A Unified Metaheuristic and System-Theoretic Framework for Petroleum Reservoir Management

Emeka Nwankwor [BEng, MSc, MEng]

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor of Philosophy

DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institutions of learning.

DEDICATION

This thesis is dedicated to *Mgbokwu be Okafor Ezeaka* – my great grandmother

Of your seed we were born
Born for greatness, no less than you and the rest of our ancestors

I pay you homage because you refused to let the dreams of your beloved husband die

A dream that he bequeaths his name and leaves a legacy unadulterated

That we may come forth and rise above the chains of ignorance

So fixated to these aspirations that even in the face of hopelessness

You looked faithfully unto Chukwu-okike – the wise God of creation

Refusing to compromise our dignity and self-respect

Today, we acknowledge that every generation is an extension of the previous

Without doubt, I acknowledge that we 'rose' from the dust of our ancestors before us

In remembrance of your life and your sufferings

I proclaim that we are forever indebted to you and indeed the rest of our ancestors

ACKNOWLEDGEMENTS

Firstly, I am profoundly grateful to *Chukwu-abiama* – the God of my ancestors, for the gift of life and for preserving me and my family during the past four 'torturous' years of this PhD. I will now proceed to express my deepest appreciation to my supervisor and Chairman Centre for Applicable Mathematics and Systems Science, Prof. Atulya K. Nagar, for his valuable guidance, encouragement and unflinching support during this academic journey. I am thankful for his patience, understanding and support during some of the most difficult and challenging periods of my life. There is no gainsaying the fact that without his supervision and financial support, this thesis would not have been possible. I am indebted to him more than he can ever think. I would also like to extend my appreciations to Dr. David Reid and Dr. Steven Presland for the roles they played as co-supervisors.

During the course of my studies, I immensely benefited from the academic benevolence of a number of individuals. In this regards, I am particularly grateful to Prof. Jan Dirk Jansen for providing me with his lecture notes; Prof. Knut-Andreas Lie and Dr. Bård Skaflestad of SINTEF for their help and the useful discussions on **MRST®**; Dr. Kusum Deep, Dr. Millie Pant and Dr. Jagdish Bansal for their valuable contributions and suggestions. I would also want to gratefully acknowledge Mr. Satley Lengmang, Mr. Joseph Umeh and Engr. C.A.O Eze. Every one of you is a clear testimony of the age-long saying "the good teacher explains, the superior teacher demonstrates while the great teacher inspires".

I am also indebted to many friends and colleagues at the department research laboratory for providing the perfect, stimulating and fun environment to work in the past four years. In this regards, I can never re-pay Riccardo, Obinna, Rentian and Hakim. My gratitude also goes to Prof. Michael Pegg and Dr. Mumuni Amadu of the Petroleum Engineering Department at Dalhousie University for their assistance and support during the time I spent with them – like a candle, you consumed yourselves to light the way for me.

Finally, I must pay tribute to my family, their love and supports have been crucial. I would like to express my gratitude to my parents – they nurtured, taught and encouraged me; my siblings – they have been very supportive; *Ijeoma* – for providing a bedrock of love and stability; *Chidera* and *Nnabike* – for their uncommon strength and endurance. Forgive me for I know I am guilty of dereliction of duty, I will certainly make up for all the deprivations.

TABLE OF CONTENTS

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BHP	Bottom Hole Pressure
BTU	British Thermal Unit
CAPEX	Capital Expenditure
CF	Cash Flow
CLoReM	Closed-Loop Reservoir Management
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
DCT	Discrete Cosine Transform
DE	Differential Evolution
E&P	Exploration and Production
EKF	Extended Kalman Filter
EnKF	Ensemble Kalman Filters
EOR	Enhanced Oil Recovery
ES	Evolutionary Strategies
FDG	Finite Difference Gradient
FDP	Field Development Planning
GA	Genetic Algorithm
GDM	Gradual Deformation Method
GRG	Generalised Reduced Gradient
HPSDE	Hybrid Particle Swarm Differential Evolution
LTI	Linear Time Invariant
MIMO	Multiple Inputs Multiple Outputs
MINLP	Mixed Integer Nonlinear Problem
MPC	Model Predictive Control
NLP	Nonlinear Programming Problem
NMPC	Nonlinear Model Predictive Control
NN	Neural Networks
NPV	Net Present Value
O&G	Oil and Gas
OECD	Organization for Economic Co-operation and Development
OPEX	Operation Cost
PCA	Principal Component Analysis

PMP	Pontryagin's Maximum Principle
PPM	Probability Perturbation Method
PSO	Particle Swarm Optimization
QP	Quadratic Programming
R&M	Refining and Marketing
REV	Revenue
ROI	Return on Investment
SA	Simulated Annealing
SADC	Southern African Development Community
SPSA	Simultaneous Perturbation Stochastic Approximation
UKF	Unscented Kalman Filters
USEIA	United States Energy Information Agency
VFSA	Very Fast Simulated Annealing

ABSTRACT

With phenomenal rise in world population as well as robust economic growth in China, India and other emerging economies; the global demand for energy continues to grow in monumental proportions. Owing to its wide end-use capabilities, petroleum is without doubt, the world's number one energy resource. The present demand for oil and credible future forecasts – which point to the fact that the demand is expected to increase in the coming decades – make it imperative that the E&P industry must device means to improve the present low recovery factor of hydrocarbon reservoirs. Efficiently tailored model-based optimization, estimation and control techniques within the ambit of a closed-loop reservoir management framework can play a significant role in achieving this objective.

In this thesis, some fundamental reservoir engineering problems such as field development planning, production scheduling and control are formulated into different optimization problems. In this regard, field development optimization identifies the well placements that best maximizes hydrocarbon recovery, while production optimization identifies reservoir well-settings that maximizes total oil recovery or asset value, and finally, the implementation of a predictive controller algorithm which computes corrected well controls that minimizes the difference between actual outputs and simulated (or optimal) reference trajectory. We employ either deterministic or metaheuristic optimization algorithms, such that the choice of algorithm is purely based on the peculiarity of the underlying optimization problem.

Altogether, we present a unified metaheuristic and system-theoretic framework for petroleum reservoir management. The proposed framework is essentially a closed-loop reservoir management approach with four key elements, namely: a new metaheuristic technique for field development optimization, a gradient-based adjoint formulation for well rates control, an effective predictive control strategy for tracking the gradient-based optimal production trajectory and an efficient model-updating (or history matching) – where well production data are used to systematically recalibrate reservoir model parameters in order to minimize the mismatch between actual and simulated measurements.

Central to all of these problems is the use of white-box reservoir models which are employed in the well placement optimization and production settings optimization. However, a simple data-driven black-box model which results from the linearization of an identified nonlinear model is employed in the predictive controller algorithm. The benefits and efficiency of the approach in our work is demonstrated through the maximization of the NPV of waterflooded reservoir models that are subject to production and geological uncertainty. Our procedure provides an improvement in the NPV, and importantly, the predictive control algorithm ensures that this improved NPV are attainable as nearly as possible in practice.

CHAPTER 1

A man who pays respect to the great paves way for his own greatness – Chinua Achebe

INTRODUCTION

In this introductory chapter, detailed background information required to understand the need for this research is presented. The age-long dependence of mankind on diverse energy resources is highlighted, and the relationship between energy utilization capacity and economic prosperity is highlighted with the focus on petroleum. Finally, the motivations, objectives and contributions of this thesis are presented.

1.1 Energy Utilization and Quality of Life

Without fear of contradiction, it can be said with some degree of accuracy that the history of mankind is a story of a continual search of energy and better means of livelihood. Over the centuries, man's quality of life has been measured in terms of increased availability at the point of need, the basic essentials of life – food, clothing and shelter. This goes hand-in-hand with better healthcare services, more effective and efficient transportation, better means of communication and telecommunication, more availability of potable water, better leisure and entertainment services, as well as reliable municipal services. The quest for better living has resulted in the invention and manufacture of a wide variety of tools and equipment; it has led to the birth of an incredible array of devices, gadgets and machines that aim to make life easy and enjoyable. These devices and machines would be nothing but lifeless chunks of matter if some driving impetus were not provided – Eze (2002). That driving impetus is energy. It is available in variety of forms, and can be broadly classified into – renewable (wind, solar, biomass) and non-renewable (natural gas, oil, coal, nuclear[1]) energy.

[1] Although nuclear energy is a low carbon power generation source, its categorization as a renewable energy power source has been the subject of much debate. In this work, we categorize it as non-renewable.

In rural parts of the developing world for example, life is nothing but a simple affair. Most communities are largely without electricity, and the common fuels are dry vegetative material, wood, charcoal, and kerosene. Very few activities are mechanized, and as a result, the standard of living is pretty low. Agriculture is the predominant occupation, and the tools used for this purpose are simple hoes and machetes; thus, the primary source of energy is manual labour. Although animal-drawn farm implements are increasingly available, there is no gainsaying that the amount of arable land that can be cultivated by these sources of energy (on a per capita basis) is minimal. It is accepted that an increased availability of more energy resources (in this case, for mechanization of agriculture) will no doubt bring those part of the world much closer to achieving the all-important goal of self-sufficiency in food production. This is so because productivity is always enhanced upon the substitution of muscular effort with machines. In other words, the mechanization of human activities often leads to tremendous improvements in productivity, economic development and empowerment. It therefore follows by logic that the level of economic empowerment in any given society, as measured in monetary terms by its per capita income, correlates quite well with the per capita energy utilization of that society – Eze (2002).

Available data from Key World Energy Statistics (KWES) show that the developed nations of the world have much higher per capita income as well as higher per capita energy utilization than the less developed nations of sub-Saharan Africa and indeed the Third World. Therefore, one can infer that the living standard and quality of life of any given economy go pari-passu with its per capita energy usage. A good illustration of this point is highlighted if we undertake a historical survey of energy demand and utilization across different geo-political regions of the world. In 1973, the approximately 750 million citizens of the Organization for Economic Co-operation and Development (OECD)[2] bloc (which constitute about 19% of the world's total population at that time) consumed no less than 11.12×10^{16} BTU of the world's total energy supply. This represents over 60% of total global energy (18.5×10^{16} BTU) supply, and a per capita energy utilization of 150×10^{6} BTU. Thirty-five years later, the OECD bloc consumed 14.7×10^{16} BTU – which represents 44% of total global

[2] The Organization for Economic Co-operation and Development (OECD) consist of 24 countries as at 1973. They include: Australia, Austria, Belgium, Canada, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Japan, Luxembourg, Netherland, New Zealand, Norway, Portugal, Spain, Sweden, Switzerland, Turkey, UK and USA. By the year 2008, countries such as Czech, Hungary, South Korea, Mexico, Poland and Slovakia had become member countries of the OECD.

energy supply (33.4×10^{16} BTU), and a per capita energy utilization of 122.5×10^{6} BTU for the 1.2 billion citizens of the bloc.

In 1973, China and her 882 million citizens consumed 1.46×10^{16} BTU – which represents 7.9% of global energy supply, and a per capita energy utilization of 16.57×10^{6} BTU. With a population of 1.33 billion citizens in 2008, China's energy consumption rose to 5.48×10^{16} BTU – i.e. 16.4% of the global energy supply; and by all indications, an energy utilization increase of monumental proportions. In other words, China's energy consumption more than tripled between the periods 1973 and 2008; and within the same time frame, the per capita energy utilization of the average Chinese more than doubled from 16.57×10^{6} BTU to 41.2×10^{6} BTU.

