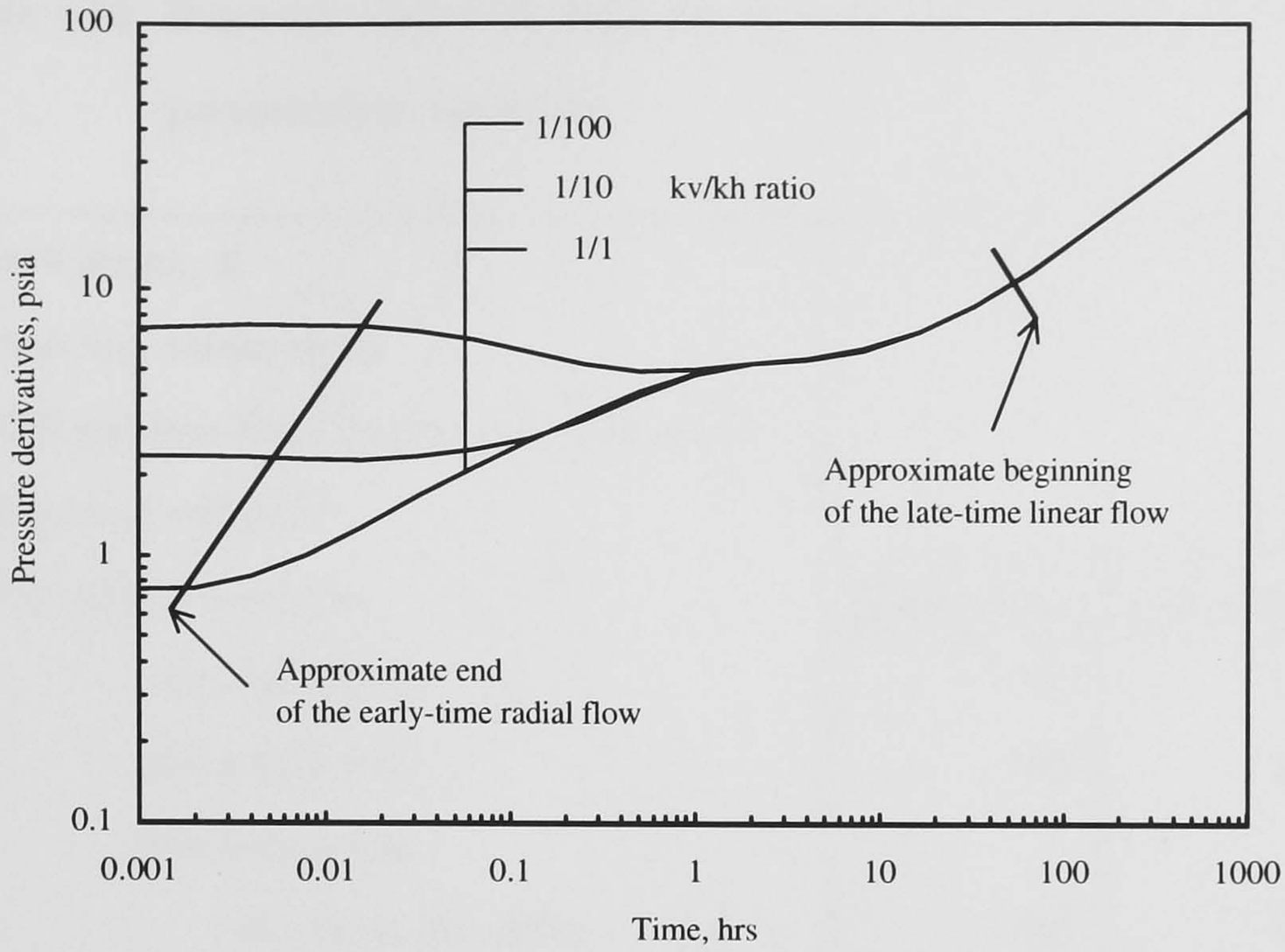


Figure 4.26: Effect of the vertical to horizontal permeability ratio in the reservoir with a permeability contrast.

log-log plot can be observed. However, it disappears for large thickness cases because the effect of the pseudo-radial flow combined with the effect of the crossflow from the lower permeability region to the higher permeability region becomes dominant before the pressure disturbance reaches the top and the bottom reservoir boundaries. Finally, the late-time linear flow period can be seen in each case.

Vertical to Horizontal Permeability Ratio

Changing the vertical to horizontal permeability ratio has the similar effect on pressure responses as changing the reservoir thickness (Fig. 4.26). The reservoir model is the same as that in Fig. 4.25 except for the reservoir thickness fixed at 50 ft and the varying permeabilities in the z direction. The other parameters are shown in Table 4.10.

If the pressure in the early-time flow period is analysed, the averaged permeability given by Eq. 4.165 can be obtained. In the cases where the k_v/k_h ratio is 1/1 or 1/10, pressure derivative increases after the early-time flow regime ends since the pressure disturbance

Table 4.10: Reservoir and well data for the varying vertical to horizontal permeability ratio case.

Well length, ft			500
Reservoir thickness, ft			50
Well position from the reservoir bottom, ft			25
Reservoir width, ft			2,500
Permeability, md, in		High k Zone	Low k Zone
the x direction		500	100
the y direction		500	100
the z direction			
for the k_v/k_h ratio,	1/1	500	100
	1/10	50	10
	1/100	5	1

reaches the top and the bottom boundary of the reservoir. However, the case where the ratio is 1/100 shows the decrease of the pressure derivative due to the effect of the pseudo-radial flow combined with the crossflow effect between the low and the high permeability regions. This behaviour is similar to the thick reservoir case where the reservoir thickness is 500 ft in the previous example (See Fig. 4.24). In the late-time linear flow period, the derivatives for all cases become the same since the permeabilities in the y direction are the same. From a pressure versus square-root time plot, the average permeability expressed by Eq. 4.164 can be obtained by using Eq. 4.29.

Anisotropic Horizontal Permeabilities

In the homogeneous reservoir case, the permeability in the x direction affects the onset of the pseudo-radial flow period. The similar effect on the heterogeneous reservoir case is examined next. Fig. 4.27 and Table 4.11 show a schematic model and reservoir parameters for this example.

As shown in Fig. 4.28, the pressure response is greatly affected by the anisotropic

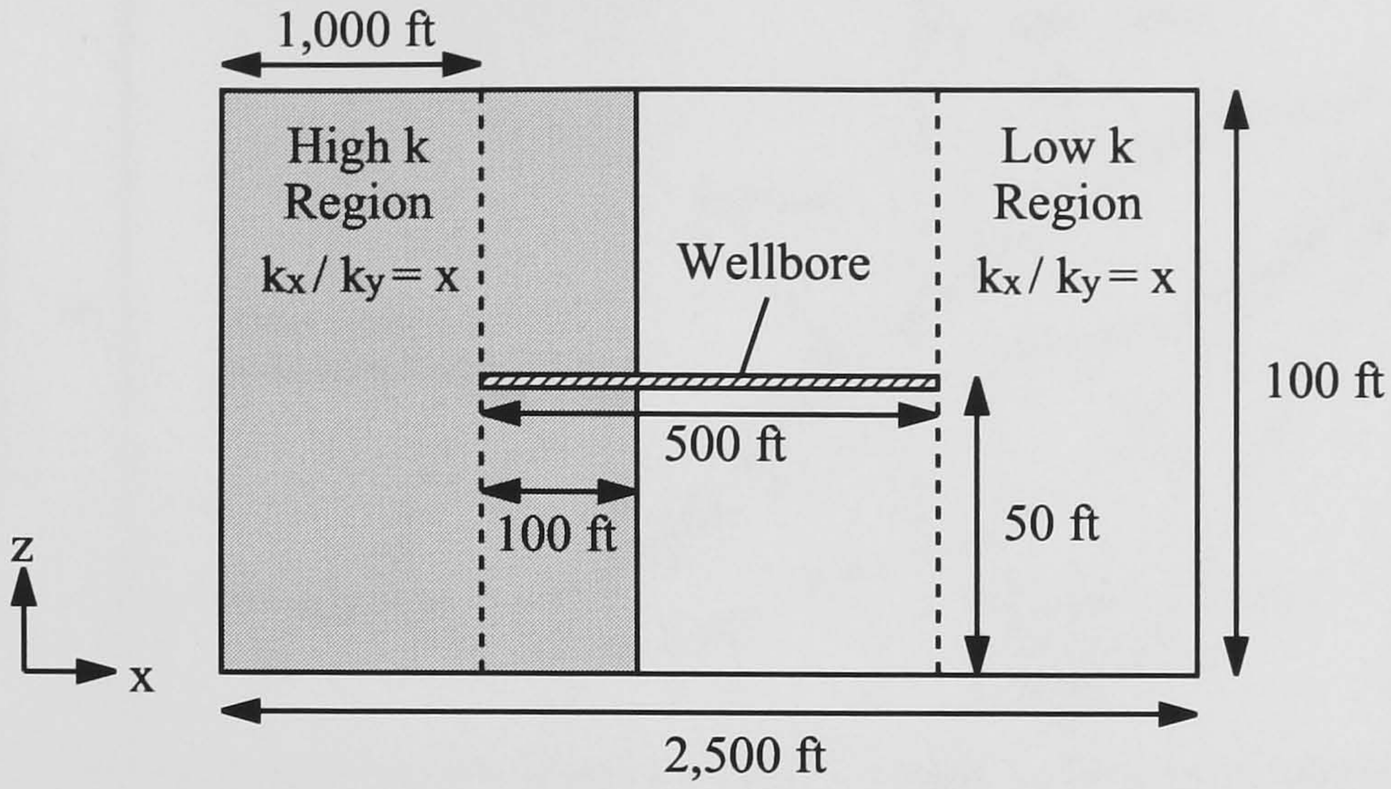


Figure 4.27: Schematic model for the varying k_x/k_y ratio case.

Table 4.11: Reservoir and well data for the case in Fig. 4.27.

Well length, ft			500
Reservoir thickness, ft			100
Well position from the reservoir bottom, ft			50
Reservoir width, ft			2,500
Permeability, md, in		High k Zone	Low k Zone
the x direction			
for the k_x/k_y ratio,	1/1	500	100
	1/10	50	10
	1/100	5	1
the y direction		500	100
the z direction		50	10

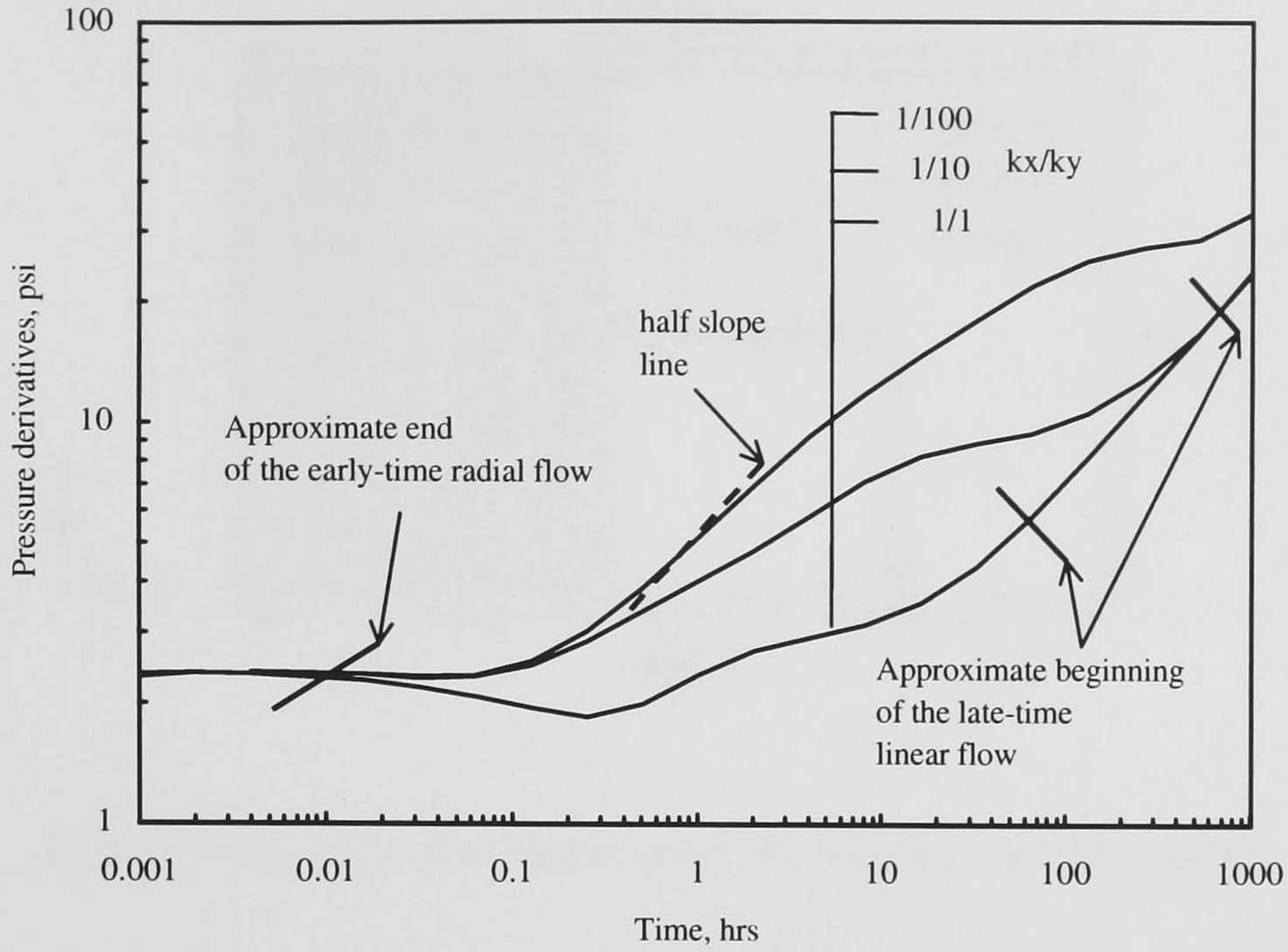


Figure 4.28: Effect of the anisotropic horizontal permeabilities in the reservoir with a permeability contrast.

horizontal permeabilities. At early times, the early-time radial flow can be observed in each case. Then, if the k_x/k_y is small, the intermediate-time linear flow similar to that of a homogeneous reservoir can occur which is shown by the half-slope line in the figure. If the k_x/k_y ratio is large, the effect of the pseudo-radial flow combined with the effect of the crossflow between the high and the low permeability zones start affecting the pressure response, which causes the derivative curve to flatten. The late-time linear flow can be observed for the large k_x/k_y cases. However, the beginning of the late-time linear flow is delayed if the k_x/k_y becomes small, since it takes longer for the pressure disturbance to reach the two reservoir edge boundaries.

Reservoir Width

The reservoir width affects the onset of the late-time linear flow. This is similar to the effect of the anisotropic horizontal permeabilities as shown in the previous example. The schematic model and parameters are shown in Fig. 4.29 and Table 4.12. The results are

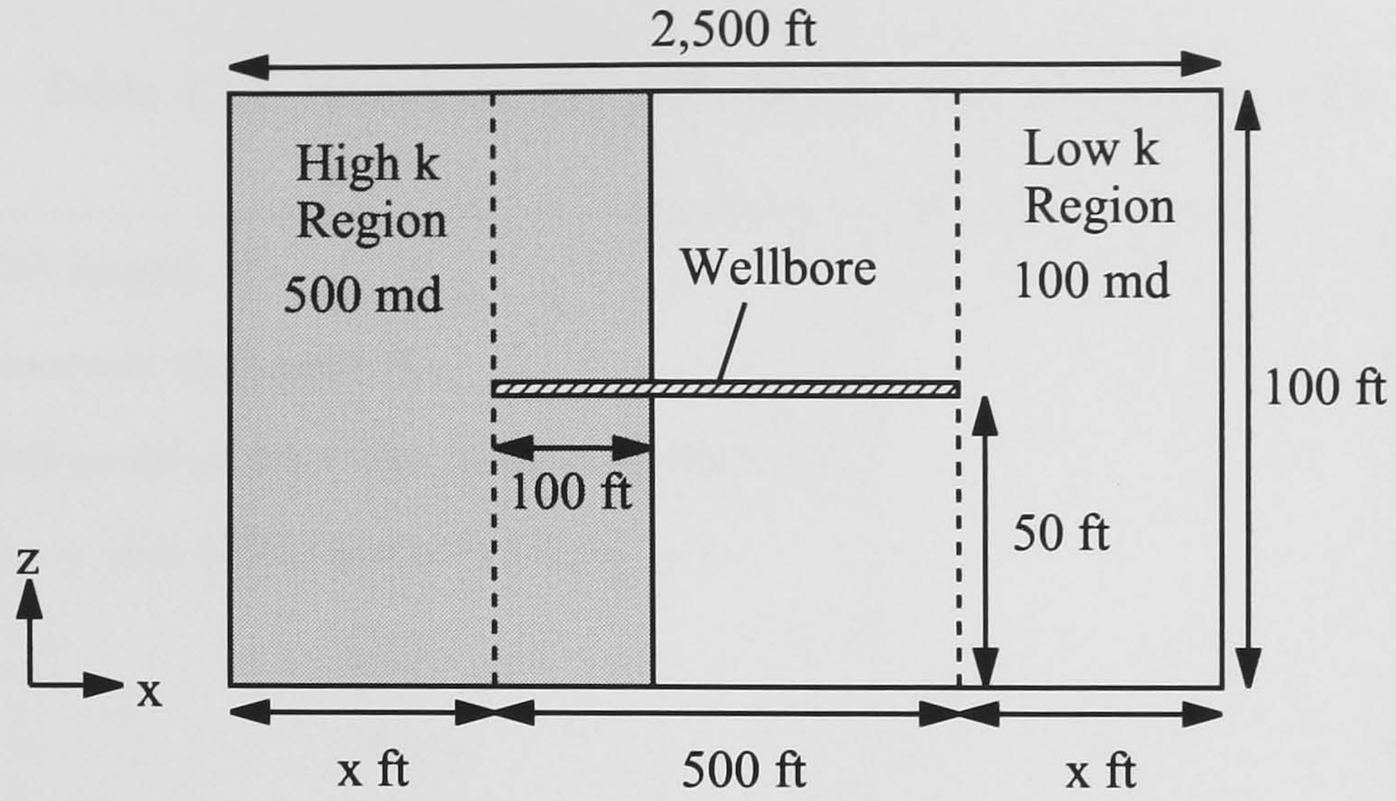


Figure 4.29: Schematic model with the varying reservoir width.

shown in Fig. 4.30. Note that the zero reservoir edge width means the reservoir width is the same as the well length. This is the same case as a vertical well in elongated linear flow systems discussed by Ehlig-Economides and Economides [32], except the reservoir is heterogeneous in our case.

For the zero reservoir edge width case, only the early-time radial flow period and the intermediate-time linear flow period (or the late-time linear flow period) exist. As the edge width increases, the pseudo-radial flow affected by the crossflow between the high and the low permeability zones becomes apparent. Eqs. 4.17 and 4.21 derived for the homogeneous reservoir case cannot be used since the crossflow effect between the zones exists. However, the averaged permeabilities of Eqs. 4.165 and 4.164 can be obtained from the slope of the early-time radial flow and the late-time linear flow periods using Eqs. 4.6 and 4.29, respectively.

Porosity Ratio Between the High and the Low Permeability Regions

Fig. 4.31 shows the effect of porosity ratio between the high and low permeability regions on the pressure derivative. The reservoir model is the same as that in Fig. 4.25. The reservoir thickness is 100 ft, and the wellbore is located at the centre of the reservoir. The reservoir and the well parameters are shown in Table 4.13. The porosities for the high and

Table 4.12: Reservoir and well data for the case in Fig. 4.29.

Well length, ft	500	
Reservoir thickness, ft	100	
Well position from the reservoir bottom, ft	50	
The width of the reservoir edges, x ft		
Case 1	0	
Case 2	200	
Case 3	500	
Case 4	1,000	
Case 5	2,000	
Isotropic permeability, md	High k Zone	Low k Zone
	500	100

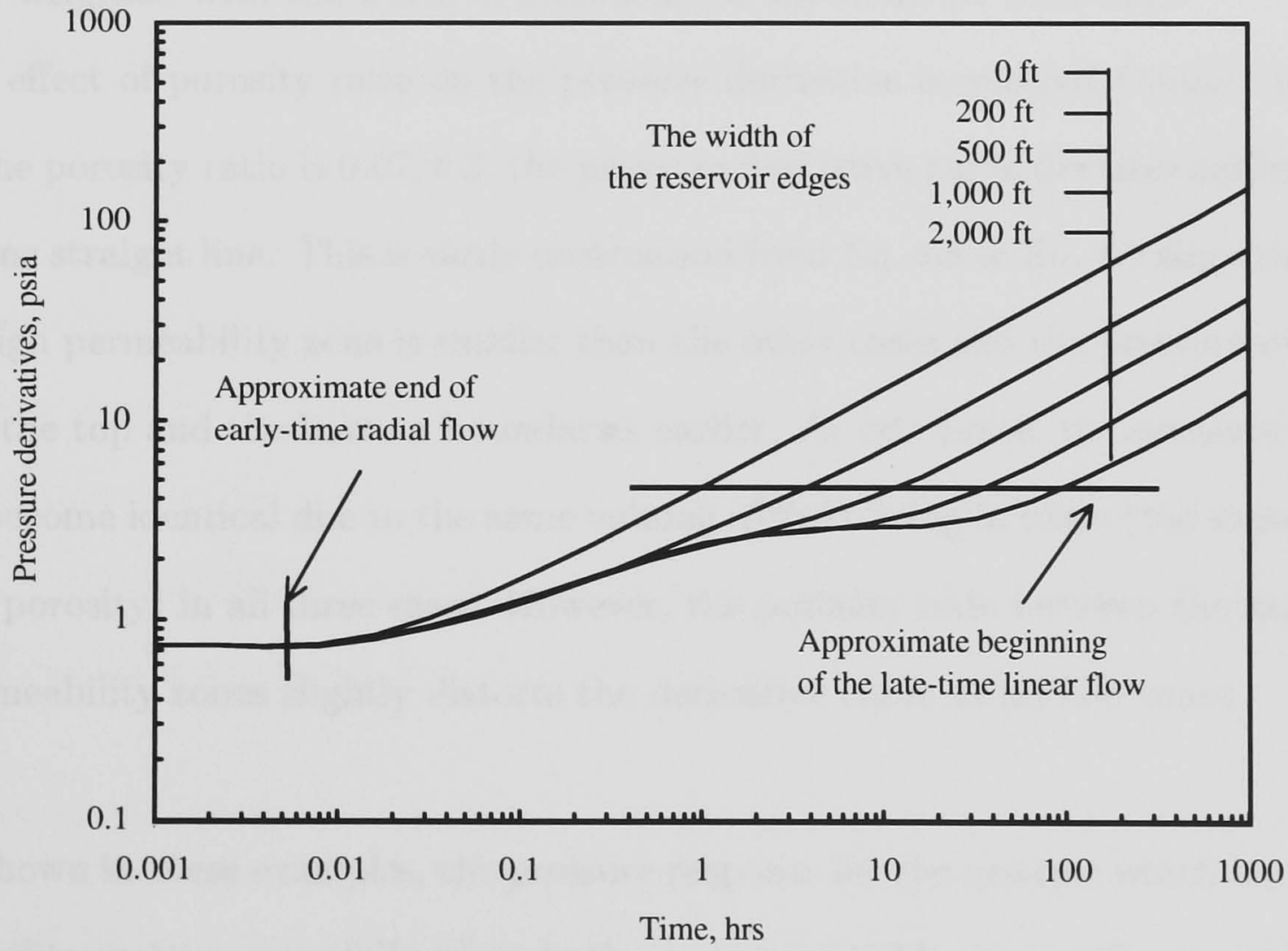


Figure 4.30: Effect of the reservoir width in the reservoir with a permeability contrast.

Table 4.13: Reservoir and well data for the reservoir with the varying porosity contrast.

Well length, ft		500
Reservoir thickness, ft		100
Well position from the reservoir bottom, ft		50
Reservoir width, ft		2,500
Isotropic permeability, md	High k Zone	Low k Zone
	500	100
Porosity $\frac{\phi_h}{\phi_l}$, fraction		
$\frac{0.33}{0.1}$	0.33	0.1
$\frac{0.2}{0.2}$	0.2	0.2
$\frac{0.07}{0.3}$	0.07	0.3

the low permeability zones are determined in the way that total fluid in place (or averaged porosity weighted with the width of each zone) is the same for each case.

The effect of porosity ratio on the pressure derivative is relatively small. In the case where the porosity ratio is 0.07/0.3, the pressure derivative curve deviates earlier from the early-time straight line. This is easily understood from Eq. 4.8 or Eq. 4.9 since the porosity of the high permeability zone is smaller than the other cases and the pressure disturbance reaches the top and the bottom boundaries earlier. At late times, the pressure derivative curves become identical due to the same volume of fluid being in place (the same weighted average porosity) in all three cases. However, the porosity ratio between the high and the low permeability zones slightly distorts the derivative curve at middle times.

As shown in these examples, the pressure response for the systems which have different permeability regions, especially when both are penetrated by the wellbore, exhibit a very complicated behaviour. Explicit analyses are possible only in the early-time radial flow or the late-time linear flow period (if observed). Information from the pressure behaviour in the intermediate time period can only be obtained by interactive or automatic pressure-

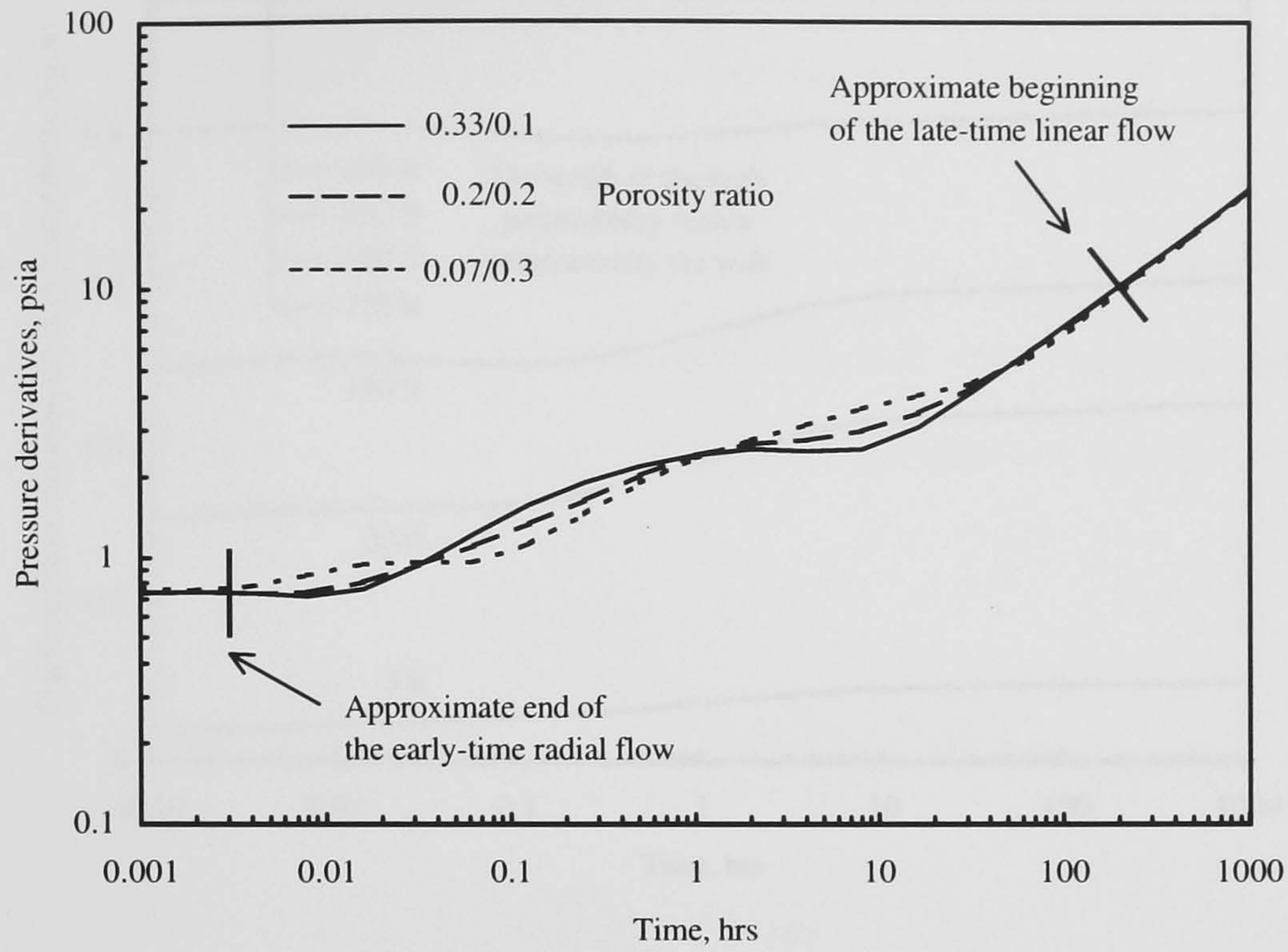


Figure 4.31: Effect of the porosity ratio in the reservoir with a permeability contrast.

pressure derivative type-curve matching procedure.

4.3.3 The Effect of Heterogeneities on Flow Rate Responses

When available, downhole flow rate data are important so as to reduce the non-uniqueness of pressure transient response. The synthetic fractional flow rate from each well segment can be obtained by Eq. 4.160. Since Eq. 4.160 yields a total flow rate within a certain interval, a continuous flow rate data as would be obtained by a production logging test should be integrated for each interval.

The Size of the High Permeability Region Penetrated by the Wellbore

The change in fractional flow rate of the high permeability region with time for the case presented in Fig. 4.19 and Table 4.6 is shown in Fig. 4.32. The effect of the size of the high permeability region penetrated by the well is clearly demonstrated in the figure.

In this case, it is observed that the fractional rate from the high permeability region

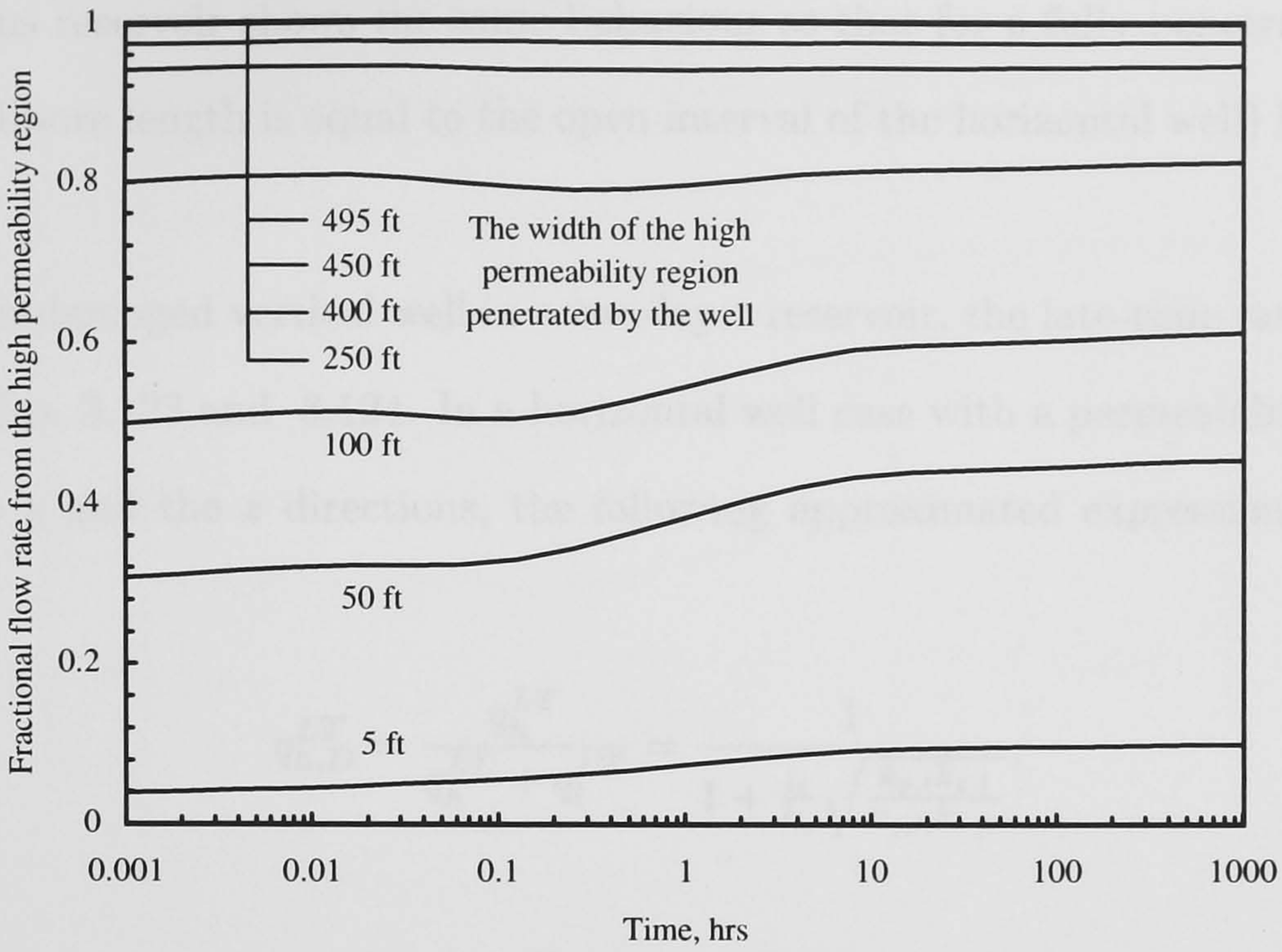


Figure 4.32: Fractional flow rate from the high permeability zone for the case in Fig. 4.19.

becomes over 0.5 when only a length of 100 ft in the high permeability region is penetrated by the wellbore (the wellbore length is 500 ft). If the well penetrates a length of 250 ft (half the well length), the fractional rate reaches approximately 0.8. At late times, the rate approaches a constant value in each case. If the high permeability region penetrated by the well is over 400 ft, the fractional flow rate becomes almost constant at all times.

For a vertical well case in a layered reservoir with crossflow, the early-time and the late-time responses can be calculated from Eq. 3.118 to Eq. 3.127 as shown in Section 3.2.6. However, for the case of a horizontal well in a laterally heterogeneous reservoir, it seems quite difficult to obtain such equations due to the complexity of the problem.

At early times before the pressure disturbance reaches the top or bottom reservoir boundary, the behaviour of the horizontal well case should be the same as that for a partially penetrating vertical well intersecting several layers. Moreover, at very early times, the response of a partially penetrating well shows a radial flow where the effective thickness is equal to the interval penetrated by the well [57]. From these facts, it is expected that at very early times, the flow rate transition of a horizontal well in a laterally

heterogeneous reservoir shows the same behaviour as that for a fully penetrating vertical well (the wellbore length is equal to the open interval of the horizontal well) in a crossflow reservoir.

For a non-damaged vertical well in a two-layer reservoir, the late-time rate can be calculated by Eqs. 3.123 and 3.124. In a horizontal well case with a permeability anisotropy between the y and the z directions, the following approximated expressions can be obtained, i.e.,

$$q_{h,D}^{LT} = \frac{q_h^{LT}}{q_h^{LT} + q_l^{LT}} = \frac{1}{1 + \frac{l_l}{l_h}\sqrt{\frac{k_{y,l}k_{z,l}}{k_{y,h}k_{z,h}}}}, \tag{4.166}$$

$$q_{l,D}^{LT} = 1 - q_{h,D}^{LT}, \tag{4.167}$$

where $q_{h,D}^{LT}$ and $q_{l,D}^{LT}$ are the late-time fractional flow rates from the high and the low permeability regions, and l_h and l_l ($l_h + l_l = L_w$) are the wellbore length penetrating the high and the low permeability regions, respectively. To obtain Eqs. 4.166 and 4.167, the same porosity was assumed for each region, and the isotropic permeability in Eq. 3.120 was replaced by the geometric mean permeability $\sqrt{k_y k_z}$. The effect of apparent skin due to the elliptic flow [120],

$$S_a = -\ln\frac{1}{2}\left(\sqrt[4]{\frac{k_y}{k_z}} + \sqrt[4]{\frac{k_z}{k_y}}\right), \tag{4.168}$$

was ignored since even though the reservoir anisotropy is relatively high, the effect of it on the skin is usually small (e.g., $S_a = -0.55$ for the case $k_z/k_y = 0.01$).

If Eq. 4.166 is applied to the case where the length of the penetration of the high permeability region is 100 ft in Fig. 4.32, the fractional flow rate becomes 0.56. In this case, the early-time radial flow represented by the straight line of the pressure derivative in Fig. 4.20 ends at approximately 0.005 hrs, and the fractional rate at the time is about 0.51 (Fig. 4.32), which is close to the value (0.56) calculated by Eq. 4.166.

It has been observed that in most cases, the fractional flow rate at the end of the early-time radial flow is nearly equal to that calculated by Eq. 4.166. The fractional rate calculated by Eq. 4.166 overestimated the fractional rate at the end of the early-time radial

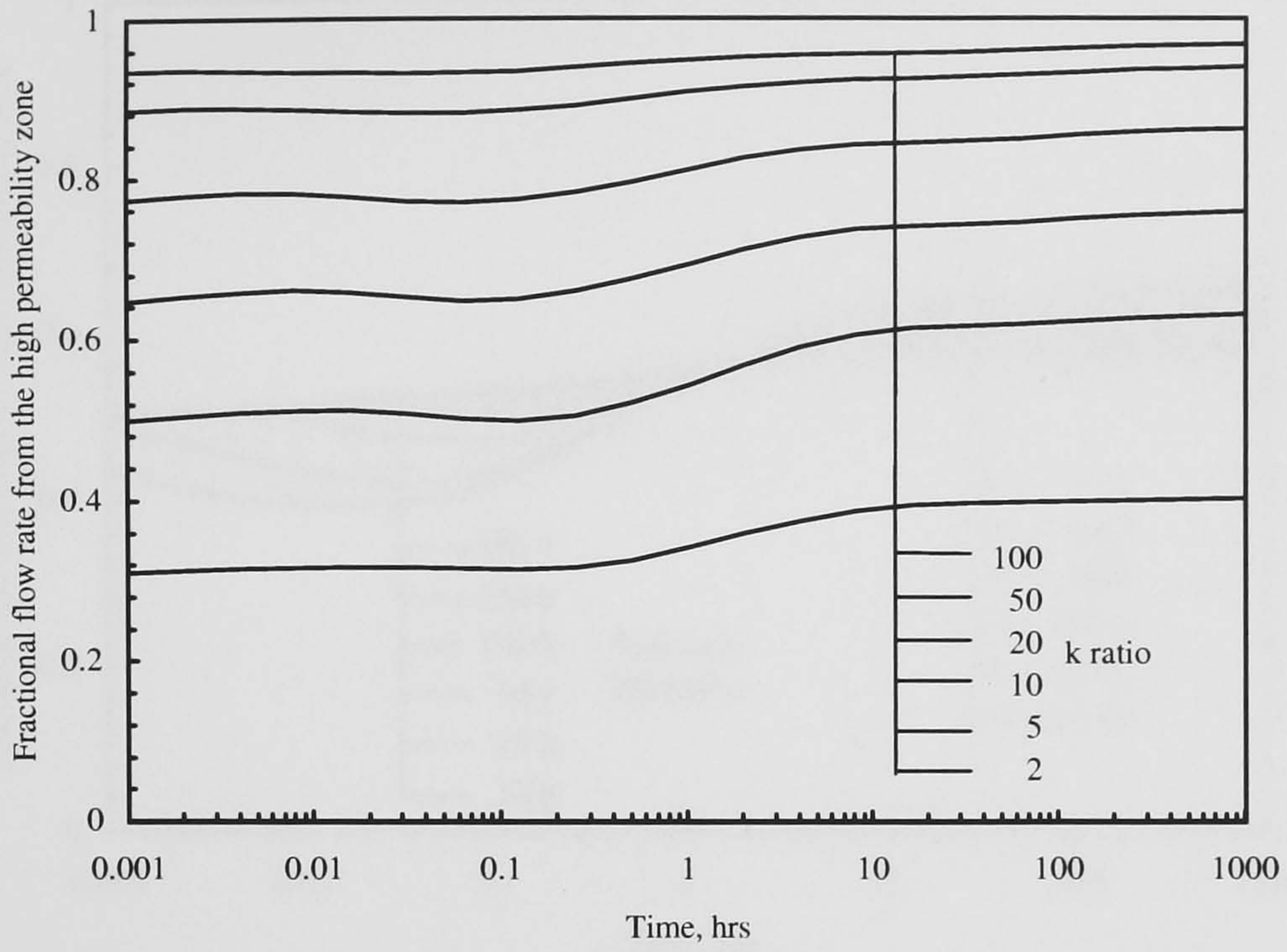


Figure 4.33: Fractional flow rate from the high permeability zone for the case in Fig. 4.23.

flow by maximum 10% for cases shown in this section. However, the error can increase if the rate is read at much earlier time in the early-time radial flow period. For the late-time fractional flow rate, no simple relation has been found yet.

Permeability Ratio Between the High and the Low Permeability Regions

Fig. 4.33 shows the fractional rate transition for the case of Fig. 4.23. The response is very similar to the previous case. It is observed that even if the permeability ratio is only 5, over 50% of the fluid is produced from only 20% (100 ft) of the wellbore penetrating the high permeability zone. The difference of the fractional rates between the early-time and the late-time periods disappears as the permeability contrast becomes large. The early-time rate at the end of the early-time radial flow period can be calculated by Eq. 4.166 within 10% error for these cases.