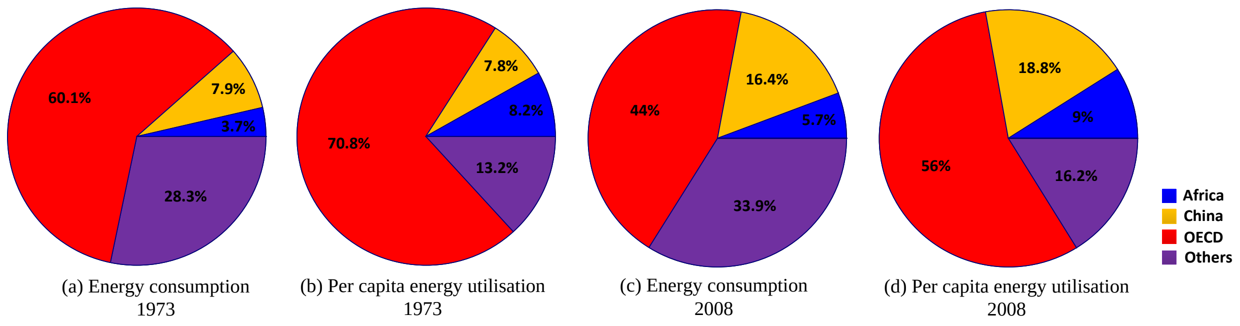


Figure 1.1: Global energy utilization at geo-political and per capita level in 1973 and 2008

Interestingly, this historical survey and comparison get a lot more meaningful if we throw-in the energy consumption and per capita energy utilization of Africa into the mix. With a population of 390 million inhabitants, Africa consumed 0.68×10^{16} BTU or 3.7% of global energy supply in 1973. This translates to a per capita energy utilization of 17.5×10^{6} BTU, which was marginally better (or at least comparable) to the per capita energy utilization of China at that time. However, the 975 million individuals that peopled Africa in 2008, consumed 1.9×10^{16} BTU (or 5.7%) of the entire global energy supply for that year. This represents a per capita energy utilization of 19.62×10^{6} BTU – which is less than half the per capita energy utilization of China, and less than a sixth of the per capita energy utilization of the OECD countries at that time. Figure 1.1 shows a pictorial representation of the global energy utilization at geo-political and per capita levels in 1973 and 2008. If we put the above statistical analyses side-by-side with the tremendous technological and economic growth that China has recorded in the past quarter-of-a-century; one does not struggle to understand why China's energy consumption almost quadrupled in the period between 1973 and 2008. It underscores the point that technological advancement is coterminous with demand for, and usage of, more and more energy. It also reinforces the logic that the level of economic empowerment of any given society correlates quite well with the per capita energy utilization of that society.

1.2 Energy Resources

There are a number of energy resources; and in many instances, these energy resources are often classified into two broad classes as depicted in Figure 1.2. One class is made of all those resources which are derived from natural processes that are replenishable in 'real-time'. This energy class is not subject to any threat of depletion; and is therefore, referred to as *renewable* energy resources – a testimony of the fact that they are constantly replenished, or perhaps in recognition of the fact that these energy resources are in a continual state of flux, Eze (2002). Examples include: solar energy, wood[3], biomass, wind, hydropower[4], etc.

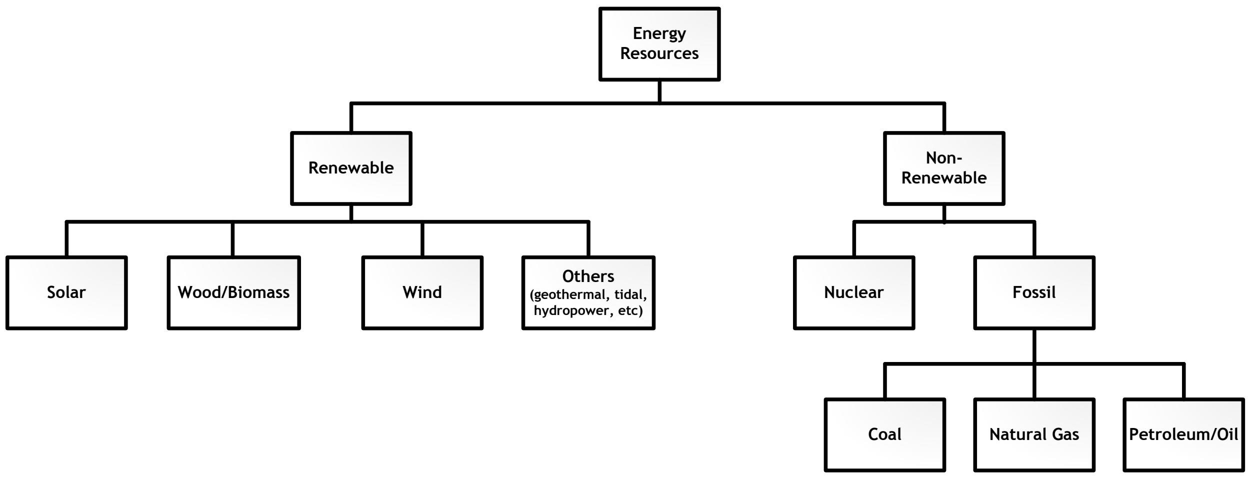


Figure 1.2: Classification of Energy Resources

[3] The membership of wood in the class of renewable energy resources is suspect. When carefully exploited with effective afforestation and re-forestation programmes in place, wood is a renewable energy resource. However, experience in various parts of the world has shown quite clearly that if these conditions are not strictly met, wood would fail to qualify as a renewable energy resource. Desertification would be a natural consequence of an extreme case of an indiscriminate exploitation of this energy resource.

[4] In principle, hydropower would appear at any rate to be a renewable energy resource (so long as it is as a result of the natural hydrologic cycle of evaporation and precipitation). However, the issue of siltation which naturally accompanies the creation of large artificial lakes (by impounding a rapidly-flowing river) is of enormous significance. The siltation process gradually accumulates silts which tend to decrease the effective depth of the man-made lake, and consequently, the suitability of the site in question for hydro-electric power generation. For this reason, hydropower sites must be carefully chosen; otherwise that energy resource would fail to be truly renewable.

The second class include all the energy resources that are found in the earth's crust in the form of material substances that are known to occur in specific or unspecific finite quantities. Though neither the deposits of these materials on a global basis, nor the actual quantities to be found in the deposits or reserves may be completely known; what is known with certainty is the fact that there is only a finite quantity of each of the energy resources in this class. An indiscriminate exploitation poses a risk of complete depletion; therefore, the resources in this class are the energy resource equivalent of 'endangered species'. As it were, they stand the risk of 'extinction', because the rate of their exploitation far out-weighs the rate at which they are replenished by relevant natural processes.

On account of this great discrepancy between the rate of replenishment and exploitation, these energy resources are referred to as *non-renewable* energy resources – a nomenclature which clearly underscores their proneness to depletion. Examples of this category of energy resources include nuclear and fossil fuels such as coal, natural gas and oil. As the title of this thesis suggests, the scope of this work is on non-renewable energy; and the focus is primarily on petroleum or crude oil.

1.2.1 Liquid Petroleum (Crude Oil)

Petroleum, also known as crude oil, is a fossil fuel and liquid mineral whose origin has been ascribed to marine organisms that were deposited in the earth crust many eons ago. These deposited organisms were subsequently transformed (under anaerobic and extreme high temperature and pressure conditions) into the mineral which is at present so valuable to modern technology and to world economy that it is often referred to as *liquid gold*. The word 'petroleum', which literally means 'rock oil', is a combination of two Latin words *petrus* (meaning rock) and *oleum* (meaning oil) – an allusion to the fact that the earliest finds of this energy resource were in the form of seepages from sedimentary rock outcrops.

Besides the well-known conventional crude oil, petroleum also occurs in the form of *tar sands* (oil sands or bituminous sands) and *oil shale*. The former occur in commercial quantities in Venezuela (Orinoco), Canada (Alberta), Russia, the USA and Madagascar to name but a few; and the latter is found in commercial quantities at various locations across the globe. Global reserves of oil in 2012 are estimated at 1600 billion barrels, Oil & Gas

Journal (2011) and USEIA (2011). Table 1.1 shows the distribution of the reserves in different geographical regions.

Geographic Regions	Oil Reserves (10^9 bbl of Oil)	% of Global Reserve
Africa	144	9.0
America (North)	234	14.6
America (Others)	237	14.8
Asia + Eurasia	140	8.7
Middle East	834	52.1
OECD Europe	11	0.7
Total	**1600**	**99.9**

Table 1.1: Geographical distribution of global oil reserves

Over the past half a century, the global economy and indeed mankind have become addicted to oil; and to satisfy this ‘addiction’, this non-renewable resource is exploited at a phenomenal rate. With this neck-breaking rate at which known reserves are depleted, and the enormous challenge of finding new ones, serious concerns have been raised to the effect that global reserves could run out in less than a century. To this end, a lot of research has been dedicated to finding alternative energy sources that would wean the world of its dependence on oil. More importantly, a greater number of researches are firmly focused on finding and developing newer and effective techniques and methodologies that would improve production and enhance recoverability of this essential energy resource.

1.2.2 Natural Gas

Natural gas is a finite energy resource which shares a common geological history with oil. Depending on the prevailing temperature and pressure condition (with respect to the fluid critical temperature and oil bubble point pressure respectively) at which the hydrocarbon maturation or catagenesis occurred; natural gas could exists together with oil (gas-cap oil reservoirs) and can therefore be referred to as *associated gas*.

In situations where the prevailing reservoir temperature is greater than the fluid critical temperature, natural gas occur as *non-associated gas* as evident in various isolated gas fields across the world. The main constituent of this energy resource is methane (CH_4) – which constitutes no less than 80% by volume of natural gas. The remainder is made of varying percentages of ethane (C_2H_6), propylene (C_3H_6), hydrogen (H_2), carbon dioxide (CO_2), carbon monoxide (CO) and other gases. The global reserves for natural gas are estimated at 6,675 trillion cubic feet, Oil & Gas Journal (2010); and the distribution is given in Table 1.2.

Geographic Regions	**Natural Gas Reserves (10^{12}) ft^3**	**% of Global Reserve**
Africa	518	7.7
America (North)	346	5.2
America (South + Central)	269	4.0
Asia + Eurasia + Europe	2856	42.8
Middle East	2686	40.2
Total	**6675**	**99.9**

Table 1.2: Geographical distribution of global natural gas reserves

1.3 Global Demand of Energy Resources

From the foregoing, it has been clearly established that man and energy are inseparable. We highlighted in section 1.1 that global energy consumption almost doubled from 18.5×10^{16} BTU in 1973 to 33.4×10^{16} BTU in 2008. This pattern of increase in global energy demand and consumption is not expected to change anytime in the distant future; in fact, the trend is expected to increase. According to the 2011 International Energy Outlook, the US Energy Information Agency (USEIA) posits that global energy consumption will significantly increase in the next quarter of a century. This unprecedented rise in demand of energy will be as a result of robust economic growth in China and India; as well as increased usage in other rapidly developing economies. Figure 1.3 depicts a summary of the projected energy demand for various energy resources as contained in the USEIA 2011 report.

For reasons bordering on environmental friendliness, and need to implement the United Nation (UN) Kyoto Protocol, renewable energy resources are becoming more and more economically competitive with fossil fuels (coal, natural gas and oil) – as evident in Figure 1.3. However, oil, coal and natural gas will certainly remain the dominant source of energy until 2035, and possibly beyond. The reason behind this reality is not far-fetched. The developmental stage and capacity of most renewable-resource technologies are still in their infancy; and others like wood-fuel (biomass) have low calorific value and thermal efficiency which rules them out as serious alternatives. On its part, nuclear energy is both unpopular and unattractive – no thanks to its potential hazards and the enormous problem associated with radioactive waste disposal. The 1979 Three Mile Island nuclear melt-down, the Chernobyl nuclear disaster of 1986, and the recent Fukushima nuclear accident are all pointers to how dangerous and problematic nuclear energy technology can be.

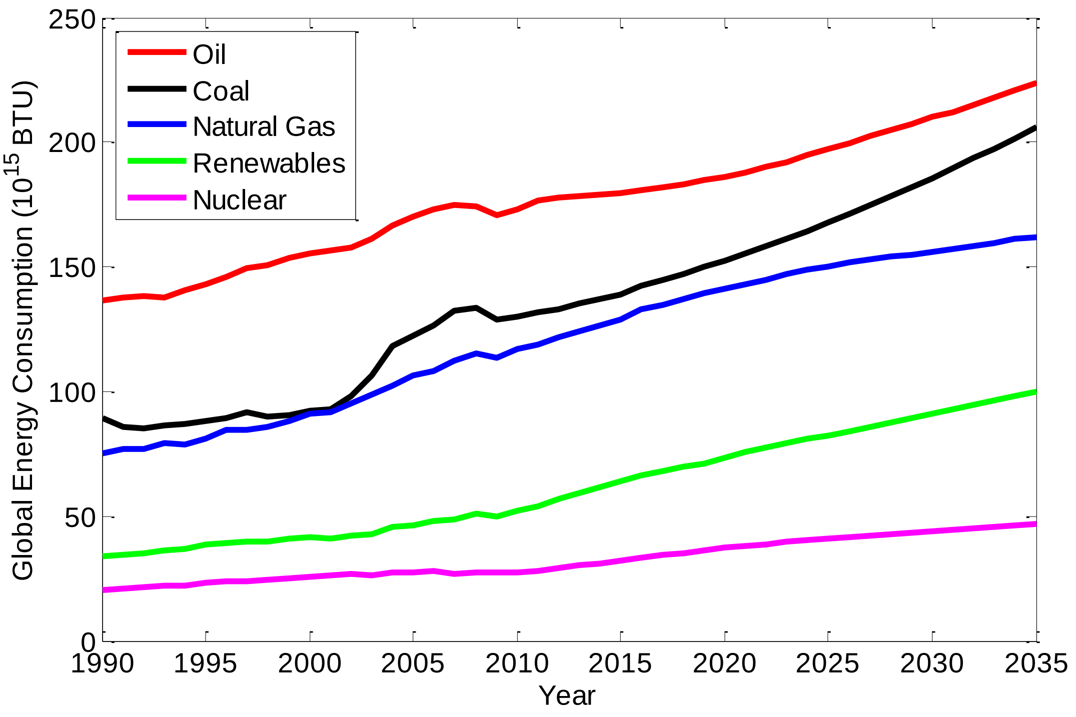


Figure 1.3: Global energy use by energy type, 1990 – 2035, raw data courtesy of USEIA (2011)

In effect, this makes the fossil fuels – oil, natural gas and coal – the most viable or feasible energy resources that can meet man's ever-growing demand for energy. Moreover, with its enormous direct-end-use capacity and on the evidence of Figure 1.3; it is pretty clear that oil is the number one energy resource in the world. Indeed, the global economy is dependent on oil; the demand for this energy commodity has continued to rise in significant proportions.