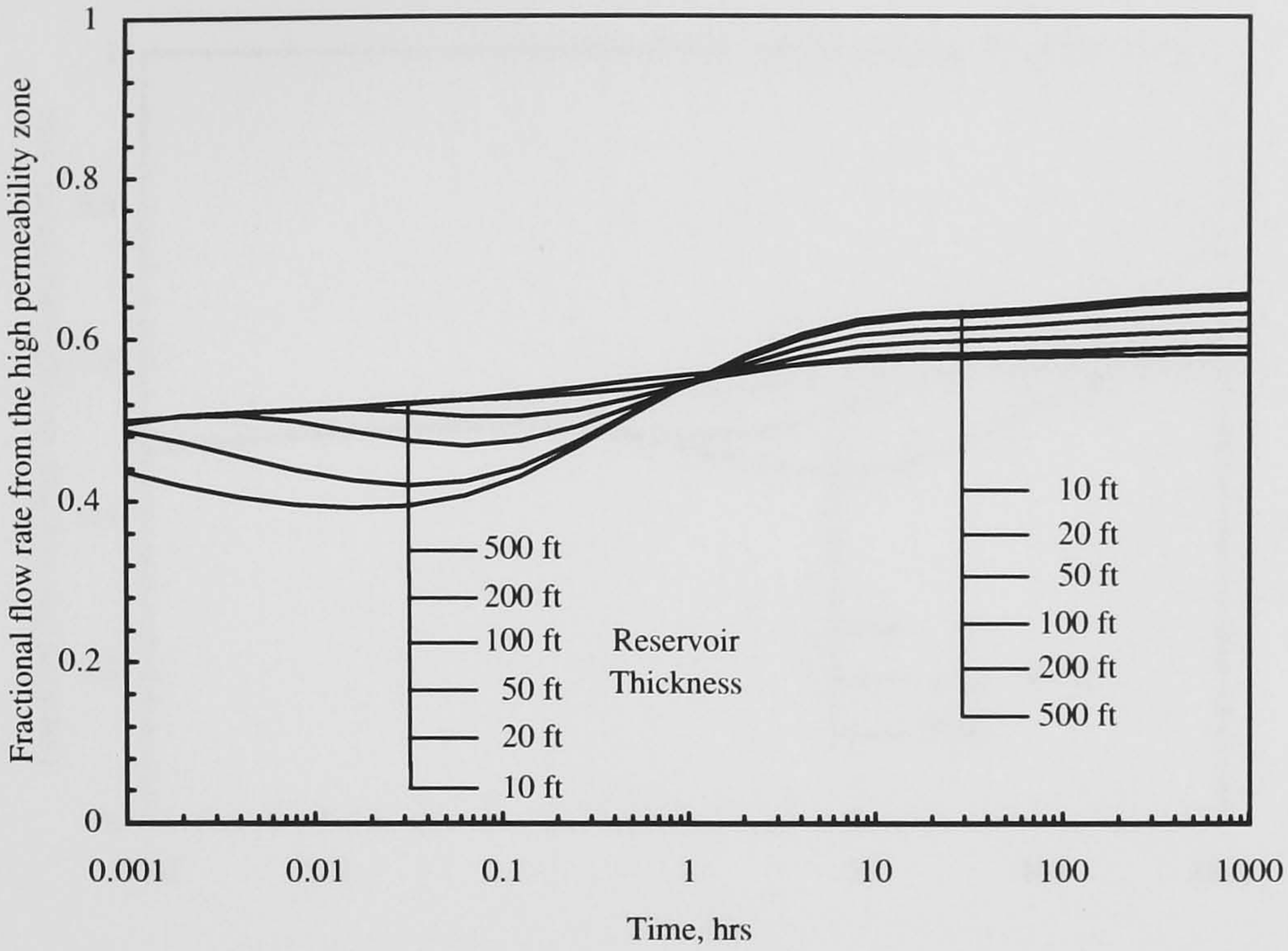


Figure 4.34: Fractional flow rate from the high permeability zone for the case in Fig. 4.25.

Reservoir Thickness

The next case shows the effect of the reservoir thickness on the flow rate transition (Fig. 4.34). The reservoir model is the same as in Fig. 4.25. Unlike the previous two cases, a concavity grows at early times as the reservoir thickness becomes small. This is considered to be the effect of the top and the bottom reservoir boundaries, and once the pressure disturbance has reached those boundaries, the rate calculation of the Eq. 4.166 becomes no longer valid. The late-time fractional flow rate is also affected by the reservoir thickness. As opposed to the early-time behaviour, the late-time fractional flow rate becomes larger as the reservoir thickness decreases.

Anisotropic Horizontal Permeabilities

The effect of the anisotropic horizontal permeability is shown in Fig. 4.35. The reservoir model is the same as in Fig. 4.27. As in the case of pressure derivative in Fig. 4.28, the same flow rate response can be observed at early times before the flow from the x direction starts affecting the rate response. If k_x/k_y is small, the rate decreases temporary at middle

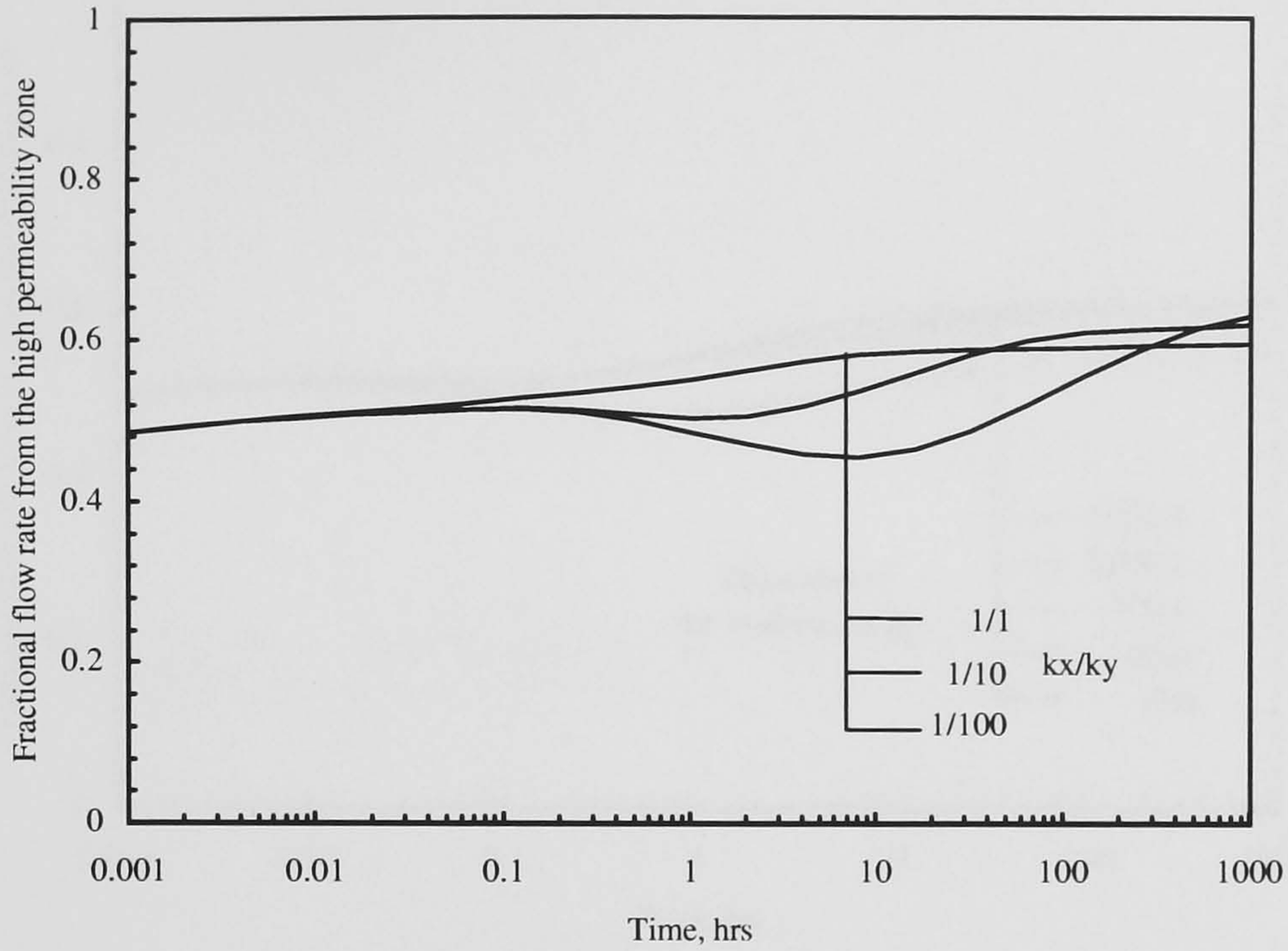


Figure 4.35: Fractional flow rate from the high permeability zone for the case in Fig. 4.27.

times due to the lack of support from the x direction. However, at late times, it increases as the flow from the x direction starts developing.

Reservoir Width

The final example shows the effect of the reservoir width on the fractional flow rate transition. Fig. 4.36 shows the results of the flow rate calculation for the same case as shown in Fig. 4.29.

It can be observed that the rate transition curve diverges from that of the largest reservoir width case (the width of the reservoir edge region is 2,000 ft) according to the width of each case. The divergence starts when the pressure disturbance reaches the reservoir edge boundaries of the linear reservoir (see Fig. 4.30). In the case of Fig. 4.36, the rate is almost indistinguishable if the width of the reservoir edge region is over 1,000 ft.

Through these examples, it has been shown the flow rate transition shows complicated

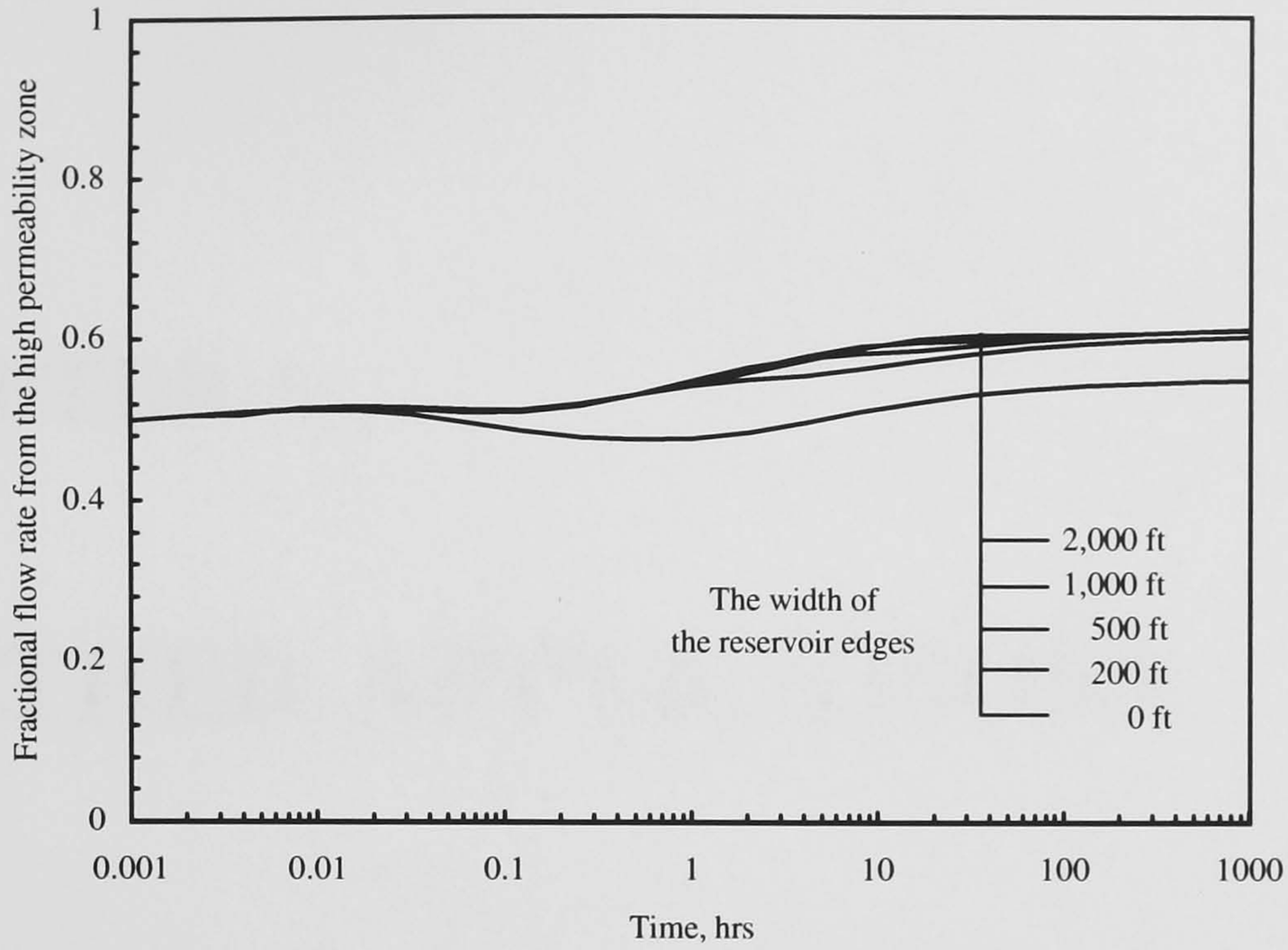


Figure 4.36: Fractional flow rate from the high permeability zone for the case in Fig. 4.29.

responses as in the case of the pressure response. However, such complexities are preferable for identifying various reservoir parameters. Since the reservoir parameters cannot be determined uniquely simply from the pressure-pressure derivative curve matching, the combined analysis with the fractional flow rate data will contribute to reduce the uncertainty of the problem.

CHAPTER 5

FURTHER APPLICATIONS

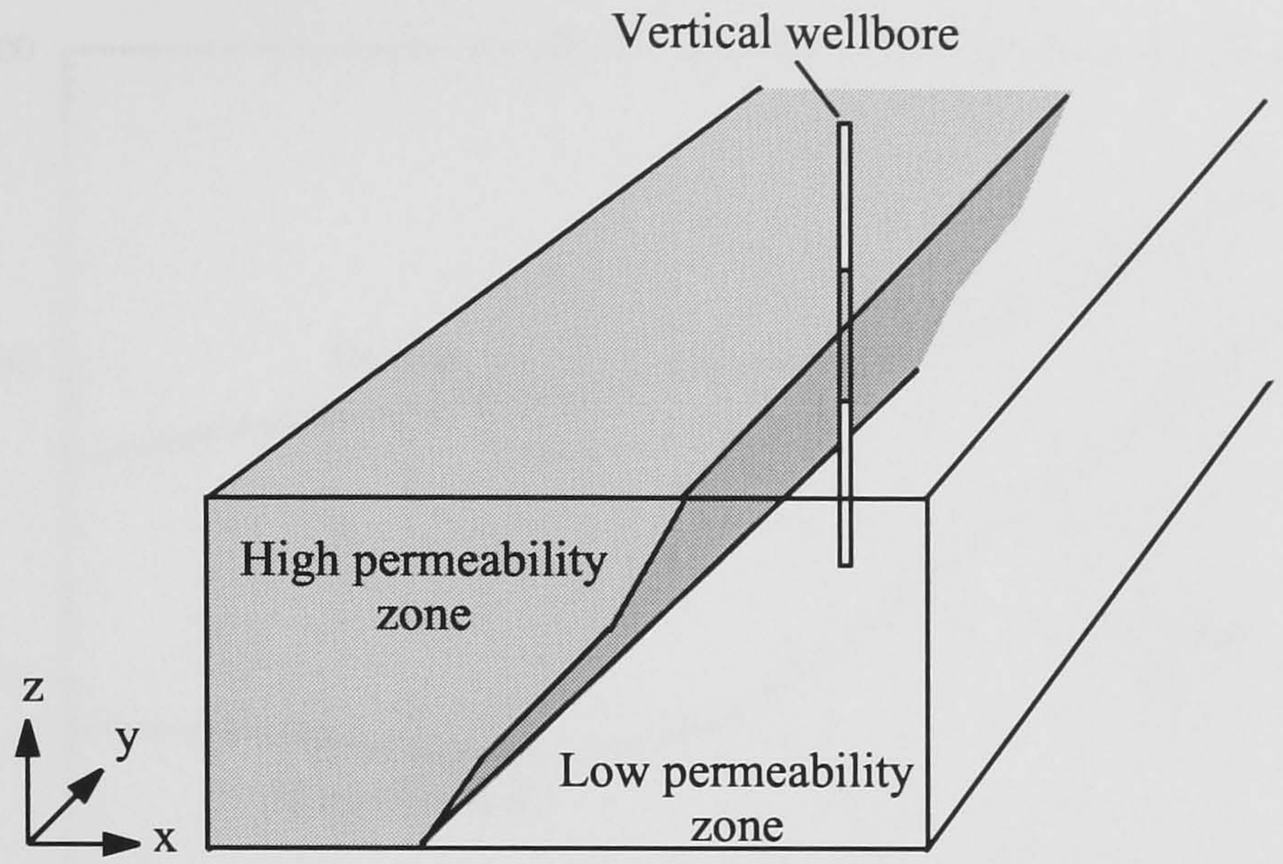


Figure 5.1: Schematic linear heterogeneous reservoir model with a vertical well penetrating heterogeneous zones.

In Chapter 4, the analytical model to produce synthetic pressure and fractional flow rate responses for a horizontal well in a heterogeneous linear reservoir was developed. However, the application of this model is not restricted only to that problem. The semi-permeable wall model combined with the strip-wellbore model extends the applicability of this model to various problems. In this chapter, the further applicability of the model is discussed.

5.1 Vertical Well Case in a Heterogeneous Linear Reservoir

The solution developed in Chapter 4 can be directly used for a vertical well case in a infinitely elongated heterogeneous linear reservoir (Fig. 5.1). Larsen [68] presented an analytical solution for a vertical well in layered crossflow reservoirs with linear boundaries using the image well technique. Bourgeois et al. [15] developed a three-composite linear reservoir model using the line-source well approximation. The model developed in Chapter 4 is more general in terms of the ability to handle any two-dimensional heterogeneity on the $x - z$ plane.

Fig. 5.2 shows a response of a vertical well in a three-layer crossflow reservoir. The

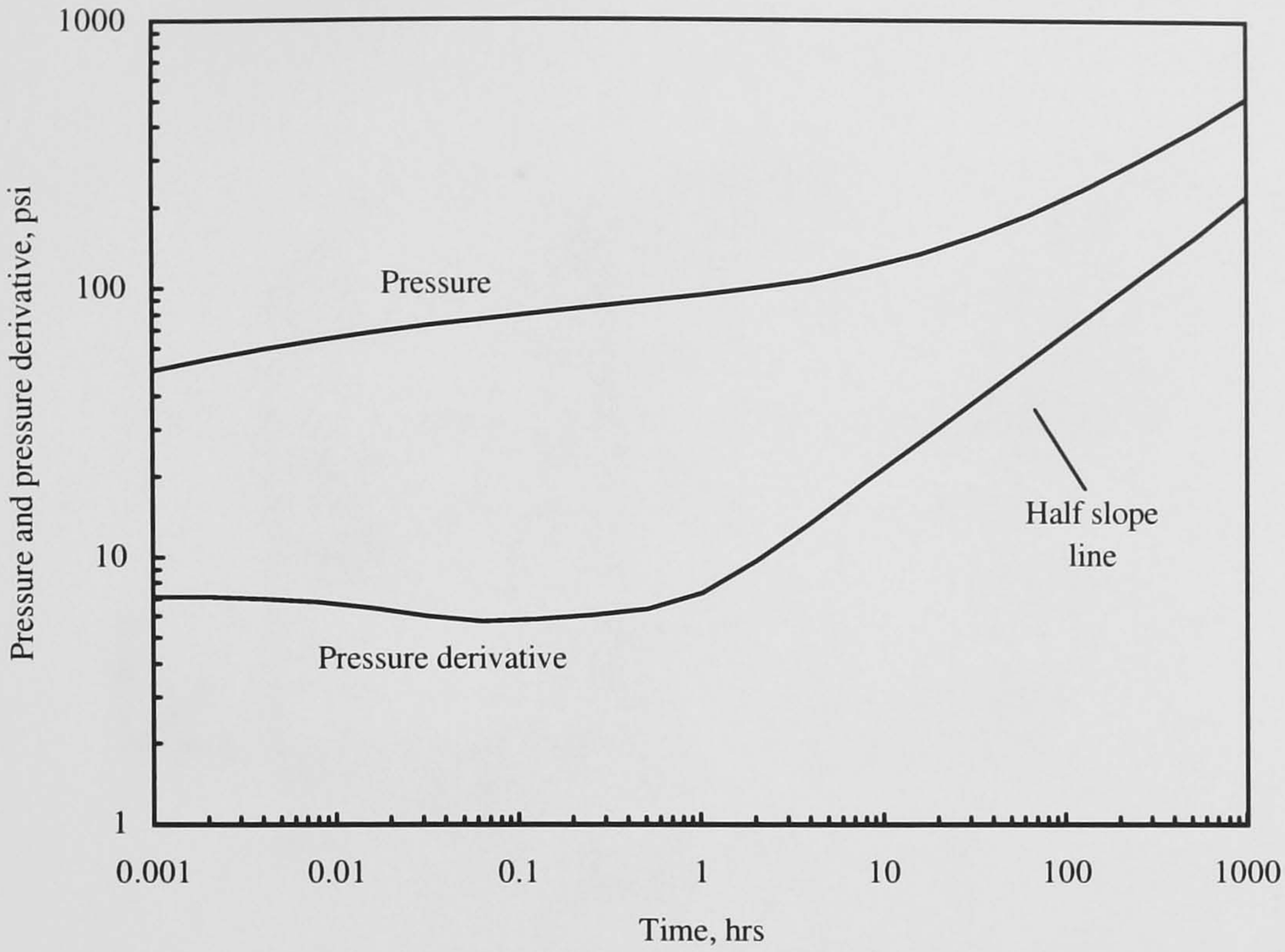


Figure 5.2: Pressure response of a vertical well in the three-layer crossflow linear reservoir.

reservoir and well data are shown in Fig. 5.3 and Table 5.1. The rest of the data is the same as that in Table 4.2. It is observed that at early times, pressure derivative shows a straight line whose level corresponds to a thickness weighted arithmetic averaged permeability among the three layers. Then, crossflow represented by a downward concavity can be observed. Finally, the linear flow develops as shown by a half-slope pressure derivative straight line.

Fig. 5.4 shows a composite reservoir case. Although Bourgeois et al. [15] shows a three-zone composite reservoir case, an extended case to a five-zone composite reservoir is shown here. The reservoir, well, and fluid data are shown in Fig. 5.5, and Tables 4.2 and 5.2. Since the reservoir includes high permeability channels away from the well, the downward concavity of the pressure derivative curve can be observed after a straight line portion. At late times, a linear flow which is typical for a linear reservoir, develops as shown by a half-slope straight line.

The next example is for a high permeability linear channel embedded in a linear reser-

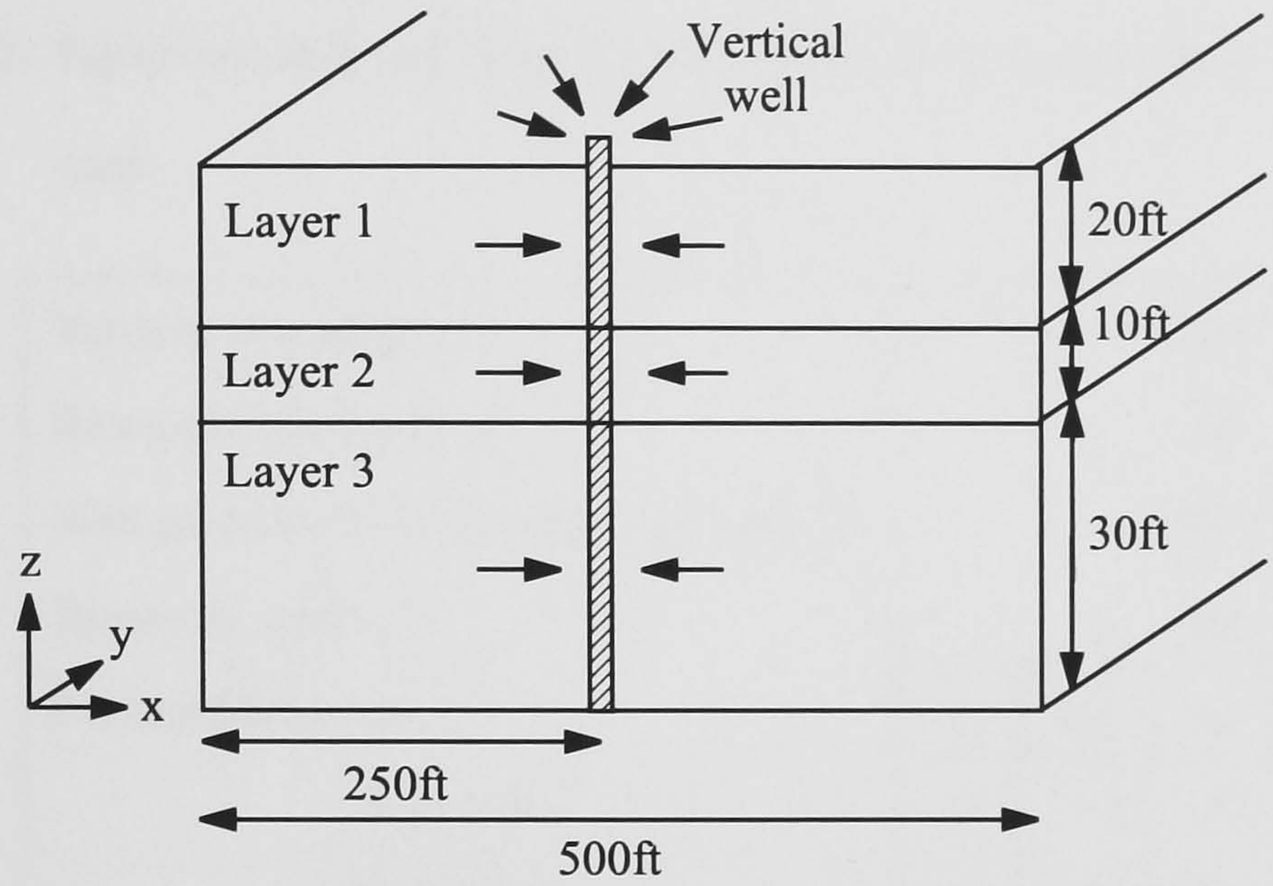


Figure 5.3: Schematic linear three-layer crossflow reservoir model.

Table 5.1: Reservoir and well data for the linear three-layer crossflow reservoir case.

Vertical well length, ft	60		
Reservoir thickness, ft	60		
Well position from the reservoir edge, ft	250		
Reservoir width, ft	500		
Permeability, md	k_x	k_y	k_z
Layer 1	50	100	10
Layer 2	500	1,000	100
Layer 3	25	50	5

Table 5.2: Reservoir and well data for the linear five-zone composite reservoir case.

Vertical well length, ft			60
Reservoir thickness, ft			60
Well position from the reservoir edge, ft			250
Reservoir width, ft			500
Permeability, md	k_x	k_y	k_z
Zone 1 & 5	10	10	10
Zone 2 & 4	500	500	500
Zone 3	50	50	50

Table 5.3: Reservoir and well data for the embedded linear channel reservoir case.

Vertical well length, ft		60
Reservoir thickness, ft		60
Well position from the reservoir edge, ft		250
Reservoir width, ft		500
Permeability, md, in	Reservoir matrix	Channel
the x direction	100	500
the y direction	100	500
the z direction	100	500

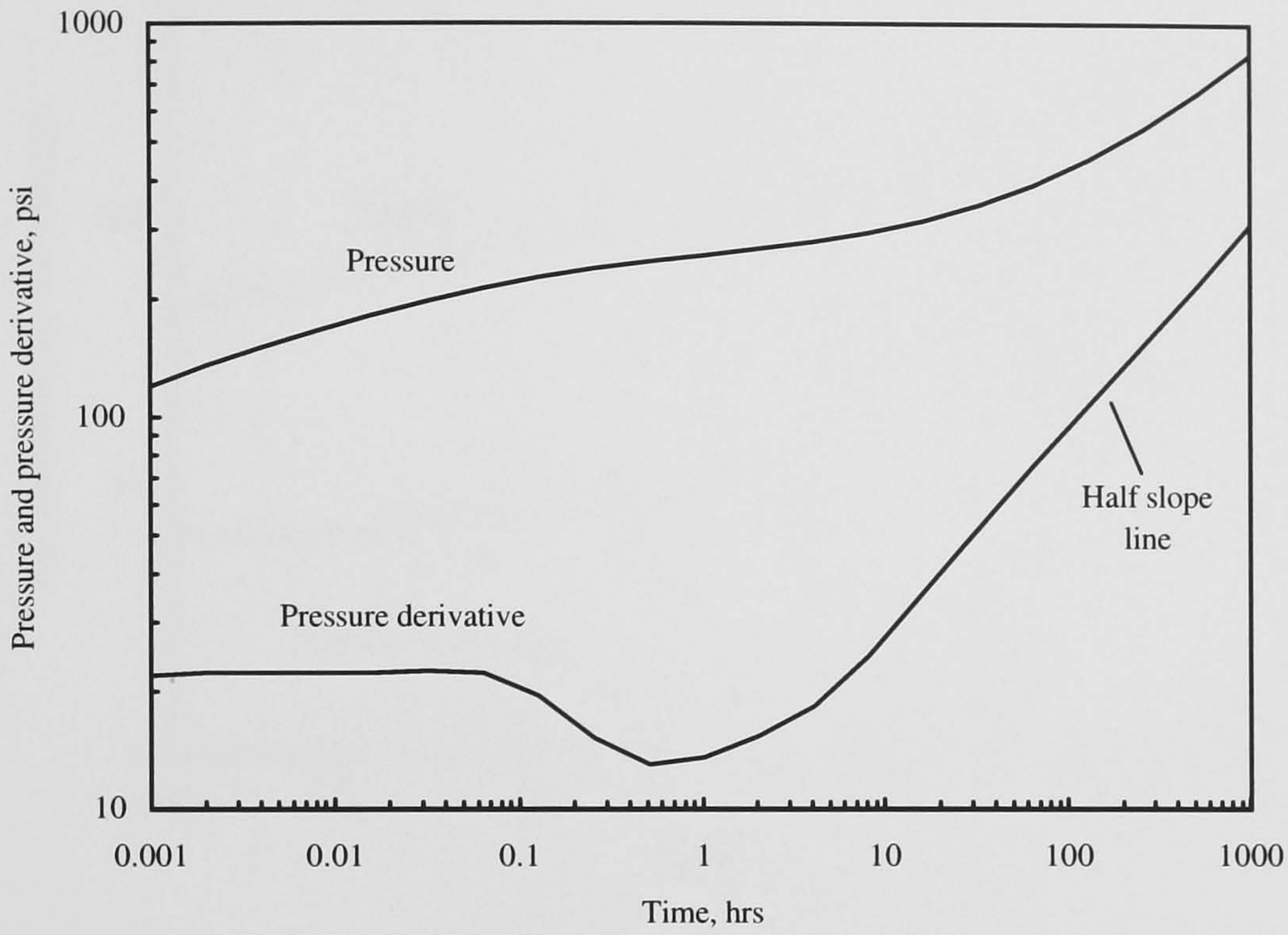


Figure 5.4: Pressure response of a vertical well in the five-zone linear composite reservoir.

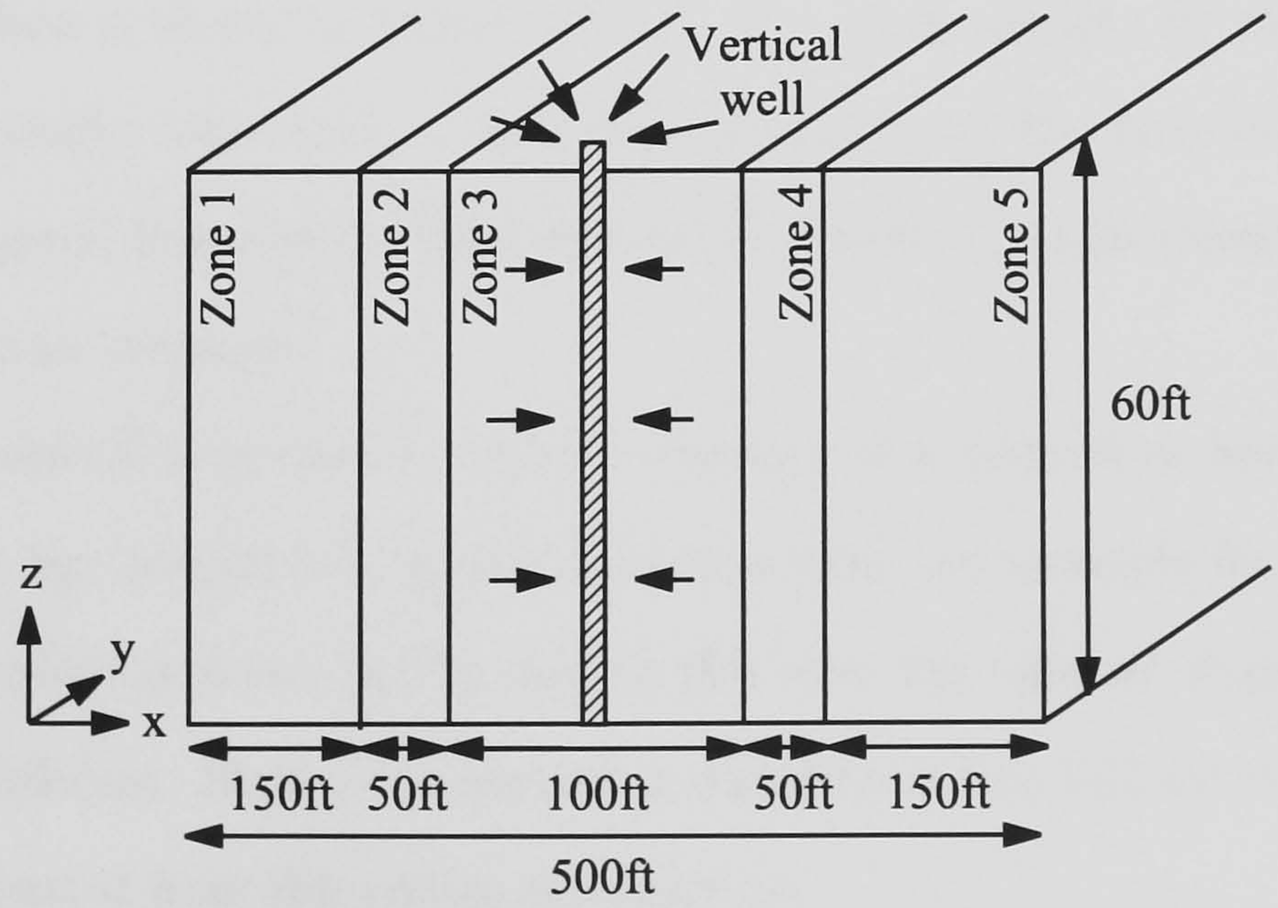


Figure 5.5: Schematic linear five-zone composite reservoir model.

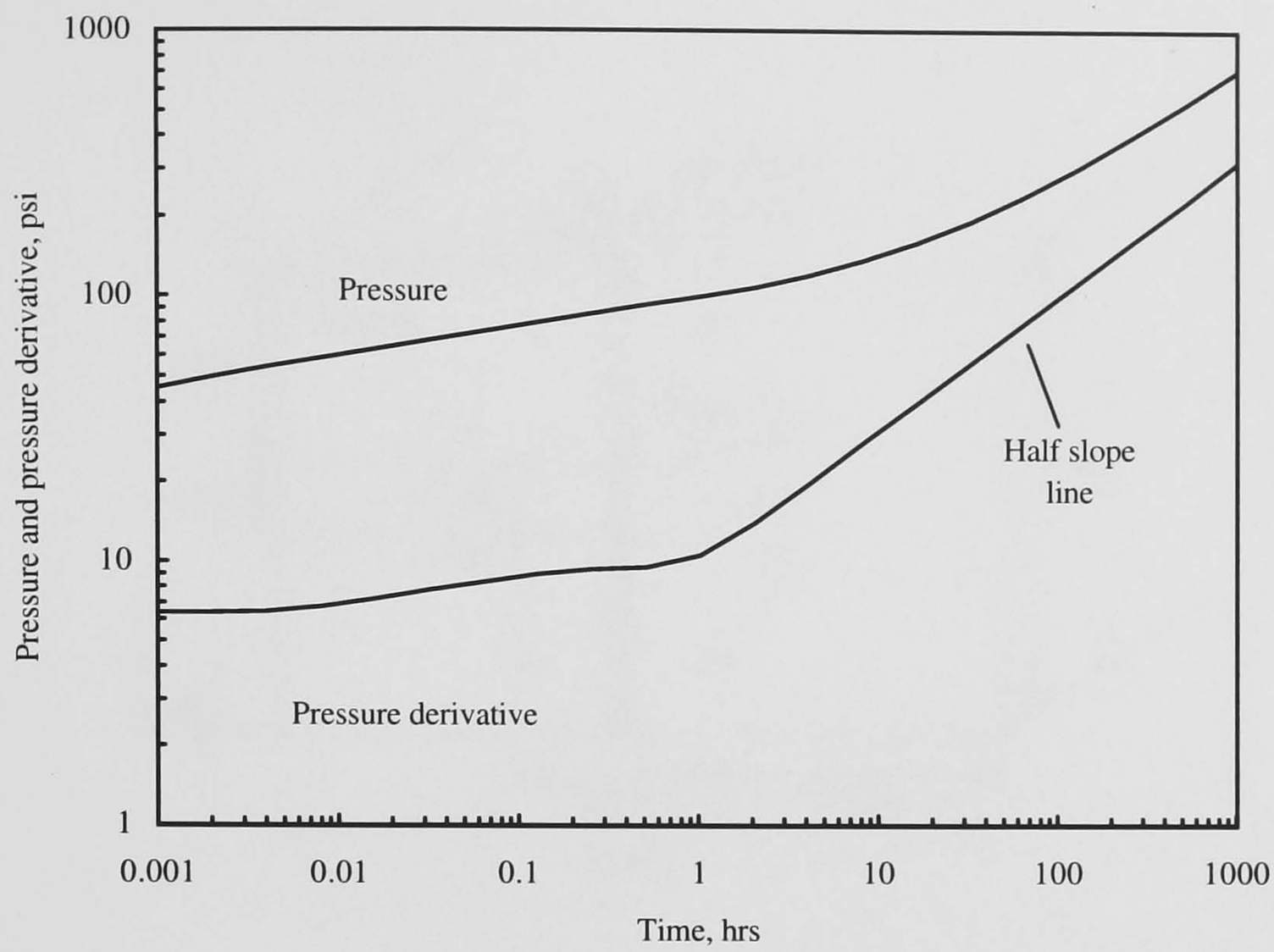


Figure 5.6: Pressure response of a vertical well in the embedded linear channel reservoir.

voir matrix (Fig. 5.7). This case is similar to that discussed in Chapter 3, however, both the channel and the reservoir are linearly elongated in this case. The rest of the reservoir and well data is shown in Tables 4.2 and 5.3. Even though the channel is linearly elongated, the similar behaviour to the radial lens case (e.g., the early-time stratified flow and the convergence flow into the channel) can be observed. At late times, the linear flow of the total system develops.

It is also possible to generate pressure response of a vertical or horizontal well in a reservoir which has complicated geological geometries. An example for a heterogeneous braided river system is shown in Fig. 5.8. In this case, the number of grid blocks in each layer may be different. Hence, the governing equation of Eq. 4.32 for the missing blocks should be eliminated from the system of equations.

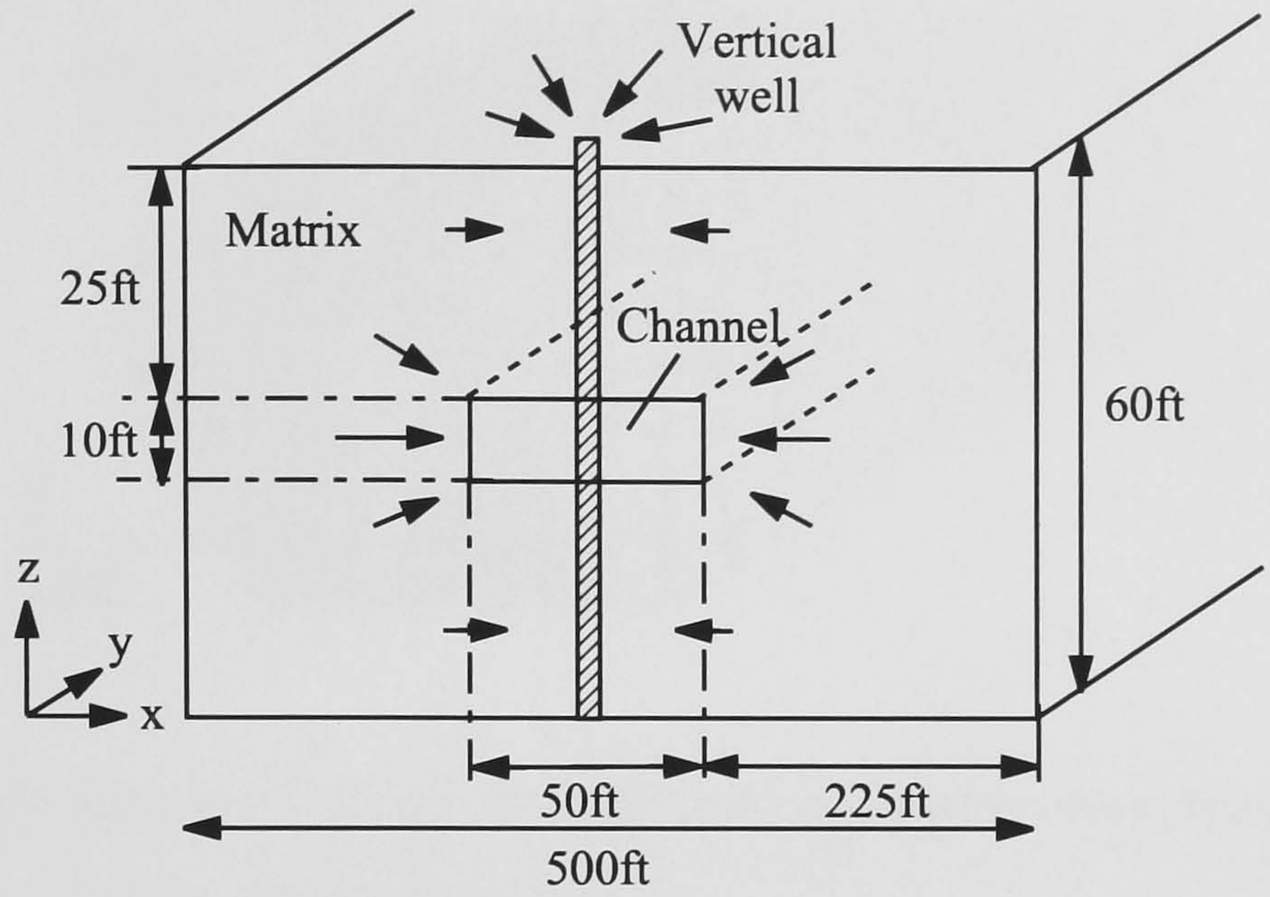


Figure 5.7: Schematic embedded linear channel reservoir model.

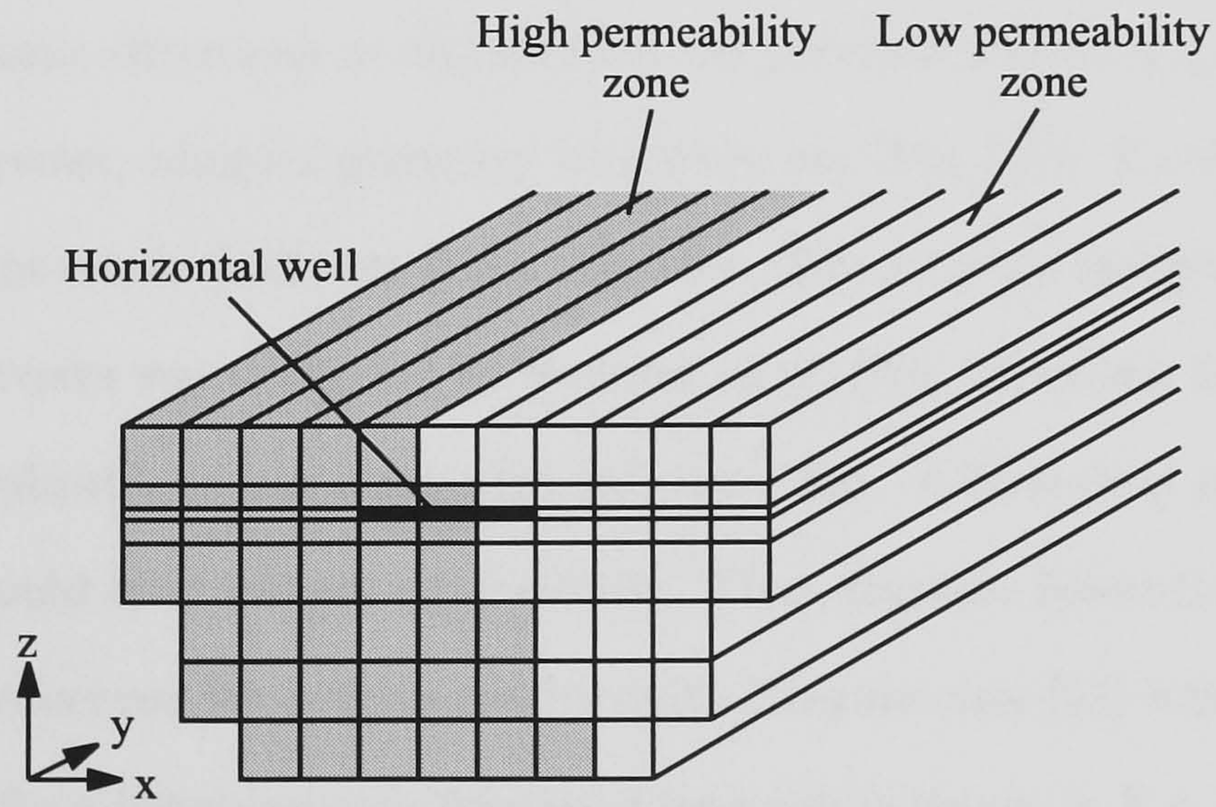


Figure 5.8: Schematic model of a horizontal well in the linear complex reservoir.

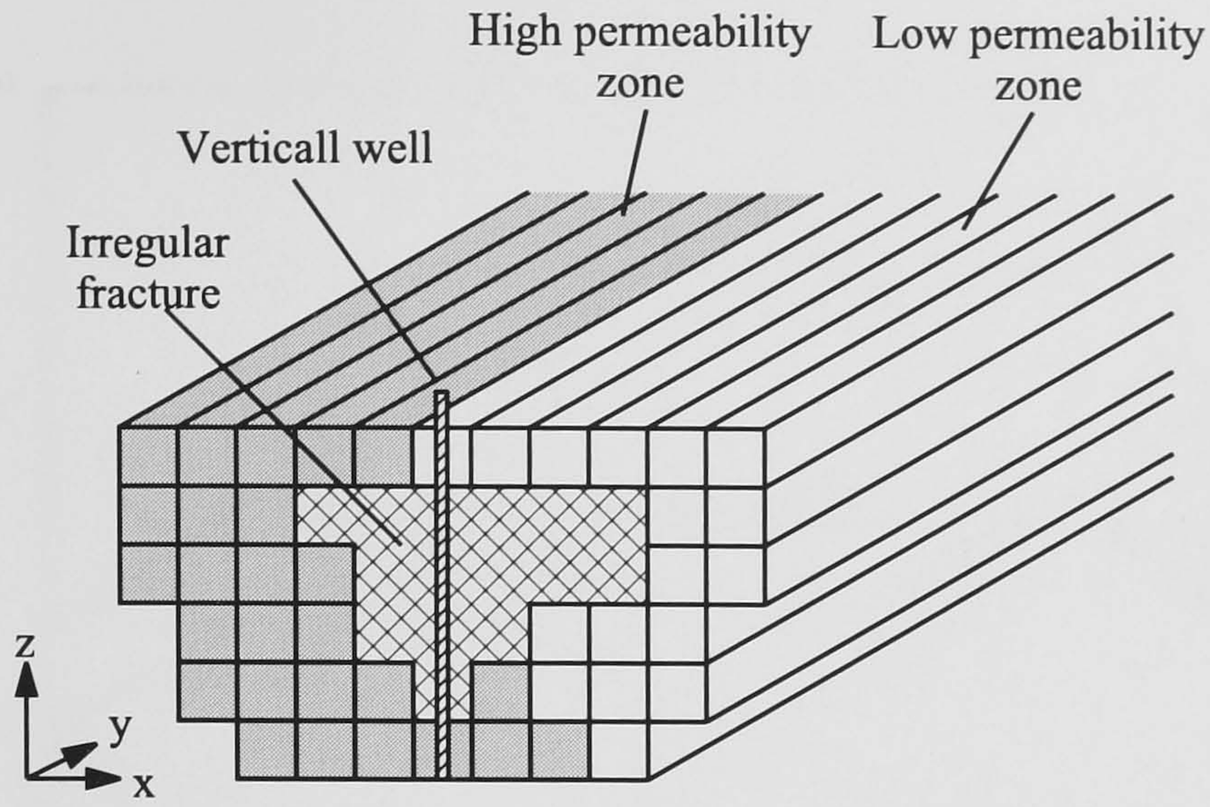


Figure 5.9: Schematic model of a well with a irregular-shape fracture in the linear complex reservoir.

5.2 Fractured Well Case in a Heterogeneous Linear Reservoir

The application to the fractured well case is quite straightforward since the fracture face is modelled easily on the $x - z$ plane in the model developed in Chapter 4. The model is applicable to many situations as discussed in the previous section, e.g., layered reservoirs, composite reservoirs, complex geometry reservoirs, etc (Fig. 5.9). The fracture may or may not penetrate the whole thickness of the reservoir. The pressure analysis of a fractured well in layered reservoirs was discussed by Bennett et al. [10]. However, the model developed here can be applicable to more complicated situations. A limitation of this model is that the fracture should have infinite conductivity. Thus, fracture linear flow and bilinear flow which may be observed for a finite conductivity fracture case [19] will not develop.

A example for a heterogeneous fractured reservoir is shown in Fig. 5.10. The reservoir, fluid, and well data are shown in Fig. 5.11 and Tables 4.2, 5.4. In Fig. 5.10, the two half-slope lines of pressure derivative curve which correspond to the early-time fracture linear flow and the late-time reservoir linear flow can be observed. At middle times, the pressure disturbance reaches the top and the bottom reservoir boundaries, and the flow

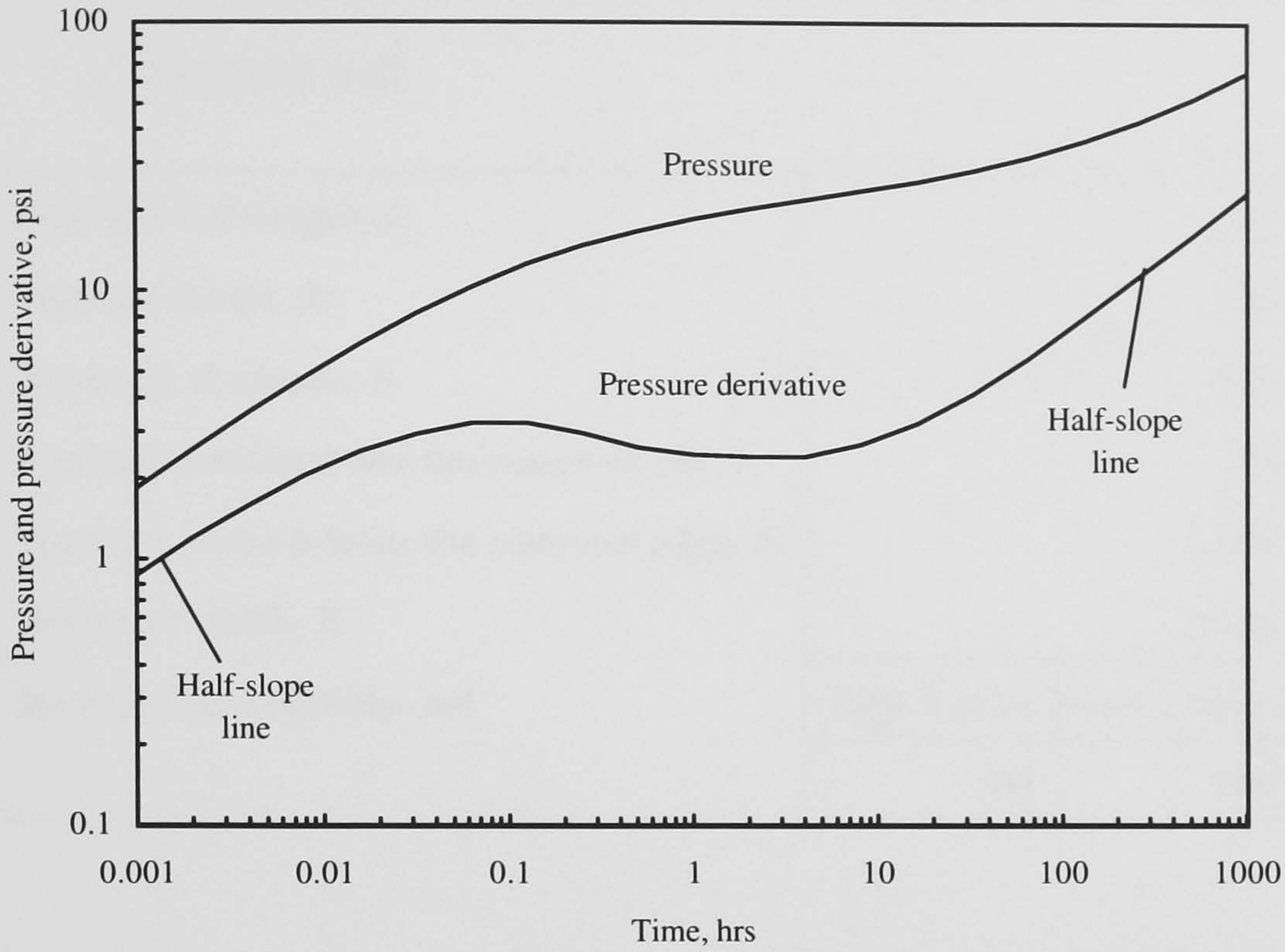


Figure 5.10: Pressure response of a fractured well in the linear composite channel reservoir.

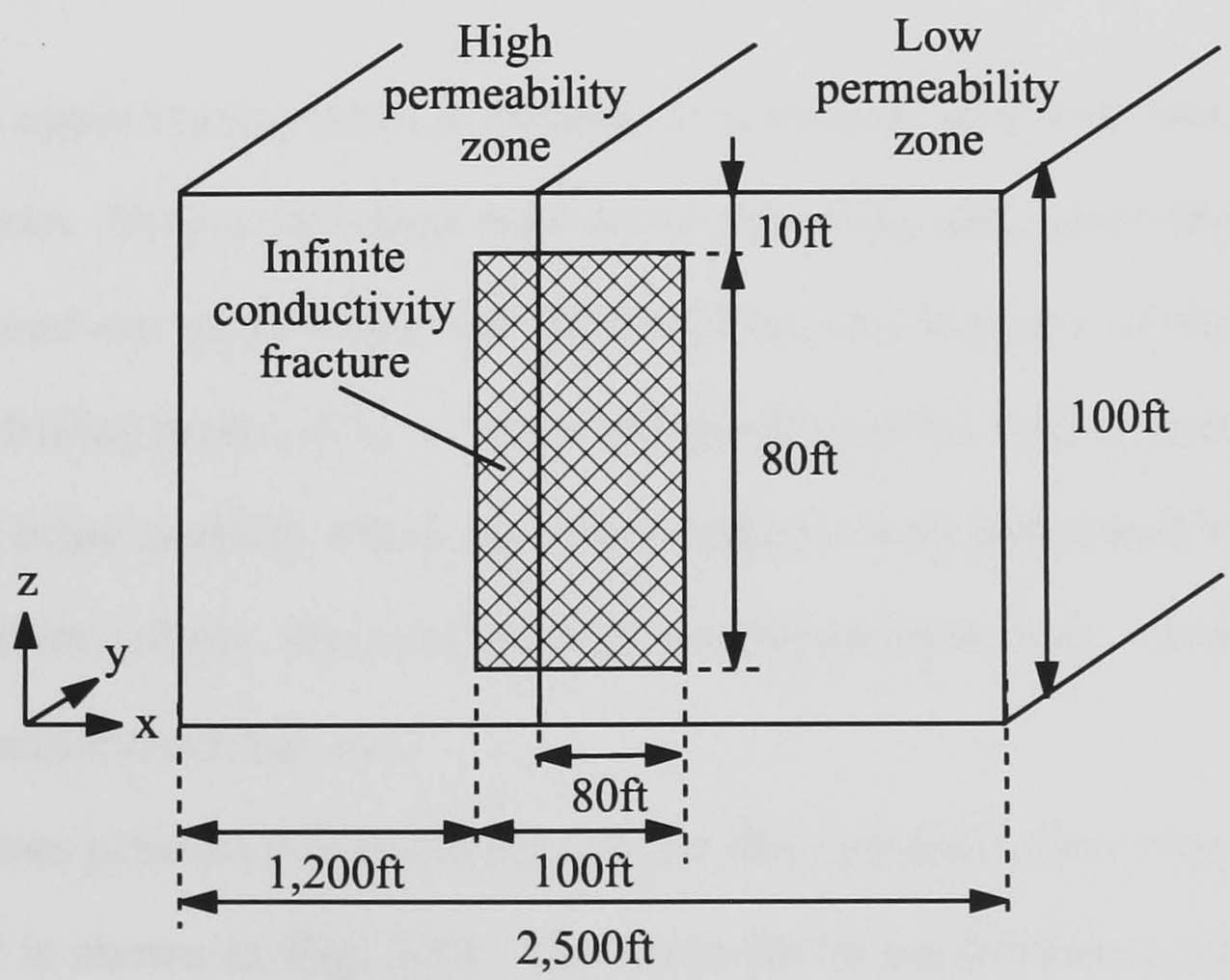


Figure 5.11: Schematic linear composite channel reservoir with a fractured well.

Table 5.4: Reservoir and well data for the linear composite reservoir with a fractured well.

Fracture half length, ft		50
Fracture height, ft		80
Reservoir thickness, ft		100
Fracture position from the reservoir top, ft		10
Fracture position from the reservoir edge, ft		1,200
Reservoir width, ft		2,500
Isotropic permeability, md	High k zone	Low k zone
	500	100

from the lateral direction starts affecting. In addition, crossflow between the high and the low permeability zones starts developing. In such cases, pressure analysis can only be done by an interactive or automatic pressure-pressure derivative type-curve matching.

5.3 Damaged or Stimulated Well Case

It is possible to apply the model to a vertical or horizontal well with varying skin damage along the wellbore. Such a situation may occur when the well penetrates different zones with different reservoir properties (rock permeability, the presence of clay minerals which are reactive to drilling muds, etc.), or when the portion of the well exposed to drilling mud longer than the other portion, which may occur when a long horizontal well is drilled [36]. As in the cases shown above, the reservoir may be a layered reservoir, a composite reservoir, a embedded-channel reservoir, etc.

Fig. 5.12 shows pressure responses for various skin combinations. The schematic model of the reservoir is shown in Fig. 5.13. The reservoir is homogeneous and isotropic. The reservoir, fluid, well data are shown in Tables 4.2 and 5.5.

It can be observed that the large variation of pressure response can occur according to the skin combination. At early times, the stimulated well case ($S_1 = -4, S_2 = -2$)

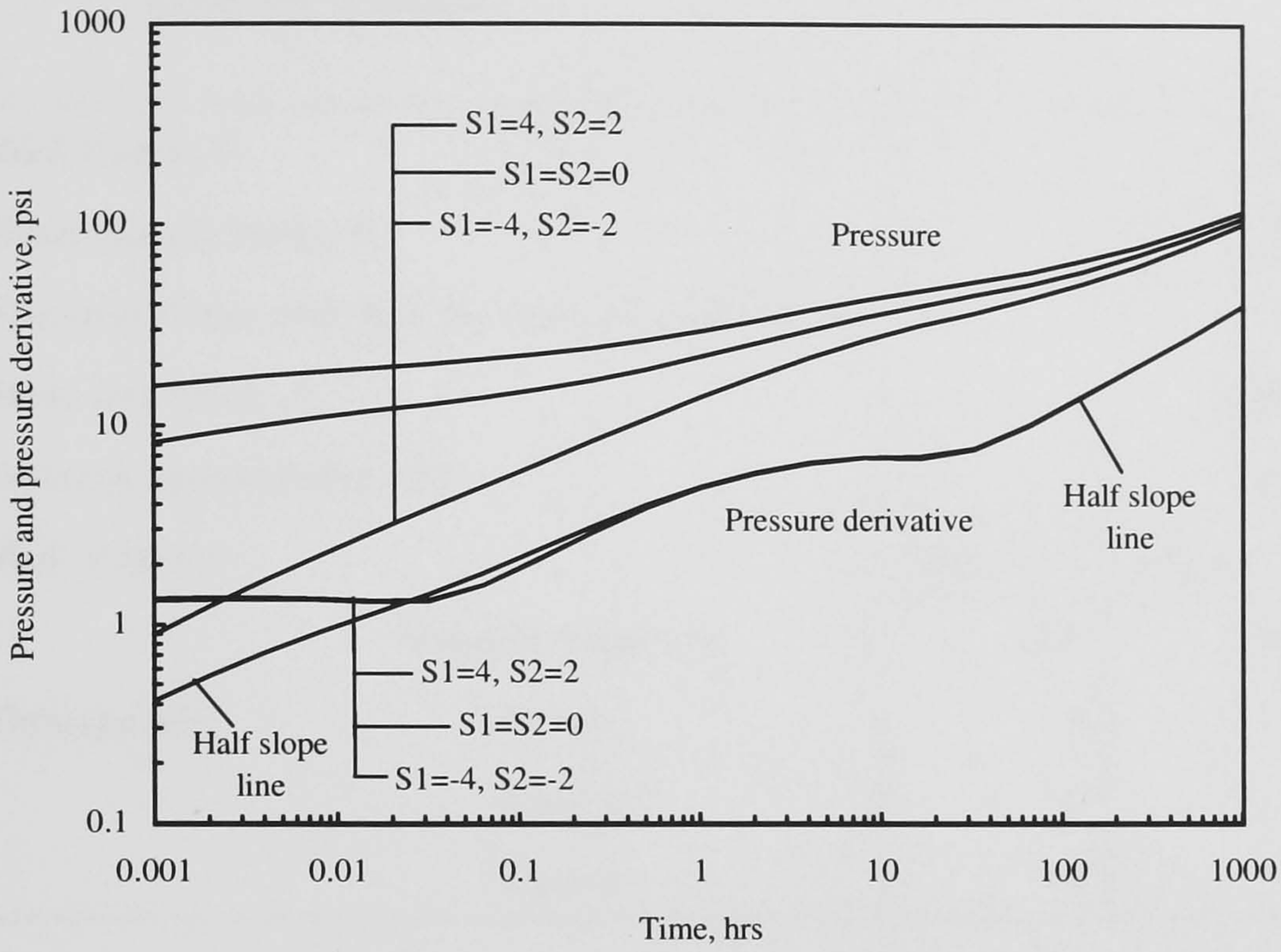


Figure 5.12: Pressure response of a horizontal well with varying skin along the wellbore in the linear homogeneous reservoir.

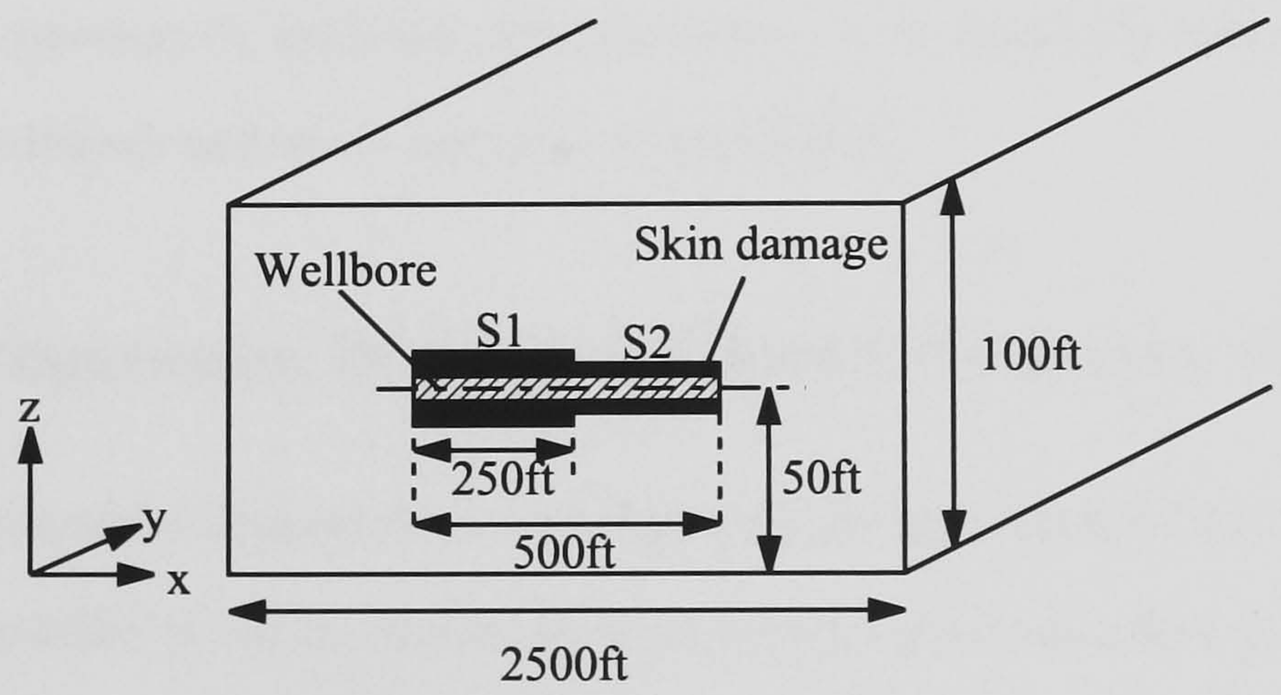


Figure 5.13: Schematic linear reservoir model for a well with the varying skin damage along the wellbore.

Table 5.5: Reservoir and well data for a well with the varying skin damage along the wellbore.

Well length, ft			500
Reservoir thickness, ft			100
Distance of the well from the reservoir bottom, ft			50
Reservoir width, ft			2,500
Isotropic permeability, md			100
Well segment		Segment 1	Segment 2
	Segment length, ft	250	250
Damage skin	Case 1	4	2
	Case 2	0	0
	Case 3	-4	-2

doesn't show the early-time radial flow period. Instead, linear flow develops represented by the half-slope line of the pressure derivative. However, all the derivative curves become identical as the time increases, and the pseudo-radial flow and the late-time linear flow which are similar to those of a homogeneous linear reservoir can be observed. The capability of modelling the early-time linear flow is one of the advantageous points of this model over the line-source wellbore approximation. It is especially useful when pressure analysis of an acidised carbonate reservoir is performed.

5.4 Heterogeneous Box-Type Closed Reservoir Case

The model developed in Chapter 4 is for an infinitely elongated reservoirs in the y direction. However, it is possible to obtain a solution for box-type closed reservoirs whether the well is vertical or horizontal. A schematic model of the horizontal well case is shown in Figs. 5.14 and 5.15. Issaka and Ambastha [54] showed the effect of a non-centred horizontal well in a homogeneous box-type reservoir. However, the heterogeneous case has not been discussed before.

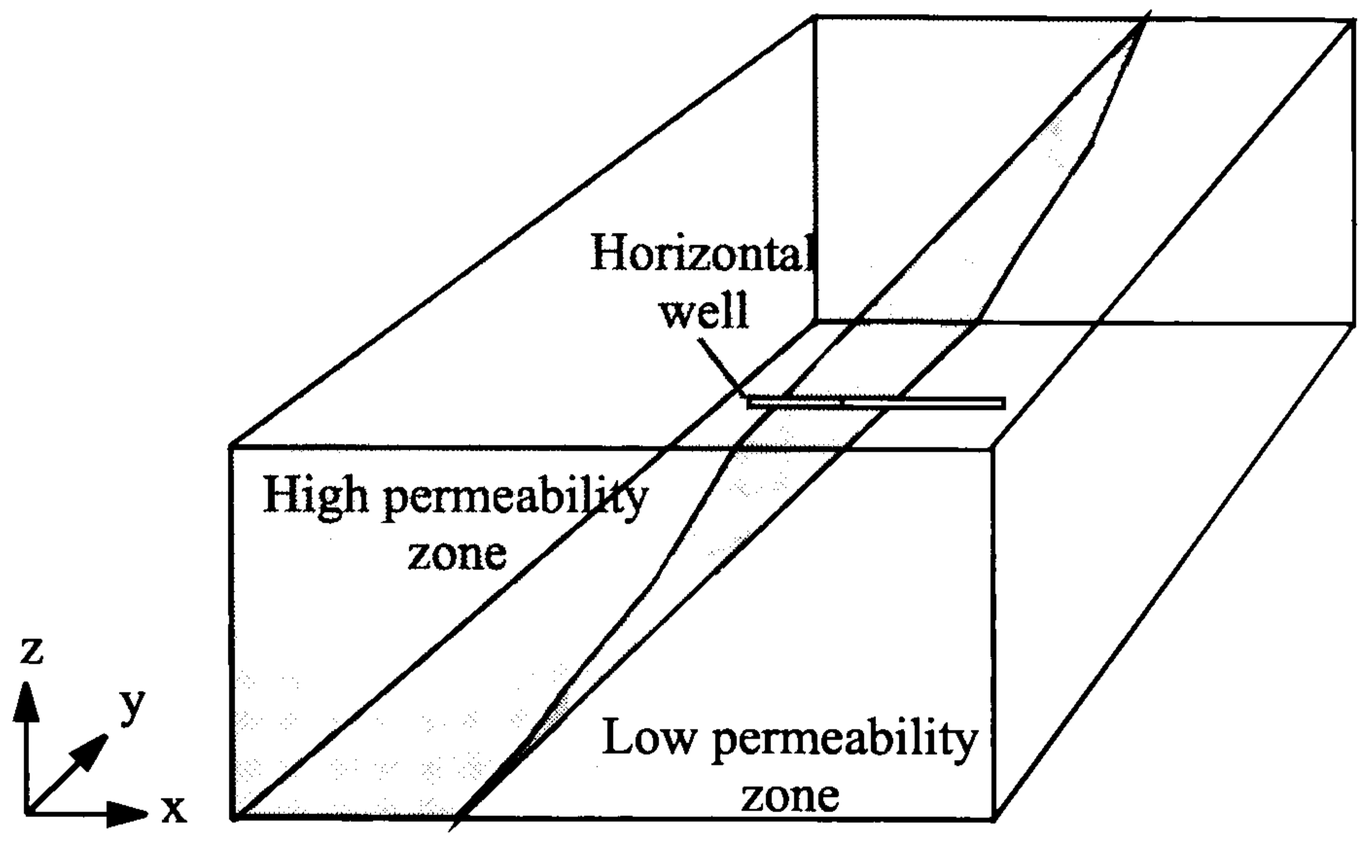


Figure 5.14: Schematic box-type reservoir with a horizontal well penetrating heterogeneous zones.

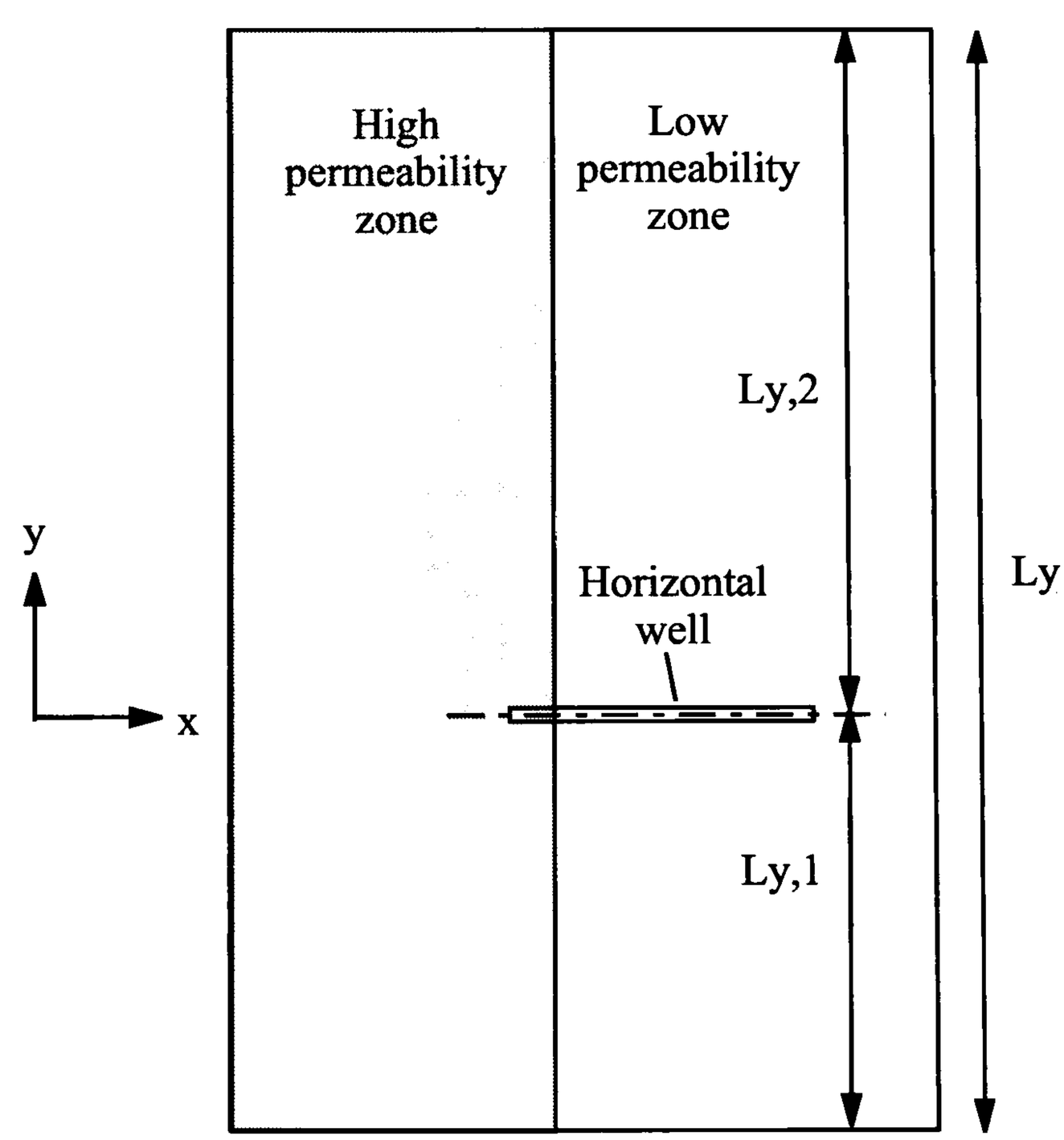


Figure 5.15: Coordinates for the box-type reservoir with a horizontal well penetrating heterogeneous zones.

For this case, no-flow boundary conditions for the $x-z$ planes at $y=-L_{y,1}$, and $L_{y,2}$ can be expressed as

$$\left.\frac{\partial p_{i,j}}{\partial y}\right|_{y=-L_{y,1}} = 0, \tag{5.1}$$

$$\left.\frac{\partial p_{i,j}}{\partial y}\right|_{y=L_{y,2}} = 0, \tag{5.2}$$

for $1 \leq i \leq m, 1 \leq j \leq n$.

For the infinitely elongated reservoir case, the infinite Fourier cosine transform was used in order to remove the partial derivative on the y direction (Eq. 4.117). On the other hand, the finite Fourier cosine transform [17] should be used for the box-type reservoir case. In this case, the dimensionless distance in the y direction is defined by

$$\hat{y}_D = \frac{\pi}{L_y}(y + L_{y,1}), \tag{5.3}$$

where L_y is the length of the box reservoir in the y direction. This procedure is similar to that presented by Bourgeois et al. [15] for their three-zone linear composite reservoirs. The horizontal wellbore should be modelled as a plane sink, and the wellbore boundary condition of Eq. 4.59 should be expressed using the Dirac delta function [91].

The rest of the calculation is similar to that for the infinitely elongated reservoir case except that the following inverse finite cosine transform should be used instead of the inverse infinite Fourier transform given by Eq. 4.153, i.e.,

$$\overline{p}_{D,i,j} = \frac{F_0}{\pi} + \frac{2}{\pi}\sum_{\tau=1}^{\infty} F_\tau \cos(\tau \hat{y}_D), \tag{5.4}$$

for $1 \leq i \leq m, 1 \leq j \leq n$ where F_τ is the Fourier transformed pressure expressed as

$$F_\tau = \int_0^\pi \overline{p}_{D,i,j} \cos(\tau \hat{y}_D)\, d\hat{y}_D. \tag{5.5}$$

The wellbore pressure can be obtained in the same way as the infinitely elongated reservoir case.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

The objective of this study is to develop new methods for pressure analysis in heterogeneous reservoirs. Since reservoirs are rarely homogeneous, the pressure analysis of heterogeneous reservoirs has been become an important issue. The effects of lens reservoirs, as found in the fluvial environments, and heterogeneous linear reservoirs, possible in a wide range of environments (e.g., fluvial, barrier island, or turbidite environments), have been studied in this thesis.

6.1 High Permeability Lens Intersected by a Vertical Well

In the lens reservoir case, it was found that the multi-layer composite reservoir model developed by Gomes and Ambastha [41] can be directly used for generating the synthetic pressure and flow rate response of the lens reservoir. The behaviour of the reservoir containing a high permeability lens was examined, and the following new insights were reported in this thesis.

1. The multi-layer composite model is useful for pressure analysis of reservoirs including a lens or lenses. Layer refinement is necessary if the crossflow between the lens and the matrix is significant. The computation time is much smaller than the numerical simulation and the reservoir modelling is easier.

2. The early time pressure drawdown response of the lens reservoir is identical to that of the layered reservoir with crossflow. However, once the pressure disturbance reaches the edge of the lens, convergence flow into the lens begins. Finally, pseudo-radial flow can be observed. When the lens is thin and the permeability contrast between the lens and the matrix is high enough, the behaviour approaches that of the infinite-conductivity horizontally fractured reservoir.

3. The presence of skin, wellbore storage, or non-Darcy effect may mask the early time period (i). In this case, the pressure response becomes similar to a homogeneous reservoir with the reservoir matrix permeability and a constant skin. This causes difficulties when we analyse real data from this kind of reservoir.

4. Sensitivities to the various dimensionless variables on pressure derivative curve were investigated. Such type-curves provide us useful information when the interactive pressure-pressure derivative type-curve matching is performed.

5. The fractional flow rate from the lens approaches the value calculated for the cross-flow layered reservoir as the lens radius increases. If the permeability contrast between the lens and the matrix is large, the lens radius required to reach the value of a layered reservoir becomes large.

6. Even when the lens shape is not circular or the well position is not in the centre of the lens, the similar behaviour to that for a circular lens with the well in the centre can be observed. The pressure behaviour of a lens elongated in one direction with the aspect ratio up to 1:5 and an arbitrary well position can be approximated by that of a circular lens penetrated by a well in the centre with the same equivalent lens radius. The same pseudo-skin can be obtained for the rectangular lenses with the same equivalent lens radius and the same lens thickness if the aspect ratio of the lens is below 1:5.

7. For a given lens volume, the well performance of a short thick lens is better than that of a long thin lens in terms of the pseudo-skin calculated in the pseudo-radial flow period.

8. The influence of the lens away from the well becomes small as the distance from the well to the lens increases. The negative pseudo-skin obtained by the analysis in the late-time pseudo-radial flow period approaches zero as the distance increases (when the damage skin effect is ignored). It is critical to penetrate part of the lens since the absolute value of the negative pseudo-skin for the penetrated lens case increases remarkably as compared with the non-penetrated lens case.

9. The method to calculate the well productivity for the lens reservoir was presented. It is only applicable if the pseudo-radial flow period (iv) exists since the pseudo-skin must be calculated from this flow period.

10. A real field case was analysed and the size of the lens was estimated. The presence of the lens reduced the skin and improved the well productivity. The damage skin calculated with the model (5.0) was significantly higher than the pseudo-skin estimated from the pseudo-radial flow interpretation (0.2). This confirms that the well is more productive due to the presence of the lens.

11. The existence of several lenses complicates the pressure response. For such cases, characteristic flow periods may not be apparent and the interactive or the automatic pressure-pressure derivative type-curve matching is necessary to analyse the pressure data.

12. In multiple lens reservoir case where the rock properties are the same for each lens, the thickness weighted arithmetic averaged lens radius defined by Eq. 3.132 can be roughly estimated by the single-lens approximation if the pseudo-radial flow period can be observed at late times. This observation is valid for the non-circular lens case and the case where the well is not located in the centre of the lenses if the aspect ratio of the lenses are below 1:5.

In this thesis, a comprehensive study of the pressure and the fractional flow rate behaviours in the lens reservoir was attempted. However, the following topics were not completed and are recommended for further work.

1. Early-time or late-time limiting form of the analytical solutions for the pressure and the fractional flow rate behaviours should be investigated. In particular, the limiting form for the late-time pressure may be useful when the productivity index for this reservoir is required.

2. Since the pressure behaviour is highly complicated (especially when the reservoir contains several lenses), an automatic type-curve matching program should be implemented.

3. Although a field example was presented in this study, the insights obtained should be established further by applying them to many field examples.

6.2 Horizontal Well in Discontinuous Permeability Reservoirs

In the heterogeneous reservoir case, the pressure and the fractional flow rate behaviour of a horizontal well in a reservoir with two permeability regions was analysed. Although the interpretation of the results is not easy due to the complexity of the system, the following conclusions were reached.

1. A model which can calculate pressure and the flow rate responses of a horizontal well in a heterogeneous linear reservoir was developed using successive Laplace and Fourier transforms. The semi-permeable wall model and the strip wellbore model were adopted for the development. The strip wellbore model allows us to deal with a negative skin and model the early-time linear flow, as may occur in carbonate reservoirs if Eq. 4.67 is satisfied. The validity was confirmed thorough comparisons with numerical simulation and with published results.