To meet the projected increase in global oil demand by 2035, the exploration and production (E&P) industry must fashion out ways to increase the present (2012) global output of 87.4 million barrels per day by another 23 million barrels per day. In the light of the fact that the industry is already struggling to meet present day demand, it is easy to see that the challenge of meeting the projected increase in demand is by all ramifications, an onerous task. The problem becomes even more exacerbated if we consider the fact that most of the existing oilfields are already at a mature stage, and the discovery of large new oilfields are becoming fewer and far between. This reality is perfectly captured in the words of the CEOs of two of the world's leading E&P companies (Total SA and Royal Dutch Shell) who posit that the days of easy oil are gone, Voss and Patel (2007).

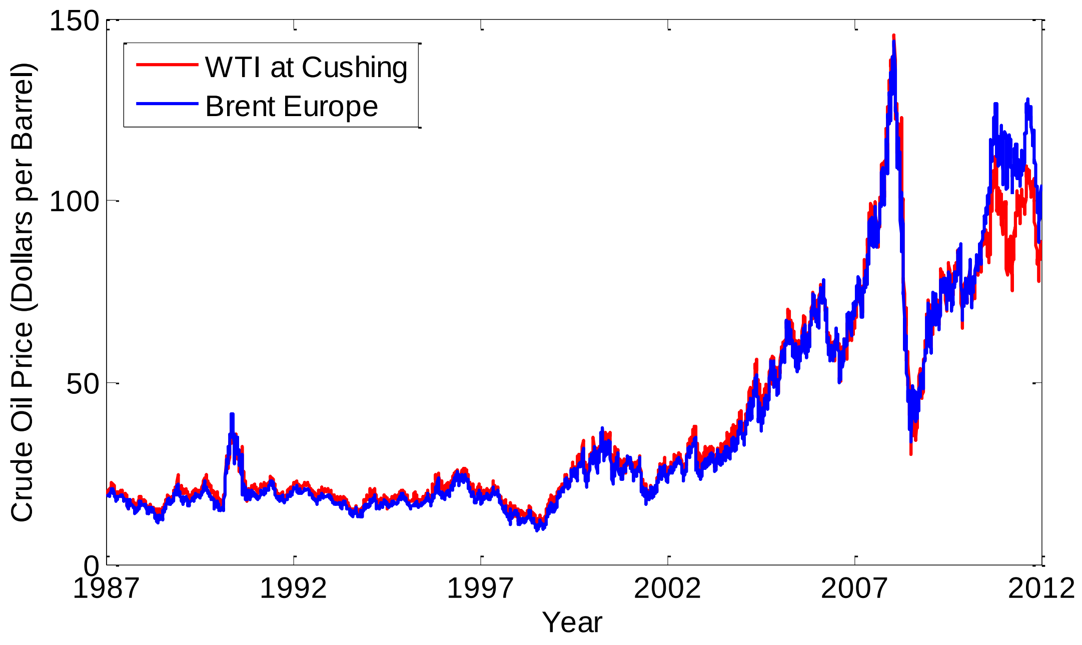


Figure 1.4: Prices of crude oil from May 1987 to July 2012 (raw data courtesy of http://www.eia.gov)

Inevitably, this has led to a disproportionate relationship in the market forces of demand and supply; and therefore, it has resulted in significant increase in the price of crude oil in the global market – as shown in Figure 1.4. It is against this backdrop that we underline the imperativeness to devise means that would help bridge the gap between global demand and supply of oil. In other words, it is essential to develop effective techniques that would improve the recovery factor of existing and new oilfields. A rapidly emerging methodology to achieving this goal is the application of mathematical optimization techniques in order to improve decision-based inputs from the cradle of petroleum reservoirs to its grave.

Particularly promising in this context is the use of intelligent computing, high end reservoir modelling and simulations, estimation, control and optimization techniques. In the oil industry, this technology is known under various names such as smart fields (Shell), intelligent fields (Chevron), e-field (BP), closed-loop reservoir management – Jansen et al. (2005) or real-time reservoir management – Sarma et al. (2006).

1.4 Natural Occurrence of Hydrocarbon

Both oil and natural gas share a common geological history (they are usually referred to as geological cousins); they originate from the remains of pre-historic plants and animals organic materials deposited underneath earth materials at different sites. Over a period of eons, these layers or sediments containing organic materials gradually stacked up from a few centimetres to hundreds and in some cases, thousands of meters. The resultant increase in temperature and pressure, as well as other severe environmental and geological activities transformed them into kerogens. The thermal maturation of these kerogens (catagenesis) yielded oil, gas and water; with the hydrocarbon components (oil and gas) separated from water by gravity. Because oil and gas are less dense than water, they tend to migrate from the source rocks where they are formed; however, these hydrocarbon component forms an accumulation (or a reservoir) if during the course of their migration, they get entrapped under a layer of low-permeable or impermeable rock material, which acts as a seal rock – thereby forming a system referred to as a petroleum trap.

1.5 Hydrocarbon Reservoir Life Cycle

The lifecycle of a hydrocarbon reservoir, from its cradle (exploration) to its grave (abandonment) is in the order of decades. Beginning from the exploration phase, it passes through the appraisal phase, development phase, production phase, and ends in the abandonment phase as depicted in Figure 1.5. It must be noted that this process diagram can be further refined to display sub-activities at deeper levels of each phase, Currie and Jansen (2004). In other words, Figure 1.5 is too simplistic a representation, as the lifecycle is not a simple sequential process devoid of feedback and iterative repetition of activities. Take for example, several cycles of appraisals and re-appraisals (based on production performance or new seismic data), development and re-development (through completion of wells, or in-fill drilling of new ones), and production may take place during the life time of a typical oilfield.

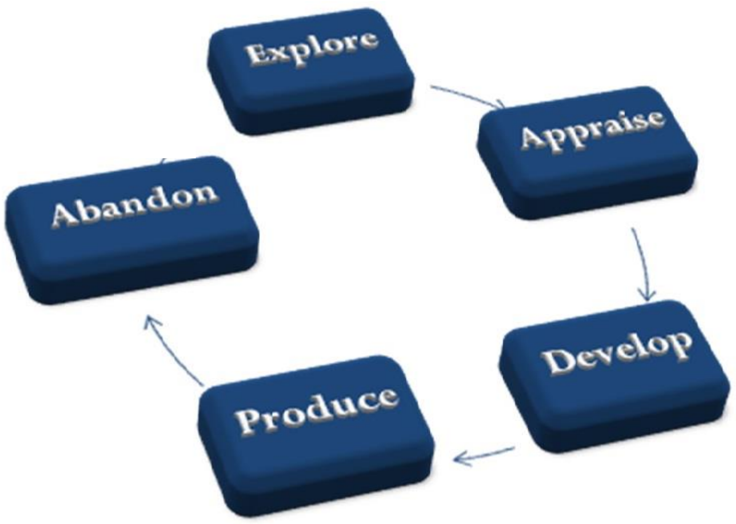


Figure 1.5: Petroleum lifecycle illustrating the cradle-to-grave process of an oil field

Though we will briefly discuss all five phases; we note however, that the main focus and contributions in this thesis are centred on the development and production phases.

1.5.1 Exploration Phase

The exploration phase involves finding and locating oil and gas reservoirs. Exploration can be very challenging because subsurface reservoirs can be located at great depths beneath the earth surface, and sometimes in some of the most politically unstable, or environmentally unfriendly or inaccessible areas of the world. Usually, sound waves generated by exploding dynamite in shallow holes or from vibrator trucks (or with the aid of airguns in offshore sites) are sent through the subsurface; and the refraction and reflection of such waves are measured (with geophones or hydrophones) and analyzed to determine if the subsurface structures therein can serve as a hydrocarbon trap. With the aid of a velocity model of the rock and the seismic velocity log obtained via well bores, the arrival times are subsequently converted into depths. Geoscientists interpret these seismic data and create 3D maps of the subsurface in a process called *seismic imaging*. If a promising reservoir structure is on the card, an exploration well is drilled to ascertain if indeed petroleum is present. It is important to note that an exploration well is not the same as a discovery well. This is so because the exploration well may indeed not contain oil; however, the exploration well is referred to a discovery well only if it contains oil. As a high risk venture, risk assessment and risk management expertise are fundamental to successful exploration portfolio management.

1.5.2 Appraisal Phase

In the appraisal phase, various feasibility studies are performed, and the major concerns are the economic as well as technical and environmental cost of producing the content of the reservoir. Appraisal and delineation wells are drilled to determine the size of the oilfield and how to develop it most efficiently. The core samples and logs from wells are combined with available seismic data to form models which can be used to roughly predict how field development decisions affect future production. Obviously, a reservoir or field (a collection of reservoirs related to the same geological structure) will only be developed if these assessments are promising enough.

1.5.3 Development Phase

The key objective of field development and indeed during the entire lifecycle of the oil reservoir is the maximization of economic or financial value of the assets, subject to prevailing project constraints. Well placements and configuration as well as design of surface facilities from which produced oil can be stored and transported, and other key operational logistics are determined at this phase.

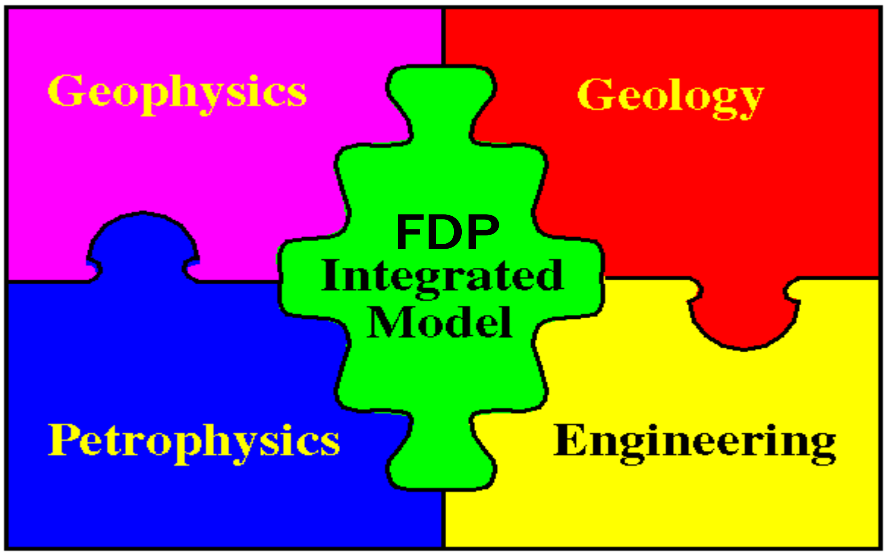


Figure 1.6: Visual Description of field development phase – no single discipline fits the bill (courtesy Schlumberger public)

It often involves the comparison of large number of development scenarios, usually in combination with a large number of subsurface models to account for the effects of geological uncertainty in the working reservoir model. As no single discipline fits the bill, immense cross-fertilization of ideas between the geophysicists, geologists, reservoir engineers, and well production engineers is common at this phase; they are usually supported with appropriate and integrated work-flow systems (software) as depicted in Figure 1.6.