2. Grid refinement is necessary even though the reservoir is homogeneous. A sensitivity check for the grid refinement is required to obtain a reliable pressure or flow rate response.

3. For the heterogeneous reservoir case, a straight line of the pressure derivative can be observed on semi-log or log-log pressure versus time plots during the early time period, which is similar to the homogeneous cases. The averaged permeability given by Eq. 4.165 can be calculated from this line.

4. The half-slope line of the pressure derivative curve can be observed on log-log pressure versus time plots during the late time period. From this line, the averaged permeability in the y direction, given by Eq. 4.164, can be calculated.

5. It is very difficult to see typical flow regimes at middle times since crossflow occurs in addition to the intermediate-time linear flow regime or the pseudo-radial flow regime. The data in the intermediate time period can be analysed only by interactive or automatic pressure-pressure derivative type-curve matching procedure.

6. The fractional flow rate transition from the high permeability zone shows a variety of responses according to the various reservoir parameters. At early times, before the pressure disturbance reaches the top or the bottom reservoir boundary, or the flow in the x direction affects the rate response, the rate approaches the value calculated by Eq. 4.166 since the behaviour is the same as that for a vertical well case in crossflow reservoirs. At late times, the rate also becomes constant. However, a simple expression to predict it has not been found yet.

7. The combined type-curve matching of the pressure-pressure derivative and the fractional flow rate transition is important for the estimation of reservoir properties to reduce the uncertainty of the reservoir model.

8. The heterogeneous linear reservoir model is quite versatile. The model can be also used to generate synthetic pressure or flow rate response of a vertical well, a fractured well with infinite conductivity, a damaged or stimulated well, all in heterogeneous linear reservoirs. In addition, the box-type heterogeneous reservoirs can be modelled using the finite Fourier transform.

The further developments of the heterogeneous linear reservoir model were outlined in Chapter 5. Some of the models are easily obtained with minor modification. The followings are recommendations to be done as the next step of this study.

1. The model should be extended in order to generate the synthetic pressure response for the irregular boundary linear reservoirs. The effect of the irregular boundary should be examined.

2. The computer program for the box-type heterogeneous linear reservoir should be implemented. The effect of the well position on pressure behaviour, well productivity, etc. should then be investigated.

3. Since the pressure behaviour is highly complicated, an automatic type-curve matching program should be implemented.

4. The computer program developed during this work (especially, the multiple precision arithmetic program) should be rewritten more efficiently to further reduce computation time.

5. The model developed here should be applied to actual field examples to show its effectiveness and an actual procedure to analyse the behaviour of the heterogeneous linear reservoirs.

NOMENCLATURE

A	Drainage area
$A_{i,j}$, $B_{i,j}$	Constants in Eq. 3.67 and Eq. 4.125 determined by boundary conditions
A_i^k, B_i^k	Constants in Eq. 3.78 and Eq. 3.79 determined by boundary conditions
A_j^k, B_j^k	Constants in Eq. 4.136 and Eq. 4.137 determined by boundary conditions
$A_{i,j}^k$, $B_{i,j}^k$	Constants in Eq. 3.71 and Eq. 4.129 determined by boundary conditions
$\mathbf{A}_i^k$, $\mathbf{B}_i^k$	Constant vectors defined by Eq. 3.75 and Eq. 3.76
$\mathbf{A}_j^k$, $\mathbf{B}_j^k$	Constant vectors defined by Eq. 4.133 and Eq. 4.134
B	Fluid formation volume factor
c_t	Total compressibility
C	Wellbore storage constant
C_A	Diets shape factor
$C_{i,j}$	Particular solutions of Eq. 4.116
$D_{i,j}$	Laplace pressure gradients defined by Eq. 4.110
$E_{i,j}^k$	Constants defined by the recursive formula of Eq. 3.77 or Eq. 4.135
$E_{i,j}^{k\ \prime}$	Constants defined by Eq. 3.82 and Eq. 3.83 or Eq. 4.139 and Eq. 4.140
$g_{i,i'}^j$	Matrix elements of $\mathbf{G}_j$ defined by Eq. 4.128
$g_{i,i'}^{j,k}$	Matrix elements of $\mathbf{G}_j^k$ defined by Eq. 4.132

$g^i_{j,j'}$	Matrix elements of $\mathbf{G}_i$ defined by Eq. 3.70
$g^{i,k}_{j,j'}$	Matrix elements of $\mathbf{G}^k_i$ defined by Eq. 3.74
$\mathbf{G}_i$	$n \times n$ matrix for the zone i defined by Eq. 3.70
$\mathbf{G}^k_i$	$n \times n$ matrix for the zone i defined by Eq. 3.74
$\mathbf{G}_j$	$m \times m$ matrix for the layer j defined by Eq. 4.128
$\mathbf{G}^k_j$	$m \times m$ matrix for the layer j defined by Eq. 4.132
h	Thickness
$i_{w,1}$, $i_{w,2}$	Indices for the left-most well segment and the right-most well segment
$\mathbf{I}$	Identity matrix
$I_0(x)$, $I_1(x)$	Modified Bessel functions of the first kind, order zero and order one
$j_{w,1}(i)$, $j_{w,2}(i)$	Indices for the bottom-most well grid and the top-most well grid in the well segment i
J	Well productivity index
k_h, k_l	Permeabilities of the high permeability zone and the low permeability zone
k_x, k_y, k_z	Principal permeabilities in the Cartesian coordinates for the heterogeneous linear reservoir case
$\overline{\sqrt{k_y k_z}}$	Average permeability defined by Eq. 4.73
k_r, k_z	Principal permeabilities in the cylindrical polar coordinates for the lens reservoir case
$K_0(x)$, $K_1(x)$	Modified Bessel functions of the second kind, order zero and order one
l	Laplace variable
l_h	Well length penetrating the high permeability region
L_w	Horizontal well length
m, n	Number of the grid blocks

m_{hr}	Slope of the straight line on pressure versus logarithmic time plots during the hemi-radial flow period
m_{1l}	Slope of the straight line on pressure versus square-root time plots during the intermediate-time linear flow period
m_{1r}	Slope of the straight line on pressure versus logarithmic time plots during the early-time radial flow period
m_{2l}	Slope of the straight line on pressure versus square-root time plots during the late-time linear flow period
m_{2r}	Slope of the straight line on pressure versus logarithmic time plots during the pseudo-radial flow period
$M_{i,j}$	Mobility ratio between the grids (i, j) and $(i+1, j)$ in Eq. 3.15
N	Even integer used in Stehfest algorithm
p	Pressure
p_{in}	Initial pressure
p_{wbs}	Pressure with wellbore storage effect
p_{wf}	Wellbore flowing pressure
p_{ws}	Wellbore shut-in pressure
$\langle p \rangle$	Vertically averaged pressure of Eq. 3.16 or horizontally averaged pressure of Eq. 4.43
q	Production rate
q_i, q_j	Production rates of the well segment i and of the layer j
r, z	Cylindrical polar coordinates for the lens reservoir case
r_w	Wellbore radius
$r_w{}'$	Equivalent wellbore radius
$\tilde{r}_l$	Equivalent radius for the non-circular lens case
$\hat{r}_l$	Equivalent radius for the multi-lens case
s, S	Skin factors
S_m	Mechanical damage skin

S_x	Skin due to the partial penetration in the x direction
S_z	Skin due to the partial penetration in the z direction
t	Time
t_{brf2}	Time at which the pseudo-radial flow begins
t_{elf1}	Time at which the intermediate-time linear flow ends
t_{erf1}	Time at which the early-time radial flow ends
t_{erf2}	Time at which the pseudo-radial flow ends
t_p	Producing time
V_i	Constants defined by Eq. 3.97 in the Stehfest algorithm
w_h	Width of the high permeability zone in the x direction
w_x	Width of the reservoir in the x direction
W	Strip wellbore width
x, y, z	Cartesian coordinates for the heterogeneous linear reservoir case
$\hat{y}$	Distance in the y direction for the box-type reservoir case

GREEK SYMBOLS

$\alpha_{i,j}$	Dimensionless flow capacity in the y direction defined by Eq. 4.82
$\beta_{i,j}$	Dimensionless flow capacity in the z direction defined by Eq. 4.83
γ	Euler's constant
δx_i	Block size defined by Eq. 4.44
δz_j	Block size defined by Eq. 3.17 or Eq. 4.60
$\delta z_{w,i}$	Wellbore strip width defined by Eq. 4.57
Δt	Shut-in time
$\zeta_{i,j}$	Permeability ratio between the grids (i,j) and $(i,j+1)$ defined by Eq. 4.41
$\kappa_{i,j}$	Dimensionless flow capacity defined by Eq. 3.32
$\lambda_{i,j}$	Interlayer or interzone flow parameter defined by Eq. 3.20 or Eq. 4.46

$\tilde{\lambda}_{i,j}$	Dimensionless interlayer or interzone flow parameter defined by Eq. 3.34 or Eq. 4.85
μ	Fluid viscosity
σ_i^k	Square root of the eigenvalues obtained from Eq. 3.69
σ_j^k	Square root of the eigenvalues obtained from Eq. 4.127
τ	Fourier variable
$\phi_{i,j}$	Porosity
$\overline{\phi c_t}$	Average storage capacity defined by Eq. 4.76
$\Psi(\eta)$	Spence function defined by Eq. 4.16
$\omega_{i,j}$	Dimensionless storage capacity defined by Eq. 3.33 or Eq. 4.84

SUBSCRIPTS

BU	Buildup
D	Dimensionless
DD	Drawdown
h	Horizontal
i, j	Indices for the grid blocks
l	Properties of the lens
m	Properties of the reservoir matrix
sf	Sand face
v	Vertical

SUPERSCRIPTS

ET	Early-time
LT	Late-time
$\overline{p}$	Laplace transform of p
$\overline{\overline{p}}$	Successive Laplace-Fourier transform of p

APPENDIX A

COMPUTER PROGRAM FOR LENS RESERVOIRS

In appendix A, a computer program for generating synthetic pressure or flow rate response of a reservoir including a high permeability lens is presented. The mathematical formulation is based on the work by Gomes and Ambastha [41] as shown in Section 3.1. C language was used for the implementation of the algorithm.

A.1 Description of the Program

The model allows to calculate pressure responses in many situations (Fig. A.1), i.e.,

- Infinite, no-flow, or constant pressure outer boundary conditions. These boundary conditions can be assigned independently for each layer.
- No-flow or constant pressure top and bottom boundary conditions. These boundary conditions can be assigned independently in each grid block of the top or bottom layer.
- Partially completed wellbore
- Skin and wellbore storage effects

Pressure buildup responses can be generated by superposition of the pressure drawdown responses by using Eqs. 3.110 and 3.112.

The following procedures are implemented in this program.

1. The eigenvalues $(\sigma_i^k)^2$ for each zone i are calculated from Eq. 3.69 using QL algorithm with implicit shifts [95]. The constant $E_{i,j}^k$ is computed using the recursive formula of Eq. 3.77.
2. $2mn$ simultaneous equations are set up from the boundary conditions and solved for the constants A_i^k and B_i^k using Gauss-Jordan elimination with full pivoting [95]. Modified Bessel functions are evaluated using algorithms based on polynomial coefficients given by Abramowitz and Stegun [4].
3. The wellbore pressure or the fractional flow rate in the Laplace domain is calculated using Eq. 3.94 or Eq. 3.95. The wellbore storage effect can be taken into account

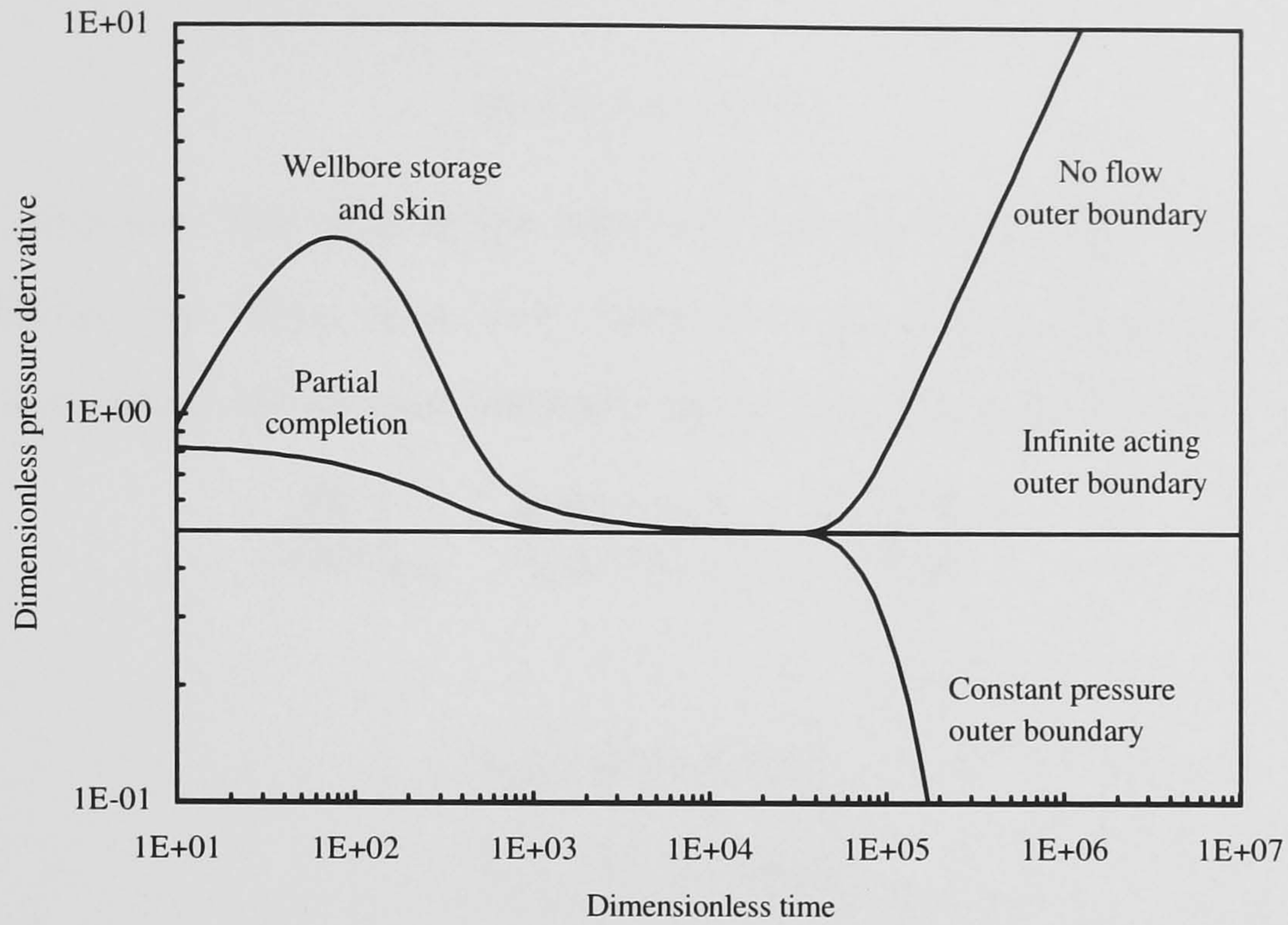


Figure A.1: Various boundary conditions implemented in the program.

using Eq. 3.61.

4. The wellbore pressure or the flow rate is inverted to the real domain using the Stehfest algorithm [111]. The constant N in Eq. 3.96 was set to 8 for both drawdown and buildup cases.

5. For pressure buildup tests, the superposition of pressure is performed using Eq. 3.110.

If the argument of modified Bessel functions is very small or very large, an overflow or underflow problem my occur during the computation. To prevent this, a dimensionless radius based on the minimum block radius instead of the wellbore radius is used following the suggestion by Gomes and Ambastha [41].

In this program, the pressure derivative is computed using rational function interpolation of pressure [95]. The rational function $R_{i(i+1)\ldots(i+m)}$ passing through the $m+1$ points (x_i, y_i), (x_{i+1}, y_{i+1}), $\ldots$, (x_{i+m}, y_{i+m}) is given by

$$R_{i(i+1)\ldots(i+m)} = \frac{A_\mu(x)}{B_\nu(x)} = \frac{a_0 + a_1 x + \cdots + a_\mu x^\mu}{b_0 + b_1 x + \cdots + b_\nu x^\nu}, \tag{A.1}$$

where

$$m + 1 = \mu + \nu + 1, \tag{A.2}$$

and b_0 is arbitrary. The order of the numerator and the denominator must be given to specify the function. Here, m, μ, and ν were set to 4, 2, and 2, respectively. Once the function is obtained, the pressure derivative at the time t_i may be computed by

$$\left.\frac{dp}{d\ln t}\right|_{t=t_i} = \frac{p_{i+1} - p_{i-1}}{t_{i+1} - t_{i-1}} t_i = \frac{p_{i+1} - p_{i-1}}{2\Delta t}, \tag{A.3}$$

where

$$t_{i+1} = t_i + t_i \Delta t, \tag{A.4}$$

$$t_{i-1} = t_i - t_i \Delta t, \tag{A.5}$$

and $\Delta t \ll 1$. In this program, Δt was set to 10^{-5}.

A.2 Variables and Functions

In this section, variables and functions used in the computer program are explained. The principal variables are listed in Table A.1 with their descriptions.

The function names and the explanations are shown below.

- **int main()**: This function calls other functions and defines the procedure of the calculation.
- **void read_data()**: Reservoir, well, fluid data, and time interval to be calculated are read from an input file.
- **void print_data()**: The input data are printed on the screen.
- **void field_to_darcy()**: The units of input data (field units) are transformed to the Darcy units.
- **void dimless_var()**: The dimensionless variables ($\kappa_{i,j}$, $\omega_{i,j}$, $\tilde{\lambda}_{i,j}$ of Eqs. 3.32, 3.33, 3.34, the dimensionless radius r_D of Eq. 3.29, the mobility ratio $M_{i,j}$ in Eq. 3.15, the dimensionless wellbore storage constant C_D of Eq. 3.38) are defined.

Table A.1: Variable table for the program.

Variable name	Description
M, N	The number of grid blocks in the r and z directions
test_type	Test type, 0=Drawdown test, 1=Buildup test
calc_type	Type of calculation, 0=Pressure, 1=Fractional flow rate
output_units	Output units, 0=Dimensionless units based on the original paper by Gomes and Ambastha [41], 1=Field units, 2=Dimensionless units calculated from Eqs. 3.27 and 3.28
im, jm	Index of a reservoir matrix block for calculating dimensionless units. The dimensionless pressure and time are calculated based on the properties of this block.
lap_count	The number of time steps in Laplace domain
real_count	The number of time steps in real domain
layer_q	Layer number to be used for the fractional flow rate calculation
outer_bc[j]	Outer boundary condition for the layer j, 0=Infinite outer boundary, 1=No-flow outer boundary, 2=Constant pressure outer boundary
top_bc[i]	Reservoir top boundary condition for the zone i, 0=No-flow boundary, 1=Constant pressure boundary
bottom_bc[i]	Reservoir bottom boundary condition for the zone i, 0=No-flow boundary, 1=Constant pressure boundary
part_comp[j]	Well completion for the layer j, 0=Not completed, 1=Completed
kh[i][j], kv[i][j]	Horizontal and vertical permeabilities for the block (i, j)
phi[i][j]	Porosity for the block (i, j)
r[i]	Radius of the concentric zone i
rd[i]	Dimensionless radius of the concentric zone i
h[j]	Thickness of the layer j
mu[i][j]	Fluid viscosity in the block (i, j)
ct[i][j]	Total compressibility for the block (i, j)
skin[j]	Skin factor for the layer j
K[i][j]	Mobility ratio between the blocks (i, j) and $(i + 1, j)$
kappa[i][j], omega[i][j], lambda[i][j]	Dimensionless parameters defined by Eqs, 3.32, 3.33, and 3.34.
sigma[i][k], E[i][j][k]	Square root of the eigenvalues of the system described by Eq. 3.68, and constants in Eqs. 3.78 and 3.79
A[i][j], B[i][j]	Constants of Eq. 3.67 for the block (i, j) determined by boundary conditions
q	Total flow rate
bo	Formation volume factor
cs	Wellbore storage constant
cd	Dimensionless wellbore storage constant
pin	Initial reservoir pressure
khmu, phich	Sum of the transmissibility and storage coefficient of the inner-most zone
tp	Producing time for buildup analysis
lap_t[k], lap_sol[k]	Time and solution (pressure or fractional flow rate) in Laplace domain
t[k], sol[k], dp[k]	Time, solution (pressure or fractional flow rate), and pressure derivative in real domain
min_t, max_t	Minimum and maximum times of the time interval to be calculated

- **void tstep_real()**: Time steps in the real domain are constructed between `min_t` and `max_t`. Each time step increases geometrically by the factor `R_GEOFAC` which is defined in the header file `vert.h`.

- **void tstep_laplace()**: Time steps in the Laplace domain which correspond to the real domain time steps are constructed.

- **void eigenvalues()**: The eigenvalues for each concentric zone is calculated from Eq. 3.69. $\mathbf{G_i}$ in Eq. 3.69 is a $n \times n$ real symmetric tridiagonal matrix. QL algorithm with implicit shifts [95] is used to obtain the eigenvalues.

- **void calc_E()**: The constants $E_{i,j}^k$ in Eqs. 3.78 and 3.79 are calculated using the recursive formula of Eq. 3.77.

- **void coef_ab()**: The coefficients $A_{i,j}$ and $B_{i,j}$ in Eq. 3.84 are calculated using boundary conditions. The system of equations with the dimension $2mn \times 2mn$ is set up, and is solved using Gauss-Jordan elimination with full pivoting [95] (the function `void gaussj()`).

- **double lap_sol_p()**: The pressure at the wellbore in Laplace domain is calculated. A table which consists of the Laplace time and the Laplace pressure is made.

- **double lap_sol_q()**: The fractional flow rate in Laplace domain for the selected layer is calculated. A table which consists of the Laplace time and the Laplace fractional flow rate is made.

- **void calc_pressure()**: The wellbore pressure in the real domain is calculated using the Stehfest algorithm [111]. The constant N in Eq. 3.96 is set to 8 for both the pressure drawdown case and the pressure buildup case.

- **void calc_pressure_derivative()**: The pressure derivative is computed using the rational function interpolation (the function `void ratint()`). The function `double diff_press()` provides the pressure derivative on logarithmic time using the finite

difference approximation of Eq. A.3. For the pressure buildup case, the derivative based on the Horner method (Eq. 3.112) is used instead.

- **void calc_rate()**: The fractional flow rate for the selected layer in the real domain is computed. The Stehfest algorithm is used for converting the flow rate in the Laplace domain.

- **void output_in_field_units()**: The units of the time, pressure, and pressure derivative are converted to the field units for output.

- **void output_in_mat_units()**: The units of the time, pressure, and pressure derivative are converted to the dimensionless units based on the reservoir matrix properties for output.

- **double bessi0(), double bessk0(), double bessi1(), double bessk1()**: These functions return the value of the modified Bessel functions $I_0(x)$, $K_0(x)$, $I_1(x)$, and $K_1(x)$, respectively, for given x.

A.3 Input Data

A sample input file is shown below with comments. The grid number is assigned for each grid block in the same way as shown in Fig. 3.1. The reservoir and the fluid parameters for each grid block should be entered from the left block to the right, and then the top block to the bottom. Field units must be used for data in the input file.

```
Sample input file                          \Title
0                                          \Test type: 0=Drawdown, 1=Buildup
1                                          \Type of calculation: 0=Pressure, 1=Fractional flow rate
2                                          \Output units: 0=Dimensionless(1), 1=Field, 2=Dimensionless(2)
9e-4 1e5                                   \Calculation interval, dimensionless or hrs
3 3                                        \Grid number: M x N
1                                          \Well layer for fractional flow rate calculation
1 3                                        \Matrix block indices used for Dimensionless(2)
100 100 100                                \kr, md
100 100 100
10000 10000 100
1 1 1                                      \kz, md
1 1 1
100 100 1
0.2 0.2 0.2                                \Porosity
```

```
0.2 0.2 0.2
0.2 0.2 0.2
0.2 1.0 1000 4000          \Well radius and zone radius, ft
30 10 10                   \Layer thickness, ft
500                        \Total flow rate, bbl/d
1.5                        \Formation volume factor
1.0 1.0 1.0                \Viscosity, cp
1.0 1.0 1.0
1.0 1.0 1.0
1e-5 1e-5 1e-5             \Compressibility, 1/psi
1e-5 1e-5 1e-5
1e-5 1e-5 1e-5
3000                       \Initial pressure, psi
30.0                       \Producing time before shut-in, hrs
0.0                        \Wellbore storage, bbl/psi
0.0 0.0 0.0                \Skin factor
0 0 0                      \Outer boundary condition: 0=Infinite, 1=No-flow, 2=Constant pressure
0 0 0                      \Reservoir top boundary condition: 0=No-flow, 1=Constant pressure
0 0 0                      \Reservoir bottom boundary condition: 0=No-flow, 1=Constant pressure
1 1 1                      \Well completion: 0=Not completed, 1=Completed
```

A.4 Header File

This header file (`vert.h`) defines constants, macros, and the prototypes of functions. This file is included in the main program at its beginning.

vert.h

```
/*
Header file for the wellbore pressure
calculation in lens reservoirs
Programmed by Atsushi Sagawa
*/

/*** pi ***/
#define PI 3.141592653589793
/*** length of a file name ***/
#define MAXLEN 30
/*** maximum block size ***/
#define MAXI 10
#define MAXJ 10
/*** maximum time step ***/
#define MAX_TIME_STEP 1000
/*** time step increment ***/
#define R_GEOFAC 1.5
/*** constants for Stehfest's algorithm ***/
#define NSTEH_DD 8
#define NSTEH_BU 8
/*** small value for derivative calculation ***/
#define DT 1.0e-5
#define TINY 1.0e-20
/*** small value used for table pick up ***/
#define D_EPS 1e-5
/*** positive sign ***/
#define SIGN(a, b) ((b) ≥ 0.0 ? fabs(a) : - fabs(a))
/*** square ***/
```

```
static double sqrarg;
#define SQR(a) ((sqrarg = (a)) == 0.0 ? 0.0 : sqrarg * sqrarg)
/*** swap two values ***/
#define SWAP(a, b) {temp = (a); (a) = (b); (b) = temp;}
/*** error exit ***/
#define ERROR_EXIT(s) {printf("%s\n", s); exit(0);}

/*
prototypes of functions
*/
void read_data(int *test_type, int *calc_type, int *output_units,
               int *layer_q, int *M, int *N, double kh[][MAXJ],
               double kv[][MAXJ], double phi[][MAXJ],
               double r[], double h[], double *q,
               double *bo, double mu[][MAXJ],
               double ct[][MAXJ], double *pin, double *cs,
               double skin[], int outer_bc[], int top_bc[],
               int bottom_bc[], int part_comp[], double *tp,
               double *min_t, double *max_t, int *im, int *jm);
void print_data(int test_type, int calc_type, int output_units,
                int layer_q, int M, int N, double kh[][MAXJ],
                double kv[][MAXJ], double phi[][MAXJ],
                double r[], double h[], double q,
                double bo, double mu[][MAXJ], double ct[][MAXJ],
                double pin, double cs, double skin[],
                int outer_bc[], int top_bc[], int bottom_bc[],
                int part_comp[], double tp, double min_t,
                double max_t, int im, int jm);
void dimless_var(int M, int N, double kh[][MAXJ],
                 double kv[][MAXJ], double phi[][MAXJ],
                 double r[], double h[], double mu[][MAXJ],
                 double ct[][MAXJ], double cs,
                 double kappa[][MAXJ], double omega[][MAXJ],
                 double lambda[][MAXJ+1], double *cd,
                 double rd[], double K[][MAXJ],
                 double *khmu, double *phich, int top_bc[],
                 int bottom_bc[], double *tp, double *min_t,
                 double *max_t);
void eigenvalues(int M, int N, double kappa[][MAXJ],
                 double omega[][MAXJ], double lambda[][MAXJ+1],
                 double lap_t, double sigma[][MAXJ]);
void calc_E(int M, int N, double kappa[][MAXJ],
            double lambda[][MAXJ+1], double omega[][MAXJ],
            double lap_t, double sigma[][MAXJ],
            double E[][MAXJ][MAXJ]);
void coef_ab(int M, int N, double kappa[][MAXJ], double sigma[][MAXJ],
             double E[][MAXJ][MAXJ], double lap_t, double rd[],
             double skin[], double K[][MAXJ], double A[][MAXJ],
             double B[][MAXJ], int outer_bc[], int part_comp[]);
void gaussj(int n, double a[][2*MAXI*MAXJ], double b[]);
double lap_sol_p(int N, double A[][MAXJ], double B[][MAXJ],
                 double E[][MAXJ][MAXJ], double sigma[][MAXJ],
                 double rd[], double skin[], int part_comp[]);
double lap_sol_q(int N, double A[][MAXJ], double B[][MAXJ],
                 double E[][MAXJ][MAXJ], double sigma[][MAXJ],
                 double rd[], int layer_q);
int lap_table(int lap_count, double lap_t[], double s);
void output_in_field_units(int real_count, double q, double r[],
                           double pin, double khmu, double phich,
                           double t[], double sol[], double dp[],
                           double bo);
void output_in_mat_units(int N, int real_count, double q, double r[],
                         double pin, double khmu, double phich,
                         double t[], double sol[], double dp[],
                         double bo, double kh[][MAXJ], double h[],
                         double mu[][MAXJ], double ct[][MAXJ],
                         double phi[][MAXJ], int im, int jm);
void field_to_darcy(int M, int N, double kh[][MAXJ],
                    double kv[][MAXJ], double r[],
```

```
                    double h[], double *q,
                    double ct[][MAXJ], double *pin,
                    double *cs, double *tp, double *min_t,
                    double *max_t);
void calc_pressure(int test_type, int lap_count, int real_count,
                   double lap_sol[], double lap_t[], double sol[],
                   double t[], double tp);
void calc_rate(int test_type, int layer_q, int lap_count,
               int real_count, double lap_sol[], double lap_t[],
               double sol[], double t[], double min_t, double max_t,
               double rd[], double khmu, double kh[][MAXJ],
               double h[], double mu[][MAXJ]);
void calc_pressure_derivative(int test_type, int real_count, double sol[],
                              double dp[], double t[], double tp);
double diff_press(int i, int real_count, double sol[], double t[]);
void ratint(double xa[], double ya[], int r, double x, double *y,
            double *dy);
void tstep_real(int test_type, int *real_count, double t[],
                double min_t, double max_t);
void tstep_laplace(int test_type, int real_count, int *lap_count,
                   double lap_t[], double t[], double tp);
double factrl(int n);
double bessi0(double x);
double bessk0(double x);
double bessi1(double x);
double bessk1(double x);
```

A.5 Main Program

The main program (`vert.c`) is listed below.

vert.c

```
/*
Main program for wellbore pressure
calculation in lens reservoirs
Programmed by Atsushi Sagawa
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "vert.h"

main()
{
    int i;
    /*** block size ***/
int M, N;
/*** test type, 0:drawdown, 1:buildup ***/
int test_type;
/*** calculation type, 0:pressure, 1:flow rate ***/
int calc_type;
/*** output units,                                    ***/
/*** 0:dimensionless based on the averaged properties, ***/
/*** 1:field,                                         ***/
/*** 2:dimensionless based on the matrix properties   ***/
int output_units;
/*** counter of arrays ***/
int lap_count, real_count;
/*** layer where flow rate is calculated ***/
int layer_q;
/*** reservoir matrix block ***/
```

```
int im, jm;
/*** outer boundary condition, ***/
/*** 0:infinite,               ***/
/*** 1:no flow,                ***/
/*** 2:constant presure        ***/
int outer_bc[MAXJ];
/*** top boundary condition,         ***/
/*** 0:no flow, 1:constant pressure ***/
int top_bc[MAXI];
/*** bottom boundary condition,      ***/
/*** 0:no flow, 1:constant pressure ***/
int bottom_bc[MAXI];
/*** well completion, ***/
/*** 0:not completed, 1:completed ***/
int part_comp[MAXJ];
/*** horizontal and vertical permeabilities ***/
double kh[MAXI][MAXJ], kv[MAXI][MAXJ];
/*** porosity ***/
double phi[MAXI][MAXJ];
/*** block radius ***/
double r[MAXI+1];
/*** block height ***/
double h[MAXJ];
/*** viscosity ***/
double mu[MAXI][MAXJ];
/*** compressibility ***/
double ct[MAXI][MAXJ];
/*** skin factor ***/
double skin[MAXJ];
/*** horizontal mobility ratio ***/
double K[MAXI][MAXJ];
/*** dimensionless parameters ***/
double kappa[MAXI][MAXJ], omega[MAXI][MAXJ];
double lambda[MAXI][MAXJ+1];
/*** eigenvalues and eigenvectors ***/
double sigma[MAXI][MAXJ], E[MAXI][MAXJ][MAXJ];
/*** dimensionless radius ***/
double rd[MAXI+1];
/*** constants A and B ***/
double A[MAXI][MAXJ], B[MAXI][MAXJ];
/*** total flow rate ***/
double q;
/*** formation volume factor ***/
double bo;
/*** wellbore storage ***/
double cs;
/*** reservoir initial pressure ***/
double pin;
/*** dimensionless wellbore storage ***/
double cd;
/*** total transmissibility and storage coefficient ***/
double khmu, phich;
/*** producing time for buildup ***/
double tp;
/*** Laplace time and solution ***/
double lap_t[MAX_TIME_STEP], lap_sol[MAX_TIME_STEP];
/*** Real time solution and pressure derivative ***/
double sol[MAX_TIME_STEP], dp[MAX_TIME_STEP];
/*** time in real domain ***/
double t[MAX_TIME_STEP];
/*** minimum and maximum time to be calculated ***/
double min_t, max_t;
/*** string length ***/
char line[MAXLEN];
/*** file pointer ***/
FILE *fp;

/*** read input data ***/
read_data(&test_type, &calc_type, &output_units, &layer_q, &M,
```

```
    &N, kh, kv, phi, r, h, &q, &bo, mu, ct, &pin, &cs, skin,
    outer_bc, top_bc, bottom_bc, part_comp, &tp, &min_t,
    &max_t, &im, &jm);

/*** specify output file ***/
printf("output file name = ");
scanf("%s", line);
if((fp = fopen(line, "w")) == 0)
    ERROR_EXIT("Output file open error!");

/*** print input data ***/
print_data(test_type, calc_type, output_units, layer_q, M, N,
    kh, kv, phi, r, h, q, bo, mu, ct, pin, cs, skin, outer_bc,
    top_bc, bottom_bc, part_comp, tp, min_t, max_t, im, jm);

/*** unit conversion, Field -> Darcy ***/
field_to_darcy(M, N, kh, kv, r, h, &q, ct, &pin, &cs, &tp,
    &min_t, &max_t);

/*** calculate dimensionless variables ***/
dimless_var(M, N, kh, kv, phi, r, h, mu, ct, cs, kappa, omega,
    lambda, &cd, rd, K, &khmu, &phich, top_bc, bottom_bc,
    &tp, &min_t, &max_t);

/*** set up time steps in real domain ***/
tstep_real(test_type, &real_count, t, min_t, max_t);

/*** set up time steps in Laplace domain ***/
tstep_laplace(test_type, real_count, &lap_count, lap_t, t, tp);

printf("calculating...\n");

for (i = 0; i < lap_count; i++) {

/*** calculate eigenvalues ***/
eigenvalues(M, N, kappa, omega, lambda, lap_t[i], sigma);

/*** calculate coefficient E ***/
calc_E(M, N, kappa, lambda, omega, lap_t[i], sigma, E);

/*** calculate coefficient A and B ***/
coef_ab(M, N, kappa, sigma, E, lap_t[i], rd, skin,
    K, A, B, outer_bc, part_comp);

if (calc_type == 0) {

    /*** calcurate solution in Laplacian domain ***/
    lap_sol[i] = lap_sol_p(N, A, B, E, sigma,
        rd, skin, part_comp);

    /*** solution with wellbore storage ***/
    lap_sol[i] = 1.0 / (cd * lap_t[i] * lap_t[i]
        + 1.0 / lap_sol[i]);

} else if (calc_type == 1) {

        /*** calcurate solution in Laplacian domain ***/
        lap_sol[i] = lap_sol_q(N, A, B, E, sigma, rd,
            layer_q);
    }
}

if (calc_type == 0) {

    /*** pressure ***/
    calc_pressure(test_type, lap_count, real_count, lap_sol,
        lap_t, sol, t, tp);

    /*** pressure derivative ***/
```

```
calc_pressure_derivative(test_type, real_count, sol, dp, t, tp);

/*** output in field units ***/
if (output_units == 1)
    output_in_field_units(real_count, q, r, pin, khmu,
        phich, t, sol, dp, bo);

/*** output in dimensionless form based on matrix properties ***/
else if (output_units == 2)
    output_in_mat_units(N, real_count, q, r, pin, khmu,
        phich, t, sol, dp, bo, kh, h, mu, ct, phi, im, jm);

/*** output results ***/
    for (i = 1; i < real_count - 1; i++)
        fprintf(fp, "%e %e %e\n", t[i], sol[i], dp[i]);

} else if (calc_type == 1) {

    /*** flow rate ***/
    calc_rate(test_type, layer_q, lap_count, real_count, lap_sol,
        lap_t, sol, t, min_t, max_t, rd, khmu, kh, h, mu);

    /*** output in field units ***/
    if (output_units == 1)
        for (i = 0; i < real_count; i++)
            t[i] *= (phich / khmu * r[1] * r[1] / 3600.0);

    /*** output in dimensionless form based on matrix properties ***/
    else if (output_units == 2)
        for (i = 0; i < real_count; i++)
            t[i] *= (phich / khmu * r[1] * r[1] * kh[jm][im]
            / phi[jm][im] / mu[jm][im] / ct[jm][im] / r[0] / r[0]);

    /*** output results ***/
        for (i = 1; i < real_count - 1;    i++)
            fprintf(fp, "%e %e\n", t[i], sol[i]);
    }

    /*** output file close ***/
    printf("Output was written in the file \"%s\".\n", line);
    fclose(fp);
}

/*
set up time steps in Laplace domain
*/
void tstep_laplace(int test_type, int real_count, int *lap_count,
                   double lap_t[], double t[], double tp)
{
    int i, j;
    double nsteh;

    /*** constant for Stehfest algorithm ***/
    if (test_type == 0)
        nsteh = NSTEH_DD;
else if (test_type == 1)
    nsteh = NSTEH_BU;

*lap_count = 0;
for (i = 0; i < real_count; i++)
    for (j = 0; j < nsteh; j++) {
        lap_t[*lap_count] = ((double) j + 1.0) * log(2.0) / t[i];
        (*lap_count)++;

        if (*lap_count > MAX_TIME_STEP)
            ERROR_EXIT("increase MAX_TIME_STEP!");
    }

/*** for buildup ***/
```

```
    if (test_type == 1) {

        for (i = 0; i < real_count; i++)
            for (j = 0; j < nsteh; j++) {
                lap_t[*lap_count] = ((double) j + 1.0) * log(2.0)
                    / (tp + t[i]);
                (*lap_count)++;

                if (*lap_count > MAX_TIME_STEP)
                    ERROR_EXIT("increase MAX_TIME_STEP!");
            }

        /*** Laplace time for tp ***/
            for (j = 0; j < nsteh; j++) {
                lap_t[*lap_count] = ((double) j + 1.0) * log(2.0) / tp;
                (*lap_count)++;

                if (*lap_count > MAX_TIME_STEP)
                    ERROR_EXIT("increase MAX_TIME_STEP!");
            }

        }
    }

    /*
    set up time steps in real domain
    */
    void tstep_real(int test_type, int *real_count, double t[],
                    double min_t, double max_t)
    {
        t[0] = min_t / R_GEOFAC;
        *real_count = 1;
        do {
            t[*real_count] = t[*real_count-1] * R_GEOFAC;
            (*real_count)++;
        } while (t[*real_count-1] < max_t * R_GEOFAC * R_GEOFAC);
    }

    /*
    output in dimensionless units based on the matrix properties
    */
    void output_in_mat_units(int N, int real_count, double q, double r[],
                             double pin, double khmu, double phich,
                             double t[], double sol[], double dp[],
                             double bo, double kh[][MAXJ], double h[],
                             double mu[][MAXJ], double ct[][MAXJ],
                             double phi[][MAXJ], int im, int jm)
    {
        int i;
        double dummy;

        dummy = 0.0;
        for (i = 0; i < N; i++)
            dummy += h[i];
        dummy *= (1.0 / khmu * kh[im][jm] / mu[im][jm]);
        for (i = 0; i < real_count; i++) {
            t[i] *= (phich / khmu * r[1] * r[1] * kh[im][jm]
                / phi[im][jm] / mu[im][jm] / ct[im][jm]
                / r[0] / r[0]);
            sol[i] *= dummy;
            dp[i] *= dummy;
        }
    }

    /*
    unit conversion from Darcy units to field units
    */
    void output_in_field_units(int real_count, double q, double r[],
                               double pin, double khmu, double phich,
```

```
                    double t[], double sol[], double dp[],
                    double bo)
{
    int i;
    double dummy;

    for (i = 0; i < real_count; i++) {
        t[i] *= (phich / khmu * r[1] * r[1] / 3600.0);
        dummy = q * bo / khmu / 2.0 / PI * 14.696;
        sol[i] *= dummy;
        dp[i] *= dummy;
    }
}

/*
calculate pressure derivatives
*/
void calc_pressure_derivative(int test_type, int real_count, double sol[],
                              double dp[], double t[], double tp)
{
    int i;

    /*** for drawdown testing ***/
    if (test_type == 0) {
        for (i = 1; i < real_count - 1;    i++)
            dp[i] = diff_press(i, real_count, sol, t);
    } else

    /*** for buildup testing ***/
    if (test_type == 1)
        for (i = 1; i < real_count - 1; i++)
            dp[i] = (tp + t[i]) / tp
                * diff_press(i, real_count, sol, t);
}

/*
differencial pressure calculation
*/
double diff_press(int i, int real_count, double sol[], double t[])
{
    double p1, p2, error;

    if (i == 1) {
        ratint(&t[0], &sol[0], 5, t[i] * (1.0 - DT), &p1, &error);
        ratint(&t[0], &sol[0], 5, t[i] * (1.0 + DT), &p2, &error);
    } else if (i == real_count - 2) {
        ratint(&t[real_count-5], &sol[real_count-5], 5,
            t[i] * (1.0 - DT), &p1, &error);
        ratint(&t[real_count-5], &sol[real_count-5], 5,
            t[i] * (1.0 + DT), &p2, &error);
    } else {
        ratint(&t[i-2], &sol[i-2], 5, t[i] * (1.0 - DT), &p1,
            &error);
        ratint(&t[i-2], &sol[i-2], 5, t[i] * (1.0 + DT), &p2,
            &error);
    }

    return (p2 - p1) / 2.0 / DT;
}

/*
rational function interpolation
*/
void ratint(double xa[], double ya[], int r, double x, double *y,
            double *dy)
{
    int p, i, ns = 0;
    double w, t, hh, h, dd;
    double c[5], d[5];
```

```
    hh = fabs(x - xa[0]);

    for (i = 0; i < r; i++) {
        h = fabs(x - xa[i]);
        if (h == 0.0) {
            *y = ya[i];
            *dy = 0.0;
            return;
        } else if (h < hh) {
            ns = i;
            hh = h;
        }
        c[i] = ya[i];
        d[i] = ya[i] + TINY;
    }
    *y = ya[ns--];
    for (p = 0; p < r - 1; p++) {
        for (i = 0; i < r - p - 1; i++) {
            w = c[i+1] - d[i];
            h = xa[i+p+1] - x;
            t = (xa[i] - x) * d[i] / h;
            dd = t - c[i+1];
            if (dd == 0.0)
                ERROR_EXIT("Error in routine ratint.");
            dd = w / dd;
            d[i] = c[i+1] * dd;
            c[i] = t * dd;
        }
        *y += (*dy = (2 * ns < (r - p - 1) ? c[ns+1] : d[ns--]));
    }
}

/*
calculate pressure
*/
void calc_pressure(int test_type, int lap_count, int real_count,
                   double lap_sol[], double lap_t[], double sol[],
                   double t[], double tp)
{
        int i, j, k, nsteh;
        double v[NSTEH_DD], dummy, sum, ptpdt[MAX_TIME_STEP];
        double s, ptp;

        /*** constant for Stehfest algorithm ***/
        if (test_type == 0)
            nsteh = NSTEH_DD;
        else if (test_type == 1)
            nsteh = NSTEH_BU;

        /*** calculate weight V ***/
        for (i = 1; i ≤ nsteh; i++) {
                if (i > nsteh / 2)
                        j = nsteh / 2;
                else
                        j = i;
        sum = 0.0;
        for (k = (i + 1) / 2; k ≤ j; k++) {
                dummy = pow((double) k,
                        (double) (nsteh / 2))
                        * factrl(2 * k)
                        / factrl(nsteh / 2 - k)
                        / factrl(k) / factrl(k - 1)
                        / factrl(i - k)
                        / factrl(2 * k - i);
                sum += dummy;
        }
        if (((nsteh / 2 + i) % 2) == 1)
                v[i - 1] = - sum;
```

```
        else
                v[i - 1] = sum;
}

/*** time loop ***/
for (j = 0; j < real_count; j++) {

    sum = 0.0;

    for (i = 1; i ≤ nsteh; i++) {

        /*** Laplace time ***/
        s = (double) i * log(2.0) / t[j];

        /*** pick up the correct index ***/
        if ((k = lap_table(lap_count, lap_t, s)) < 0)
            ERROR_EXIT("Error in lap_table!");

        sum += v[i-1] * lap_sol[k];
    }

    /*** values in real domain ***/
    sol[j] = log(2.0) / t[j] * sum;
}

/*** for buildup test ***/
if (test_type == 1) {

    sum = 0.0;

    for (i = 1; i ≤ nsteh; i++) {

        s = (double) i * log(2.0) / tp;

    /*** pick up the correct index ***/
    if ((k = lap_table(lap_count, lap_t, s)) < 0)
        ERROR_EXIT("Error in lap_table!");

    sum += v[i-1] * lap_sol[k];
}

ptp = log(2.0) / tp * sum;

/*** time loop ***/
for (j = 0; j < real_count; j++) {

    sum = 0.0;

    for (i = 1; i ≤ nsteh; i++) {

        /*** Laplace time ***/
        s = (double) i * log(2.0) / (t[j] + tp);

        /*** pick up the correct index ***/
        if ((k = lap_table(lap_count, lap_t, s)) < 0)
            ERROR_EXIT("Error in lap_table!");

            sum += v[i-1] * lap_sol[k];
        }

        /*** values in real domain ***/
        ptpdt[j] = log(2.0) / (t[j] + tp) * sum;
    }

    /*** superposition in time ***/
    for (i = 0; i < real_count; i++)
        sol[i] += (ptp - ptpdt[i]);
    }
}
```

```
/*
pick up a value from the Laplace domain table
*/
int lap_table(int lap_count, double lap_t[], double s)
{
    int i, index;

    for (i = 0; i < lap_count; i++)
        if (fabs(lap_t[i] - s) / s < D_EPS) {
            index = i;
            break;
        } else
            index = -1;

    return index;
}

/*
calculate flow rate
*/
void calc_rate(int test_type, int layer_q, int lap_count,
               int real_count, double lap_sol[], double lap_t[],
               double sol[], double t[], double min_t, double max_t,
               double rd[], double khmu, double kh[][MAXJ],
               double h[], double mu[][MAXJ])
{
        int i, j, k, nsteh;
        double v[NSTEH_DD], dummy, sum, s;

        /*** constant for Stehfest algorithm ***/
        if (test_type == 0)
            nsteh = NSTEH_DD;

        /*** calculate weight V ***/
        for (i = 1; i <= nsteh; i++) {
                if (i > nsteh / 2)
                        j = nsteh / 2;
                else
                        j = i;
                sum = 0.0;
                for (k = (i + 1) / 2; k <= j; k++) {
                        dummy = pow((double) k,
                                (double) (nsteh / 2))
                                * factrl(2 * k)
                                / factrl(nsteh / 2 - k)
                                / factrl(k) / factrl(k - 1)
                                / factrl(i - k)
                                / factrl(2 * k - i);
                        sum += dummy;
                }
                if (((nsteh / 2 + i) % 2) == 1)
                        v[i - 1] = - sum;
                else
                        v[i - 1] = sum;
        }

        /*** time loop ***/
        for (j = 0; j < real_count; j++) {

            sum = 0.0;

            for (i = 1; i <= nsteh; i++) {

                /*** Laplace time ***/
                s = (double) i * log(2.0) / t[j];

                /*** pick up the correct index ***/
                if ((k = lap_table(lap_count, lap_t, s)) < 0)
```

```
                ERROR_EXIT("Error in lap_table!");

            sum += v[i-1] * lap_sol[k];
        }

        /*** values in real domain ***/
        sol[j] = log(2.0) / t[j] * sum;

        /*** dimension ***/
        sol[j] *= (rd[0] * kh[0][layer_q] * h[layer_q]
            / mu[0][layer_q] / khmu);
    }
}

/*
unit conversion from field units to Darcy units
*/
void field_to_darcy(int M, int N, double kh[][MAXJ],
                    double kv[][MAXJ], double r[],
                    double h[], double *q, double ct[][MAXJ],
                    double *pin, double *cs, double *tp,
                    double *min_t, double *max_t)
{
    int i, j;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++) {
            kh[i][j] *= 0.001;
            kv[i][j] *= 0.001;
            ct[i][j] *= 14.696;
        }

    for (i = 0; i < M + 1; i++)
        r[i] *= 30.48;

    for (j = 0; j < N; j++)
        h[j] *= 30.48;

    *q *= 1.840131;
    *pin *= 0.068046;
    *cs *= (14.696 * 1.589873e5);
    *tp *= 3600.0;
    *min_t *= 3600.0;
    *max_t *= 3600.0;
}

/*
calculate factorials
*/
double factrl(int n)
{
    double x = n;

    if (n < 0)
        ERROR_EXIT("n < 0 in factrl!");

    if (n == 0)
        return 1.0;
    else {
        n--;
        while (n > 0) {
            x *= (double) n;
            n--;
        }

        return (double) x;
    }
}
```

```
/*
pressure solution in the Laplace domain
*/
double lap_sol_p(int N, double A[][MAXJ], double B[][MAXJ],
                 double E[][MAXJ][MAXJ], double sigma[][MAXJ],
                 double rd[], double skin[], int part_comp[])
{
    int j, jj, k;
    double dummy;

    /*** find open layer ***/
    for (j = 0; j < N; j++)
      if (part_comp[j] == 1) {
        jj = j;
        break;
      }

    dummy = 0.0;
    for (k = 0; k < N; k++)
        dummy += (E[0][jj][k] * ((bessk0(sigma[0][k] * rd[0])
        + skin[jj] * rd[0] * sigma[0][k] * bessk1(sigma[0][k] * rd[0]))
        * A[0][k] + (bessi0(sigma[0][k] * rd[0]) - skin[jj] * rd[0]
        * sigma[0][k] * bessi1(sigma[0][k] * rd[0])) * B[0][k]));

    return dummy;
}

/*
flow rate solution in the Laplace domain
*/
double lap_sol_q(int N, double A[][MAXJ], double B[][MAXJ],
                 double E[][MAXJ][MAXJ], double sigma[][MAXJ],
                 double rd[], int layer_q)
{
    int k;
    double dummy;

    dummy = 0.0;

    for (k = 0; k < N; k++)
        dummy += (sigma[0][k] * E[0][layer_q][k] *
            (A[0][k] * bessk1(sigma[0][k] * rd[0])
            - B[0][k] * bessi1(sigma[0][k] * rd[0])));

    return dummy;
}

/*
determine coefficients A and B
*/
void coef_ab(int M, int N, double kappa[][MAXJ], double sigma[][MAXJ],
             double E[][MAXJ][MAXJ], double lap_t, double rd[],
             double skin[], double K[][MAXJ], double A[][MAXJ],
             double B[][MAXJ], int outer_bc[], int part_comp[])
{
int i, j, k, jj;
double dummy;
double a[2*MAXI*MAXJ][2*MAXI*MAXJ], b[2*MAXI*MAXJ];

/*** clear matrix ***/
for (i = 0; i < 2*M*N; i++) {
    for (j = 0; j < 2*M*N; j++)
        a[i][j] = 0.0;
    b[i] = 0.0;
}

/*** constant rate inner boundary condition ***/
for (k = 0; k < N; k++) {
    dummy = 0.0;
```

```
    for (j = 0; j < N; j++)
        if (part_comp[j] == 1)
            dummy += kappa[0][j] * E[0][j][k];

    a[0][k] = dummy * sigma[0][k] * rd[0]
        * bessk1(sigma[0][k] * rd[0]);
    a[0][k+N] = - dummy * sigma[0][k] * rd[0]
        * bessi1(sigma[0][k] * rd[0]);
}
b[0] = 1.0 / lap_t;

/*** constant pressure inner boundary condition ***/
jj = -1;
for (j = 0; j < N; j++)
    if ((part_comp[j] == 0) && (jj < 0))
        for (k = 0; k < N; k++) {
            a[j+1][k] = - E[0][j][k] * sigma[0][k]
                  * bessk1(sigma[0][k] * rd[0]);
            a[j+1][k+N] = E[0][j][k] * sigma[0][k]
                  * bessi1(sigma[0][k] * rd[0]);
        } else if ((part_comp[j] == 0) && (jj ≥ 0))
            for (k = 0; k < N; k++) {
                a[j][k] = - E[0][j][k] * sigma[0][k]
                    * bessk1(sigma[0][k] * rd[0]);
                a[j][k+N] = E[0][j][k] * sigma[0][k]
                    * bessi1(sigma[0][k] * rd[0]);
            } else if ((jj ≥ 0) && (part_comp[j] == 1))
                for (k = 0; k < N; k++) {
                    a[j][k] = bessk0(sigma[0][k] * rd[0])
                        * (E[0][jj][k] - E[0][j][k])
                        + sigma[0][k] * bessk1(sigma[0][k] * rd[0])
                        * rd[0] * (skin[jj] * E[0][jj][k]
                        - skin[j] * E[0][j][k]);
                    a[j][k+N] = bessi0(sigma[0][k] * rd[0])
                        * (E[0][jj][k] - E[0][j][k])
                        - sigma[0][k] * bessi1(sigma[0][k] * rd[0])
                        * rd[0] * (skin[jj] * E[0][jj][k]
                        - skin[j] * E[0][j][k]);
                } else if ((jj < 0) && (part_comp[j] == 1))
                    jj = j;

/*** pressure and flow rate continuity ***/
for (i = 0; i < M - 1; i++)
    for (j = 0; j < N; j++)
        for (k = 0; k < N; k++) {

            /*** pressure continuity ***/
            a[j+N+i*N][k+i*2*N] = E[i][j][k]
                * bessk0(sigma[i][k] * rd[i+1]);
            a[j+N+i*N][k+N+i*2*N] = E[i][j][k]
                * bessi0(sigma[i][k] * rd[i+1]);
            a[j+N+i*N][k+i*2*N+2*N] = - E[i+1][j][k]
                * bessk0(sigma[i+1][k] * rd[i+1]);
            a[j+N+i*N][k+3*N+i*2*N] = - E[i+1][j][k]
                * bessi0(sigma[i+1][k] * rd[i+1]);

            /*** flow rate continuity ***/
            a[j+M*N+i*N][k+i*2*N] = sigma[i][k] * E[i][j][k]
                * bessk1(sigma[i][k] * rd[i+1]);
            a[j+M*N+i*N][k+N+i*2*N] = - sigma[i][k] * E[i][j][k]
                * bessi1(sigma[i][k] * rd[i+1]);
            a[j+M*N+i*N][k+i*2*N+2*N] = - K[i][j] * sigma[i+1][k]
                * E[i+1][j][k] * bessk1(sigma[i+1][k] * rd[i+1]);
            a[j+M*N+i*N][k+3*N+i*2*N] = K[i][j] * sigma[i+1][k]
                * E[i+1][j][k] * bessi1(sigma[i+1][k] * rd[i+1]);
        }

/*** outer boundary condition ***/
for (j = 0; j < N; j++)
```

```
switch (outer_bc[j]) {

    /*** infinite ***/
    case 0:
        for (k = 0; k < N; k++)
            a[j+N+2*N*(M-1)][k+2*N*(M-1)] = 0.0;
        break;

    /*** no flow ***/
    case 1:
        for (k = 0; k < N; k++) {
            a[j+N+2*N*(M-1)][k+2*N*(M-1)]
                = sigma[M-1][k] * E[M-1][j][k]
                * besski(sigma[M-1][k] * rd[M]);
            a[j+N+2*N*(M-1)][k+N+2*N*(M-1)]
                = - sigma[M-1][k] * E[M-1][j][k]
                * bessi1(sigma[M-1][k] * rd[M]);
        }
        break;

    /*** constant p ***/
    case 2:
        for (k = 0; k < N; k++) {
            a[j+N+2*N*(M-1)][k+2*N*(M-1)]
                = E[M-1][j][k] * bessk0(sigma[M-1][k] * rd[M]);
            a[j+N+2*N*(M-1)][k+N+2*N*(M-1)]
                = E[M-1][j][k] * bessi0(sigma[M-1][k] * rd[M]);
        }
        break;
    }

/*** solve for A, B ***/
if (outer_bc[0] == 0)
    gaussj(2*M*N-N, a, b);
else
    gaussj(2*M*N, a, b);

/*** arrange solution vector, A and B ***/
for (i = 0; i < M; i++)
    for (k = 0; k < N; k++) {
        A[i][k] = b[k+2*i*N];
        B[i][k] = b[k+2*i*N+N];
        }
}

/*
Gauss-Jordan elimination with full pivoting
*/
void gaussj(int n, double a[][2*MAXI*MAXJ], double b[])
{
    int i, icol, irow, j, k, l, ll;
    int ipiv[2*MAXI*MAXJ];
    double big, dum, pivinv, temp;

for (j = 0; j < n; j++)
    ipiv[j] = 0;

for (i = 0; i < n; i++) {
    big = 0.0;
    for (j = 0; j < n; j++)
        if (ipiv[j] ≠ 1)
            for (k = 0; k < n; k++) {

                if (ipiv[k] == 0) {

                    if (fabs(a[j][k]) ≥ big) {
                        big = fabs(a[j][k]);
                        irow = j;
```

```
            icol = k;
        }

    } else
        if (ipiv[k] > 1)
            ERROR_EXIT("gaussj:singular matrix-1");
}
++(ipiv[icol]);

if (irow ≠ icol) {
    for (l = 0; l < n; l++)
        SWAP(a[irow][l], a[icol][l]);
    SWAP(b[irow], b[icol]);
}

if (a[icol][icol] == 0.0)
    ERROR_EXIT("gaussj:singular matrix-2");

pivinv = 1.0 / a[icol][icol];
a[icol][icol] = 1.0;
for (l = 0; l < n; l++)
    a[icol][l] *= pivinv;
b[icol] *= pivinv;

for (ll = 0; ll < n; ll++)
    if (ll ≠ icol) {
        dum = a[ll][icol];
        a[ll][icol] = 0.0;
        for (l = 0; l < n; l++)
            a[ll][l] -= (a[icol][l] * dum);
        b[ll] -= (b[icol] * dum);
    }
}

/*
calculate constants E
*/
void calc_E(int M, int N, double kappa[][MAXJ],
            double lambda[][MAXJ+1], double omega[][MAXJ],
            double lap_t, double sigma[][MAXJ],
            double E[][MAXJ][MAXJ])
{
    int i, j, k;
    double a[MAXJ], b[MAXJ], c[MAXJ];

    /*** loop for each block ***/
    for (i = 0; i < M; i++)

        /*** loop for each eigenvalue ***/
        for (k = 0; k < N; k++) {
            for (j = 0; j < N - 1; j++) {
                a[j] = 1.0 / kappa[i][j] * (omega[i][j] * lap_t
                + lambda[i][j] + lambda[i][j+1]) - sigma[i][k] * sigma[i][k];
                b[j] = - lambda[i][j+1] / kappa[i][j];
                c[j+1] = - lambda[i][j+1] / kappa[i][j+1];
            }

            a[N-1] = 1.0 / kappa[i][N-1] * (omega[i][N-1] * lap_t
                + lambda[i][N-1] + lambda[i][N]) - sigma[i][k] * sigma[i][k];

            E[i][0][k] = 1.0;
            E[i][1][k] = - a[0] / b[0];
            for (j = 2; j < N; j++)
                E[i][j][k] = - 1.0 / b[j-1] * (c[j-1] * E[i][j-2][k]
                    + a[j-1] * E[i][j-1][k]);
        }
}
```

```
/*
modified Bessel functions K1(x)
*/
double besskl(double x)
{
    double ans, y;
    double bessi1(double x);

    if (x ≤ 2.0) {
        y = x * x / 4.0;
        ans = (log(x / 2.0) * bessi1(x)) + (1.0 / x)
            * (1.0 + y * (0.15443144 + y * (−0.67278579
            + y * (−0.18156897 + y * (−0.1919402e−1
            + y * (−0.110404e−2 + y * (−0.4686e−4)))))));
        } else {
            y = 2.0 / x;
            ans = (exp(−x) / sqrt(x)) * (1.25331414
                + y * (0.23498619 + y * (−0.3655620e−1
                + y * (0.1504268e−1 + y * (−0.780353e−2
                + y * (0.325614e−2 + y * (−0.68245e−3)))))));
        }

    return ans;
}

/*
modified Besssel functions I1(x)
*/
double bessi1(double x)
{
    double ax, ans, y;

    if ((ax = fabs(x)) < 3.75) {
        y = x / 3.75;
        y *= y;
        ans = ax * (0.5 + y * (0.87890594 + y * (0.51498869
            + y * (0.15084934 + y * (0.2658733e−1
            + y * (0.301532e−2 + y * 0.32411e−3))))));
        } else {
            y = 3.75 / ax;
            ans = 0.2282967e−1 + y * (−0.2895312e−1
                + y * (0.1787654e−1 − y * 0.420059e−2));
            ans = 0.39894228 + y * (−0.3988024e−1
                + y * (−0.362018e−2 + y * (0.163801e−2
                + y * (−0.1031555e−1 + y * ans))));
            ans *= (exp(ax) / sqrt(ax));
        }

    return x < 0.0 ? −ans : ans;
}

/*
modified Bessel functions K0(x)
*/
double bessk0(double x)
{
    double ans, y;
    double bessi0(double x);

    if (x ≤ 2.0) {
        y = x * x / 4.0;
        ans = (−log(x/2.0) * bessi0(x)) + (−0.57721566
            + y * (0.42278420 + y * (0.23069756
            + y * (0.3488590e−1 + y * (0.262698e−2
            + y * (0.10750e−3 + y * 0.74e−5))))));
        } else {
            y = 2.0 / x;
            ans = (exp(−x) / sqrt(x)) * (1.25331414 + y
```

```
                * (-0.7832358e-1 + y * (0.2189568e-1
                + y * (-0.1062446e-1 + y * (0.587872e-2
                + y * (-0.251540e-2 + y * 0.53208e-3))))));
        }

    return ans;
}

/*
modified Bessel functions I0(x)
*/
double bessi0(double x)
{
    double ax, ans, y;

    if ((ax = fabs(x)) < 3.75) {
        y = x / 3.75;
        y *= y;
        ans = 1.0 + y * (3.5156229 + y * (3.0899424
            + y * (1.2067492 + y * (0.2659732
            + y * (0.360768e-1 + y * 0.45813e-2)))));
    } else {
        y = 3.75 / ax;
        ans = (exp(ax) / sqrt(ax)) * (0.39894228
            + y * (0.1328592e-1 + y * (0.225319e-2
            + y * (-0.157565e-2 + y * (0.916281e-2
            + y * (-0.2057706e-1 + y * (0.2635537e-1
            + y * (-0.1647633e-1 + y * 0.392377e-2)))))));
    }

    return ans;
}

/*
calculate eigenvalues
*/
void eigenvalues(int M, int N, double kappa[][MAXJ],
                 double omega[][MAXJ], double lambda[][MAXJ+1],
                 double lap_t, double sigma[][MAXJ])
{
    int i, j;
    double alpha[MAXJ], beta[MAXJ];
    void tqli(double d[], double e[], int n);

    for (i = 0; i < M; i++) {

        for (j = 0; j < N; j++)
            alpha[j] = 1.0 / kappa[i][j] * (omega[i][j] * lap_t
                + lambda[i][j+1] + lambda[i][j]);

        for (j = 1; j < N; j++)
            beta[j] = - lambda[i][j] / sqrt(kappa[i][j-1] * kappa[i][j]);

        /*** QL algorithm ***/
        tqli(alpha, beta, N);

        /*** set eigenvalues ***/
        for (j = 0; j < N; j++)
            sigma[i][j] = sqrt(alpha[j]);
    }
}

/*
return square root of a^2+b^2
*/
double pythag(double a, double b)
{
    double absa, absb;
```

```
    absa = fabs(a);
    absb = fabs(b);
    if (absa > absb)
        return absa * sqrt(1.0 + SQR(absb/absa));
    else
        return (absb == 0.0 ? 0.0 :
            absb * sqrt(1.0 + SQR(absa/absb)));
}

/*
QL algorithm to determine eigenvalues
*/
void tqli(double d[], double e[], int n)
{
    double pythag(double a, double b);
    int m, l, iter, i;
    double s, r, p, g, f, dd, c, b;

    for (i = 1; i < n; i++)
        e[i - 1] = e[i];
    e[n-1] = 0.0;

    for (l = 0; l < n; l++) {
        iter = 0;
        do {
            for (m = l; m < n - 1; m++) {
                dd = fabs(d[m]) + fabs(d[m+1]);
                if ((fabs(e[m]) + dd) == dd)
                    break;
            }

            if (m ≠ l) {
                if (iter++ == 30)
                    ERROR_EXIT("Too many iterations in tqli");
                g = (d[l+1] - d[l]) / (2.0 * e[l]);
                r = pythag(g, 1.0);
                g = d[m] - d[l] + e[l] / (g + SIGN(r, g));
                s = c = 1.0;
                p = 0.0;

                for (i = m - 1; i ≥ l; i--) {
                    f = s * e[i];
                    b = c * e[i];
                    e[i+1] = (r = pythag(f, g));
                    if (r == 0.0) {
                        d[i+1] -= p;
                        e[m] = 0.0;
                        break;
                    }

                    s = f / r;
                    c = g / r;
                    g = d[i+1] - p;
                    r = (d[i] - g) * s + 2.0 * c * b;
                    d[i+1] = g + (p = s * r);
                    g = c * r - b;

                }

                if (r == 0.0 && i ≥ l)
                    continue;
                d[l] -= p;
                e[l] = g;
                e[m] = 0.0;
            }
        } while (m ≠ l);
    }
}
```

```
/*
define dimensionless variables
*/
void dimless_var(int M, int N, double kh[][MAXJ],
                 double kv[][MAXJ], double phi[][MAXJ],
                 double r[], double h[], double mu[][MAXJ],
                 double ct[][MAXJ], double cs,
                 double kappa[][MAXJ], double omega[][MAXJ],
                 double lambda[][MAXJ+1], double *cd,
                 double rd[], double K[][MAXJ],
                 double *khmu, double *phich, int top_bc[],
                 int bottom_bc[], double *tp, double *min_t,
                 double *max_t)
{
    int i, j;
    double X;

    /*** average permeability and porosity ***/
    *khmu = *phich = 0.0;
    for (j = 0; j < N; j++) {
        *khmu += (kh[0][j] / mu[0][j] * h[j]);
        *phich += (phi[0][j] * ct[0][j] * h[j]);
    }

    /*** kappa and omega ***/
    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++) {
            kappa[i][j] = kh[i][j] / mu[i][j]
            / (*khmu) * h[j];
            omega[i][j] = phi[i][j] / (*phich)
            * ct[i][j] * h[j];
        }

    /*** upper and lower boundary condition ***/
    for (i = 0; i < M; i++) {
        if (bottom_bc[i] == 0)
            lambda[i][0] = 0.0;
        else
            lambda[i][0] = 2.0 / h[0]
                / mu[i][0] * kv[i][0];
        if (top_bc[i] == 0)
            lambda[i][N] = 0.0;
        else
            lambda[i][N] = 2.0 / h[N-1]
                / mu[i][N-1] * kv[i][N-1];
    }

    for (j = 1; j < N; j++)
        for (i = 0; i < M; i++) {
            X = 2.0 / (h[j-1] * mu[i][j-1]
            / kv[i][j-1] + h[j] * mu[i][j]
            / kv[i][j]);
            lambda[i][j] = r[1] * r[1] / (*khmu)
            * X;
        }

    /*** based on r[1] ***/
    for (i = 0; i < M + 1; i++)
        rd[i] = r[i] / r[1];

    for (i = 0; i < M - 1; i++)
        for (j = 0; j < N; j++)
            K[i][j] = kh[i+1][j] / kh[i][j]
            * mu[i][j] / mu[i+1][j];

    /*** wellbore storage ***/
    *cd = cs / 2.0 / PI / r[1] / r[1] / (*phich);
```

```
    /*** productiopn time before shutin ***/
    *tp *= (*khmu) / (*phich) / r[1] / r[1];

    /*** time range in real domain ***/
    *min_t *= (*khmu) / (*phich) / r[1] / r[1];
    *max_t *= (*khmu) / (*phich) / r[1] / r[1];
}

/*
print input data
*/
void print_data(int test_type, int calc_type, int output_units,
                int layer_q, int M, int N, double kh[][MAXJ],
                double kv[][MAXJ], double phi[][MAXJ],
                double r[], double h[], double q,
                double bo, double mu[][MAXJ], double ct[][MAXJ],
                double pin, double cs, double skin[],
                int outer_bc[], int top_bc[], int bottom_bc[],
                int part_comp[], double tp, double min_t,
                double max_t, int im, int jm)
{
    int i, j;

    printf("*****************************************\n");
    printf("                INPUT DATA                \n");
    printf("*****************************************\n");

    /*** test type ***/
    if (test_type == 0)
        printf("Pressure drawdown testing.\n");
    else if (test_type == 1)
        printf("Pressure buildup testing.\n");
    else
        ERROR_EXIT("Enter a correct test type!");

    /*** calculation type ***/
    if (calc_type == 0)
        printf("Wellbore pressure calculation.\n");
    else if (calc_type == 1)
        printf("Flow rate calculation for  the layer %d.\n",
        layer_q + 1);
    else
        ERROR_EXIT("Enter a correct calculation type!");

    /*** reservoir matrix block ***/
    if (output_units == 2) {
        printf("Dimensionless variables are based on the ");
        printf("reservoir matrix properties ");
        printf("of the block (%d,%d).\n", im + 1, jm + 1);
    }

    /*** output units ***/
    if (output_units == 0)
        printf("Output in Gemes et al dimensionless form.\n");
    else if (output_units == 1)
        printf("Output in field units.\n");
    else if (output_units == 2) {
        printf("Output in dimensionless form based ");
        printf("on matrix properties.\n");
    } else
        ERROR_EXIT("Enter a correct output units!");

    /*** time to be calculated ***/
    printf("Time interval to be calcurated = %e - %e hrs\n",
        min_t, max_t);

    /*** number of grid blocks ***/
    printf("Reservoir Dimension (I x J) = %d x %d\n", M, N);
```

```
/*** grid size ***/
printf("Grid size\n");
printf("ri = ");
for (i = 0; i ≤ M; i++)
    printf("%6.2lf ", r[i]);
printf("\n");
printf("hj = ");
for (j = 0; j < N; j++)
    printf("%6.2lf ", h[j]);
printf("\n");

/*** permeability ***/
printf("Permeability, md\n");
printf("kh =\n");
for (j = N − 1; j ≥ 0; j−−) {
    for (i = 0; i < M; i++)
        printf("%6.2lf ", kh[i][j]);
    printf("\n");
}
printf("kv =\n");
for (j = N − 1; j ≥ 0; j−−) {
    for (i = 0; i < M; i++)
        printf("%6.2lf ", kv[i][j]);
    printf("\n");
}

/*** porosity ***/
printf("Porosity\n");
for (j = N − 1; j ≥ 0; j−−) {
    for (i = 0; i < M; i++)
        printf("%6.2lf ", phi[i][j]);
    printf("\n");
}

/*** viscosity ***/
printf("Viscosity, cp\n");
for (j = N − 1; j ≥ 0; j−−) {
    for (i = 0; i < M; i++)
        printf("%6.2lf ", mu[i][j]);
    printf("\n");
}

/*** compressibility ***/
printf("Compressibility, /psi\n");
for (j = N − 1; j ≥ 0; j−−) {
    for (i = 0; i < M; i++)
        printf("%6.2e ", ct[i][j]);
    printf("\n");
}

/*** flow rate ***/
printf("Flow rate, bbl = %6.2lf\n", q);

/*** formation volume factor ***/
printf("Formation vol. factor = %6.2lf\n", bo);

/*** initial pressure and producing time ***/
printf("Initial pressure, psi = %lf\n", pin);
if (test_type == 1)
    printf("Producing time before shutin, hrs = %lf\n", tp);

/*** wellbore storage ***/
printf("Wellbore storage constant = %6.2lf\n", cs);

/*** skin factors ***/
printf("Skin for each layer =\n");
for (j = 0; j < N; j++)
    printf("%6.2lf ", skin[j]);
printf("\n");
```

```
/*** outer boundary conditions ***/
printf("Outer boundary condition.\n");
for (j = 0; j < N; j++) {
    printf("Layer %d = ", j);
    switch (outer_bc[j]) {
        case 0:
        printf("Infinite outer boundary.\n");
        break;
        case 1:
        printf("No flow outer boundary.\n");
        break;
        case 2:
        printf("Constant pressure boundary.\n");
        break;
        default:
        ERROR_EXIT("Enter a correct boundary condition!");
    }
}

/*** upper and lower boundary conditions ***/
printf("Upper boundary.\n");
for (i = 0; i < M; i++) {
    printf("Block %d = ", i);
    if (top_bc[i] == 0)
        printf("Closed.\n");
    else
        printf("Gascap drive.\n");
}
printf("Lower boundary.\n");
for (i = 0; i < M; i++) {
    printf("Block %d = ", i);
    if (bottom_bc[i] == 0)
        printf("Closed.\n");
    else
        printf("Bottom water drive.\n");
    }

    /*** well completion ***/
    printf("Completion.\n");
    for (j = 0; j < N; j++) {
        printf("Layer %d = ", j);
        if (part_comp[j] == 0)
            printf("Not completed.\n");
        else
            printf("Completed.\n");
    }
}

/*
read input data
*/
void read_data(int *test_type, int *calc_type, int *output_units,
               int *layer_q, int *M, int *N, double kh[][MAXJ],
               double kv[][MAXJ], double phi[][MAXJ],
               double r[], double h[], double *q,
               double *bo, double mu[][MAXJ],
               double ct[][MAXJ], double *pin, double *cs,
               double skin[], int outer_bc[], int top_bc[],
               int bottom_bc[], int part_comp[], double *tp,
               double *min_t, double *max_t, int *im, int *jm)
{
    int i, j;
    char line[MAXLEN];
    FILE *fp;

    /*** input file name ***/
    printf("Input file name = ");
    scanf("%s", line);
```

```
if((fp = fopen(line, "r")) == 0)
    ERROR_EXIT("Input file open error!");

/*** notes ***/
fgets(line, MAXLEN, fp);
printf("Reading data file...\n%s\n", line);

/*** drawdown or buildup ***/
fscanf(fp, "%d", test_type);

/*** pressure or flow rate ***/
fscanf(fp, "%d", calc_type);
if ((*test_type == 1) && (*calc_type == 1))
    ERROR_EXIT("Rates cannot be computed in buildup tests!");

/*** output units ***/
fscanf(fp, "%d", output_units);

/*** time interval in real domain ***/
fscanf(fp, "%lf %lf", min_t, max_t);

/*** block size ***/
fscanf(fp, "%d %d", M, N);

/*** a layer where the rate is calculated ***/
fscanf(fp, "%d", layer_q);
(*layer_q)--;
if ((*layer_q < 0) || (*layer_q ≥ *N))
    ERROR_EXIT("Enter a correct layer number!");

/*** reservoir matrix block ***/
fscanf(fp, "%d %d", im, jm);
(*im)--;
(*jm)--;
if ((*im < 0) || (*im ≥ *M) || (*jm < 0) || (*jm ≥ *N))
    ERROR_EXIT("Enter a correct matrix index!");

/*** permeability ***/
for (j = *N - 1; j ≥ 0; j--)
      for (i = 0; i < *M; i++)
          fscanf(fp, "%lf", &(kh[i][j]));

for (j = *N - 1; j ≥ 0; j--)
    for (i = 0; i < *M; i++)
        fscanf(fp, "%lf", &(kv[i][j]));

/*** porosity ***/
for (j = *N - 1; j ≥ 0; j--)
    for (i = 0; i < *M; i++)
        fscanf(fp, "%lf", &(phi[i][j]));

/*** block radius ***/
for (i = 0; i ≤ *M; i++)
    fscanf(fp, "%lf", &(r[i]));

/*** block thickness ***/
for (j = *N - 1; j ≥ 0; j--)
    fscanf(fp, "%lf", &(h[j]));

/*** production rate ***/
fscanf(fp, "%lf", q);

/*** formation vol. factor ***/
fscanf(fp, "%lf", bo);

/*** viscosity ***/
for (j = *N - 1; j ≥ 0; j--)
    for (i = 0; i < *M; i++)
        fscanf(fp, "%lf", &(mu[i][j]));
```

```
/*** compressibility ***/
for (j = *N - 1; j ≥ 0; j--)
    for (i = 0; i < *M; i++)
        fscanf(fp, "%lf", &(ct[i][j]));

/*** initial pressure ***/
fscanf(fp, "%lf", pin);

/*** producing time before shutin ***/
fscanf(fp, "%lf", tp);

/*** wellbore storage ***/
fscanf(fp, "%lf", cs);

/*** skin ***/
for (j = *N - 1; j ≥ 0; j--)
    fscanf(fp, "%lf", &(skin[j]));

/*** outer boundary condition ***/
for (j = *N - 1; j ≥ 0; j--)
    fscanf(fp, "%d", &(outer_bc[j]));

/*** upper boundary condition ***/
for (i = 0; i < *M; i++)
    fscanf(fp, "%d", &(top_bc[i]));

/*** lower boundary condition ***/
for (i = 0; i < *M; i++)
    fscanf(fp, "%d", &(bottom_bc[i]));

/*** partially penetrating well ***/
for (j = *N - 1; j ≥ 0; j--)
    fscanf(fp, "%d", &(part_comp[j]));

    /*** close file ***/
    fclose(fp);
}
```

APPENDIX B

COMPUTER PROGRAM FOR HETEROGENEOUS LINEAR RESERVOIRS

A computer program for pressure analysis of a horizontal well in heterogeneous linear reservoirs is presented in Appendix B. As shown in Chapter 5, this program can be directly used to generate the synthetic pressure and the fractional flow rate responses of a vertical well in heterogeneous linear reservoirs. It can be also used for a fractured well case with small modification. The box-type reservoir case which uses the finite Fourier cosine transform has not been implemented yet. Although the constant pressure top or bottom reservoir boundary case can be easily implemented, such option has not been included in this program. C language was used for this program. However, multiple precision arithmetic program written in C++ language was incorporated in the program.

B.1 Description of the Program

A procedure to calculate the pressure or the fractional flow rate response are provided in Fig. B.1. Part of the Laplace pressure/fractional flow rate calculation is shown separately in Fig. B.2.

Basically, the program consists of the following sequence of execution.

1. Using input data, grid blocks (reservoir and the wellbore blocks) are re-constructed according to the size of the wellbore strip.

2. The eigenvalues $(\sigma_j^k)^2$ and the constants $E_{i,j}^k$ for the layer j are computed from Eq. 4.127 using QL algorithm with implicit shifts [95].

3. The pressure gradient at the wellbore D_i is assumed for each well segment. Then, the particular solution $C_{i,j}$ is computed by solving the symmetric tridiagonal system of equations (Eqs. 4.142 to 4.145).

4. $2mn$ simultaneous equations are set up from the boundary conditions and solved for the constants A_j^k and B_j^k using Gauss-Jordan elimination with partial pivoting.

5. The inverse Fourier transform is numerically performed using the Gauss-Legendre integration [95]. The wellbore pressure for each well segment is computed using the pressure averaging method proposed by Kuchuk et al. [62].

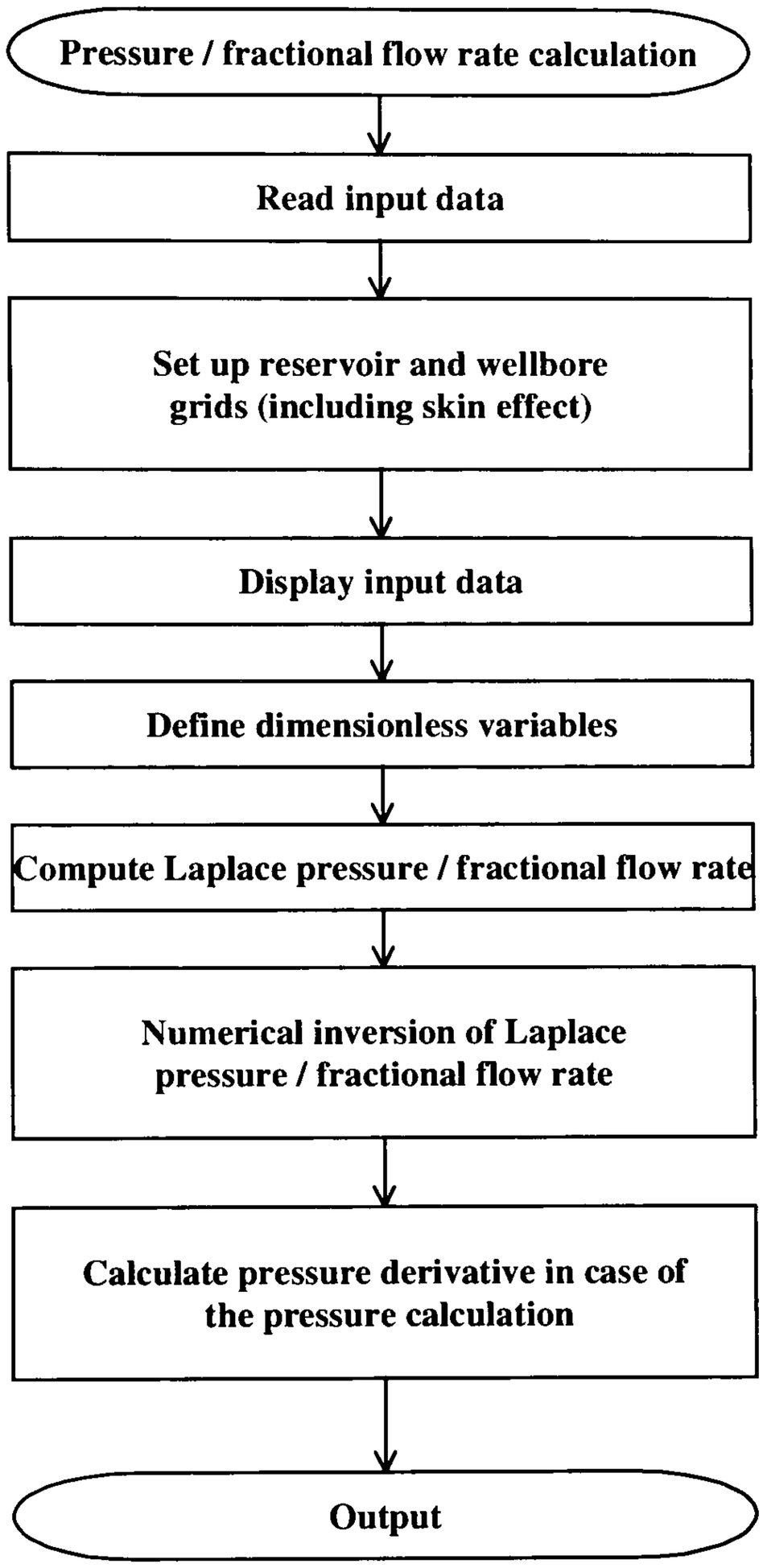


Figure B.1: Flow chart for the wellbore pressure or the fractional flow rate calculation.