1.5.4 Production Phase

The production phase involves all the processes that are aimed at depleting or draining the reservoir. Economically, it is by far the most important phase of the hydrocarbon reservoir lifecycle, as oil and gas are produced at this phase. Thus, it is the phase that brings return on investment (ROI) to fruition, as the previously mentioned phases (exploration, appraisal and development) as well as the last phase (abandonment) are capital expenditures. Therefore, it is during the production phase that the project can break even, and more importantly, make profit. Usually, the recovery of oil in the production phase is divided into primary, secondary and tertiary recovery processes.

In its initial state, the reservoir and its content are in a state of high pressure equilibrium that has been established for eons. When a well is drilled through the impermeable seal rock and into hydrocarbon bearing reservoir rock, this equilibrium is disturbed; and if not properly controlled by pressure valves (connecting reservoir to wells, and surface facilities), can lead to *blow-out* which can be catastrophic. The hydrocarbon is subsequently produced as a result of the existence of high differential in pressure which drives the fluid toward the wellbore and surface facility in a process referred to as ***primary recovery*** by natural drives. The drive mechanisms that powers primary recovery include – depletion drive (solution gas drive and gas cap drive), water drive (from active aquifer or artesian water), compaction drive, gravity drainage and combination drives. During this recovery stage, only a small percentage of the original oil in place is produced. Referred to as the recovery factor, this is perhaps about 20% for most oil reservoirs; after which there is a significant drop in the reservoir pressure. This drop in pressure leads to reduced flow of oil; and eventually, the production process will no longer be economically sustainable – as a new equilibrium is attained in which the reservoir pressure is equivalent or near-equivalent to surface pressure.

This marks the beginning of ***secondary recovery*** by engineered drives. Secondary recovery often involves the injection of water or gas into the reservoir. Waterflooding, which involves the injection of water (via injection wells) into the subsurface reservoir, is arguably the most commonly deployed secondary recovery operation. The reason for water injection is twofold – to re-build the lost reservoir pressure, and to sweep out more hydrocarbons from the injection wells towards designated production wells. Depending on the formation of the reservoir, it is perhaps possible to produce another 15-35% of the original oil-in-place by this process, Rossi et al. (2002), Golder Associates (2000). Waterflooding engineered drives are standard procedure at most E&P locations around the world, this recovery procedure is the focus in this thesis.

Finally, in order to produce even more oil, ***tertiary recovery*** or enhanced oil recovery (EOR) techniques are employed. This refers to techniques that alter the original properties of oil; for example, using chemicals such as solvents and polymers, or steam heating the reservoir. At present, such EOR techniques are considered too expensive for large scale commercial use. Several studies and research are being conducted in this area, with a view of investigating its mathematical foundations and economic viability, Aarnes et al, (2007).

1.5.5 Abandonment Phase

In broad terms, hydrocarbon reservoirs are said to have reached their *economic limit* when the most efficient production rate does not cover the operating expenses and the prevailing tax regime. When the economic limit is reached, the project becomes a liability and the process which involves reversing the field (as close as possible) to its initial environmental condition is initiated. Because there is often a significant amount of ‘unrecoverable’ oil left in the reservoir at this stage, physical abandonment is often deferred for extended period of time. The reason for deferment is the overriding hope that the price of oil may shoot-up to justify production, or that newer and effective recovery techniques will emerge. In some jurisdiction, however, lease provisions as well as governmental regulations usually require complete physical abandonment.

1.6 Field Development Plan and Production Optimization

As mentioned earlier in section 1.5, the phases that are relevant to this work are the development and production phases. Field development planning or FDP encompasses all activities and processes required to develop a field. Often, the economic and technical targets as well as development and production strategies are clearly defined at this stage. The goal is simply to maximize an economic criterion such as return on investment (ROI) or net present value (NPV) of the asset, while minimizing capital expenditure and environmental impact. FDP involves critical investment and operational decisions that can potentially make or mar the project. These decisions are based on field constraints, company's policy, technical information, economic judgments and even political factors.

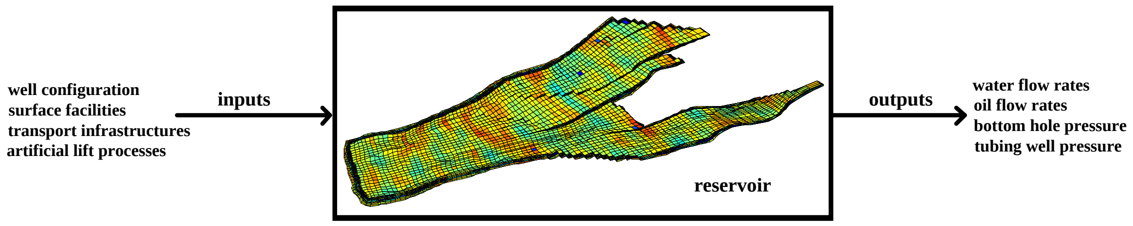


Figure 1.7: An open-loop input-reservoir-output representation of a reservoir system, SINTEF

Therefore, to ensure the highest return on investment, optimal operational decisions are crucial in FDP. These critical decisions are generally referred to as inputs, and they include well type, spacing, completion design, lift strategies, surface facilities and infrastructure. The effects of these inputs (which are referred to as outputs) include the pressures in the wells and the flow rates of the produced fluid. Thus, the inputs represent external functions, forces or sequences that are acting upon the reservoir; while the outputs represent the measurable or observable behaviour of the reservoir. In this sense, the input and output constitutes a cause–effect relationship; and this is schematically illustrated in Figure 1.7. To facilitate the study of the interaction between the inputs, the reservoir, and corresponding outputs; the reservoir is often conceptualized as a dynamical system[5]. This system represents the reservoir in terms of some mathematical relationships between the inputs, the outputs, and the states – which are the time-varying properties of the reservoir. In the context of reservoir engineering, the states are usually the pressures and saturations. The fluid properties (viscosities and densities) as well as the geological properties (permeability and porosity) are assumed to be time-invariant; they are generally referred to as parameters. Making appropriate input decisions that will maximize the potential output of a reservoir is by no means trivial. One possible

[5] In this context, a system is a mathematical abstraction that is devised to serve as a model of a dynamic phenomenon of interest.

approach to tackle this challenge is through a wide array of techniques collectively termed *production optimization*, Sarma (2006).

In this regards, production optimization refers to the overall maximization of the performance of hydrocarbon reservoirs through optimal development and production decision inputs. Thus, production optimization is closely knitted to FDP as it is pretty much impossible to divorce one from the other. Production optimization and FDP are iteratively related in the sense that they involve repetition of sequence of activities that transform measured or collected data into optimal decision-inputs for enhanced reservoir productivity as schematically illustrated shown in Figure 1.8. Clearly, at the heart of the success or otherwise of the iterative processes in field development (or redevelopment) planning and production optimization is the reservoir simulation model. These reservoir models are of crucial importance, as they often play key roles in all the activities and sub-activities of FDP and production optimization alike. They seek to describe the effect of decisions on petroleum reservoirs and are firmly based on physical conservation laws as well as other simplifying assumptions.

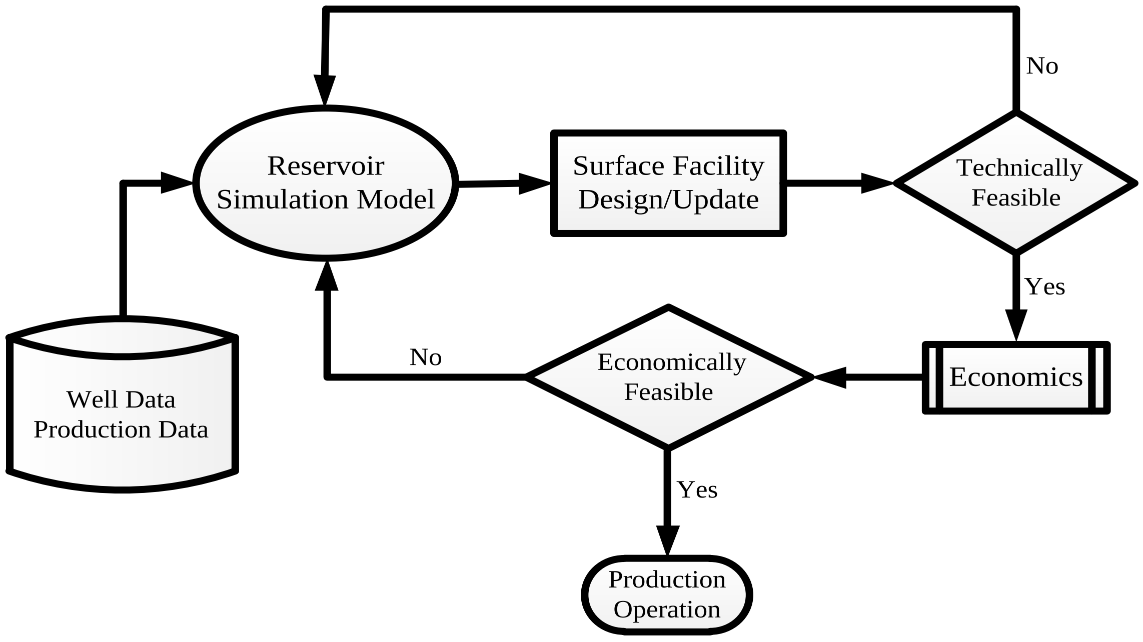

No
Reservoir Simulation Model
Surface Facility Design/Update
Technically Feasible
Yes
No
Economically Feasible
Economics
Well Data Production Data
Yes
Production Operation

Figure 1.8: Illustration of the iterative relationship between FDP and production optimization

Although reservoir model parameters are considered to be time-invariant, it is important to note that these parameters (especially the geological properties) are spatial-varying. In other words, they vary quite significantly over the space of the reservoir. These variations are usually referred to as heterogeneities, and in a waterflooding production secondary recovery operation, these heterogeneities can result to preferential flow paths as shown in Figure 1.9.

Indeed, significant quantities of oil can be bypassed inside the reservoir as a result of the irregular water-oil front that characterizes preferential flow path. In order to reduce this phenomenon and its attendant effects, it is essential to factor-in these geological heterogeneities in reservoir modelling. To this end, it is common to divide the reservoir model into a finite number of coarse grids blocks whose geological and fluid properties are assumed to be homogeneous, Aarnes et al. (2007).

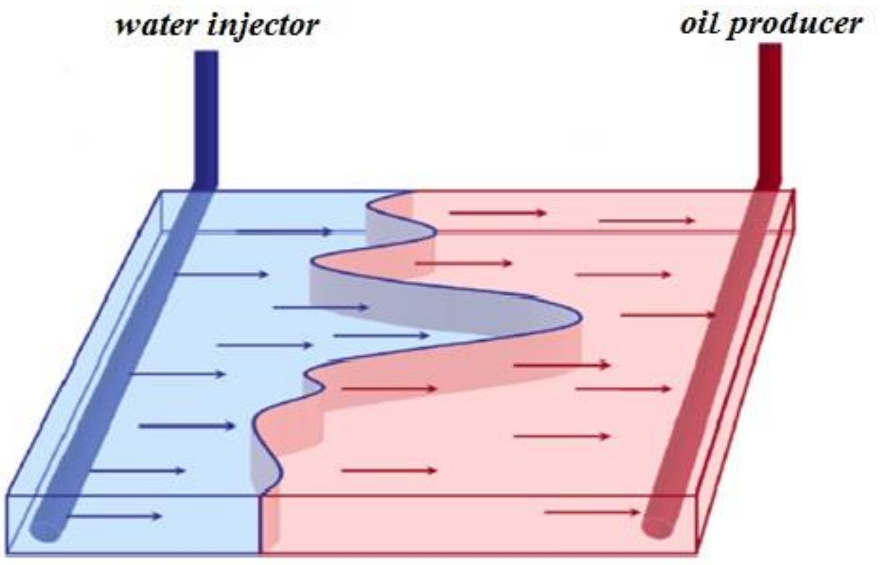


Figure 1.9: A simple waterflooding production operation – water is injected through the injection well on the left in order to flood the reservoir and sweep the oil to the production well on the right – adapted from Jansen et al. (2005).