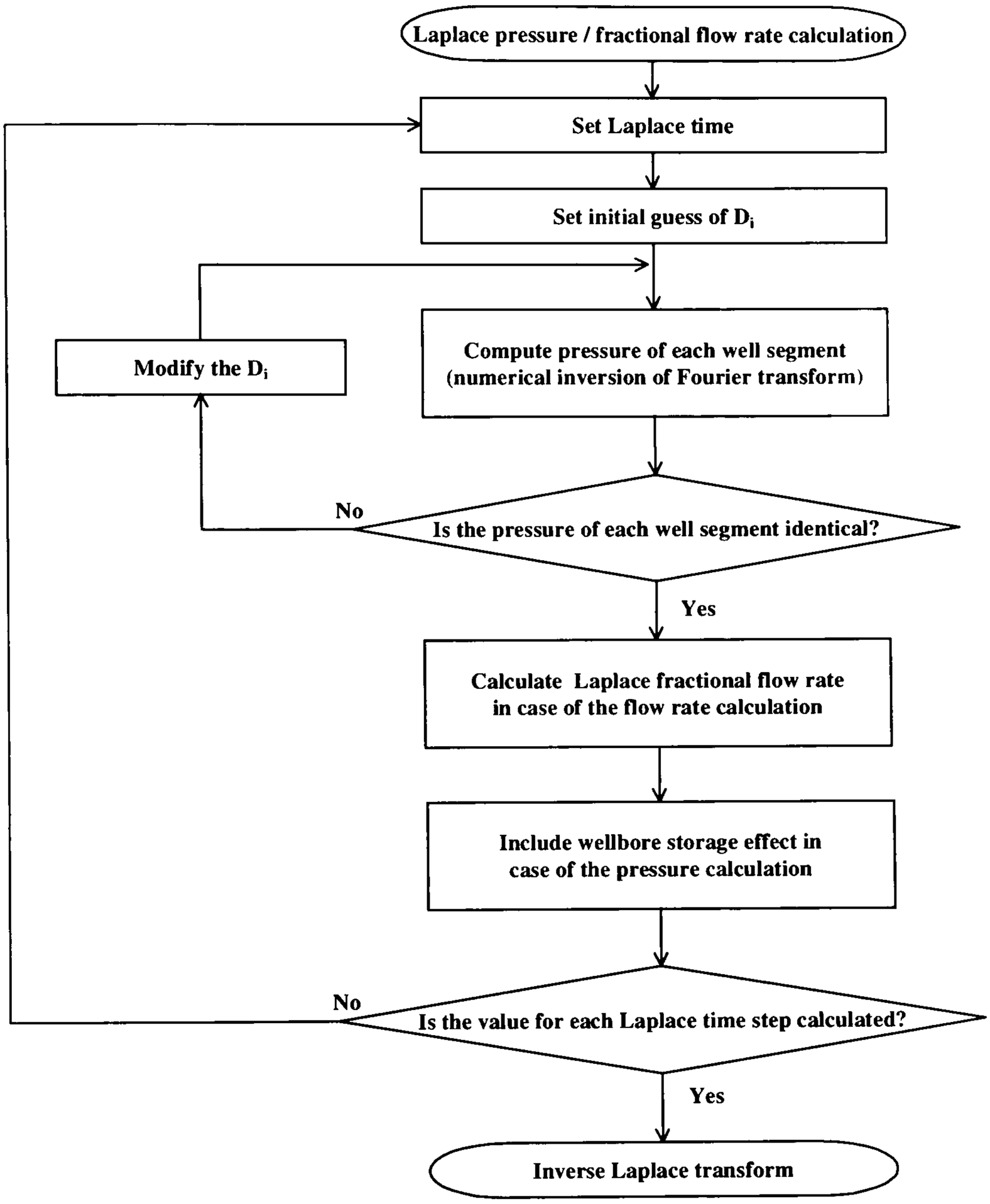


Figure B.2: Flow chart for the Laplace pressure or the Laplace fractional flow rate calculation.

6. Step 3 to Step 5 are repeated until the wellbore pressure of each well segment becomes identical. Since the governing system of equations is non-linear, Newton-Raphson method with a globally convergent strategy developed by Press et al. [95] was adopted to solve it.

7. For the fractional flow rate calculation, the Laplace fractional flow rate of selected well segments is computed from Eq. 4.160.

8. The wellbore pressure or the fractional flow rate is inverted to the real domain using the Stehfest algorithm [111].

The general solution of Eq. 4.141 includes two exponential terms which may cause overflow or underflow problems during the calculation. For such cases, a class library of multiple precision arithmetic (written in C++ language) developed by Fukuda [37] was implemented for the main program. This library enables us to handle arbitrary digits with arbitrary precision. The standard mathematical functions defined in ANSI C `<math.h>` are available in the multiple precision arithmetic library. The number of digits k is defined by the keyword `lb_digits(k)`, and it is set to 20 in this program. This option is used only when the overflow or underflow problem occurs during the calculation. The library is not listed in this thesis. However, it can be available freely from the Internet website of Ref. [37].

In Step 5, an integration with an infinite upper limit must be evaluated, which is not suitable for numerical calculation. However, the integral for the region where τ in Eq. 4.155 is large can be neglected because the integrand decreases rapidly as τ becomes large. Four-point and ten-point Gauss-Legendre integration were used for small τ and large τ, respectively. Although the ten-point integration is more accurate, the four-point integration is considered to be enough because the interval to be integrated is divided into many small time intervals for the small τ case. The above statement can be expressed

mathematically as

$$\begin{aligned} F(\tau) &= \int_0^\infty f(\tau)\,d\tau \\ &\simeq \underbrace{\int_0^{\delta\tau_1} f(\tau)\,d\tau + \int_{\delta\tau_1}^{\delta\tau_1+\delta\tau_2} f(\tau)\,d\tau + \cdots + \int_{\sum_{i=1}^{k-1}\delta\tau_i}^{\sum_{i=1}^{k}\delta\tau_i} f(\tau)\,d\tau + \cdots}_{\text{4 point integration}} \\ &\quad + \underbrace{\int_{\tau_s}^{\tau_s/\sqrt{\overline{\delta z_w}}} f(\tau)\,d\tau}_{\text{10 point integration}}, \end{aligned} \tag{B.1}$$

where $f(\tau)$ is the integrand in Eq. 4.155 and $\delta\tau_i$ is the time step determined by

$$\delta\tau_i = 0.005 e^{\frac{0.005}{\overline{\delta z_w}^{0.125}} i}. \tag{B.2}$$

In this expression, i is a loop counter for the inverse transform, and $\overline{\delta z_w}$ is a dimensionless average wellbore strip width expressed (using the notation in Chapter 4) as

$$\overline{\delta z_w} = \frac{2\sum_{i=i_{w,1}}^{i_{w,2}} \delta x_i \delta z_{w,i}}{L_w^2}. \tag{B.3}$$

The numerical integration of $f(\tau)$ is considered to be converged when the following condition is satisfied, i.e.,

$$\frac{\int_{\sum_{i=1}^{k-1}\delta\tau_i}^{\sum_{i=1}^{k}\delta\tau_i} f(\tau)\,d\tau}{\int_0^{\delta\tau_1} f(\tau)\,d\tau + \int_{\delta\tau_1}^{\delta\tau_1+\delta\tau_2} f(\tau)\,d\tau + \cdots + \int_{\sum_{i=1}^{k-1}\delta\tau_i}^{\sum_{i=1}^{k}\delta\tau_i} f(\tau)\,d\tau} < 10^{-5}. \tag{B.4}$$

If the overflow or the underflow problem occurs before reaching this criterion, the program switches to the multiple precision arithmetic algorithm, and the calculation continues. When it occurs at $\tau = \tau_s$, the four-point integration is replaced by the ten-point integration. In that case, the integration is performed only once between the points $\tau = \tau_s$ and $\tau = \tau_s/\sqrt{\overline{\delta z_w}}$. It has been empirically found that the above settings work well for applications in practical time range.

Another feature of this program is the reduction of Laplace time steps needed for the calculation of the real domain pressure or fractional flow rate. As shown in Section 3.1.7, a value in the real domain $f(t)$ at the time t is computed from several values in the Laplace domain $\overline{f}(l)$ at $l_i = i\ln 2/t$ $(1 \le i \le N)$. A schematic diagram for $N = 4$ is shown in Fig. B.3.

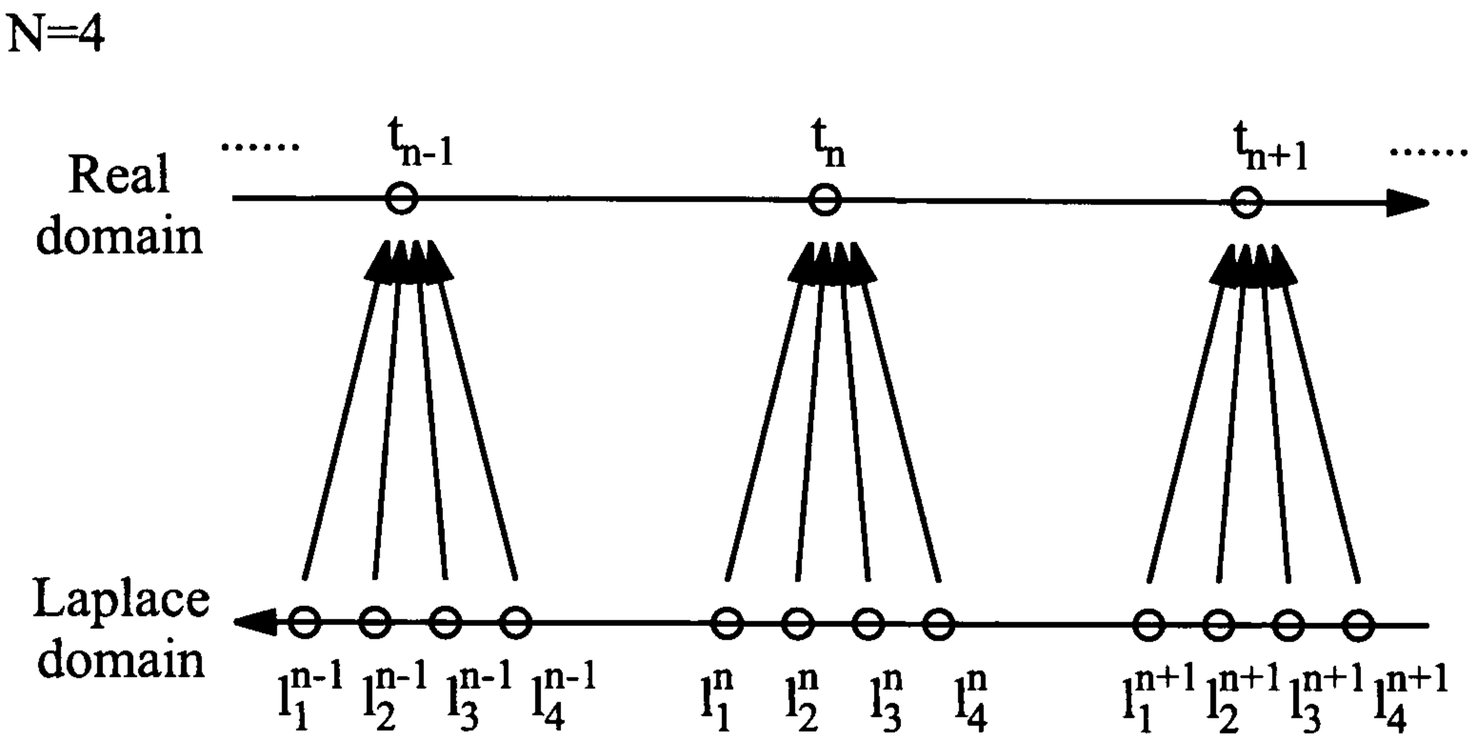


Figure B.3: The relationship between the real domain time steps and the Laplace domain time steps.

Table B.1: Laplace time steps required for calculating real domain values for $N = 4$.

Time step	1	2	3	...	n_{max}
$i = 1$	$l_1^1 = \ln 2/t_1$	$l_1^2 = \ln 2/2t_1$	$l_1^3 = \ln 2/4t_1$	...	$l_1^{n_{max}} = \ln 2/2^{n_{max}} t_1$
$i = 2$	$l_2^1 = 2\ln 2/t_1$	$l_2^2 = l_1^1$	$l_2^3 = l_1^2$	...	$l_2^{n_{max}} = l_1^{n_{max}-1}$
$i = 3$	$l_3^1 = 3\ln 2/t_1$	$l_3^2 = 3\ln 2/2t_1$	$l_3^3 = 3\ln 2/4t_1$	...	$l_3^{n_{max}} = 3\ln 2/2^{n_{max}} t_1$
$i = 4$	$l_4^1 = 4\ln 2/t_1$	$l_4^2 = l_2^1$	$l_4^3 = l_2^2$	...	$l_4^{n_{max}} = l_2^{n_{max}-1}$

In general, the Laplace time steps used for calculating values in the real domain doesn't overlap each other. However, if the real time steps t_n are chosen in the way that $t_n = 2t_{n-1}$ $(2 \leq n \leq n_{max})$, not a few time steps can overlap each other. Such situation is explained in Table B.1 for $N = 4$. Since the Laplace pressures should be calculated at each Laplace time step, the sharing of the Laplace time steps contributes the reduction of computation time considerably. If the real time steps are set in this way, the number of points in every log cycle becomes 3 or 4, which is considered to be enough in most cases when the logarithmic time is considered. N was set to 6 in this program.

The globally convergent method used in Step 6 converges to a solution from almost any starting point [95]. The method uses the Newton-Raphson method for a rapid local

convergence, combined with a globally convergent strategy which will guarantee some progress toward the solution at each iteration. The goal is to find variables x_i $(1 \leq i \leq N)$ which satisfy the relations,

$$F_i(x_1, x_2, \ldots, x_N) = 0 \qquad \text{for } i = 1, 2, \ldots, N. \tag{B.5}$$

In this method, the above condition is transformed to

$$f = \sum_{i=1}^{N} F_i^2, \tag{B.6}$$

and the variables x_i to minimise f are found. The Newton-Raphson method is applied first, and the reduction of f is checked at each iteration. In the Newton-Raphson method, the variables x_i moves to a descent direction for f. If f does not decrease, the searching direction of x_i is backtracked until an acceptable step is found. Press et al. [95] mentioned that this method can occasionally fail by landing on a local minimum of f, however, this is quite rare in practice.

To compute pressure derivative, rational function interpolation of pressure shown in Section A.1 was adopted.

B.2 Variables and Functions

In this section, variables and functions used in the computer program are explained. The principal variables are listed in Table B.2 with their descriptions.

The name of functions and explanations are shown below.

- **int main()**: This function calls other functions and defines the procedure of the calculation.

- **void read_data()**: Reservoir, well, fluid data, and time interval to be calculated are read from an input file.

- **void set_well_block()**: The grid blocks are re-constructed according to the strip-wellbore width. If the reservoir is symmetric, the number of grid blocks is reduced in the function `void lat_half_res()` (symmetric on the vertical $y-z$ plane), `void`

Table B.2: Variable table for the program.

Variable name	Description
m, n	The number of grid blocks in the x and z directions
output_units	Output units, 0=Dimensionless units calculated from Eqs. 4.71 and 4.74, 1=Field units
calc_type	Type of calculation, 0=Pressure, 1=Fractional flow rate
r_mode	Region of the reservoir to be calculated, 0=Full reservoir, 1=laterally half, 2=vertically half, 3=quater. This option is used to reduce the number of the grid blocks by considering the symmetry of the reservoir.
iw1, iw2	Grid numbers i in the x direction for the left-most and the right-most well segments
iw1_q, iw2_q	Minimum and maximum well segment grid numbers i in the x direction considered in the fractional flow rate calculation
jw1[i], jw2[i]	The smallest and the largest vertical grid numbers j for the well segment i
dx[i]	Grid size in the x direction
dz[j], dzd[j]	Dimensional and dimensionless grid size in the z direction
dzw[i], dzwd[i]	Dimensional and dimensionless strip wellbore width for the well segment i
h	Total thickness of the reservoir
zw	Well vertical position in the well blocks
kx[i][j], ky[i][j], kz[i][j]	Directional permeabilities for the grid block (i, j)
phi[i][j]	Porosity for the grid block (i, j)
ct[i][j]	Total compressibility for the grid block (i, j)
mu	Viscosity
q	Total production rate
bo	Formation volume factor
pin	Initial reservoir pressure
rw	Wellbore radius
skin[i]	Skin factor for the well segment i
wbs, wbsd	Dimensional and dimensionless wellbore storage constant
ave_k, ave_phic	Average permeability and porosity-compressibility product defined by Eqs. 4.73 and 4.76
ave_dzwd	Average dimensionless strip wellbore width used to determine time step increments
alpha[i][j], beta[i][j], lambda[i][j], omega[i][j]	Dimensionless variables for each grid block or each interface between the blocks defined by Eqs. 4.82, 4.83, 4.85, and 4.84.
zeta[i][j]	Mobility ratio between the blocks (i, j) and $(i, j+1)$.
sigma[i][j], E[i][j][k]	Square root of the eigenvalues of the system described by Eq. 4.126, and constants $E^k_{i,j}{}'$ in Eqs. 4.139 and 4.140 for the block (i, j)
rho[i][j]	$\tilde{\lambda}_{i+1,j} + \tilde{\lambda}_{i,j} + \omega_{i,j} l$ in Eq. 4.116
nu[i][j]	$\alpha_{i,j} r^2 +$ rho[i][j] in Eq. 4.116
psi[i][j]	$-\tilde{\lambda}_{i,j} / \sqrt{\beta_{i-1,j} \beta_{i,j}}$ in Eq. 4.126
real_min_t, real_max_t	Minimum and maximum times of the time interval to be calculated
real_count	The number of time steps
t[k], sol[k], dp[k]	Time, solution (pressure or fractional flow rate), and pressure derivative

`vert_half_res()` (symmetric on the horizontal $x-y$ plane), `void quater_res()` (symmetric on both of the $x-y$ and $y-z$ planes).

- **void print_input_data()**: The input data and the re-constructed grid blocks are printed on the screen.

- **void input_unit_conv()**: The units of input data (field units) is transformed to the Darcy units.

- **void dimless_var()**: The dimensionless variables ($\alpha_{i,j}$, $\beta_{i,j}$, $\omega_{i,j}$, $\tilde{\lambda}_{i,j}$ of Eqs. 4.82, 4.83, 4.84, and 4.85, the dimensionless layer thickness z_D and the dimensionless strip-wellbore width $z_{w,D}$ of Eq. 4.79, the mobility ratio $\zeta_{i,j}$ of Eq. 4.41, the dimensionless wellbore storage constant C_D of Eq. 4.80) are defined.

- **void tstep_real()**: In this function, time steps in the real domain are constructed between `real_min_t` and `real_max_t`. Each time step is determined to be two times larger than the previous value.

- **void initialize_D()**: This function sets an initial guess of the pressure gradients D_i for the Newton-Raphson iteration, where all the D_i are set to an identical value.

- **void get_solution()**: The pressure or the fractional flow rate is computed using the Newton-Raphson method with a globally convergent strategy (the function `void newt()`). For the next Laplace time step, the pressure gradients D_i are initialised by keeping the same proportions among the D_i as those of the previous time step. The Laplace pressure or fractional flow rare is transformed to that in the real domain in the function `void stehfest()`. The pressure derivative is computed in the function `void calc_deriv()` using the rational function interpolation (the function `void ratint()`).

- **void newt()**: The Newton-Raphson method with a globally convergent strategy was implemented [95]. This function calls functions `void ludcmp()` and `void lubksb()` for solving linear equations by LU decomposition, `void lnsrch()` for finding the

new x_i in Eq. B.5, `void fdjac()` for calculating Jacobian, and `double fmin()` for computing f of Eq. B.6.

- **void wellbore_pressure()**: The system of equations which consists of the wellbore boundary condition of Eq. 4.158 is set up.

- **void inv_fourier()**: The inverse Fourier transform of Eq. 4.155 is performed using the Gauss-Legendre integration. The pressure after being applied the pressure averaging method ($\overline{\overline{p}}_{D,wf,i}$ where $i_{w,1} \leq i \leq i_{w,2}$) is obtained in the function `int calc_lap_p()` or `int lb_calc_lap_p()`. The function `int lb_calc_lap_p()` is used only if the overflow or underflow problem occurs.

- **int calc_lap_p(), void lb_calc_lap_p()**: The function `int calc_lap_p()` calls functions `void calc_particular_sol()`, `void eigenvalues()`, `int const_ab()`, and `void calc_dlap_p()`. The exponential terms in Eq. 4.146 are checked, and if the overflow or underflow occurs, the function returns -1 as the sign of the problem. In that case, the function is replaced by the function `int lb_calc_lap_p()` which utilises the multiple precision arithmetic.

- **void calc_particular_sol()**: The particular solutions $C_{i,j}$ are computed from Eq. 4.142. The function `void tridag()` is used to obtain the solution of the symmetric tridiagonal system of equations.

- **void eigenvalues()**: The square root of the eigenvalues $(\sigma_j^k)^2$ and the constant $E_{i,j}^{k\ \prime}$ related to the eigenvectors are computed. The function `void tqli()` is used to calculate the eigenvalues and the eigenvectors.

- **void tqli()**: This function was adopted from Ref. [95]. The eigenvalues and the eigenvectors are computed by QL algorithm with implicit shifts.

- **int const_ab(), void lb_const_ab()**: In the function `int const_ab()`, the constants $A_{j,k}$ and $B_{j,k}$ in Eq. 4.146 are calculated using the boundary conditions from Eqs. 4.147 to 4.152. To solve the system of equations, the Gauss-Jordan

elimination with partial pivoting (the function `void gauss()` or `void lb_gauss()` if the overflow or the underflow occurs) is used. If the overflow or the underflow problem occurs during the calculation, the function returns -1 as the sign of the problem. In that case, the function is replaced by the function `int lb_const_ab()` which utilises the multiple precision arithmetic.

- **void calc_dlap_p(), void lb_calc_dlap_p()**: The pressure averaging method of Eq. 4.114 is applied for the well segment i. The Laplace pressure $\overline{p}_{D,wf,i}$ for the well segment i is computed, and then the value is used in the function `void inv_fourier()`. When the overflow or the underflow problem occurs, the function `void lb_calc_dlap_p()` is used instead.

- **double calc_lap_q()**: This function returns the value of the Laplace fractional flow rate from the well segments `iw1_q` $\leq i \leq$ `iw2_q`.

- **void output_unit_conv()**: In this function, the dimensionless units of the pressure, the pressure derivative, and the time are converted to the field units using the functions `void dimensional_var()` and `void unit_conv_darcy_to_field()`.

B.3 Input Data

A sample input file is shown below with comments for data. The grid number is assigned for each grid block in the same way as shown in Fig. 4.5. The reservoir parameters for each grid block should be entered from the left block to the right, and then the top block to the bottom. Field units must be used for the data in the input file.

Sample input file	**\Title**
2	**\Calculation mode: 0=Full, 1=Laterally half, 2=Vertically half, 3=Quater**
1	**\Output units: 0=Dimensionless, 1=Field**
0	**\Type of calculation: 0=Pressure, 1=Flow rate**
0.001 1000	**\Calculation interval, dimensionless or hrs**
8 1	**\Grid number: m × n**
3 6 1	**\Well grids: i_min, i_max, j**
3 4	**\Well segments for rate calculation**
0.33	**\Wellbore radius, ft**
50	**\Well vertical distance from the bottom of the well block, ft**
1000 1000 1000 1000 100 100 100 100	**\kx, md**
1000 1000 1000 1000 100 100 100 100	**\ky, md**

```
50 50 50 50 20 20 20 20                    \kz, md
750 250 70 30 150 250 250 750              \Lateral grid size
100                                        \Vertical grid size
0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2            \Porosity
1e-5 1e-5 1e-5 1e-5 1e-5 1e-5 1e-5 1e-5    \Compressibility, 1/psi
0.8                                        \Viscosity, cp
1000                                       \Total flow rate, bbl/d
1.2                                        \Formation volume factor
3000                                       \Initial pressure, psi
0.0 0.0 0.0 0.0                            \Skin factor
0.0                                        \Wellbore storage, bbl/psi
```

B.4 Header File

This header file (`horiz.h`) defines constants, macros, the prototypes of functions, and global variables. This file is included in the main program at its beginning.

horiz.h

```
/*
header file for horiz.cpp
A.Sagawa
*/
/*** pi ***/
#define PI 3.141592653589793
/*** maximum length of a file name ***/
#define MAXLEN 30
/*** maximum block size ***/
#define MAXI 10
#define MAXJ 10
/*** inverse Fourier variable increment ***/
#define DL 0.005
#define CONST_F 0.005 /* fixed */
#define POW_INDEX 0.125
/*** small value for derivative calculation ***/
#define DT 1e-5
/*** tolerance for value identification ***/
#define D_EPS 1e-5
/*** convergence criteria of inverse Fourier transform ***/
#define INVF_EPS 1e-6 /* Fixed */
/*** maximum iteration for inverse Fourier transform ***/
#define INVF_MAXITR 10000
/*** maximum time step in Laplace domain ***/
#define L_MAX_TSTEP 100
/*** maximum time step in real domain ***/
#define R_MAX_TSTEP 30
/*** constant for Stehfest's algorithm ***/
#define NSTEH 6
/*** positive sign ***/
#define SIGN(a, b) ((b) ≥ 0.0 ? fabs(a) : - fabs(a))
/*** accuracy of multiple precision arithmetic ***/
#define LB_ACC 20 /*** fixed ***/
/*** constants for global convergent method ***/
#define MAXITS 200
#define EPS 1.0e-4
#define TOLF 1.0e-6
#define TOLX 1.0e-10
#define STPMX 100.0
```

```
#define TOLMIN 1.0e-6
#define ALF 1.0e-4
#define TINY 1.0e-20
/*** square ***/
static double sqrarg;
#define SQR(a) ((sqrarg = (a)) == 0.0 ? 0.0 : sqrarg * sqrarg)
/*** swap two values ***/
#define SWAP(a, b) {temp = (a); (a) = (b); (b) = temp;}
/*** return the larger value ***/
static double maxarg1, maxarg2;
#define DMAX(a,b) (maxarg1=(a),maxarg2=(b),(maxarg1) > (maxarg2) ?\
    (maxarg1) : (maxarg2))
/*** error exit ***/
#define ERROR_EXIT(s) {printf("%s\n", s); exit(0);}

/*
prototypes of functions
*/
void read_data(double kx[][MAXJ], double ky[][MAXJ], double kz[][MAXJ],
               double phi[][MAXJ], double dz[], double *h, double *mu,
               double ct[][MAXJ], double *q, double *bo,
               double *pin, double *rw, double *zw, double skin[],
               double *wbs, double *real_min_t, double *real_max_t,
               int *output_units, int *r_mode, int *calc_type,
               int *iw1_q, int *iw2_q);
void set_well_block(double kx[][MAXJ], double ky[][MAXJ],
                    double kz[][MAXJ], double phi[][MAXJ],
                    double ct[][MAXJ], double dz[],
                    double dzw[], double rw, double zw,
                    double skin[], int r_mode, int *iw2_q);
void print_input_data(double kx[][MAXJ], double ky[][MAXJ],
                      double kz[][MAXJ], double phi[][MAXJ],
                      double dz[], double mu, double ct[][MAXJ],
                      double q, double bo, double pin, double rw,
                      double skin[], double wbs, double real_min_t,
                      double real_max_t, int output_units,
                      int calc_type);
void input_unit_conv(double kx[][MAXJ], double ky[][MAXJ],
                     double kz[][MAXJ], double dz[],
                     double dzw[], double ct[][MAXJ],
                     double *q, double *pin, double *rw,
                     double *wbs, double *h, double *real_min_t,
                     double *real_max_t, int output_units);
void dimless_var(double kx[][MAXJ], double ky[][MAXJ],
                 double kz[][MAXJ], double phi[][MAXJ],
                 double dz[], double dzw[], double h,
                 double mu, double ct[][MAXJ], double rw,
                 double wbs, double *wbsd, double *ave_k,
                 double *ave_phic, double *real_min_t,
                 double *real_max_t, int output_units);
void dimensional_var(int real_count, double t[], double p[],
                     double dp[], double h, double q, double mu,
                     double bo, double ave_k, double ave_phic);
void unit_conv_darcy_to_field(int real_count, double t[],
                              double p[], double dp[]);
void output_unit_conv(int output_units, int r_mode, int real_count,
                      double t[], double sol[], double dp[], double h,
                      double q, double mu, double bo, double ave_k,
                      double ave_phic, int calc_type);
void get_solution(double t[], double sol[], double dp[],
                  int real_count, double wbsd, int calc_type,
                  int iw1_q, int iw2_q);
double calc_lap_q(double D[], int iw1_q, int iw2_q);
void tstep_real(int *real_count, double real_min_t, double real_max_t,
                double t[]);
void piksrt(int r, double arr[]);
void tridag(double a[], double b[], double r[], double C[][MAXJ],
            int j, int rank);
int gauss(int r, double a[][2*MAXI*MAXJ], double b[]);
```

```
void lb_gauss(int r, lb a[][2*MAXI*MAXJ], lb b[]);
void define_rho();
void stehfest(int laplace_count, int real_count, double lap_p[],
              double lap_t[], double t[], double sol[]);
void calc_deriv(int real_count, double sol[], double t[], double dp[]);
int calc_lap_p(double l, double l_p[], double D[]);
void lb_calc_lap_p(double l, double l_p[], double D[]);
void set_lap_t(int real_count, int *laplace_count, double t[],
               double lap_t[]);
void eigenvalues();
int const_ab(double C[][MAXJ], double sol[], double expn[][MAXI]);
void lb_const_ab(double C[][MAXJ], lb sol[], lb expn[][MAXI]);
void calc_particular_sol(double C[][MAXJ], double D[]);
void calc_dlap_p(double sol[], double C[][MAXJ], double expn[][MAXI],
                 double dl_p[]);
void lb_calc_dlap_p(lb sol[], double C[][MAXJ], lb expn[][MAXI],
                    double dl_p[]);
void tqli(double d[], double e[], int rk, double z[][MAXI]);
double pythag(double a, double b);
void calc_deriv(int real_count, double p[], double t[], double dp[]);
void ratint(double xa[], double ya[], int r, double x, double *y,
            double *dy);
void inv_fourier(double D[]);
void initialize_D(double D[], double lap_t0);
void newt(double x[], int rank, int *check,
          void (*vecfunc)(double [], int, double []));
void ludcmp(double a[][MAXI], int r, int indx[], double *d);
void lubksb(double a[][MAXI], int r, int indx[], double b[]);
void fdjac(int r, double x[], double fvec[], double df[][MAXI],
           void (*vecfunc)(double [], int, double []));
double fmin(double x[]);
void lnsrch(int rank, double xold[], double fold, double g[],
            double p[], double x[], double *f, double stpmax,
            int *check, double (*func)(double []));
void lat_half_res(int *iw2_q);
void vert_half_res(double dz[], double dzw[]);
void quarter_res(double dz[], double dzw[], int *iw2_q);
void wellbore_pressure(double D[], int r, double vec[]);
void (*nrfuncv)(double D[], int rank, double vec[]);

/*
global variables
*/
/*** block size ***/
int m, n;
/*** well position ***/
int iw1, iw2, jw1[MAXI], jw2[MAXI];
/*** dimensionless variables ***/
double alpha[MAXI][MAXJ], beta[MAXI][MAXJ];
double lambda[MAXI+1][MAXJ], omega[MAXI][MAXJ];
/*** mobility ratio ***/
double zeta[MAXI][MAXJ+1];
/*** dimensionless parameters ***/
double E[MAXI][MAXJ][MAXI], sigma[MAXJ][MAXI], nu[MAXI][MAXJ];
double psi[MAXI+1][MAXJ], rho[MAXI][MAXJ];
/*** x block size ***/
double dx[MAXI];
/*** strip width of the well blocks ***/
double dzwd[MAXI];
/*** dimensionless z block size ***/
double dzd[MAXJ];
/*** well length ***/
double lw;
/*** average wellbore strip width ***/
double ave_dzwd;
/*** minmum and maximum j of well blocks ***/
int minjw, maxjw;
/*** index for solution matrix ***/
int offset_A[MAXJ], offset_B[MAXJ];
```

```
/*** Laplace time and pressure ***/
double lap_p_b[MAXI], lap_time;
/*** global variables needed for global convergent method ***/
int rr;
double fvec[MAXI];
```

B.5 Main Program

The main program (`horiz.cpp`) is listed below. This main program uses the multiple precision arithmetic program `lb.cpp`, which is not listed here. The header file `lb.h` for the program is included in the first part of this program.