However, it should be noted that this measure only serves to mitigate the issues associated with the effects of heterogeneities; it does not remove them entirely. The idea of mitigating or containing the effects of heterogeneities stems from the fact that it is impossible to obtain a complete and accurate characterization of rock parameters and dynamical states that influence flow in porous media. And even if we did, it would be impossible to simulate reservoir models that are based on precise geological grid blocks. This would certainly require a tremendous amount of computational resources which would exceed by far the capabilities of modern multi-processor computers. Thus, coarse grid models with grid-blocks that are typically ten to hundred times larger than actual geological grid models are built using some kind of up-scaling of the geophysical parameters, Aarnes et al. (2007). In other words, it is fair to say that reservoir simulation models are only but a crude approximation of subsurface geological reality. Their predictions of reservoir performance viz-a-viz field (re)development decisions as well as production optimization and the resulting oil production profiles are prone to errors and uncertainties.

1.6.1 Production Operation and Reservoir Management

In production operation, the primary concern is usually to meet some pre-defined daily production targets; while the primary concern in reservoir management is the maximization of reservoir's recovery or project's asset value during its entire lifecycle. In other words, the time-scale in the production operation domain is in the order of days or weeks, while the time-scale in reservoir management domain is in the order of years or decades. A clear distinction in the spatial and time domains of these processes is depicted in Figure 1.10.

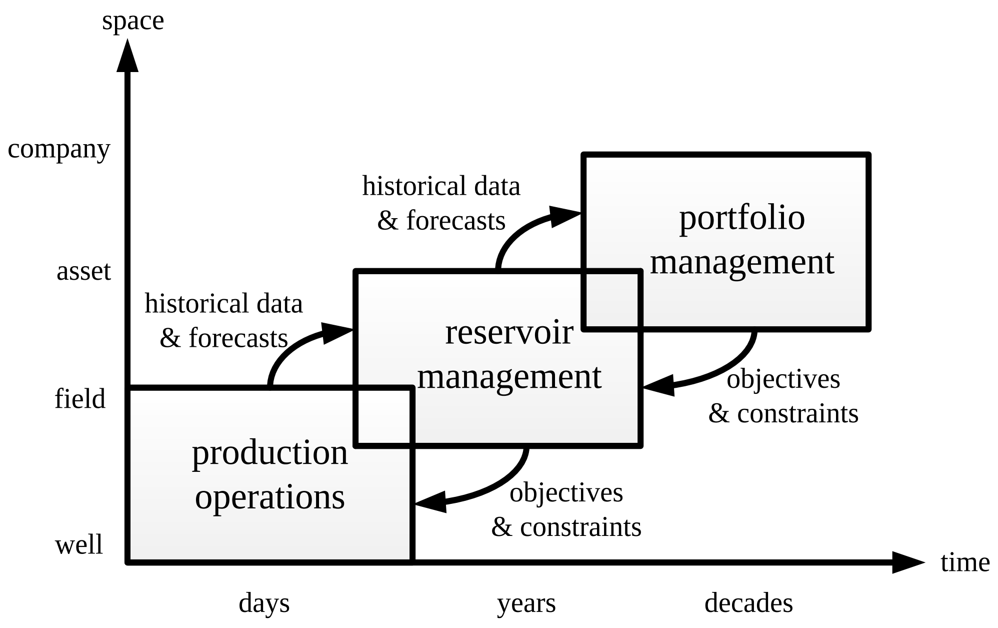


Figure 1.10 E&P activity spatial and temporal domain as adapted from ISAPP–TNO

Each domain provides both the objectives and the constraints to the domain below it in the sequence, and at the same time providing historic data and forecasts to the domain above it. If for example, we consider the simple waterflooding production strategy shown in Figure 1.9; the task in production operation would be to meet a stipulated daily production target; whereas that of reservoir management would be to maximize total oil production or reservoir recovery, while minimizing total cost of production or staying within operational constraints.

Usually, the production operation decisions which include individual well production settings, well workovers and interventions; are performed in a manual or ad-hoc fashion, which is often based on rule-of-thumb, operator's experience and the prevailing production scenario realities. This invariably translates to the fact that production operations decisions are implemented in an open-loop (input-reservoir-output as illustrated in Figure 1.7) manner; therefore, for the simple waterflooding production operation scenario illustrated in Figure 1.9, such open-loop production system means that water would eventually breakthrough at the producer, even as unproduced oil is bye-passed parts of the reservoir.

Thus, traditional production operation strategies as described above are generally reactive, unreliable and sub-optimal. In an attempt to move from reactive production operation, field data gathered over long periods of production operations, via reservoir surveillance, are subsequently incorporated into the reservoir model in a special 'feedback' mechanism known as *production history matching*. History matching is a notorious time-consuming exercise; it is often carried out only when the underlying reservoir is considered for re-development. The aim is to tune the reservoir model so that it is consistent with field performance, and the ultimate goal is to have a re-calibrated reservoir model with high prediction capability.

1.6.2 Closed-loop Reservoir Management

In the last sub-section, the goals and operational time-scales that are relevant in production operation and reservoir management domains were underscored. It was also highlighted that data gathered from daily production operation via reservoir surveillance are incorporated into the reservoir model to re-calibrate it for better performance. In Chierici (1992), the need to bridge the gap between the goals and time-scales by some kind of continuous feedback mechanism was underscored. The author posits that a "continuous feedback process" is required throughout the lifecycle of the field (as against the campaign-based approach of traditional reservoir management) in order to maximize its recovery factor.

It is therefore important to underline that it is the essence to bridge the short-term production operation goals and long-term reservoir management goals that gave birth to the concept of closed-loop reservoir management. Also referred to as intelligent fields, smart fields, self-learning reservoir management or real-time reservoir management; the underlying principle is based on the theory that there exists a significant potential to increase hydrocarbon recovery through continuous optimization and continuous updating of the reservoir model – see Jansen et al. (2005, 2009). Depicted in Figure 1.11, closed-loop reservoir management draws inspiration from basic concepts in system and control theories, as successfully applied in the process and chemical industry.

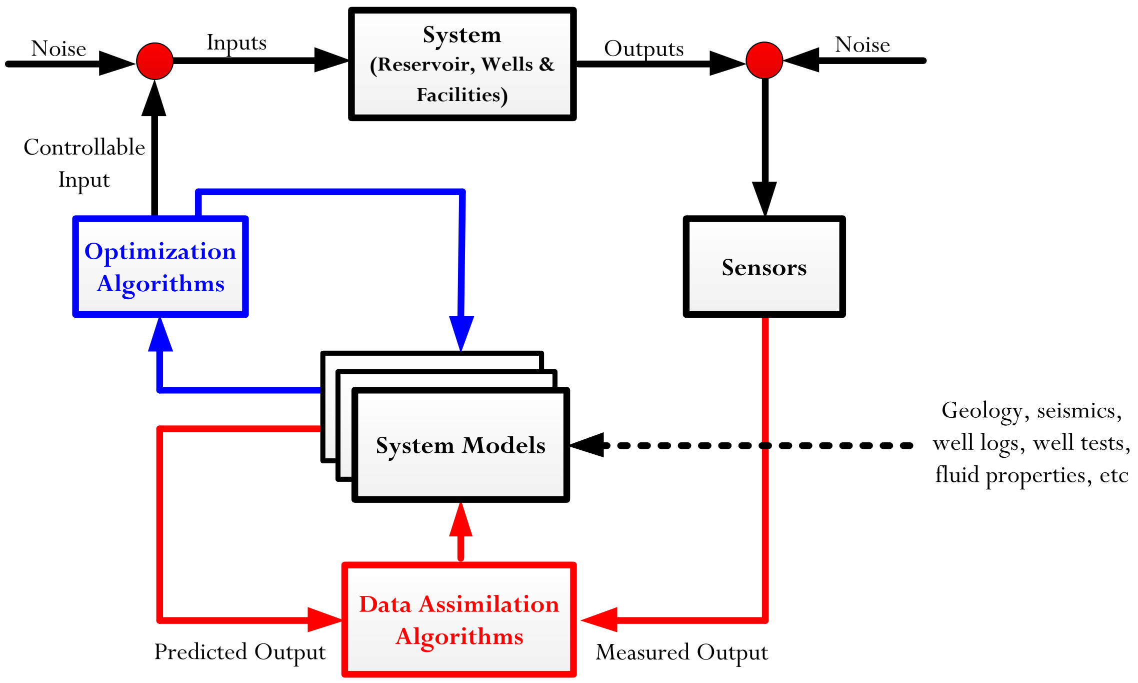

Noise
Inputs
System (Reservoir, Wells & Facilities)
Outputs
Noise
Controllable Input
Optimization Algorithms
Sensors
System Models
Geology, seismics, well logs, well tests, fluid properties, etc
Data Assimilation Algorithms
Predicted Output
Measured Output

Figure 1.11: Closed-loop Reservoir Management (CLoReM) courtesy Jansen (2009)

Although the fundamental idea behind closed-loop reservoir management can be traced back to Chierici (1992), the fact cannot be overlooked that the 'reawakening' of this concept is credited to Jansen et al. (2005). Prior to this reawakening, the idea of and research in closed-loop reservoir management was as it were, in a state of dormancy. This is mainly because practical implementation of a continuous feedback mechanism in reservoir management cannot be feasible if its underlying components – field development plan, production operation and history matching – are not efficient. This is clearly understandable because the reason why continuous feedback is not applied in the first place is that they are notoriously time-consuming, a testimony of the fact that the techniques or algorithms employed for that purpose are not suitable. Using Figure 1.11 as a point of reference, the closed-loop reservoir management framework begins with the optimization loop (marked in blue colour) that is performed on the current reservoir model. Using well rates and bottom hole pressures (BHP) as control variables, the objective of the optimization is often to maximize a performance measure such as net present value (NPV) of the reservoir. In other words, the optimization provides optimal settings of the controls for the next step. These optimal control settings are subsequently applied to the physical reservoir as input; and the corresponding outcome of these inputs are measured outputs such as BHP, oil production profile or water cuts.

With the aid of these output measurements, new information about the reservoir model can be inferred; and the reservoir model is subsequently updated or re-calibrated on the basis of the inference made from the measured outputs. The model-updating or history matching loop (marked in red) is a feedback estimation process; it serves the purpose of updating or estimating the reservoir model so as to enhance its performance. In the next control step, the optimization loop is performed on the updated model, the resulting control is applied to the real reservoir as inputs, and the model is again updated using information or data from the measured output. This process continues iteratively throughout the production life of the reservoir. It is however important to note that this continuous optimization and estimation approach to reservoir management (i.e. closed-loop reservoir management) is generally impracticable. The complex nature of the physics behind porous media flow and the inherent geological uncertainties in reservoir models are significant debilitating factors.

1.7 Research Aims and Motivations

In the light of the foregoing, a number of points have been established. These include:

1. Man and civilization are inseparable from energy, as the energy utilization capacity of any given population or economy is strongly correlated to its quality of life
2. Global energy demand would continue to rise in the foreseeable future
3. Owing to its enormous direct-end-use capacity and its high caloric value, there is no realistic hope that oil would relinquish its position as the number one energy resource in the next quarter-of-a-century, and maybe beyond
4. As long as the E&P industry continues to struggle to meet the ever increasing demand for oil, price will continue to rise and
5. As far as primary recovery is concerned, the current industry average for recovery factor is between 15% (for naturally fractured reservoirs) and 35% (for reservoirs with favourable production conditions)

Having established the above, it is now imperative to explain in clear terms the underlying motivation for this work. The motivation for this research is borne out of the desire to develop reliable and efficient tools that would lead to an increase in the present meager recovery factor of existing and new petroleum reservoirs, so as to increase the cumulative production of oil. This, in the opinion of the author is imperative, if meeting the challenges of rising global demand of oil is anything to go by.

Surely, the use of innovative solutions and advanced technology are crucial to achieving this goal. Thus, many commercial E&P companies have over the past decades invested heavily in the research of innovative solutions and the development of state-of-the-art technology that are geared towards enhancing reservoir fluid recovery, and improving the overall efficiency of the many processes a typical reservoir is subjected to during its operational life span. In this regard, the high price of crude oil (which is as a result of the disproportionate relationship in the market forces of demand and supply) has been the saving grace of the industry. It has provided the availability of funds for the research and development of innovative solutions. In this work, we focus squarely on deploying reliable optimization, control and estimation tools that would make the constituting elements of reservoir management as efficient as possible. Although we acknowledged that there are various techniques (such as enhanced oil recovery, development of non-conventional resources, etc.) that can equally lead to increased cumulative production of oil on the global stage; it is however noted, that all such techniques fall outside the scope of this thesis.

1.7.1 Research Objective

Having highlighted the motivation of this work, the research objective is therefore summarised as follows:

> ***to develop and deploy efficient optimization, control and estimation techniques that would lead to the maximization of hydrocarbon reservoir recovery factor within the ambit of model-based closed-loop reservoir management framework***.

1.7.2 Research Approach

As stated earlier, the guiding principle of the thesis is centred on the maximization of recovery factor of petroleum reservoirs within an efficiently tailored optimization, control and estimation framework. In other words, we seek to develop efficient decision support work flows for optimizing reservoir performance via model-based optimization, control and estimation techniques. To this end, we propose a metaheuristic and system-theoretic framework as illustrated in Figure 1.12. Modified from the original closed-loop reservoir management (Figure 1.11) and Jansen (2010); the fundamental ideas behind this framework are described as follows.

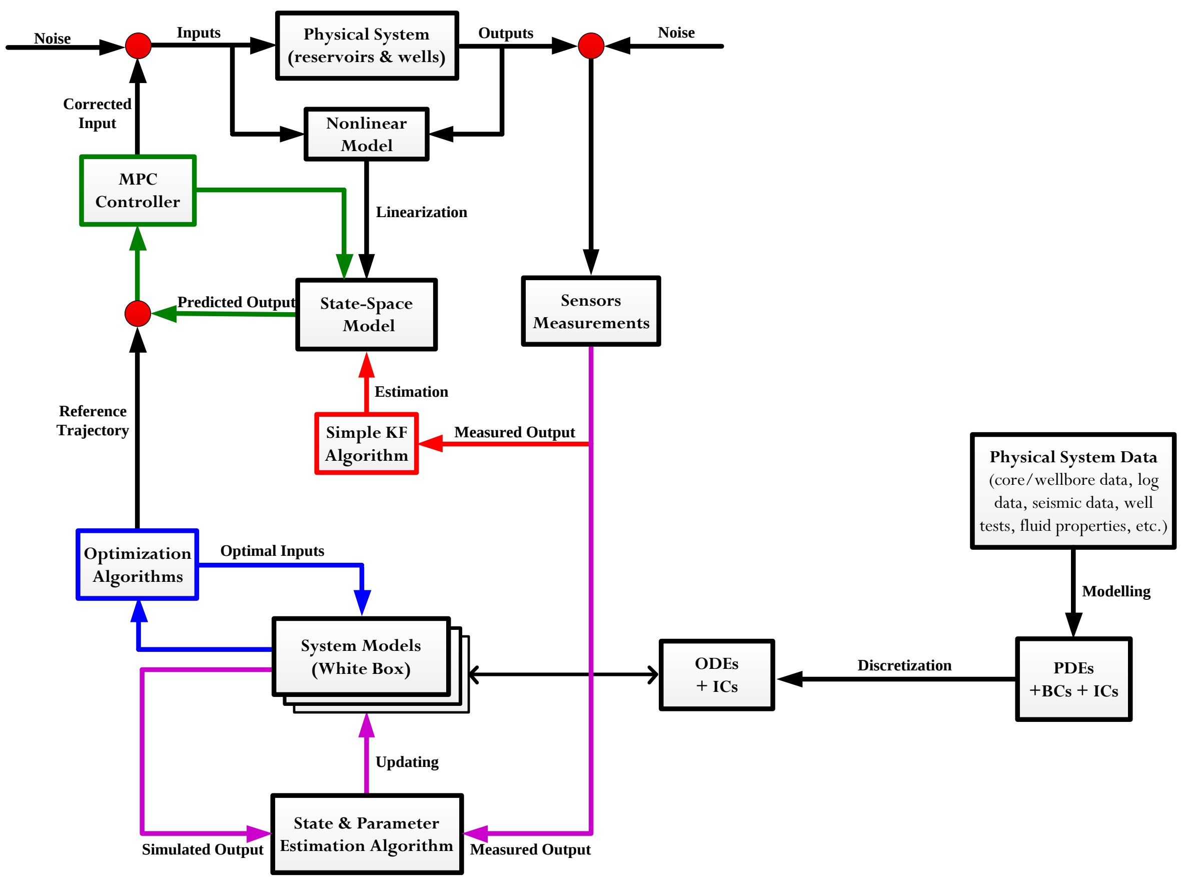


Figure 1.12: A metaheuristic and system-theoretic reservoir management framework as modified from Jansen (2010)

Firstly, we underline that continuous optimization and frequent updating of the working reservoir model (the cornerstone of the closed-loop reservoir management framework) are cumbersome and almost impracticable. The blue and magenta loops in Figure 1.12 is basically the same as Figure 1.11 – the iterative optimization and estimation processes are carried out as described in sub-section 1.6.2. The green loop is a predictive controller which serves the purpose of tracking the ‘optimal’ profile resulting from the production settings optimization based on updated reservoir model (blue loop). The goal of the green loop is to stay as close as possible to the profiles resulting from the blue loop. Evidently, the working model of the blue and the green loops are correspondingly updated via the magenta and red loops respectively. The key difference however, is the time-scale at which these updating processes are performed. While the working model of the blue loop is updated periodically (because of computational cost associated with physics-based white box model), the model of the green loop – which is a linearized nonlinear model obtained via system identification – is updated frequently. Hence, it follows that the key elements of this framework are – field development optimization (with degree of freedom restricted to well configuration), production settings optimization, model predictive control and production history matching[6].

Upon close observation and scrutiny of the aforementioned key elements, one can invariably infer that they are centred on the theories of optimization, estimation and control. Although there are numerous optimization, control and estimation techniques that are mature viz-a-viz their applicability and effectiveness in the chemical process industry; it is important however, to underline that their applicability in reservoir engineering problems is severely handicapped by the very nature of reservoir models. In the light of the above facts, it is therefore essential that any candidate optimization, estimation and control techniques that is deployed in this work, and in reservoir engineering applications as it were, must explicitly take into account that hydrocarbon reservoir models are generally a physics-based multiple input multiple output (MIMO) system of nonlinear equations with large number of time-varying states and time-invariant parameters. Unsurprisingly, many optimization, estimation and control techniques are usually not applicable to reservoir models; this underscores the need to tailor them towards suitability for and applicability in reservoir modelling and engineering.

1.7.3 Original Contribution

At this juncture, we highlight the original contribution of this thesis. The contributions as contained in this work are as follow:

- implementation of a novel hybridized metaheuristic algorithm for field development optimization (where the degree of freedom is on well placement optimization)

- use of a novel predictive control algorithm which is based on a linearized nonlinear model for tracking optimal production trajectory

[6] It is important to state that we did not carry out the history matching component of this work

1.8 Summary and Thesis Outline

In this introductory chapter, it was established that global energy demand and per capita energy utilization have continued to grow from one age to another; it was further established that the rate of increase has been more significant in the most technologically advanced and most technologically advancing culture in each particular age. Besides Malthusian[7] growth of human population, a number of factors were attributed to the unprecedented rise in energy demand. These include the ever-increasing number and variety of energy-using contrivances, as well as phenomenal economic growth in China, India and other rapid emerging economies. Of the diverse energy resources that man can harness today, oil is the most demanded and utilized – no thanks to its tremendous direct-end-use capacity. Considering that oil is a non-renewable energy resource which depletion rate is alarmingly worrying, and the fact that the so-called 'easy oil' (i.e. cheap and quick-to-recover oil) has been produced; the need to develop efficient techniques that would maximize the recovery factor of existing and new oil reservoirs becomes crucial and essential. It is noted that there are other ways to increase the cumulative production of oil; but for all intents and purposes, the focus of this thesis is on optimization and control techniques within a metaheuristic and system-theoretic closed-loop reservoir management framework. This work is structured as follow – in chapter 2, we undertake an in-depth review of literature with the view of highlighting the strengths and weaknesses of the techniques that have already been deployed in the problems set out in this thesis; and reservoir modelling is presented in Chapter 3. Chapter 4 deals with field development optimization, where the degree of freedom is on well placement optimization problem; while Chapter 5 is on production settings optimization and control technique for effective tracking of simulated optimal production profile. The thesis ends in Chapter 6 where we draw conclusions, make recommendations and suggest future work directions.

[7] The Malthusian model is basically a simple exponential growth model that is named after Reverend Thomas Malthus who authored "An Essay on the Principle of Population".

CHAPTER 2

Those whose palm kernels have been cracked by a benevolent spirit should not forget to be humble – Chinua Achebe

LITERATURE REVIEW

In this chapter, we will take a look at the literature as it relates to the relevant elements of the framework upon which this work is based. Thus, the literatures on key elements of the framework are reviewed with respect to their applicability or otherwise in reservoir engineering problems as well as other closely related fields. The aim is to note the gaps and weaknesses therein, with the view to explore the strengths and exploit same.

2.1 Well Placement Optimization

During field development planning, determining the optimal number, type, location and drilling sequence of wells is arguably the most important decision input. It is generally a non-trivial task which has a significant bearing on the asset value of the project, as it can potentially determine the recoverability (or otherwise) of the hydrocarbon in place. The fact however, is that the very nature of subsurface reservoirs make well configuration[8] optimization a very challenging problem.

Usually, it is common practice in reservoir engineering to associate wells to reservoir grid block cell centres where they are represented as source or sink terms – depending on whether they produce fluid from, or inject fluid into the reservoir. Hence, the optimization of the well trajectory and its location is typically an integer problem as established in Bangerth et al. (2006). Since determining the number of wells is an integer problem, the combination of this optimization problem with the optimization of the wells' production settings invariably leads to a mixed-integer nonlinear problem (MINLP), Kosmidis et al. (2005). However, due to

[8] Well configuration often entails the number, type, location and possibly the drilling sequence of hydrocarbon reservoir wells.

issues bothering on non-convexity, such MINLPs are extremely difficult to solve as elaborated in Haus et al. (2008). Another drawback with the application of MINLPs is that they require far too many evaluations of the objective function (which entails full reservoir simulation), and this renders it less-effective in reservoir engineering applications. To this end, most well optimization solution methods in the literature are either gradient-based local optimization methods (which require the computation of derivatives of the objective function using some numerical finite difference schemes, or the adjoint formulations) or derivative-free stochastic optimization techniques (which employ heuristic or metaheuristic algorithms).

Local optimization methods attempt to find the optimum by iteratively improving upon an initial guess (well placement) until the optimal, albeit local, is determined. The main drawback of these methods is the challenge of finding improving directions in which to alter the initial guess. Bangerth et al. (2006) compared three local techniques – the finite difference gradient (FDG), the simultaneous perturbation stochastic approximation (SPSA) and the very fast simulated annealing (VFSA) – in optimising the placement of vertical wells in a 2D reservoir model. The FDG method attempts to find improving directions by perturbing each of the well location by one grid block in every direction. The obvious drawback of this method is that in order to compute an improving direction for any *n* to-be-placed wells, there is a minimum requirement of $2n+1$ number of objective function evaluations. The SPSA method which was earlier employed in Spall (1992) is basically an approximate gradient-based technique. To compute the derivative, a random direction in which to alter the wells is generated, and if this change in position of the wells does not yield an improvement in the objective function, then the opposite direction is automatically assumed. This algorithm was shown to perform better than genetic algorithm (GA) in the optimization of vertical wells; and it is noted that the computational requirement for this method is less-expensive, as the maximum number of reservoir simulation required to determine improving direction is found in at most two reservoir simulations. However, the disadvantage of the SPSA algorithm is that the assumed optimal configuration may generally not be the 'steepest' one. Another drawback appears in the calculation of new solutions; the step size must be carefully chosen, otherwise there is a risk of finding 'solutions' that are not feasible. Thus, the assumed efficiency of this method is both suspect and questionable. The VFSA method is based on standard simulated annealing (SA). It had been previously deployed in a number of geophysical inversion applications – see Ingber (1989), and it has close semblance to most stochastic approximation algorithms. In all cases considered, both the VFSA and SPSA outperformed the FDG method.

Wang et al. (2007) applied a gradient-based steepest descent algorithm in optimizing the number and placement of injection wells in a 2D reservoir model; while Sarma and Chen (2008) applied a gradient-based algorithm where the derivative of the objective function is computed with respect to continuous well locations, thereby allowing for arbitrary step size and search directions. A gradient-based algorithm was utilized in Zandvliet et al. (2008) for the optimal placement of vertical wells in a 2D reservoir model. It is important to note that the methods employed in Sarma and Chen (2008) and Zandvliet et al. (2008) are based on the same principle. The difference however, is in the derivatives used and the method of its computation. While the derivative of the objective function in Sarma and Chen (2008) is with respect to continuous well locations, the derivative in Zandvliet et al. (2008) is with respect to flow rates. At each discrete time step, an adjoint formulation was used to compute the "rate gradients" for each of the low rate "pseudowells" that are placed at the eight neighboring grid blocks surrounding a current well position. The derivatives (with respect to flow rate) at the pseudowells are then summed, and the well is moved in the direction of the pseudowell with the largest summed gradient. In terms of computational efficiency, these gradient-based approaches are highly efficient because they often require fewer number of objective function evaluations. For example, the gradient-based methods where adjoint formulation is used for the computation of derivatives require only two objective function evaluations irrespective of the number of decision variables.