horiz.cpp

```
/*
Pressure and pressure derivative calculation
for welltest analysis of horizontal well in
reservoirs having permeability discontinuities
A. Sagawa 16/11/99
*/
#include <float.h>
#include "lb.h"
#include "horiz.h"

main()
{
    /*
    declaration of variables
    */

    int i;
    /*** permeabilities ***/
    double kx[MAXI][MAXJ], ky[MAXI][MAXJ], kz[MAXI][MAXJ];
    /*** porosity ***/
    double phi[MAXI][MAXJ];
    /*** viscosity ***/
    double mu;
    /*** total compressibility ***/
    double ct[MAXI][MAXJ];
    /*** total flow rate ***/
    double q;
    /*** formation volume factor ***/
    double bo;
    /*** initial pressure ***/
    double pin;
    /*** wellbore radius ***/
    double rw;
    /*** z block size in field units ***/
    double dz[MAXJ];
    /*** width of the well strip in field units ***/
    double dzw[MAXI];
    /*** total thickness of leyers ***/
    double h;
    /*** vertical well position ***/
    double zw;
    /*** skin factor ***/
    double skin[MAXI];
    /*** wellbore storage ***/
    double wbs;
    /*** average k ***/
```

```
double ave_k;
/*** average phi ct ***/
double ave_phic;
/*** dimensionless welbore storage ***/
double wbsd;
/*** counter of real pressure table ***/
int real_count;
/*** real time ***/
double t[R_MAX_TSTEP];
/*** solution (pressure or rate) in real domain ***/
double sol[R_MAX_TSTEP];
/*** pressure derivative in real domain ***/
double dp[R_MAX_TSTEP];
/*** minimum and maximum time in real domain ***/
double real_min_t, real_max_t;
/*** file pointer for an output file ***/
FILE *fp;
/*** output file name ***/
char line[20];
/*** units for output ***/
int output_units;
/*** reservoir mode ***/
/* 0: full reservoir,  1: laterally half  */
/* 2: vertically half, 3: quarter         */
int r_mode;
/*** calculation type, 0: pressure, 1: flow rate ***/
int calc_type;
/*** zone where flow rate is calculated ***/
/*** iw1 ≤ iw1_q, iw2_q ≤ iw2 ***/
int iw1_q, iw2_q;

/*** read input data ***/
read_data(kx, ky, kz, phi, dz, &h, &mu, ct, &q, &bo,
    &pin, &rw, &zw, skin, &wbs, &real_min_t,
    &real_max_t, &output_units, &r_mode, &calc_type,
    &iw1_q, &iw2_q);

/*** set up well blocks ***/
set_well_block(kx, ky, kz, phi, ct, dz, dzw, rw, zw,
    skin, r_mode, &iw2_q);

/*** open an output file ***/
printf("output file name = ");
scanf("%s", line);
if((fp = fopen(line, "w")) == 0)
    ERROR_EXIT("Input file open error!");

/*** print input data ***/
print_input_data(kx, ky, kz, phi, dz, mu, ct, q, bo,
    pin, rw, skin, wbs, real_min_t, real_max_t,
    output_units, calc_type);

/*** unit conversion, field -> darcy ***/
input_unit_conv(kx, ky, kz, dz, dzw, ct, &q, &pin,
    &rw, &wbs, &h, &real_min_t, &real_max_t, output_units);

/*** define dimensionless variables ***/
dimless_var(kx, ky, kz, phi, dz, dzw, h, mu, ct, rw, wbs,
    &wbsd, &ave_k, &ave_phic, &real_min_t, &real_max_t,
    output_units);

/*** dimensionless time step in real domain ***/
tstep_real(&real_count, real_min_t, real_max_t, t);

/*** calculate pressure & derivative, or rate ***/
get_solution(t, sol, dp, real_count, wbsd, calc_type,
    iw1_q, iw2_q);

/*** unit conversion of results ***/
```

```
output_unit_conv(output_units, r_mode, real_count, t, sol,
    dp, h, q, mu, bo, ave_k, ave_phic, calc_type);

/*** output data to an file ***/
if (calc_type == 0)
    for (i = 1; i < real_count - 1; i++)
        fprintf(fp, "%lf %lf %lf\n", t[i], sol[i], dp[i]);
else if (calc_type == 1)
    for (i = 1; i < real_count - 1; i++)
        fprintf(fp, "%lf %lf\n", t[i], sol[i]);

/*** output file close ***/
    printf("Output was written in the file \"%s\".\n", line);
    fclose(fp);

    return 0;
}

/*
unit conversion of results
*/
void output_unit_conv(int output_units, int r_mode, int real_count,
                double t[], double sol[], double dp[], double h,
                double q, double mu, double bo, double ave_k,
                double ave_phic, int calc_type)
{
    int i;

    /*** for pressure calculation ***/
    if (calc_type == 0) {

        /*** for a dimensionless case ***/
        if (output_units == 0) {
    if ((r_mode == 1) || (r_mode == 2))
        for (i = 0; i < real_count; i++) {
            sol[i] *= 0.5;
            dp[i] *= 0.5;
        }
    else if (r_mode == 3)
        for (i = 0; i < real_count; i++) {
            sol[i] *= 0.25;
            dp[i] *= 0.25;
        }

/*** for a dimensional case ***/
} if (output_units == 1) {
    /*** dimensional variables, dimensionless -> darcy ***/
    dimensional_var(real_count, t, sol, dp, h, q, mu, bo,
        ave_k, ave_phic);

    /*** unit conversion, darcy -> field ***/
    unit_conv_darcy_to_field(real_count, t, sol, dp);
}

    } else if (calc_type == 1) /*** for rate calculation ***/

        if (output_units == 1) {

            for (i = 0; i < real_count; i++)
                t[i] *= mu * ave_phic * lw * lw / 4.0 / ave_k /3600.0;

            if ((r_mode == 1) || (r_mode == 3))
                for (i = 0; i < real_count; i++)
                    sol[i] *= 0.5;
        }
}

/*
set time steps in real domain
```

```
*/
void tstep_real(int *real_count, double real_min_t, double real_max_t,
                double t[])
{
    t[0] = real_min_t / 2.0;
    *real_count = 1;
    do {
        t[*real_count] = t[*real_count-1] * 2.0;
        (*real_count)++;
    } while (t[*real_count-1] < real_max_t * 2.0);
}

/*
calculate pressure and pressure derivative in dimensionless units
*/
void get_solution(double t[], double sol[], double dp[],
                  int real_count, double wbsd, int calc_type,
                  int iw1_q, int iw2_q)
{
    int i, j, k, laplace_count, check;
    double lap_sol[L_MAX_TSTEP], lap_t[L_MAX_TSTEP], D[MAXI];

    /*** search well layers ***/
    minjw = jw1[iw1];
    maxjw = jw2[iw1];
    for (i = iw1 + 1; i <= iw2; i++) {
        if (jw1[i] < minjw)
            minjw = jw1[i];
        if (jw2[i] > maxjw)
            maxjw = jw2[i];
    }

    /*** calcurate offsets of grids for matrix calculation ***/
    offset_A[0] = 0;
    offset_B[0] = m;
    for (i = 1; i <= n; i++) {
        offset_A[i] = offset_A[i-1] + 2 * m;
        offset_B[i] = m + offset_A[i];
    }

    /*** set time step in Laplacian domain ***/
    set_lap_t(real_count, &laplace_count, t, lap_t);
    printf("%d time steps will be processed.\n", laplace_count);

    /*** off-diagonal elements for eigenvalue calculation ***/
    for (j = 0; j < n; j++) {
        psi[0][j] = psi[m][j] = 0.0;
        for (i = 1; i < m; i++)
            psi[i][j] = - lambda[i][j]
                / sqrt(beta[i-1][j] * beta[i][j]);
    }

    /*** initialize D's ***/
    initialize_D(D, lap_t[0]);

    /*** loop for Laplace time step ***/
    for (k = 0; k < laplace_count; k++) {

        lap_time = lap_t[k];

        printf("%2d/%2d: l_t = %e\n", k, laplace_count - 1,
            lap_time);

        /*** define rho for eigenvalue calculation ***/
        define_rho();

        /*** modify D's for the next N-R step ***/
        if (k != 0)
            for (i = 0; i < iw2 - iw1; i++)
```

```
        D[i] *= lap_t[k-1] / lap_time;

/*** global convergent method for const.press b.c. ***/
newt(D, iw2 - iw1, &check, &wellbore_pressure);

if (check == 0)
    printf("normal.\n");
else if (check == 1)
    ERROR_EXIT("local minimum!");

if (calc_type == 0) {
        /*** include wellbore storage effect ***/
        lap_sol[k] = 1.0 / (lap_time * lap_time * wbsd
                + 1.0 / lap_p_b[iw1]);

        printf("l_p = %e\n", lap_sol[k]);

    } else if (calc_type == 1) {

        /*** calculate Laplace flow rate ***/
        lap_sol[k] = calc_lap_q(D, iw1_q, iw2_q);
        printf("l_q = %e\n", lap_sol[k]);
    }
}

/*** inverse Laplace transform ***/
if (calc_type == 0) {
    /*** pressure ***/
    stehfest(laplace_count, real_count, lap_sol, lap_t, t, sol);

    /*** calculate derivative ***/
    calc_deriv(real_count, sol, t, dp);
    } else if (calc_type == 1)
        /*** flow rate ***/
        stehfest(laplace_count, real_count, lap_sol, lap_t, t, sol);
}

/*
calculate flow rate in Laplace domain
*/
double calc_lap_q(double D[], int iw1_q, int iw2_q)
{
    int i, j;
    double sum1, sum2;

    for (sum2 = 0.0, i = iw1_q; i ≤ iw2_q; i++) {
        for (sum1 = 0.0, j = jw1[i]; j ≤ jw2[i]; j++)
            sum1 += alpha[i][j] * dzd[j];
        sum2 += D[i-iw1] * sum1;
    }

    return - sum2 / PI;
}

/*
initialize D's for lap_t[0]
*/
void initialize_D(double D[], double lap_t0)
{
    int i, j;
    double sum = 0.0;

    if (iw1 ≠ iw2) {
        for (i = iw1; i ≤ iw2; i++)
            for (j = jw1[i]; j ≤ jw2[i]; j++)
                sum += dzd[j] * alpha[i][j];
        sum = - PI / lap_t0 / sum;
```

```
        /*** D[iw2-iw1] is determined from the rest ***/
        for (i = 0; i < iw2 - iw1; i++)
            D[i] = sum;
    }
}

/*
boundary condition for wellbore
*/
void wellbore_pressure(double D[], int r, double vec[])
{
int i, j;
double sum1 = 0.0, sum2 = 0.0;

/*** calculate D[iw2-iw1] from the rest of D's ***/
for (i = iw1; i < iw2; i++)
    for (j = jw1[i]; j ≤ jw2[i]; j++)
        sum1 += dzd[j] * alpha[i][j] * D[i-iw1];
for (j = jw1[iw2]; j ≤ jw2[iw2]; j++)
    sum2 += dzd[j] * alpha[iw2][j];
D[iw2-iw1] = (- PI / lap_time - sum1) / sum2;

/*** print D's ***/
printf("D: ");
for (i = iw1; i ≤ iw2; i++)
    printf("%e, ", D[i-iw1]);
printf("\n");

/*** calculate pressures for each block ***/
inv_fourier(D);

/*** define functions to be set 0 ***/
for (i = iw1 + 1; i ≤ iw2; i++)
        vec[i-iw1-1] = (lap_p_b[i] - lap_p_b[iw1])
            / lap_p_b[iw1];

    /*** print pressures in each block ***/
    printf("p: ");
    for (i = iw1; i ≤ iw2; i++)
        printf("%e, ", lap_p_b[i]);
    printf("\n");

    return;
}

/*
LU decomposition used for N-R method
*/
void ludcmp(double a[][MAXI], int r, int indx[], double *d)
{
    int i, imax, j, k;
    double big, dum, sum, temp;
    double vv[MAXI];

    *d = 1.0;
    for (i = 0; i < r; i++) {
        big = 0.0;
        for (j = 0; j < r; j++)
            if ((temp = fabs(a[i][j])) > big)
                big = temp;
        if (big == 0.0)
            ERROR_EXIT("Singular matrix in routine ludcmp");
        vv[i] = 1.0 / big;
    }
    for (j = 0; j < r; j++) {
        for (i = 0; i < j; i++) {
            sum = a[i][j];
            for (k = 0; k < i; k++)
```

```
            sum -= a[i][k] * a[k][j];
        a[i][j] = sum;
    }
    big = 0.0;
    for (i = j; i < r; i++) {
        sum = a[i][j];
        for (k = 0; k < j; k++)
            sum -= a[i][k] * a[k][j];
        a[i][j] = sum;
        if ((dum = vv[i] * fabs(sum)) >= big) {
            big = dum;
            imax = i;
        }
    }
    if (j != imax) {
        for (k = 0; k < r; k++) {
            dum = a[imax][k];
            a[imax][k] = a[j][k];
            a[j][k] = dum;
        }
        *d = -(*d);
        vv[imax] = vv[j];
    }
    indx[j] = imax + 1;
    if (a[j][j] == 0.0)
        a[j][j] = TINY;
    if (j != r - 1) {
        dum = 1.0 / (a[j][j]);
        for (i = j + 1; i < r; i++)
            a[i][j] *= dum;
    }
  }
}

/*
forward and backward substitution for LU decomposition
*/
void lubksb(double a[][MAXI], int r, int indx[], double b[])
{
    int i, ii = 0, ip, j;
    double sum;

    for (i = 0; i < r; i++) {
        ip = indx[i] - 1;
        sum = b[ip];
        b[ip] = b[i];
        if (ii)
            for (j = ii - 1; j < i; j++)
                sum -= a[i][j] * b[j];
        else if (sum)
            ii = i + 1;
        b[i] = sum;
    }
    for (i = r - 1; i >= 0; i--) {
        sum = b[i];
        for (j = i + 1; j < r; j++)
            sum -= a[i][j] * b[j];
        b[i] = sum / a[i][i];
    }
}

/*
computation of Jacobian for N-R method
*/
void fdjac(int r, double x[], double fvec[], double df[][MAXI],
           void (*vecfunc)(double [], int, double []))
{
    int i, j;
    double h, temp, f[MAXI];
```

```
for (j = 0; j < r; j++) {
    temp = x[j];
    h = EPS * fabs(temp);
    if (h == 0.0)
        h = EPS;
    x[j] = temp + h;
    h = x[j] - temp;
    (*vecfunc)(x, r, f);
    x[j] = temp;
    for (i = 0; i < r; i++)
        df[i][j] = (f[i] - fvec[i]) / h;
    }
}

/*
globally convergent method
*/
void newt(double x[], int rank, int *check,
          void (*vecfunc)(double [], int, double []))
{
    int i, its, j, indx[MAXI];
    double d, den, f, fold, stpmax, sum, temp, test;
    double fjac[MAXI][MAXI], g[MAXI], p[MAXI], xold[MAXI];

    rr = rank;
    nrfuncv = vecfunc;
    f = fmin(x);
    test = 0.0;
    for (i = 0; i < rank; i++)
        if (fabs(fvec[i]) > test)
            test = fabs(fvec[i]);
    if (test < 0.01 * TOLF) {
        *check = 0;
        return;
    }
    for (sum = 0.0, i = 0; i < rank; i++)
        sum += SQR(x[i]);
    stpmax = STPMX * DMAX(sqrt(sum), (double)rank);
    for (its = 0; its < MAXITS; its++) {
        fdjac(rank, x, fvec, fjac, vecfunc);
        for (i = 0; i < rank; i++) {
            for (sum = 0.0, j = 0; j < rank; j++)
                sum += fjac[j][i] * fvec[j];
            g[i] = sum;
        }
        for (i = 0; i < rank; i++)
            xold[i] = x[i];
        fold = f;
        for (i = 0; i < rank; i++)
            p[i] = - fvec[i];
        ludcmp(fjac, rank, indx, &d);
        lubksb(fjac, rank, indx, p);
        lnsrch(rank, xold, fold, g, p, x, &f, stpmax,
            check, fmin);
        test = 0.0;
        for (i = 0; i < rank; i++)
            if (fabs(fvec[i]) > test)
                test = fabs(fvec[i]);
        if (test < TOLF) {
            *check = 0;
            return;
        }
        if (*check) {
            test = 0.0;
            den = DMAX(f, 0.5 * rank);
            for (i = 0; i < rank; i++) {
                temp = fabs(g[i]) * DMAX(fabs(x[i]), 1.0)
                    / den;
```

```
            if (temp > test)
                test = temp;
        }
        *check = (test < TOLMIN ? 0 : 1);
        return;
    }
    test = 0.0;
    for (i = 0; i < rank; i++) {
        temp = (fabs(x[i] - xold[i])) / DMAX(fabs(x[i]), 1.0);
        if (temp > test)
            test = temp;
    }
        if (test < TOLX)
            return;
    }
    printf("MAXITS exceeded in newt\n");
}

/*
line searches and backtracking for global convergent method
*/
void lnsrch(int rank, double xold[], double fold, double g[],
            double p[], double x[], double *f, double stpmax,
            int *check, double (*func)(double []))
{
    int i;
    double a, alam, alam2, alamin, b, disc, f2, rhs1, rhs2;
    double slope, sum, temp, test, tmplam;

    *check = 0;
    for (sum = 0.0, i = 0; i < rank; i++)
        sum += p[i] * p[i];
    sum = sqrt(sum);
if (sum > stpmax)
    for (i = 0; i < rank; i++)
        p[i] *= stpmax / sum;
    for (slope = 0.0, i = 0; i < rank; i++)
        slope += g[i] * p[i];
    if (slope ≥ 0.0)
        ERROR_EXIT("Roundoff probrem in lnsrch");
    test = 0.0;
    for (i = 0; i < rank; i++) {
        temp = fabs(p[i]) / DMAX(fabs(xold[i]), 1.0);
        if (temp > test)
            test = temp;
    }
    alamin = TOLX / test;
    alam = 1.0;
    for (;;) {
        for (i = 0; i < rank; i++)
            x[i] = xold[i] + alam * p[i];
        *f = (*func)(x);
        if (alam < alamin) {
            for (i = 0; i < rank; i++)
                x[i] = xold[i];
            *check = 1;
            return;
} else if (*f ≤ fold + ALF * alam * slope)
    return;
else {
    if (alam == 1.0)
        tmplam = - slope / (2.0 * (*f - fold - slope));
    else {
        rhs1 = *f - fold - alam * slope;
        rhs2 = f2 - fold - alam2 * slope;
        a = (rhs1 / (alam * alam) - rhs2 / (alam2 * alam2))
            / (alam - alam2);
        b = (- alam2 * rhs1 / (alam * alam) + alam * rhs2
            / (alam2 * alam2)) / (alam - alam2);
```

```
if (a == 0.0)
    tmplam = - slope / (2.0 * b);
else {
    disc = b * b - 3.0 * a * slope;
    if (disc < 0.0)
        tmplam = 0.5 * alam;
    else if (b <= 0.0)
        tmplam = (- b + sqrt(disc)) / (3.0 * a);
    else
        tmplam = - slope / (b + sqrt(disc));
}
if (tmplam > 0.5 * alam)
                    tmplam = 0.5 * alam;
            }
        }
        alam2 = alam;
        f2 = *f;
        alam = DMAX(tmplam, 0.1 * alam);
    }
}

/*
function needed for global convergent method
*/
double fmin(double x[])
{
    int i;
    double sum;

    (*nrfuncv)(x, rr, fvec);
    for (sum = 0.0, i = 0; i < rr; i++)
        sum += SQR(fvec[i]);

    return 0.5 * sum;
}

/*
inverse Fourier transform
*/
void inv_fourier(double D[])
{
    int i, j, inv_fourier_count, flag_overflow;
    double l, dl, dl_p1[MAXI], dl_p2[MAXI], dlap_p_b[MAXI];
    double xr, xm, dx;
    static double x4[] = {0.339981043584856, 0.861136311594053};
    static double w4[] = {0.652145154861630, 0.347854845137448};
    static double x10[] = {0.148874338981631,
        0.433395394129247, 0.679409568299024,
        0.865063366688985, 0.973906528517172};
    static double w10[] = {0.295524224714753,
        0.269266719309992, 0.219086362515287,
        0.149451349150581, 0.066671344308684};

    /*** initialize ***/
for (i = iw1; i <= iw2; i++)
    lap_p_b[i] = 0.0;
dl = DL;

/*** loop for inverse Laplace transform ***/
for (inv_fourier_count = 0, l = 0.0;;
    inv_fourier_count++, l += dl) {

    /*** error if too many iterations ***/
    if (inv_fourier_count > INVF_MAXITR)
        ERROR_EXIT("Too many iterations in inv_fourier!");

    /*** initialize block pressures ***/
    for (i = iw1; i <= iw2; i++)
        dlap_p_b[i] = 0.0;
```

```
/*** 4 point Gauss-Legendre integration ***/
xm = 0.5 * (1 + dl + 1);
xr = 0.5 * dl;
for (j = 0; j < 2; j++) {
    dx = xr * x4[j];
    if ((flag_overflow = calc_lap_p(xm + dx, dl_p1, D))
            == -1)
        goto exit_loop;
    if ((flag_overflow = calc_lap_p(xm - dx, dl_p2, D))
            == -1)
        goto exit_loop;
    for (i = iw1; i ≤ iw2; i++)
        dlap_p_b[i] += w4[j] * (dl_p1[i] + dl_p2[i]);
}

for (i = iw1; i ≤ iw2; i++)
    lap_p_b[i] += dlap_p_b[i] * xr;

/*** convergence check ***/
for (i = iw1; i ≤ iw2; i++)
    if (fabs(dlap_p_b[i] * xr / lap_p_b[i]) > INVF_EPS)
        goto next_loop;

    exit_loop:

        /*** the case exp() is over MAX_DBL ***/
        if (flag_overflow == -1)
            l -= dl;

        /*** 10 point Gauss-Legendre integration ***/
        xm = 0.5 * (1 / sqrt(ave_dzwd) + 1);
        xr = 0.5 * (1 / sqrt(ave_dzwd) - 1);
        for (i = iw1; i ≤ iw2; i++)
            dlap_p_b[i] = 0.0;

        for (j = 0; j < 5; j++) {
            dx = xr * x10[j];
            lb_calc_lap_p(xm + dx, dl_p1, D);
            lb_calc_lap_p(xm - dx, dl_p2, D);
            for (i = iw1; i ≤ iw2; i++)
                dlap_p_b[i] += w10[j]
                    * (dl_p1[i] + dl_p2[i]);
        }

        for (i = iw1; i ≤ iw2; i++)
            lap_p_b[i] += dlap_p_b[i] * xr;

        /*** exit from the loop ***/
        break;

next_loop:

        /*** Fourier variable increment ***/
        dl = DL * exp(CONST_F / pow(ave_dzwd, POW_INDEX)
                    * inv_fourier_count);
    }
}

/*
integrand for inverse Fourier transform
*/
int calc_lap_p(double l, double dl_p[], double D[])
{
    register int i;
    register int j;
    double arg, expn[MAXJ][MAXI], sol[2*MAXI*MAXJ];
    double C[MAXI][MAXJ];
```

```
    /*** diagonal part of the tridiagonal matrix ***/
    for (j = 0; j < n; j++)
        for (i = 0; i < m; i++)
            nu[i][j] = alpha[i][j] * l * l + rho[i][j];

    /*** calculate particular solutions ***/
    calc_particular_sol(C, D);

    /*** calculate eigenvalues and eigenvectors ***/
    eigenvalues();

    /*** store exponential terms ***/
    for (j = 0; j < n; j++)
        for (i = 0; i < m; i++)
            if ((arg = sigma[j][i] * dzd[j]) > log(DBL_MAX))
                return -1;
            else
                expn[j][i] = exp(arg);

    /*** calculate constants A, B ***/
    if (const_ab(C, sol, expn) == -1)
        return -1;

    /*** calculate pressure increments in Laplacian space ***/
    calc_dlap_p(sol, C, expn, dl_p);

    return 1;
}

/*
integrand for inverse Fourier transform, lb version
*/
void lb_calc_lap_p(double l, double dl_p[], double D[])
{
    register int i;
    register int j;
    double C[MAXI][MAXJ];
    lb lb_expn[MAXJ][MAXI];
    lb lb_sol[2*MAXI*MAXJ];

    /*** accuracy of multiple precision calculation ***/
    lb_digits(LB_ACC);

    /*** diagonal part of the tridiagonal matrix ***/
    for (j = 0; j < n; j++)
        for (i = 0; i < m; i++)
            nu[i][j] = alpha[i][j] * l * l + rho[i][j];

    /*** calculate particular solutions ***/
    calc_particular_sol(C, D);

    /*** calculate eigenvalues and eigenvectors ***/
    eigenvalues();

    /*** store exponential terms ***/
    for (j = 0; j < n; j++)
        for (i = 0; i < m; i++)
            lb_expn[j][i] = exp((lb)(sigma[j][i] * dzd[j]));

    /*** calculate constants A, B ***/
    lb_const_ab(C, lb_sol, lb_expn);

    /*** calculate pressure increments in Laplacian space ***/
    lb_calc_dlap_p(lb_sol, C, lb_expn, dl_p);
}

/*
set time steps in Laplacian domain
*/
```

```
void set_lap_t(int real_count, int *l_count, double t[],
               double lap_t[])
{
    int i, j;

    /*** construct time steps in Laplacian domain ***/
    *l_count = 0;
    for (j = 0; j < (NSTEH / 2); j++) {
        for (i = 0; i < real_count; i++) {
            lap_t[*l_count] = (2.0 * (double)j + 1.0)
                * log(2.0) / t[i];
            (*l_count)++;
            /*** exit if a Laplace time step exceeds maximum ***/
            if (*l_count > L_MAX_TSTEP)
                ERROR_EXIT("increase L_MAX_TSTEP!");
        }
        lap_t[*l_count] = 2.0 * ((double)j + 1.0)
            * log(2.0) / t[0];
        (*l_count)++;
    }

    /*** sort time steps ***/
    piksrt(*l_count, lap_t);
}

/*
unit change, Darcy units to field units
*/
void unit_conv_darcy_to_field(int real_count, double t[],
                              double p[], double dp[])
{
    int i;

    for (i = 0; i < real_count; i++) {
        t[i] *= 1.0 / 3600.0;
        p[i] *= 14.696;
        dp[i] *= 14.696;
    }
}

/*
unit change, dimensionless units to Darcy units
*/
void dimensional_var(int real_count, double t[], double p[],
                     double dp[], double h, double q, double mu,
                     double bo, double ave_k, double ave_phic)
{
    int i;
    double dummy;

    dummy = q * mu * bo / 2.0 / PI / ave_k / h;
    for (i = 0; i < real_count; i++) {
        t[i] *= mu * ave_phic * lw * lw / 4.0 / ave_k;
        p[i] *= dummy;
        dp[i] *= dummy;
    }
}

/*
Stehfest's algorithm
*/
void stehfest(int laplace_count, int real_count, double lap_p[],
              double lap_t[], double t[], double sol[])
{
    int i, j, k;
    double v[NSTEH], dummy, s, sum;
    double factrl(int r);
    int lap_table(int laplace_count, double lap_t[], double s);
```

```
/*** calculate weights ***/
for (i = 1; i ≤ NSTEH; i++) {
    if (i > NSTEH / 2)
        j = NSTEH / 2;
    else
        j = i;
    sum = 0.0;
    for (k = (i + 1) / 2; k ≤ j; k++) {
        dummy = pow((double) k, (double) (NSTEH / 2))
            * factrl(2 * k) / factrl(NSTEH / 2 - k)
            / factrl(k) / factrl(k - 1) / factrl(i - k)
            / factrl(2 * k - i);
        sum += dummy;
    }
    if (((NSTEH / 2 + i) % 2) == 1)
        v[i-1] = - sum;
    else
        v[i-1] = sum;
}

/*** inverse Laplace transform ***/
for (j = 0; j < real_count; j++) {

    sum = 0.0;

    for (i = 1; i ≤ NSTEH; i++) {

        /*** Laplacian time ***/
        s = (double)i * log(2.0) / t[j];

        /*** pick up a correct index ***/
        if ((k = lap_table(laplace_count, lap_t, s)) < 0)
            ERROR_EXIT("Error in lap_table!");

        sum += v[i-1] * lap_p[k];
    }

    /*** values in real domain ***/
    sol[j] = log(2.0) / t[j] * sum;
  }
}

/*
lookup table and find a correct row
*/
int lap_table(int laplace_count, double lap_t[], double s)
{
    int i, index;

    for (i = 0; i < laplace_count; i++)
        if (fabs(lap_t[i] - s) / s < D_EPS) {
            index = i;
            break;
        } else
            index = - 1;

    return index;
}

/*
calculate factorial of r, r!
*/
double factrl(int r)
{
    double x = r;

    if (r < 0) {
        printf("r < 0!\n");
```

```
    exit(0);
}

if (r == 0)
    return 1.0;
else {
    r--;
    while (r > 0) {
        x *= (double) r;
        r--;
    }
    return (double) x;
}
}

/*
calculate pressure at the wellbore
*/
void calc_dlap_p(double sol[], double C[][MAXJ],
                 double expn[][MAXI], double dl_p[])
{
    register int k;
    register int j;
    register int i;

    /*** averaging in z direction ***/
    for (i = iw1; i ≤ iw2; i++) {
        dl_p[i] = 0.0;
        for (j = jw1[i]; j ≤ jw2[i]; j++) {
            for (k = 0; k < m; k++)
                dl_p[i] += (sol[k+offset_A[j]]
                    * (expn[j][k] - 1.0)
                    - sol[k+offset_B[j]]
                    * (1.0 / expn[j][k] - 1.0))
                    * E[i][j][k] / sigma[j][k];
            dl_p[i] += C[i][j] * dzd[j];
        }
        dl_p[i] *= 2.0 / PI / dzwd[i];
    }
}

/*
calculate pressure at the wellbore, lb version
*/
void lb_calc_dlap_p(lb sol[], double C[][MAXJ], lb expn[][MAXI],
                    double dl_p[])
{
    register int k;
    register int j;
    register int i;

    /*** averaging in z direction ***/
    for (i = iw1; i ≤ iw2; i++) {
        dl_p[i] = 0.0;
        for (j = jw1[i]; j ≤ jw2[i]; j++) {
            for (k = 0; k < m; k++)
                dl_p[i] += (double)((sol[k+offset_A[j]]
                    * (expn[j][k] - 1.0)
                    - sol[k+offset_B[j]]
                    * (1.0 / expn[j][k] - 1.0))
                    * E[i][j][k] / sigma[j][k]);
            dl_p[i] += C[i][j] * dzd[j];
        }
        dl_p[i] *= 2.0 / PI / dzwd[i];
    }
}

/*
```

```
calculate constants A, B
*/
int const_ab(double C[][MAXJ], double sol[], double expn[][MAXI])
{
    int p, q;
    register int k;
    register int i;
    register int j;
    static double a[2*MAXI*MAXJ][2*MAXI*MAXJ];
    double dum;

    /*** lower noflow boundary condition ***/
for (i = 0; i < m; i++) {

    for (k = 0; k < m; k++) {
        a[i][k] = E[i][0][k] * sigma[0][k];
        a[i][k+offset_B[0]] = - a[i][k];
    }

    for (k = offset_A[1]; k < offset_A[n]; k++)
        a[i][k] = 0.0;

    sol[i] = 0.0;
}

/*** interblock continuity ***/
for (j = 0; j < n - 1; j++)
    for (i = 0; i < m; i++) {

        /*** pressure continuity ***/
        p = m + 2 * (m * j + i);
        q = p + 1;

for (k = 0; k < offset_A[j]; k++)
    a[p][k] = 0.0;

for (k = 0; k < m; k++) {
    a[p][k+offset_A[j]] = E[i][j][k] * expn[j][k];
    a[p][k+offset_B[j]] = E[i][j][k] / expn[j][k];
    a[p][k+offset_A[j+1]] = a[p][k+offset_B[j+1]]
        = - E[i][j+1][k];
}

for (k = offset_A[j+2]; k < offset_A[n]; k++)
    a[p][k] = 0.0;

/*** sink ***/
if ((j ≥ minjw) && (j < maxjw))
    sol[p] = C[i][j+1] - C[i][j];
else if (j == minjw - 1)
    sol[p] = C[i][j+1];
else if (j == maxjw)
    sol[p] = - C[i][j];
else
    sol[p] = 0.0;

/*** flow rate continuity ***/
for (k = 0; k < offset_A[j]; k++)
    a[q][k] = 0.0;

for (k = 0; k < m; k++) {
    a[q][k+offset_A[j]] = a[p][k+offset_A[j]]
        * sigma[j][k];
    a[q][k+offset_B[j]] = - a[p][k+offset_B[j]]
        * sigma[j][k];
    a[q][k+offset_A[j+1]] = a[p][k+offset_A[j+1]]
        * zeta[i][j+1] * sigma[j+1][k];
    a[q][k+offset_B[j+1]] = - a[q][k+offset_A[j+1]];
}
```

```
        for (k = offset_A[j+2]; k < offset_A[n]; k++)
            a[q][k] = 0.0;

        sol[q] = 0.0;
    }

/*** upper noflow boundary condition ***/
for (i = 0; i < m; i++) {
    p = m * (2 * n - 1) + i;

    for (k = 0; k < offset_A[n-1]; k++)
        a[p][k] = 0.0;

    for (k = 0; k < m; k++) {
        a[p][k+offset_A[n-1]] = E[i][n-1][k]
            * sigma[n-1][k] * expn[n-1][k];
        a[p][k+offset_B[n-1]] = - E[i][n-1][k]
            * sigma[n-1][k] / expn[n-1][k];
    }

    sol[p] = 0.0;
}

/*** overflow and underflow check ***/
for (j = 0; j < 2 * m * n; j++)
    for (i = 0; i < 2 * m * n; i++)
        if (((dum = fabs(a[i][j])) != 0.0)
            && ((dum < DBL_MIN) || (dum > DBL_MAX)))
        return -1;

/*** solve the system of equations ***/
gauss(offset_A[n], a, sol);

    /*** overflow and underflow check ***/
    for (i = 0; i < 2 * m * n; i++)
        if (((dum = fabs(sol[i])) != 0.0)
            && ((dum < DBL_MIN) || (dum > DBL_MAX)))
            return -1;

    return 1;
}

/*
calculate constants A, B, lb version
*/
void lb_const_ab(double C[][MAXJ], lb sol[], lb expn[][MAXI])
{
    int p, q;
    register int k;
    register int i;
    register int j;
    static lb a[2*MAXI*MAXJ][2*MAXI*MAXJ];

    /*** lower noflow boundary condition ***/
for (i = 0; i < m; i++) {

    for (k = 0; k < m; k++) {
        a[i][k] = E[i][0][k] * sigma[0][k];
        a[i][k+offset_B[0]] = - a[i][k];
    }

    for (k = offset_A[1]; k < offset_A[n]; k++)
        a[i][k] = 0.0;

    sol[i] = 0.0;
}

/*** interblock continuity ***/
```

```
for (j = 0; j < n - 1; j++)
    for (i = 0; i < m; i++) {

        /*** pressure continuity ***/
        p = m + 2 * (m * j + i);
        q = p + 1;

        for (k = 0; k < offset_A[j]; k++)
            a[p][k] = 0.0;

        for (k = 0; k < m; k++) {
            a[p][k+offset_A[j]] = E[i][j][k] * expn[j][k];
            a[p][k+offset_B[j]] = E[i][j][k] / expn[j][k];
            a[p][k+offset_A[j+1]] = a[p][k+offset_B[j+1]]
                = - E[i][j+1][k];
        }

        for (k = offset_A[j+2]; k < offset_A[n]; k++)
            a[p][k] = 0.0;

        /*** sink ***/
        if ((j ≥ minjw) && (j < maxjw))
            sol[p] = C[i][j+1] - C[i][j];
        else if (j == minjw - 1)
            sol[p] = C[i][j+1];
        else if (j == maxjw)
            sol[p] = - C[i][j];
        else
            sol[p] = 0.0;

        /*** flow rate continuity ***/
        for (k = 0; k < offset_A[j]; k++)
            a[q][k] = 0.0;

        for (k = 0; k < m; k++) {
            a[q][k+offset_A[j]] = a[p][k+offset_A[j]]
                * sigma[j][k];
            a[q][k+offset_B[j]] = - a[p][k+offset_B[j]]
                * sigma[j][k];
            a[q][k+offset_A[j+1]] = a[p][k+offset_A[j+1]]
                * zeta[i][j+1] * sigma[j+1][k];
            a[q][k+offset_B[j+1]] = - a[q][k+offset_A[j+1]];
        }

        for (k = offset_A[j+2]; k < offset_A[n]; k++)
            a[q][k] = 0.0;

        sol[q] = 0.0;
    }

/*** upper noflow boundary condition ***/
for (i = 0; i < m; i++) {
    p = m * (2 * n - 1) + i;

    for (k = 0; k < offset_A[n-1]; k++)
        a[p][k] = 0.0;

    for (k = 0; k < m; k++) {
        a[p][k+offset_A[n-1]] = E[i][n-1][k]
            * sigma[n-1][k] * expn[n-1][k];
        a[p][k+offset_B[n-1]] = - E[i][n-1][k]
            * sigma[n-1][k] / expn[n-1][k];
    }

    sol[p] = 0.0;
}

/*** solve the system of equations ***/
lb_gauss(offset_A[n], a, sol);
```

```
}

/*
gaussian elimination with partial pivoting
*/
int gauss(int r, double a[][2*MAXI*MAXJ], double b[])
{
    register int j;
    register int i;
    register int k;
    int pv;
    double p;

for (k = 0; k < r; k++) {

    /*** partial pivoting */
    pv = k;
    p = fabs(a[k][k]);

    for (i = k + 1; i < r; i++)
        if (p < fabs(a[i][k])) {
            pv = i;
            p = fabs(a[i][k]);
        }

    if (pv ≠ k) {
        for (j = 0; j < r; j++) {
            p = a[k][j];
            a[k][j] = a[pv][j];
            a[pv][j] = p;
        }

        p = b[k];
        b[k] = b[pv];
        b[pv] = p;
    }

    /*** trianglarization ***/
    if (fabs(a[k][k]) == 0.0)
        ERROR_EXIT("0 pivot in gauss!");

    a[k][k] = 1.0 / a[k][k];

    for (i = k + 1; i < r; i++) {
        a[i][k] *= a[k][k];

        for (j = k + 1; j < r; j++)
            a[i][j] -= a[i][k] * a[k][j];

        b[i] -= a[i][k] * b[k];
    }
}

/*** back substitutuion */
for (i = r - 1; i ≥ 0; i--) {
        for (j = i + 1; j < r; j++)
            b[i] -= a[i][j] * b[j];
        b[i] *= a[i][i];
    }

    return 0;
}

/*
gaussian elimination with partial pivoting, lb version
*/
void lb_gauss(int r, lb a[][2*MAXI*MAXJ], lb b□)
{
    register int j;
```

```
register int i;
register int k;
int pv;
lb p;

for (k = 0; k < r; k++) {

    /*** partial pivoting */
    pv = k;
    p = fabs(a[k][k]);

    for (i = k + 1; i < r; i++)
    if (p < fabs(a[i][k])) {
        pv = i;
        p = fabs(a[i][k]);
    }

if (pv ≠ k) {
    for (j = 0; j < r; j++) {
        p = a[k][j];
        a[k][j] = a[pv][j];
        a[pv][j] = p;
    }

    p = b[k];
    b[k] = b[pv];
    b[pv] = p;
}

/*** trianglarization ***/
if (fabs(a[k][k]) == 0.0)
    ERROR_EXIT("0 pivot in lb_gauss!");

        a[k][k] = 1.0 / a[k][k];

        for (i = k + 1; i < r; i++) {
            a[i][k] *= a[k][k];
            for (j = k + 1; j < r; j++)
                a[i][j] -= a[i][k] * a[k][j];
            b[i] -= a[i][k] * b[k];
        }
    }

    /*** back substitutuion */
    for (i = r - 1; i ≥ 0; i--) {
        for (j = i + 1; j < r; j++)
            b[i] -= a[i][j] * b[j];
        b[i] *= a[i][i];
    }
}

/*
solve symmetric tridiagonal set of equations
*/
void tridag(double a[], double b[], double r[],
            double C[][MAXJ], int j, int rank)
{
    int i;
    double bet;
    double gam[MAXI];

    if (b[0] == 0.0)
        ERROR_EXIT("Error 1 in tridag.");

    C[0][j] = r[0] / (bet = b[0]);

    for (i = 1; i < rank; i++) {
        gam[i] = a[i] / bet;
        bet = b[i] - a[i] * gam[i];
```

```
        if (bet == 0.0)
            ERROR_EXIT("Error 2 in tridag.");
        C[i][j] = (r[i] - a[i] * C[i-1][j]) / bet;
    }

    for (i = (rank - 2); i >= 0; i--)
        C[i][j] -= gam[i+1] * C[i+1][j];

}

/*
define rho for eigenvalue calculation
*/
void define_rho()
{
    register int i;
    register int j;

    for (j = 0; j < n ; j++)
        for (i = 0; i < m; i++)
            rho[i][j] = lambda[i+1][j] + lambda[i][j]
                + omega[i][j] * lap_time;
}

/*
calcurate eigenvalues and eigenvectors
*/
void eigenvalues()
{
    int i, j, k;
    double d[MAXI], e[MAXI], id[MAXI][MAXI], root_beta;

    /*** calculate eigenvalues and eigenvectors ***/
    for (j = 0; j < n; j++) {

        for (i = 0; i < m; i++) {

            /*** set diagonal elements ***/
            d[i] = nu[i][j] / beta[i][j];

            /*** set off-diagonal elements ***/
            e[i+1] = psi[i+1][j];
        }

        /*** initialize an identity matrix ***/
        for (k = 0; k < m; k++)
            for (i = 0; i < m; i++)
                if (k == i)
                    id[k][i] = 1.0;
                else
                    id[k][i] = 0.0;

        /*** QL algorithm ***/
        tqli(d, e, m, id);

        /*** set eigenvalues ***/
        for (k = 0; k < m; k++)
            sigma[j][k] = sqrt(d[k]);

        /*** set eigenvectors ***/
        for (i = 0; i < m; i++) {
            root_beta = sqrt(beta[i][j]);
            for (k = 0; k < m; k++)
                E[i][j][k] = id[i][k] / root_beta;
        }
    }
}

/*
```

```
calculate (a^2+b^2)^0.5
*/
double pythag(double a, double b)
{
    double absa, absb;

    absa = fabs(a);
    absb = fabs(b);
    if (absa > absb)
        return absa * sqrt(1.0 + SQR(absb/absa));
    else
        return (absb == 0.0 ? 0.0 :
            absb * sqrt(1.0 + SQR(absa/absb)));
}

/*
QL algorithm with implicit shifts
*/
void tqli(double d[], double e[], int rk, double z[][MAXI])
{
    int q, l, iter, i, k;
    double s, r, p, g, f, dd, c, b;

    for (i = 1; i < rk; i++)
        e[i - 1] = e[i];
    e[rk-1] = 0.0;

    for (l = 0; l < rk; l++) {
        iter = 0;
        do {
            for (q = l; q < rk - 1; q++) {
                dd = fabs(d[q]) + fabs(d[q+1]);
                if ((double) (fabs(e[q]) + dd) == dd)
                    break;
            }

            if (q ≠ l) {
                if (iter++ == 30)
                    ERROR_EXIT("Too many iterations in tqli");

                g = (d[l+1] - d[l]) / (2.0 * e[l]);
                r = pythag(g, 1.0);
                g = d[q] - d[l] + e[l] / (g + SIGN(r, g));
                s = c = 1.0;
                p = 0.0;

                for (i = q - 1; i ≥ l; i--) {
                    f = s * e[i];
                    b = c * e[i];
                    e[i+1] = (r = pythag(f, g));
                    if (r == 0.0) {
                        d[i+1] -= p;
                        e[q] = 0.0;
                        break;
                    }
                }

                s = f / r;
                c = g / r;
                g = d[i+1] - p;
                r = (d[i] - g) * s + 2.0 * c * b;
                d[i+1] = g + (p = s * r);
                g = c * r - b;

                /*** eigenvectors ***/
                for (k = 0; k < rk; k++) {
                    f = z[k][i+1];
                    z[k][i+1] = s * z[k][i] + c * f;
                    z[k][i] = c * z[k][i] - s * f;
                }
```

```
                }

                if (r == 0.0 && i ≥ 1)
                    continue;
                d[1] −= p;
                e[1] = g;
                e[q] = 0.0;
            }
        } while (q ≠ 1);
    }
}

/*
calculate particular solutions
*/
void calc_particular_sol(double C[][MAXJ], double D[])
{
    register int i;
    register int j;
    double a[MAXI], b[MAXI], r[MAXI];

    /*** set a matrix for each well layer ***/
    for (j = minjw; j ≤ maxjw; j++) {
        for (i = 0; i < m − 1; i++) {
            b[i] = nu[i][j];
            a[i+1] = − lambda[i+1][j];
            if ((j ≥ jw1[i]) && (j ≤ jw2[i]))
                r[i] = − alpha[i][j] * D[i−iw1];
            else
                r[i] = 0.0;
        }
        b[m−1] = nu[m−1][j];
        if ((j ≥ jw1[i]) && (j ≤ jw2[i]))
            r[m−1] = − alpha[m−1][j] * D[m−1−iw1];
        else
            r[m−1] = 0.0;

        /*** solve a symmetric tridiagonal matrix ***/
        tridag(a, b, r, C, j, m);
    }
}

/*
calculate dimensionless variables
*/
void dimless_var(double kx[][MAXJ], double ky[][MAXJ],
                 double kz[][MAXJ], double phi[][MAXJ],
                 double dz[], double dzw[], double h,
                 double mu, double ct[][MAXJ], double rw,
                 double wbs, double *wbsd, double *ave_k,
                 double *ave_phic, double *real_min_t,
                 double *real_max_t, int output_units)
{
    int i, j;
    double area;

    /*** average kx and average phi ct x ***/
    *ave_k = *ave_phic = area = 0.0;
    for (j = 0; j < n; j++)
        for (i = 0; i < m; i++) {
            *ave_k += sqrt(kz[i][j] * ky[i][j]) * dx[i] * dz[j];
            *ave_phic += phi[i][j] * ct[i][j] * dx[i] * dz[j];
            area += dx[i] * dz[j];
        }
    *ave_k *= 1.0 / area;
    *ave_phic *= 1.0 / area;

    /*** dimensionless minimum and maximum real time ***/
    if (output_units == 1) {
```

```
    *real_min_t *= 4.0 * (*ave_k) / mu / (*ave_phic)
        / lw / lw;
    *real_max_t *= 4.0 * (*ave_k) / mu / (*ave_phic)
        / lw / lw;
}

/*** dimensionless variables ***/
for (j = 0; j < n; j++)
    for (i = 0; i < m; i++) {
        alpha[i][j] = ky[i][j] * dx[i] / *ave_k / h;
        beta[i][j] = kz[i][j] * dx[i] / *ave_k / h;
        omega[i][j] = phi[i][j] * ct[i][j] * dx[i]
            / *ave_phic / h;
    }

for (j = 0; j < n; j++) {
    lambda[0][j] = lambda[m][j] = 0.0;
    for (i = 1; i < m; i++)
        lambda[i][j] = lw * lw / 2.0 / *ave_k / h
            / (dx[i-1] / kx[i-1][j] + dx[i] / kx[i][j]);
}

/*** mobility ratios ***/
/* lower and upper boundary */
for (i = 0; i < m; i++)
    zeta[i][0] = zeta[i][n] = 0.0;

/* for the other layers */
for (j = 1; j < n; j++)
    for (i = 0; i < m; i++)
            zeta[i][j] = kz[i][j] / kz[i][j-1];

/*** dimensionless layer thicknesses ***/
    for (j = 0; j < n; j++)
        dzd[j] = 2.0 * dz[j] / lw;

    /*** dimensionless well strip widths ***/
    for (i = iw1; i <= iw2; i++)
        dzwd[i] = 2.0 * dzw[i] / lw;

    /*** wellbore storage ***/
    *wbsd = 2.0 * wbs / PI / mu / *ave_phic / lw / lw;
}

/*
change units, field units to Darcy units
*/
void input_unit_conv(double kx[][MAXJ], double ky[][MAXJ],
                     double kz[][MAXJ], double dz[],
                     double dzw[], double ct[][MAXJ],
                     double *q, double *pin, double *rw,
                     double *wbs, double *h,
                     double *real_min_t, double *real_max_t,
                     int output_units)
{
int i, j;

/*** minimum and maximum time ***/
if (output_units == 1) {
    *real_min_t *= 3600.0;
    *real_max_t *= 3600.0;
}

/*** permeability and total compressibility ***/
for (i = 0; i < m; i++)
    for (j = 0; j < n; j++) {
        kx[i][j] *= 0.001;
        ky[i][j] *= 0.001;
        kz[i][j] *= 0.001;
```

```
            ct[i][j] *= 14.696;
        }

    /*** block size ***/
    for (i = 0; i < m; i++)
        dx[i] *= 30.48;

    for (j = 0; j < n; j++)
        dz[j] *= 30.48;

    for (i = 0; i < m; i++)
        dzw[i] *= 30.48;

        /*** flow rate ***/
        *q *= 1.840131;

        /*** initial pressure ***/
        *pin *= 0.068046;

        /*** wellbore radius ***/
        *rw *= 30.48;

        /*** effective wellbore length ***/
        lw *= 30.48;

        /*** total layer thickness ***/
        *h *= 30.48;

        /*** wellbore storage ***/
        *wbs *= 14.696 * 1.589873e5;
}

/*
print input data
*/
void print_input_data(double kx[][MAXJ], double ky[][MAXJ],
                      double kz[][MAXJ], double phi[][MAXJ],
                      double dz[], double mu, double ct[][MAXJ],
                      double q, double bo, double pin, double rw,
                      double skin[], double wbs, double real_min_t,
                      double real_max_t, int output_units,
                      int calc_type)
{
    int i, j;
    void print_matrix(double mat[][MAXJ]);

    printf("**********************\n");
    printf("      INPUT DATA     \n");
    printf("**********************\n");

    /*** calculation type ***/
    if (calc_type == 0)
        printf("Wellbore pressure calculation.\n\n");
    else if (calc_type == 1)
        printf("Flow rate calculation.\n\n");

/*** units for output ***/
if (output_units == 0)
    printf("Output is in dimensionless units.\n");
else if (output_units == 1)
    printf("Output is in field units\n");

/*** minimum and maximum time ***/
if (output_units == 0) {
    printf("Minimum time = %3.2e\n", real_min_t);
    printf("Maximum time = %3.2e\n", real_max_t);
} else if (output_units == 1) {
    printf("Minimum time, hrs = %3.2e\n", real_min_t);
    printf("Maximum time, hrs = %3.2e\n", real_max_t);
```

```
}

/*** total block size ***/
printf("Total block size = %d x %d\n", m, n);

/*** effective well length ***/
printf("Effective well length, ft = %.1e\n", lw);

/*** block size ***/
printf("Block size, ft =\n");
for (i = 0; i < m; i++)
    printf("  dx[%d]  ", i + 1);
printf("\n");
for (i = 0; i < m; i++)
    printf("%3.2e ", dx[i]);
printf("\n");
for (j = 0; j < n; j++)
    printf("  dz[%d]  ", j + 1);
printf("\n");
for (j = 0; j < n; j++)
    printf("%3.2e ", dz[j]);
printf("\n");

/*** permeability ***/
printf("Permeability kx, md = \n");
print_matrix(kx);

printf("Permeability ky, md = \n");
print_matrix(ky);

printf("Permeability kz, md = \n");
print_matrix(kz);

/*** porosity ***/
printf("Porosity = \n");
print_matrix(phi);

/*** total compressibility ***/
printf("Total compressibility, /psi\n");
print_matrix(ct);

/*** viscosity ***/
printf("Viscosity = %6.4lf cp\n", mu);

/*** flow rate ***/
printf("Flow rate = %4.3e bbl/day\n", q);

/*** formation volume factor ***/
printf("Formation vol. factor = %5.3lf\n", bo);

/*** initial pressure ***/
printf("Initial pressure = %5.0lf psi\n", pin);

/*** wellbore radius ***/
printf("Wellbore radius = %4.3lf ft\n", rw);

    /*** skin factor ***/
    printf("Skin factors =\n");
    for (i = iw1; i ≤ iw2; i++)
        printf("%4.2lf ", skin[i]);
    printf("\n");

    /*** wellbore storage ***/
    printf("Wellbore storage = %4.2e bbl/psi\n", wbs);

    printf("------------------------\n");
}

/*
```

```
print matrix
*/
void print_matrix(double mat[][MAXJ])
{
    int i, j;

    for (j = n - 1; j ≥ 0; j--) {
        for (i = 0; i < m; i++) {

            if ((i ≥ iw1) && (i ≤ iw2) && (j ≥ jw1[i])
                && (j ≤ jw2[i]))

                /*** well block ***/
                printf("%2.1e* ", mat[i][j]);
            else
                printf("%2.1e  ", mat[i][j]);
        }
        printf("\n");
    }
}

/*
read input data
*/
void read_data(double kx[][MAXJ], double ky[][MAXJ], double kz[][MAXJ],
               double phi[][MAXJ], double dz[], double *h, double *mu,
               double ct[][MAXJ], double *q, double *bo, double *pin,
               double *rw, double *zw, double skin[], double *wbs,
               double *real_min_t, double *real_max_t, int *output_units,
               int *r_mode, int *calc_type, int *iw1_q, int *iw2_q)
{
    int i, j, jw;
    char line[MAXLEN];
    FILE *fp;

    /*** open an input file ***/
    printf("Input file name = ");
    scanf("%s", line);
    if((fp = fopen(line, "r")) == 0)
        ERROR_EXIT("Input file open error!");

    /*** notes ***/
    fgets(line, MAXLEN, fp);
    printf("Reading data file...\n%s\n", line);

    /*** reservoir mode ***/
    fscanf(fp, "%d", r_mode);
    if ((*r_mode < 0) || (*r_mode > 3))
        ERROR_EXIT("Select a correct reservoir mode!");

    /*** units for output ***/
    fscanf(fp, "%d", output_units);
    if ((*output_units ≠ 0) && (*output_units ≠ 1))
        ERROR_EXIT("Select correct output units!");

    /*** type of calculation ***/
    fscanf(fp, "%d", calc_type);

    /*** minimum and maximum time in real domain ***/
    fscanf(fp, "%lf %lf", real_min_t, real_max_t);

    /*** read block size ***/
    fscanf(fp, "%d %d", &m, &n);

    /*** well location ***/
    fscanf(fp, "%d %d %d", &iw1, &iw2, &jw);
    iw1--;
    iw2--;
    jw--;
```

```
for (i = iw1; i ≤ iw2; i++)
    jw1[i] = jw2[i] = jw;

/*** layer where flow rate is calculated ***/
fscanf(fp, "%d %d", iw1_q, iw2_q);
(*iw1_q)--;
(*iw2_q)--;
if ((*iw1_q < iw1) || (*iw2_q > iw2) || (*iw1_q > *iw2_q))
    ERROR_EXIT("Wellbore zone should be chosen!");
if ((*r_mode == 1) && (*iw2_q ≥ m / 2))
    ERROR_EXIT("Select correct well segments!");
if ((*r_mode == 3) && (*iw2_q ≥ m / 2))
    ERROR_EXIT("Select correct well segments!");

/*** wellbore radius ***/
fscanf(fp, "%lf", rw);

/*** well position in z direction ***/
fscanf(fp, "%lf", zw);

/*** permeability ***/
/* kx */
for (j = n - 1; j ≥ 0; j--)
    for (i = 0; i < m; i++)
        fscanf(fp, "%lf", &kx[i][j]);

/* ky */
for (j = n - 1; j ≥ 0; j--)
    for (i = 0; i < m; i++)
        fscanf(fp, "%lf", &ky[i][j]);

/* kz */
for (j = n - 1; j ≥ 0; j--)
    for (i = 0; i < m; i++)
        fscanf(fp, "%lf", &kz[i][j]);

/*** x grid size ***/
for (i = 0; i < m; i++)
    fscanf(fp, "%lf", &dx[i]);

/*** layer thickness ***/
for (j = 0; j < n; j++)
    fscanf(fp, "%lf", &dz[j]);

/*** total thickness ***/
for (*h = 0.0, j = 0; j < n; j++)
    *h += dz[j];

/*** porosity ***/
for (j = n - 1; j ≥ 0; j--)
    for (i = 0; i < m; i++)
        fscanf(fp, "%lf", &phi[i][j]);

/*** total compressibility ***/
for (j = n - 1; j ≥ 0; j--)
    for (i = 0; i < m; i++)
        fscanf(fp, "%lf", &ct[i][j]);

/*** viscosity ***/
fscanf(fp, "%lf", mu);

/*** flow rate ***/
fscanf(fp, "%lf", q);
if ((*r_mode == 1) || (*r_mode == 2))
    *q *= 0.5;
else if (*r_mode == 3)
    *q *= 0.25;

/*** formation volume factor ***/
```

```
    fscanf(fp, "%lf", bo);

    /*** initial pressure ***/
    fscanf(fp, "%lf", pin);

    /*** skin factor ***/
    for (i = iw1; i ≤ iw2; i++)
        fscanf(fp, "%lf", &skin[i]);

    /*** wellbore storage ***/
    fscanf(fp, "%lf", wbs);

    /*** close input file ***/
    fclose(fp);
}

/*
set up well blocks
*/
void set_well_block(double kx[][MAXJ], double ky[][MAXJ],
                    double kz[][MAXJ], double phi[][MAXJ],
                    double ct[][MAXJ], double dz[],
                    double dzw[], double rw, double zw,
                    double skin[], int r_mode, int *iw2_q)
{
    int i, j, jw, n_wlayer;
    double zp, zjw[MAXJ];

    /*** well layer ***/
    jw = jw1[iw1];

    /*** effective well length ***/
    lw = 0.0;
    for (i = iw1; i ≤ iw2; i++)
        lw += dx[i];

    /*** layer thickness for well layer ***/
    for (i = iw1; i ≤ iw2; i++)
        dzw[i] = 2.0 * rw * exp(— skin[i])
            * (1.0 + sqrt(kz[i][jw] / ky[i][jw]));

    /*** average dimensionless wellbore width ***/
    for (ave_dzwd = 0.0, i = iw1; i ≤ iw2; i++)
        ave_dzwd += dzw[i] * dx[i];
    ave_dzwd *= 2.0 / lw / lw;

    /*** calculate z coordinate in well blocks ***/
    zjw[0] = 0.0;
    zjw[1] = zw — dzw[iw1] / 2.0;
    if (zjw[1] < 0.0)
        ERROR_EXIT("Too wide strip width downward!");
    zjw[2] = zw + dzw[iw1] / 2.0;
    if (zjw[2] > dz[jw])
        ERROR_EXIT("Too wide strip width upward!");
    zjw[3] = dz[jw];

    /*** number of layers in well blocks ***/
    n_wlayer = 3;

    for (i = iw1 + 1; i ≤ iw2; i++) {

        /* find the same dzw */
        for (j = iw1; j < i; j++)
            if (dzw[i] == dzw[j])
                goto next_i;

        /* set z coordinates */
        zjw[++n_wlayer] = zw — dzw[i] / 2.0;
        zjw[++n_wlayer] = zw + dzw[i] / 2.0;
```

```
    next_i:
        ;
}

/*** sort zjw's in acending order ***/
piksrt(n_wlayer + 1, zjw);

/*** calculate dz ***/
for (j = jw + 1; j < n; j++)
    dz[j+n_wlayer-1] = dz[j];
for (j = 0; j < n_wlayer; j++)
    dz[jw+j] = zjw[j+1] - zjw[j];

/*** modify permeability and porosity matrix ***/
for (j = jw; j < n; j++)
    for (i = 0; i < m; i++) {
        kx[i][j+n_wlayer-1] = kx[i][j];
        ky[i][j+n_wlayer-1] = ky[i][j];
        kz[i][j+n_wlayer-1] = kz[i][j];
        phi[i][j+n_wlayer-1] = phi[i][j];
        ct[i][j+n_wlayer-1] = ct[i][j];
    }
for (j = 0; j < n_wlayer; j++)
    for (i = 0; i < m; i++) {
        kx[i][jw+j] = kx[i][jw];
        ky[i][jw+j] = ky[i][jw];
        kz[i][jw+j] = kz[i][jw];
        phi[i][jw+j] = phi[i][jw];
        ct[i][jw+j] = ct[i][jw];
    }

/*** jw's in well blocks ***/
/* for non-well layers */
for (i = 0; i < m; i++)
    jw1[i] = jw2[i] = -1;

/* well layers */
for (i = iw1; i <= iw2; i++) {
    zp = zw - dzw[i] / 2.0;
    for (j = 1; j <= (n_wlayer - 1) / 2; j++)
        if (fabs(zp - zjw[j]) / zp < D_EPS) {
            jw1[i] = j + jw;
            jw2[i] = n_wlayer - 1 - j + jw;
        }
}

/*** modify vertical block number ***/
n += n_wlayer - 1;

/*** reservior reconstruction ***/
if (r_mode == 1)
    lat_half_res(iw2_q);
else if (r_mode == 2)
    vert_half_res(dz, dzw);
else if (r_mode == 3)
    quarter_res(dz, dzw, iw2_q);
}

/*
set up a lateral half reservoir model
*/
void lat_half_res(int *iw2_q)
{
    /*** modify the lateral grid number ***/
    if (m % 2 == 0)
        m = m / 2;
    else if (m % 2 == 1) {
        m = m / 2 + 1;
```

```
    /*** modify the length of the middle grid ***/
    dx[m-1] *= 0.5;
}

/*** modify the well index ***/
iw2 = m - 1;

/*** modify the well index fow rate calculation ***/
if (*iw2_q > iw2)
    *iw2_q = iw2;
}

/*
set up a vertical half reservoir model
*/
void vert_half_res(double dz[], double dzw[])
{
    int i;
    double dz_old;

    /*** modify the vertical grid number ***/
    n = n / 2 + 1;

    /*** modify the width of the middle grid ***/
    dz_old = dz[n-1];
    dz[n-1] *= 0.5;

    /*** modify the well strip width ***/
    for (i = iw1; i <= iw2; i++)
        dzw[i] = (dzw[i] - dz_old) / 2.0 + dz[n-1];

    /*** modify the well index ***/
    for (i = iw1; i <= iw2; i++)
        jw2[i] = n - 1;
}

/*
set up quarter reservoir model
*/
void quarter_res(double dz[], double dzw[], int *iw2_q)
{
    int i;
    double dz_old;

    /*** modify the lateral grid number ***/
    if (m % 2 == 0)
        m = m / 2;
    else if (m % 2 == 1) {
        m = m / 2 + 1;

        /*** modify the length of the middle grid ***/
        dx[m-1] *= 0.5;
    }

    /*** modify the vertical grid number ***/
    n = n / 2 + 1;

    /*** modify the width of the middle grid ***/
    dz_old = dz[n-1];
    dz[n-1] *= 0.5;

    /*** modify the well strip width ***/
    for (i = iw1; i <= iw2; i++)
        dzw[i] = (dzw[i] - dz_old) / 2.0 + dz[n-1];

    /*** modify the well index ***/
    iw2 = m - 1;
    for (i = iw1; i <= iw2; i++)
```

```
            jw2[i] = n - 1;

        /*** modify the well index for rate calculation ***/
        if (*iw2_q > iw2)
            *iw2_q = iw2;
}

/*
sort array in accending order
*/
void piksrt(int r, double arr[])
{
    int i, j;
    double a;

    for(j = 1; j < r; j++) {
        a = arr[j];
        i = j - 1;
        while ((i >= 0) && (arr[i] > a)) {
            arr[i+1] = arr[i];
            i--;
        }
        arr[i+1] = a;
    }
}

/*
calculate pressure derivative
*/
void calc_deriv(int r_count, double sol[], double t[], double dp[])
{
    int i;
    double p1, p2, error;

    for (i = 1; i < r_count - 1; i++) {
        if (i == 1) {
            ratint(&t[0], &sol[0], 5, t[i] * (1.0 - DT), &p1, &error);
            ratint(&t[0], &sol[0], 5, t[i] * (1.0 + DT), &p2, &error);
        } else if (i == r_count - 2) {
            ratint(&t[r_count-5], &sol[r_count-5], 5,
                t[i] * (1.0 - DT), &p1, &error);
            ratint(&t[r_count-5], &sol[r_count-5], 5,
                t[i] * (1.0 + DT), &p2, &error);
        } else {
            ratint(&t[i-2], &sol[i-2], 5, t[i] * (1.0 - DT), &p1,
                &error);
            ratint(&t[i-2], &sol[i-2], 5, t[i] * (1.0 + DT), &p2,
                &error);
        }

        dp[i] = (p2 - p1) / 2.0 / DT;
    }
}

/*
rational function interpolation
*/
void ratint(double xa[], double ya[], int r, double x, double *y,
            double *dy)
{
    int p, i, ns = 0;
    double w, t, hh, h, dd;
    double c[R_MAX_TSTEP], d[R_MAX_TSTEP];

    hh = fabs(x - xa[0]);

    for (i = 0; i < r; i++) {
        h = fabs(x - xa[i]);
        if (h == 0.0) {
```

```
            *y = ya[i];
            *dy = 0.0;
            return;
        } else if (h < hh) {
            ns = i;
            hh = h;
        }
        c[i] = ya[i];
        d[i] = ya[i] + TINY;
    }
    *y = ya[ns--];
    for (p = 0; p < r - 1; p++) {
        for (i = 0; i < r - p - 1; i++) {
            w = c[i+1] - d[i];
            h = xa[i+p+1] - x;
            t = (xa[i] - x) * d[i] / h;
            dd = t - c[i+1];
            if (dd == 0.0)
                ERROR_EXIT("Error in routine ratint.");
            dd = w / dd;
            d[i] = c[i+1] * dd;
            c[i] = t * dd;
        }
        *y += (*dy = (2 * ns < (r - p - 1) ? c[ns+1] : d[ns--]));
    }
}
```