Nevertheless, it is important to note that these gradient-based techniques have their own drawbacks. The non-convex nature of the underlying optimization problem inevitably means that they generally contain multiple optima; hence, the gradient-based methods are prone to be trapped in local solutions. In addition, discontinuous derivatives arising from the non-smooth nature of the optimization surface may also pose significant problems. It is also important to recognize the fact that gradient information is often not readily available. Adjoint formulations, which are a popular and efficient way of computing derivatives, are invasive with respect to the flow simulator; they are therefore only feasible with full access to, and detailed knowledge of, the simulator source code, Ciaurri et al. (2011). Besides, the objective function value may be computed with some noise, and this therefore means that any computation of derivative estimates is susceptible to lots of inaccuracies.

The derivative-free stochastic optimization techniques by their very nature are generally non-invasive with respect to the flow simulator. They treat the simulator as a black-box, as only cost function values are required and no explicit gradient computations are involved. These methods are therefore easier to implement than, for example, the adjoint-based techniques. However, in a typical '*no free lunch*' fashion – see Wolpert and Macready (1997); this advantage is counterbalanced by a significant deterioration in computational efficiency when compared to gradient-based approaches. In other words, the derivative-free global methods will tolerate lower performance measures in the hope of finding the global optimum, as opposed to the computationally efficient gradient-based local optimal solutions. This appears to be the reason why there are many applications of gradient-free metaheuristic optimization algorithms in well placement optimization literature. In this regard, GA appears to be the most frequently used technique.

Beckner and Song (1995) employed simulated annealing (SA) algorithm in the optimization of the placement and schedule of horizontal wells. The optimization problem was cast as a traveling salesman problem in which the potential well locations are represented as cities on the salesman's tour, and the drilling schedule was represented as the sequence for visiting those cities. The ensuing travel salesman problem was subsequently solved using the SA algorithm. In the study of Centilmen et al. (1999), a neuro-simulation technique that is based on fully trained artificial neural networks (ANNs) was employed. The network neurons are trained using data arising from simulation results of randomly selected well placement scenarios that are entirely based on rule of thumb but supported by engineering judgment. The authors applied this technique in well placement problems of different complexities, and the results showed that the use of simulated-results trained ANNs was effective in terms of reducing computational cost and maintaining accurate predictive capabilities. As stated earlier, GA appears to be most utilized stochastic algorithm in the well placement optimization literature. In this regards, the studies by Bittencourt and Horne (1997), Guyaguler et al. (2000), Montes et al. (2001), Aitokhuehi et al. (2004), Onwunalu (2006) and Farshi (2008) were reviewed. In Bittencourt and Horne (1997), a hybrid GA that is based on polytope search and tabu search was developed and applied in the development of a real oil field; and the result led to a reduction in the total number of well originally earmarked for the project – thereby leading to a 6% rise in the profit of the project. Another hybrid GA which is based on the polytope method and the kriging algorithm (proxy approach) was deployed by Guyaguler et al (2000) for the optimal placement of wells in the GOM Pompano field. The authors reported that the hybridization of the GA introduces hill-climbing capabilities in the algorithm, and this significantly reduces the number of simulations required; and therefore, computational time is effectively reduced. In the study of Montes et al. (2001), a simple GA was developed and applied to two well placement optimization case studies; afterwards, the authors investigated the effects of GA control parameters on the performance of the algorithm. The primary focus of the study of Aitokhuehi et al. (2004) was on the use of GAs in nonconventional well types (monobore, dual-lateral, tri-lateral, multi-lateral) and trajectory optimization problems. Basically, the approach involves the coding of well type and trajectory information on binary strings (or chromosomes) of the algorithm; and allowing the encoded information on the chromosome to evolve through the probabilistic steps of crossover, mutation and elitism over the course of the GA optimization process. In the studies of Onwunalu (2006) and Farshi (2008), a kriging algorithm was used as a proxy in the application of GA in nonconventional well optimization problems. A combination of ANN and GA was applied in Yeten et al. (2003) in the optimization of nonconventional wells. It is important to note that the study by Yeten et al. (2003) was more or less similar to the work by Aitokhuehi et al. (2004). However, the only difference was that the authors in Yeten et al. (2003) included the use of ANNs as proxy algorithm; and this significantly reduced the amount of computation required. Thus, the combination of ANN and GA in Yeten et al. (2003) led to an efficient optimization process. Onwunalu and Durlofsky (2010) and Dong et al. (2011) employed the use of particle swarm optimization (PSO) in the placement of conventional and nonconventional wells in reservoir models of varying complexities; and compared the NPV attained by this technique with results from binary GA algorithm. On the average, it was found that the PSO algorithm outperformed the GA algorithm in all the cases considered. Using PUNQ-S3 benchmark reservoir case, Bouzarkouna et al. (2011) applied covariance matrix adaptation evolution strategy (CMA-ES) to a number of well location and trajectory optimization problems and compared the results from this approach to that resulting from the use of GA. The authors demonstrated that CMA-ES generally outperformed GA in terms of net present value (NPV) attained, and importantly, the approach also led to a significant reduction in the number of function evaluations needed to reach a good well configuration.

Therefore, it follows that the use of derivative-free stochastic or metaheuristic algorithms are quite popular in well optimization problem solutions. The popularity of these methods are largely enhanced by the fact that the previously mentioned high computational demand associated with them are substantially reduced by employing multiple and parallel processors. This notwithstanding, a computational efficient metaheuristic algorithm that require limited number of simulations (objective function evaluation) is still lacking.

2.2 Production Optimization

In this thesis, all reference to the production phase is implicitly restricted to secondary recovery, unless otherwise stated. Again, it is worth noting that the relevant production operation strategy is the waterflooding process. For a given reservoir model with a specified well configuration; the well schedule that maximizes the recovery factor over its production life can be posed as a dynamic optimization problem. In every practical sense, this dynamic optimization problem is centred on finding the time-varying input settings (such as well rates, bottom-hole-pressures (BHP), valve or choke settings, etc.) that maximizes the recovery factor. Generally, there are three distinct approaches to dynamic optimization problems. These include the classical approach (which is based on the calculus of variations, and has been developed into the well-known optimal control theory), the dynamic programming approach (which is based on the principle of optimality and the Hamilton-Jacobi-Bellman (HJB) equation), and finally the Lagrangian approach – which is basically an extension of the Lagrangian technique of static optimization.

The classic approach for solving dynamic optimization problems is extremely attractive because the optimization problem is approached in its original form without any mathematical transformations, Feehery et al. (1997). It therefore follows that considering the large-scale nature of reservoir models, the classic approach i.e. optimal control theory, would be the best solution approach to production optimization problems. Interestingly, Brouwer (2004) posits that optimal control theory makes it possible to calculate the control strategy which forces the state from its initial value to its final value along a physically feasible trajectory. Regardless of this, the fact that the recovery factor of a reservoir cannot be formulated or cast into a simple quadratic objective function is a non-trivial challenge. This non-quadratic nature of the objective function is further exacerbated by the fact that the underlying dynamics of reservoirs are governed by equations that are linear in the control, but nonlinear in the state, Zandvliet (2008).

In mathematical optimization literature, the solution methods for optimal control problems involving nonlinear systems with non-quadratic objective functions can be broadly divided into direct methods (which involves the maximization or minimization of the objective function subject to prevailing constraints); and indirect methods whereby a solution that satisfies the maximum principle or related necessary conditions is sought.

Depending on how the underlying states equations are discretized, the solution techniques in the direct method approach can be simultaneous or sequential. The simultaneous technique involves a complete discretization of the states and control variables, and it is often achieved through collocation. A severe drawback that is often associated to this method is that it usually creates a multitude of additional variables which ultimately leads to large, unwieldy nonlinear programmes (NLPs) whose numerical solution is often difficult or impracticable. Thus, the use of simultaneous method requires awareness of the tradeoff between approximation and the optimization problem, Logson and Biegler (1992). The sequential method is usually achieved through control parameterization in which the control variable profiles are approximated by a series of basic functions in terms of a finite set of real parameters – see Teo et al. (1991). The parameters are then used as decision variables in a dynamic embedded nonlinear programme (NLP). Although this method has the advantage of yielding a relatively small NLP; however, its applicability in large-scale problems such as reservoir models suffers severe performance limitations.

The indirect methods often involve the reformulation of the optimal control problem using the Pontryagin's Maximum Principle (PMP). It is important to underscore that the so-called necessary condition is fundamental to the implementation of the indirect method. Essentially, the solution techniques under this method include the shooting and gradient-based techniques. With the aid of the PMP, the original optimal control problem is modified into a Hamiltonian function – see Bryson and Ho (1975) and Luenberger (1979). On one hand, this provides a closed-form expression for the optimal input as a function of the state and adjoint variables – which are essential in the shooting technique; while on the other hand, it provides derivative information which is used to generate the search directions that are required in the gradient-based technique. Usually, the state and adjoint equations are solved simultaneously, even though the boundary conditions for the state and adjoint equations are split – i.e. the initial conditions of the state equations and the terminal conditions of the adjoint equations are known. A major limitation associated with this approach is the stability of solutions, Murthy et al. (1980). Furthermore, unless good initial guesses of the adjoint states are available (which is rarely the case as the adjoint represent sensitivity functions), it is computationally expensive to find the optimal solution. It also suffers performance limitations if there are discontinuities in the adjoints, which is typical in the presence of state constraints. The gradient-based technique closely resembles the sequential approach of the direct formulation except that the derivatives are calculated using the necessary condition. It has been used in large-scale nonlinear systems, and it possesses an advantage to the effect that an initial guess of the decision variables is not detrimental to its convergence. For this reason, this method is the most viable technique – even though the computation of derivatives can be exceptionally demanding, not the least, in large-scale systems.

There are essentially three approaches in the computation of the derivatives of an objective function – the finite differences, the forward sensitivity equation, and the backward adjoint formulation. The finite difference approach has the drawback of requiring far too many objective function evaluations. In fact, a minimum of $2m+1$ objective function evaluations are required for m decision variables; and since each function evaluation entails a full reservoir simulation run, this limitation makes it impracticable in large-scale systems like reservoir models. With the forward sensitive equation approach, one simulation run of the model is required in addition to m sensitivity models. In large-scale systems, the obvious drawback of this approach is the computational memory requirement for the storage of the huge sensitivity information that is inevitably required for the computation of the objective function derivatives. Regardless of the number of decision variables, the computation of derivatives via the backward adjoint formulation requires two simulation runs only. Therefore, it is efficient and can be applied in large-scale nonlinear systems as evident in its vast application in the field of metrology and oceanography.

The use of gradient-based adjoint formulation in production optimization is not entirely new in the oil industry. Indeed, it was applied in Ramirez (1987) for enhanced oil recovery of a surfactant-flooded reservoir. Asheim (1987, 1988) applied the same principle to achieve increased sweep efficiency that led to 2–11% improvement in the NPV of different water-flooded reservoirs. In Zakirov et al. (1996), optimal well-rate allocation and improved waterflooding performance were achieved using a conjugate gradient-based optimal control technique where the derivatives are computed via adjoint formulation. However, many researchers in reservoir engineering acknowledge Sudaryanto and Yortos (2000, 2001) to be the first to systematically address the waterflooding problem using derivative-based adjoint model. By optimally allocating well rate in the injectors, the authors used optimal control theory to maximize the sweep efficiency of a multiple-injection single-producer system. They also investigated the shape of the optimal solutions, and were able to show that a bang-bang[9] control strategy was achievable in practice; however, the authors were silent on how and under what conditions the bang-bang control are achieved. This important question was addressed in Zandvliet et al. (2007), where the conditions under which bang-bang optimal solutions are attained were established. It was shown that bang-bang control has obvious advantage (over smooth solutions) in the sense that they can be implemented with simple on-off valves. This is important, because variable-setting valves are much more expensive than simple on-off ones, Zandvliet (2008).