REFERENCES

[1] Abbaszadeh, A. D. and Cinco-Ley, H. Pressure transient behavior in a reservoir with a finite-conductivity fault, paper SPE 24704. In *Proc., SPE Annual Technical Conference and Exhibition*, Washington, DC., Oct. 4–7 1992.

[2] Abbaszadeh, M. and Hegeman, P. S. Pressure-transient analysis for a slanted well in a reservoir with vertical pressure support. *SPEFE*, pages 277–284, Sept. 1990.

[3] Abbaszadeh, M. and Kamal, M. M. Automatic type-curve matching for well test analysis. *SPEFE*, pages 567–577, Sept. 1988.

[4] Abramowitz, M. and Stegun, I. A. *Handbook of Mathematical Functions*, volume 55 of *Applied Mathematical Series*. Dover Publications, New York, 1968.

[5] Allen, J. R. L. Fining upward cycles in alluvial succession. *Liverpool, Manchester Geol. Jour.*, 4:229–246, 1965.

[6] Ambastha, A. K., McLeroy, P. G., and Sageev, A. Effects of a partially communicating fault in a composite reservoir on transient pressure testing, paper SPE 16764. In *Proc., SPE Annual Technical Conference and Exhibition*, Dallas, Texas, Sept. 27–30 1987.

[7] Anbarci, K., Grader, A. S., and Ertekin, T. Determination of front locations in a multilayer composite reservoir, paper SPE 19799. In *Proc., SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, Oct. 8–11 1989.

[8] Ayestaran, L. C., Nurmi, R. D., Shehab, G. A. K., and El Sisi, W. S. Well test design and final interpretation improved by integrated well testing and geological

efforts, paper SPE 17945. In *Proc., SPE Middle East Oil Technical Conference and Exhibition*, Bahrain, March 11–14 1989.

[9] Beard, D. C. and Weyl, P. K. Influence of texture on porosity and permeability of unconsolidated sand. *AAPG Bulletin*, 57:349–369, 1973.

[10] Bennet, C. O., Camacjo-V., R. G., Reynolds, A. C., and Raghavan, R. Approximate solutions for fractured wells producing layered reservoirs. *SPEJ*, pages 729–742, Oct. 1985.

[11] Bixel, H. C., Larkin, B. K., and van Poollen, H. K. Effect of linear discontinuities on pressure build-up and drawdown bahavior. *JPT*, pages 885–895, Aug. 1963.

[12] Boggs, S. Jr. *Principles of Sedimentology and Stratigraphy.* Prentice Hall, New Jersey, 1995.

[13] Bourdet, D. Pressure behavior of layered reservoirs with crossflow, paper SPE 13628. In *Proc., SPE California Regional Meeting*, Bakersfield, California, March 27–29 1985.

[14] Bourdet, D., Ayoub, J. A., and Pirard, Y. M. Use of pressure derivative in well-test interpretation. *SPEFE*, pages 293–302, June 1989.

[15] Bourgeois, M. J., Daviau, F. H., and Boutaud de la Combe, J-L. Pressure behavior in finite channel-levee complexes, paper SPE 26461. In *Proc., SPE Annual Technical Conference and Exhibition*, Houston, Texas, Oct. 3–6 1993.

[16] Camacho-V, R., Vásquez-C, M., Roldán-C, J., Samaniego-V, F., and Maciás-C, L. New results on transient well tests analysis considering nonlaminar flow in the reservoir, paper SPE 26180. In *Proc., SPE Gas Technology Symposium*, Calgary, Alberta, June 28–30 1993.

[17] Carslaw, H. S. and Jaeger, J. C. *Conduction of Heat in Solids*, pages 455–465. Oxford University Press, New York, second edition, 1959.

[18] Chen, C. C., Chu, W-C., and Sadighi, S. Pressure transient testing of gas reservoirs with edge-water drive, paper SPE 28381. In *Proc., SPE Annual Technical Conference and Exhibition*, New Orleans, Los Angeles, Sept. 25-28 1994.

[19] Cinco-Ley, H. and Samaniego-V, F. Transient pressure analysis for fractured wells. *JPT*, pages 1749–1766, Sept. 1981.

[20] Clonts, M. D. and Ramey, H. J. Jr. Pressure transient analysis for wells with horizontal drainholes, paper SPE 15116. In *Proc., SPE California Regional Meeting*, Oakland, California, April 2–4 1986.

[21] Collinson, J. D. Alluvial sediments. In H. G. Reading, editor, *Sedimentary Environments: Processes, Facies and Stratigraphy*, pages 49–53. Blackwell Science Ltd, Oxford, third edition, 1996.

[22] Corbett, P. W. M., Mesmari, A., and Stewart, G. A method for using the naturally-occurring negative geoskin in the description of fluvial reservoirs, paper SPE 36882. In *Proc., SPE European Petroleum Conference*, Milan, Italy, Oct. 22–24 1996.

[23] Dake, L. P. *Fundamentals of Reservoir Engineering*, page 150. Elsevier Science Publishers, Amsterdam, 1994. Fourteenth impression.

[24] Daviau, F, Mouronval, G., Bourdarot, G., and Curutchet, P. Pressure analysis for horizontal wells. *SPEFE*, pages 716–724, Dec. 1988.

[25] Davies, D. K., Williams, B. P. J., and Vessell, R. K. Reservoir geometry and internal permeability distribution in fluvial, tight, gas sandstones, travis peak formation, texas. *SPERE*, pages 7–12, Feb. 1993.

[26] Davis, R. A. Jr. *Depositional Systems, A Genetic Approach to Sedimentary Geology*, pages 257–258. Prentice-Hall Inc., New Jersey, 1983.

[27] de Swaan-O, A. Analytical solutions for determinating naturally fractured reservoir properties by well testing. *SPEJ*, pages 117–122, June 1976.

[28] Deutsch, C. V. and Journel, A. G. *GSLIB, Geostatistical Software Library and User's Guide*, pages 156–158. Oxford University Press, New York, second edition, 1998.

[29] Earlougher, R. C. Jr. *Advances in Well Test Analysis*, volume 5 of *Monograph Series*, pages 126–127. Society of Petroleum Engineers of AIME, Dallas, Texas, 1977.

[30] Edinburgh Petroleum Services Ltd. *Advanced Welltest Analysis*, pages 24–29. Edinburgh, UK, Aug. 9–13 1999.

[31] Eggenschwiler, M., Satman, A., Ramey, H. J. Jr., and Cinco-Ley, H. Interpretation of injection well pressure transient data in thermal oil recovery, paper SPE 8908. In *Proc., SPE Annual California Regional Meeting*, Los Angeles, California, April 9–11 1980.

[32] Ehlig-Economides, C. and Economides, M. J. Pressure transient analysis in an elongated linear flow system. *SPEJ*, pages 839–847, Dec. 1985.

[33] Ehlig-Economides, C. A. and Joseph, J. A new test for determination of individual layer properties in a multilayered reservoir. *SPEFE*, pages 261–275, Sept. 1987.

[34] Feitosa, G. S., Chu, L., Thompson, L. G., and Reynolds, A. C. Determination of permeability distribution from well test, paper SPE 26047. In *Proc., SPE Western Regional Meeting*, Anchorage, Alaska, May 26–28 1993.

[35] Fox, M. J. and Stewart, G. Simple characterization of communication between reservoir regions, paper SPE 18360. In *Proc., SPE European Petroleum Conference*, London, UK, Oct. 16-19 1988.

[36] Frick, T. P. and Economides, M. J. Horizontal well damage characterization and removal. *SPEPF*, pages 15–22, 1993.

[37] Fukuda, H. Development of a class for multiple precision arithmetic in C/C++ language. *Review of Administration and Informatics, University of Shizuoka*, 10(2):33–40, 1998. http://kilin.u-shizuoka-ken.ac.jp/softs/softs.htm, in Japanese.

[38] Gao, C-T. Single-phase fluid flow in a stratified porous medium with crossflow. *SPEJ*, pages 97–106, Feb. 1984.

[39] Gao, C-T. and Deans, H. A. Pressure transient and crossflow in a multilayer reservoir - single phase flow, paper SPE 11966. In *Proc., SPE Annual Technical Conference and Exhibition*, San Francisco, Calfornia, Oct. 5–8 1983.

[40] Gao, C-T. and Deans, H. A. Pressure transients and crossflow caused by diffusivities in multilayer reservoirs. *SPEFE*, pages 438–448, June 1988.

[41] Gomes, E. and Ambastha, A. K. An analytical pressure-transient model for multilayered, composite reservoirs with pseudosteady-state formation crossflow, paper SPE 26049. In *Proc., SPE Western Regional Meeting*, Anchorage, Alaska, May 26–28 1993.

[42] Goode, P. A. and Thambynayagam, R. K. M. Pressure drawdown and buildup analysis of horizontal wells in anisotorpic media. *SPEFE*, pages 683–697, Dec. 1987.

[43] Goode, P. A. and Wilkinson, D. J. Inflow performance of partially open horizontal wells. *JPT*, pages 983–987, Aug. 1991.

[44] Gringarten, A. C. Interpretation of tests in fissured and multilayered reservoirs with double-porosity behavior: Theory and practice. *JPT*, pages 549–564, April 1984.

[45] Gringarten, A. C. and Ramey, H. J. Jr. The use of source and green's functions in solving unsteady-flow problems in reservoirs. *SPEJ*, pages 285–296, Oct. 1973.

[46] Gringarten, A. C. and Ramey, H. J. Jr. Unsteady-state pressure distributions created by a well with a single horizontal fracture, partial penetration, or restricted entry. *SPEJ*, pages 413–426, Aug. 1974.

[47] Gringarten, A. C. and Ramey, H. J. Jr. An approximate infinite conductivity solution for a partially penetration line-source well. *SPEJ*, pages 140–148, April 1975.

[48] Gringarten, A. C., Ramey, H. J. Jr., and Raghavan, R. Unsteady-state pressure distributions created by a well with a single infinite-conductivity vertical fracture. *SPEJ*, pages 347–360, Aug. 1974.

[49] Gringarten, A. C., Ramey, H. J. Jr., and Raghavan, R. Unsteady-state pressure distributions created by a well with a single infinite-conductivity vertical fracture. *SPEJ*, pages 347–360, Aug. 1974.

[50] Harris, D. G. Geologic input for enhanced recovery projects, a case study of an illinois basin field, paper 7605. In *Proc., Joint CIM-CSPG Convention on Enhanced Recovery*, Calgary, Alberta, June 7–11 1976.

[51] Harris, D. G. and Hewitt, C. H. Synergism in reservoir management – the geologic perspective. *JPT*, pages 761–770, July 1977.

[52] Hartsock, J. H. and Warren, J. E. The effect of horizontal hydraulic fracturing on well performance. *JPT*, pages 1050–1056, Oct. 1961.

[53] Hatzignatiou, D. G., Ogbe, D. O., Dehghani, K., and Economides, M. J. Interference pressure behavior in multilayered composite reservoirs, paper SPE 16766. In *Proc., SPE Annual Technical Conference and Exhibition*, Dallas, Texas, Sept. 27–30 1987.

[54] Issaka, M. B. and Ambastha, A. K. Drawdown and buildup pressure derivative analysis for horizontal wells, paper SPE 24323. In *Proc., SPE Rocky Mountain Regional Meeting*, Casper, Wyoming, May, 18–21 1992.

[55] Jacquard, P. Etude mathematique du drainage d'un reservoir heterogene. *Revue de l'Institute Francais du Pétrole*, XV(10):1384–1400, 1960.

[56] Katz, M. L. and Tek, M. R. A theoretical study of pressure distribution and fluid flux in bounded stratified porous systems with crossflow. *SPEJ*, pages 68–75, 1962.

[57] Kazemi, H and Seth, M. S. Effect of anisotropy and stratification on pressure transient analysis of wells with restricted flow entry. *JPT*, pages 639–647, May 1969.

[58] Kuchuk, F. and Brigham, W. E. Transient flow in elliptical systems. *SPEJ*, pages 401–410, Dec. 1979.

[59] Kuchuk, F., Karakas, M., and Ayestaran, L. Well testing and analysis techniques for layered reservoirs. *SPEFE*, pages 342–354, Aug. 1986.

[60] Kuchuk, F. J. Applications of convolution and deconvolution to transient well tests. *SPEFE*, pages 375–384, Dec. 1990.

[61] Kuchuk, F. J. Well testing and interpretation for horizontal wells. *JPT*, pages 36–41, Jan. 1995.

[62] Kuchuk, F. J., Goode, P. A., Wilkinson, D. J., and Thambynayagam, R. K. M. Pressure-transient behavior of horizontal wells with and without gas cap or aquifer. *SPEFE*, pages 86–91, March 1991.

[63] Kuchuk, F. J. and Habashy, T. Pressure behavior of horizontal wells in multilayer reservoirs with crossflow. *SPEFE*, pages 55–64, March 1996.

[64] Kuchuk, F. J. and Habashy, T. M. Pressure behavior of laterally composite reseroirs, paper SPE 24678. In *Proc., SPE Annual Technical Conference and Exhibition*, Washington, DC, Oct. 4–7 1992.

[65] Kuchuk, F. J. and Wilkinson, D. J. Transient pressure behavior of commingled reservoirs. *SPEFE*, pages 111–120, March 1991.

[66] Kui-Fu, D and Stewart, G. Analysis of transient-pressure response of horizontal wells in bounded reservoirs. *SPEFE*, pages 32–38, March 1994.

[67] Larsen, L. Similarities and differences in methods currently used to analyze pressure-transient data from layered reservoirs, paper SPE 18122. In *Proc., SPE Annual Technical Conference and Exhibition*, Houston, Texas, Oct. 2–5 1988.

[68] Larsen, L. Boundary effects in pressure-transient data from layered reservoirs, paper SPE 19797. In *Proc., SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, Oct. 8–11 1989.

[69] Larsen, L. Experience with combined analyses of PLT and pressure-transient data from layered reservoirs, paper SPE 27973. In *Proc., the University of Tulsa Centennial Petroleum Engineering Symposium*, Tulsa, Oklahoma, Aug. 29–31 1994.

[70] Larsen, L. Determination of pressure-transient and productivity data for deviated wells in layered reservoirs. *SPEREE*, pages 95–103, Feb. 1999.

[71] Le Blanc, R. J. Distribution and continuity of sandstone reservoirs - part 1. *JPT*, pages 776–792, July 1977.

[72] Le Blanc, R. J. Distribution and continuity of sandstone reservoirs - part 2. *JPT*, pages 793–804, July 1977.

[73] Lefkovits, H. C., Hazebroek, P, Allen, E. E., and Matthews, C. S. A study of the behavior of bounded reservoirs composed of stratified layers. *SPEJ*, pages 43–58, 1961.

[74] Lowe, D. R. Sediment gravity flows: II. depositional models with special reference to the deposits of high-density turbidity currents. *J. sedim. Petrol.*, 52:279–297, 1982.

[75] Lu, P. Horizontal and slanted wells in layered reservoirs with crossflow, June 1997. Master Thesis. Stanford University, Stanford, California.

[76] Massonnat, G. J., Norris, R. J., and Chalmette, J-C. Well test interpretation in geologically complex channelized reservoirs, paper SPE 26464. In *Proc., SPE Annual Technical Conference and Exhibition*, Houston, Texas, Oct. 3–6 1993.

[77] Matthews, C. S. and Russell, D. G. *Presure Buildup and Flow Tests in Wells*, pages 97–101. Monograph Series. SPE of AIME, Dallas, Texas, 1 edition, 1967.

[78] Mei, C. C. *Methematical Analysis in Engineering*, pages 192–193. Cambridge University Press, Cambridge, paperback edition, 1997.

[79] Mesmari, A. K. A. Well test modelling in formations with finite conductivity pseudo-fracture channels, Oct. 1996. Master Thesis. Heriot-Watt University, Edinburgh, UK.

[80] Miall, A. D. A review of the braided-river depositional environment. *Earth Science Rev.*, 13:1–62, 1977.

[81] Mueller, T. D. and Witherspoon, P. A. Pressure interference effects within reservoirs and aquifers. *JPT*, pages 471–474, April 1965.

[82] Muskat, M. *The flow of Homogeneous Fluids Through Porous Media*, page 273. McGraw-Hill Book Co., New York, 1949.

[83] Nagtegaal, P. J. C. Sandstone-framework instability as a function of burial diagenesis. *J. Geol. Soc., London*, 135:101–105, 1978.

[84] Niko, H. Well test interpretation in heterogeneous reservoirs with skin and afterflow: Some new theoretical solutions and general field experience, paper SPE 11964. In *Proc., SPE Annual Technical Conference and Exhibition*, San Francisco, Calfornia, Oct. 5–8 1983.

[85] Odeh, A. S. and Babu, D. K. Transient flow behavior of horizontal wells: Pressure drawdown and buildup analysis. *SPEFE*, pages 7–15, March 1990.

[86] Olarewaju, J. S. and Lee, W. J. An analytical model for composite reservoirs produced at either constant bottomhole pressure or constant rate, paper SPE 16763. In *Proc., SPE Annual Technical Conference and Exhibition*, Dallas, Texas, Sept. 27–30 1987.

[87] Oliver, D. S. The averaging process in permeability estimation from well-test data. *SPEFE*, pages 319–324, Sept. 1990.

[88] Ozkan, E., Raghavan, R., and Joshi, S. D. Horizontal-well pressure analysis. *SPEFE*, pages 567–575, Dec. 1989.

[89] Ozkan, E. and Raghavan, R. S. Some strategies to apply the stehfest algorithm for a tabulated set of numbers. *SPEJ*, pages 363–372, Sept. 1997.

[90] Park, H. and Horn, R. N. Well test analysis of a multilayered reservoir with formation crossflow, paper SPE 19800. In *Proc., SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, Oct. 8–11 1989.

[91] Pinkus, A. and Zafrany, S. *Fourier Series and Integral Transforms*, pages 160–162. Cambridge University Press, Cambridge, UK, 1997.

[92] Polubarinova-Kocina, P. Ya. *Theory of Groundwater Movement*, pages 381–395. Princeton University Press, Princeton, New Jersey, 1962. Translated from Russian by J. M. Roger de Weist.

[93] Potter, P. E. Late Mississippian sandstones of Illinois basin. *Circ. - Illi. State Geol. Surv.*, 340, 1962.

[94] Prats, M. Effect of vertical fractures on reservoir behavior - incompressible fluid case. *SPEJ*, pages 105–117, June 1961.

[95] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge University Press, Cambridge, second edition, 1997.

[96] Prijambodo, R., Raghavan, R., and Reynolds, A. C. Well test analysis for wells producing layered reservoirs with crossflow. *SPEJ*, pages 380–396, June 1985.

[97] Puchyr, P. J. A numerical well test model, paper SPE 21815. In *Proc., SPE Rocky Mountain Regional Meeting and Low-Permeability Reservoirs Symposium*, Denver, Colorado, April 15–17 1991.

[98] Raghavan, R. Behavior of wells completed in multiple producing zones. *SPEFE*, pages 219–230, June 1989.

[99] Rahaman, N. M. A. and Ambastha, A. K. A generalized transient-pressure model for areally-compartmentalized reservoirs, paper SPE 37340. In *Proc., SPE Eastern Regional Meeting*, Columbus, Ohio, Oct. 23–25 1996.

[100] Rosa, A. J. and Carvalho, R. S. A mathematical model for pressure evaluation in an infinite-conductivity horizontal well. In *SPEFE*, pages 559–566, Dec. 1989.

[101] Rosa, A. J. and Horne, R. N. Pressure transient behavior in reservoirs with an internal circular discontinuity, paper SPE 26455. In *Proc., SPE Annual Technical Conference and Exhibition*, Houston, Texas, Oct. 3–6 1993.

[102] Rupke, N. A. Growth of an ancient deep-sea fan. *Jour. Geol.*, 85:725–744, 1977.

[103] Russel, D. G. and Prats, M. Performance of layered reservoirs with crossflow – single-compressible-fluid case. *SPEJ*, pages 53–67, 1962.

[104] Russel, D. G. and Prats, M. The practical aspects of interlayer crossflow. *JPT*, pages 589–594, June 1962.

[105] Ryseth, A., Fjellbirkeland, H, Osmundsen, I. K., Skaålnes, Å, and Zachariassen, E. High-resolution stratigraphy and seismic attribute mapping of a fluvial reservoir: Middle Jurassic Ness Formation, Oseberg Field. *AAPG Bulletin*, 82(9):1627–1651, Sept. 1998.

[106] Sagawa, A., Corbett, P. W. M., and Davies, D. R. Pressure transient analysis of reservoirs with a high permeability lens intersected by the wellbore. *J. Pet. Sci. & Eng.*, 1999. In review.

[107] Satman, A. and Ramey, H. J. Jr. An analytical study of transient flow in systems with radial discontinuities, paper SPE 9399. In *Proc., SPE Annual Technical Conference and Exhibition*, Dallas, Texas, Sept. 21-24 1980.

[108] Selley, R. C. *Ancient Sedimentary Environments and Thier Sub-surface Diagnosis.* Chapman & Hall, London, fourth edition, 1996.

[109] Selley, R. C. *Elements of Petroleum Geology*, pages 256–281. Academic Press, San Diego, California, second edition, 1998.

[110] Sneider, R. M., Richardson, F. H., Paynter, D. D., Eddy, R. E., and Wyant, I. A. Predicting reservoir rock geometry and continuity in Pennsylvanian reservoirs, Elk City field, Oklahoma. *JPT*, pages 851–866, July 1977.

[111] Stehfest, H. Numerical inversion of Laplace transforms. *Communications of the ACM*, 13(1):47–49, 1970.

[112] Stewart, G. and Whaballa, A. E. Pressure behavior of compartmentalized reservoirs, paper SPE 19779. In *Proc., SPE Annual Technical Conference and Exhibition*, San Antonio, Texas, Oct. 8–11 1989.

[113] Stow, D. A. V., Reading, H. G., and Collinson, J. D. Deep seas. In H. G. Reading, editor, *Sedimentary Environments: Processes, Facies and Stratigraphy*, pages 415–418. Blackwell Science Ltd, Oxford, third edition, 1996.

[114] Streltsova, T. D. Pressure drawdown in a well with limited flow entry. *JPT*, pages 1469–1476, Nov. 1979.

[115] Streltsova, T. D. *Well Testing in Heterogeneous Formations*, page 48. John Wiley & Sons, Inc., New York, 1988.

[116] Suzuki, K. and Nanba, T. Horizontal well pressure transient behavior in stratified reservoirs, paper SPE 22732. In *Proc., SPE Annual Technical Conference and Exhibition*, Dallas, Texas, Oct. 6–9 1991.

[117] Tariq, S. M. and Ramey, H. J. Jr. Drawdown behavoir of a well with storage and skin effect communicating with layers of different radii and other characteristics, paper SPE 7453. In *Proc., SPE Annual Technical Conference and Exhibition*, Houston, Texas, Oct. 1–3 1978.

[118] Valkó, P. and Economides, M. J. Transient behavior of finite conductivity horizontal fractures. *SPEJ*, pages 213–222, June 1997.

[119] Walker, R. G. and Cant, D. J. Sandy fluvial systems. In R. G. Walker, editor, *Facies Models: Geoscience Canada Reprint Series*, 1, pages 71–89, 1984.

[120] Warren, J. E. and Root, P. J. The behavior of naturally fractured reservoirs. *SPEJ*, pages 245–255, Sept. 1963.

[121] Wendt, W. A., Sakurai, S., and Nelson, P. H. Permeability prediction from well logs using multiple regression. In L. W. Lake and H. B. Carroll, Jr., editors, *Reservoir Characterization*, pages 181–221. Academic Press, Inc., Orlando, Florida, 1986.

[122] Whyatt, M., Bowen, J. M., and Rhodes, D. N. Nelson successful application of a development geoseismic model in North Sea exploration. *First Break*, 9:265–280, 1991.

[123] Wilkinson, D. J. and Hammond, P. A perturbation method for mixed boundary value problems in pressure transient testing. *Transport in Porous Media*, 5:609–636, 1990.

[124] Yaxley, L, M. Effect of a partially communicating fault on transient pressure behavior. *SPEFE*, pages 590–598, Dec. 1987.

[125] Zheng, S-Y, Corbett, P. W. M., Ryseth, A., and Stewart, G. Uncertainty in well test and core permeability analysis : A case study in fluvial channel reservoirs, Northern North Sea, Norway. *AAPG Bulletin*, 1999. In review.

[126] Zhu, D, Hill, A. D., and Looney, M. D. Evaluation of acid treatments in horizontal wells, paper SPE 56782. In *Proc., SPE Annual Technical Conference and Exhibition*, Houston, Texas, Oct. 3–6 1999.