At this juncture, it is important to note that most of the works mentioned above assumed constant production rates. However, Brouwer and Jansen (2004) underscored the fact that this was hardly common practice. They therefore investigated the problem further by comparing the constant production rate scenario with the constant bottom-hole-pressure (BHP) scenario. Both scenarios were argued to illustrate the two extremes of well operating conditions – thus, practical production planning need to take both into consideration. Brouwer and Jansen (2004) also focused on the production potential of using smart well control. They considered optimizing individual rates and valve settings in the waterflooding problem, and demonstrated the possibility to significantly increase recovery by using smart wells in reservoirs with heterogeneous permeability fields. There have been numerous applications ever since, and the problem has been extended to include the notoriously difficult issue of path constraints (state or output constraints) handling – i.e. bounds on reservoir pressure or amount of water produced in the production wells. In this regard, Montleau et al. (2006) and Kraaijevanger et al. (2007) employed a generalised reduced gradient (as proposed in Mehra and Davis (1972)), by using a control-input and state variable combination of well rates and bottom hole pressure (BHP). The apparent limitation of this method is the difficulty (or impracticability) of extending it to other control-input or state variable combination, such as reservoir saturation or total amount of water produced in the production wells. A feasible direction optimization in combination with state constraint "lumping" approach was applied

[9] In systems and control theory, a bang-bang control strategy often entails a feedback controller that switches between two – on and off – states.

by Sarma et al. (2008a); while a Lagrangian barrier method which uses Lagragian multipliers estimate to identify active constraints was applied by Suwartadi et al. (2010).

Despite these aforementioned developments in waterflooding production optimization, it is virtually next to impossible for the physical reservoir to attain the resultant optimal trajectory. In other words, the supposed optimal production profile is not attainable in reality.

2.3 Model Predictive Control

Model Predictive Control (MPC) encompasses a wide array of control algorithms that make explicit use of process data-driven models to obtain corrected control input by minimizing an objective function; and using a receding strategy such that after each sampling instance, the horizon is displaced towards the future. In other words, an optimal control problem is solved repeatedly at specific sampling instants of the current system state. The first part of the input is applied to the system until the next sampling instant, at which the optimal control problem for the new system state is solved again. Since the optimal control problem is solved at every sampling instant for one fixed initial condition, the solution is much easier to obtain than to derive a solution of the HJB partial differential equation (for all possible initial conditions) of the original optimal control problem, Findeisen et al. (2007).

Introduced as a control algorithm in the 1970s, MPC was at that time considered a major advancement in process control; and it is presently considered to be a matured technology in the chemical process and petrochemical industries where it originated. Over the years, it has grown to become one of the most popular and attractive control strategy for linear and near-linear processes; and its application has been well-established in the downstream sector of the oil industry. The main reason for its widespread acceptance is tied to the fact that MPC combines the principles of optimality with robustness of closed-loop control, while efficiently handling constraints on system inputs and outputs at the same time, Meum et al. (2008). Besides this, it was shown in Keerthi and Gilbert (1988) that the MPC algorithms provides an efficient way to obtain constrained optimal control while avoiding the notoriously difficult HJB partial differential equations.

Crucial to effective and successful implementation of MPC are validated linear empirical models, which are developed (usually via system identification) specifically for the type of dynamic process to be controlled, and efficient state observers for real-time model state estimation. Note that both the model and state estimation components are inter-twined in the sense that the performance of the observer is strongly dependent on the accuracy and validity of the identified model. In other words, the MPC performance is strongly dependent on how accurate the identified model describes the real process. Most process models, which are developed from limited quality and quantity of experimental observations, are often inaccurate; it is therefore, important to factor-in model uncertainty in the analysis and design of MPC controllers.

In upstream research, the first application of MPC was in the study of Saputelli et al. (2006). Using a 30-day control horizon, the authors implemented a moving horizon MPC where the performance measure – defined over 2200 days – is maximized by varying the production settings in the wells. A similar MPC strategy was also employed in Gildin and Wheeler (2008) to show that production is improved when compared to an uncontrolled reactive waterflooding scenario. In Rezapour (2009), a simple LTI-based MPC strategy was used to enhance the performance of waterflooding operation in homogenous 2D and heterogeneous 3D reservoir models. A similar approach was employed in van Essen et al. (2010) where an MPC-based proactive flooding strategy was implemented by introducing feedback into the control structure by using locally identified linear models.

It is important to note that the dynamics of the waterflooding process is highly nonlinear, thus, it is essential that such nonlinearities are reflected in the underlying models for MPC controllers deployed in this process. However, the identification of nonlinear models for MPC strategies generally leads to the more computationally demanding nonlinear model predictive control (NMPC) – the nonlinear extension of MPC. The high computational cost of NMPC stems from the fact that the often easy-to-solve quadratic programming (QP) control problem that results from linear MPC changes to a more challenging nonlinear programming (NLP) control problem when the underlying model is nonlinear. Thus, as much as we underscore the importance of employing nonlinear models (from the view point of accuracy); it is however, attractive, from the view-point of control, to employ simple linear models that possess easy-to-implement controller-design properties. Striking the appropriate balance between accuracy and control is still lacking in reservoir engineering applications where predictive control strategies has been deployed.

Across process engineering academic literature, a number of techniques have been developed for the sole purpose of circumventing the computationally demanding NLP that results from NMPC. Generally speaking, these techniques are based on the linearization of the nonlinear models. One of the techniques involves the use of a prediction horizon that is equal to one, Wang and Hendriksen (1994) and Haber et al. (1998). Thus, for a single input process (whose nonlinearity is a polynomial) with no input, state and output constraints, the optimal solution can be found by solving a polynomial equation in one variable. However, the requirements of a control horizon equal to one and the absence of constraints are very restrictive for practical purposes – see Bloemen (2002). Another possibility is the sequential quadratic programming approach. This technique involves the linearization of the problem around a control sequence, obtained from previous iterations. In other words, the supposed NLP problem becomes a quadratic programming problem. Note that the linearization error (the difference between the calculated optimal input sequence and the output sequence around which the predictions are linearized) can be decreased by using several iterations within one sampling interval as shown by Gerksic et al. (2000). Again, it is important to underline that this approach is vulnerable to local optima convergence as well as slow convergence when trying to reduce the linearization error. In another technique, the structural property of a specific class of nonlinear models known as Hammerstein-Wiener models are exploited in circumventing the computational demand of NLPs. This method which involves the inversion of the static nonlinearity in these nonlinear models was employed in the studies of Norquay et al. (1998) and Kouvaritakis et al. (2000). It effectively removes the nonlinearity in the control problem; hence, provided there are no constraints, a linear MPC technique can be applied to the remaining linear block. A major drawback in inverting the static nonlinearity is that the effect of the nonlinearity on the input-output behavior of the process is lost, and therefore, not taken into account by the controller. In Bloemen (2002), an algorithm which takes into account the effect of the input and output nonlinearities of Hammerstein-Wiener model, while still retaining a convex optimization problem was presented. This was achieved through the transformation of the static nonlinearities into some form of polytopic descriptions. The nonlinear model is therefore, represented as an uncertain linear model in which a robust linear MPC strategy can be applied. Therefore, it is safe to say that there exist ample techniques that could be employed to plug the gap between accuracy and control in reservoir engineering applications where these control strategies are deployed.

2.3.1 System Identification

In applied mathematics, and indeed, systems theory, the building of mathematical models from measured experimental data of dynamical systems is generally referred to as system identification. It is an important sub-set of statistics; hence, many identification techniques as well as the tools for analyzing their inherent properties are very much rooted in statistical theory. A review of the literature suggests that research in this area of system theory can be traced back to the mid-1960s – perhaps to the works of Åström and Bohlin (1965), and Ho and Kalman (1965). In the former, the very fundamental principle behind the *maximum likelihood* methods for parametric input-output models (which later became known as *prediction error identification*) was presented; while the latter presented the first known solution of *state-space realization theory*, which subsequently led to *stochastic realization* and finally to the birth of *subspace identification* methods. Ever since, system identification has experienced tremendous growth, which can rightly be said to have been spurred by the enormous interest in model-based control strategies as well as the advancements in optimal control theory by Rudolf Kalman and his peers.

Evidently, building mathematical models from first principles material and energy conservations often require considerable expert knowledge, and can be expensive in terms of man-hour requirements. The resulting models are referred to as white-box, and are often characterized by high complexity which makes them very unsuitable for real-time model-based control applications. In other words, physics-based white-box models are able to capture process behavior over a wide range of operation; they are however, not suitable in applications where small computation times are crucial. In contrast, systems identification provides a suitable alternative to these so-called first principle models. It results in simple compact models that can be used in real-time model-based controllers; it also provides an enablement for the construction of process models that are able to reproduce process data and therefore exhibit accurate description of the local behavior of the system. To this end, identified models are often used in control design, in the adjustment of free parameters in first principle models, in fault detection techniques and process monitoring, Larimore (1997). Based on the physical interpretation of the parameters of the identified model, the resulting model may be referred to as black-box (if there is no physical interpretation of parameters), or grey-box (i.e. if there is little meaning of model parameter).

System identification of linear time-invariant (LTI) models can be broadly classified as parametric or non-parametric. The parametric approach includes the prediction error methods – which have strong relations to maximum likelihood estimation, the output error methods – where the governing criterion is to minimize the error of the output measurements, and the subspace methods – which intersect system theory, numerical linear algebra and geometry. The idea behind SubID is based on the possibility of retrieving certain subspaces which are related to the system state-space matrices, according to block-Hankel matrices, structured from input output data, Overschee and de Moor (1996) and Verhaegen and Verdult (2007). Non-parametric approach to system identification includes frequency domain identification techniques such as the correlation and spectral analysis methods, Brillinger (1981) and Pintelon and Schouken (2012). By and large, the theory of systems identification LTI systems is considered mature, Ljung (1987, 1999). One reason attributed to this fact is the simplicity of modeling and implementation of black-box linear models in several control applications.

However, it is important to note that the dynamics of most industrial processes, for example, the waterflooding process, are governed by highly nonlinear equations; therefore, the use of data-driven LTI models for model-based control strategy in such nonlinear dynamical processes suffer severe performance limitations. Thus, while LTI models can be sufficient for the purpose of control in some nonlinear dynamical systems, they usually come short in situation where the underlying system is highly nonlinear, or where the dynamics of the system varies a lot for different operating point, Tayamon (2012). To this end, nonlinear system identification is becoming popular, and the identification of nonlinear black-box model has received much attention in the past decade.

The resulting models from nonlinear system identification include the Nonlinear AutoRegressive eXogenous input (NARX) model, Nonlinear AutoRegressive Moving Average eXogenous input (NARMAX) model, Volterra model; which are considered to be the nonlinear extension of the popular AutoRegressive eXogenous input (ARX), AutoRegressive Moving Average eXogenous input (ARMAX) and Finite Impulse Response (FIR) models respectively. A special sub-group of nonlinear models often referred to as block-oriented class include the Hammerstein, Wiener and Hammerstein-Wiener model structures. This family of nonlinear models consist of linear dynamic and nonlinear static blocks connected in series; it is important to note that the nomenclature of the resulting model structure is solely based on the relative position of the linear dynamic block with respect to the nonlinear static block. If the linear dynamic block is preceded by a static input nonlinear block, the model is referred to as a Hammerstein; however, if the linear dynamic block is followed by the static output nonlinear block, the model is referred to as a Wiener. Because both Hammerstein and Wiener models are basically composed of the same components connected in reverse order, one is in every sense the dual of the other. When the linear dynamic block of the model is sandwiched in between the static nonlinear blocks, we obtain a Hammerstein-Wiener model.

Generally, the various system identification techniques often involve the following steps:

1. An appropriate experiment is designed and executed on the system such that those properties that are deemed to be relevant for the model are excited.

2. A set of candidate models or model structure has to be chosen which consists usually of a dynamic model that connects the excited inputs with the measured outputs and contains unknown parameters or free variables on various locations inside the model.

3. To determine the best model in the set, some criterion function is chosen that measures the distance between model predictions and the process measurements as a function of the free variables. By some mathematical optimization procedure this cost function is minimized to find optimal parameter values.

4. The last step is the model validation step. This aims to assess whether the model is "good enough" for its purpose. Common validation tools are *residual analysis* and the so-called *cross-validation* – where the identified model is simulated using new data and the output compared to the measured output.

Note that if the initial model fails to pass the validation tests, some or all of the above steps have to be repeated iteratively, until a model that passes the validation tests is found.

2.3.2 State Estimation

Usually, current estimate of the model state is a *sine-quo-non* for effective implementation of MPC strategies. In other words, since MPC is based on a mathematical model that represents or describes a physical process, the model state is necessary for prediction and therefore has to be known. In some cases it is accessible through measurements, but in other cases, a state observer must be explicitly included in the control loop for effective state estimation. This is exactly the case for most MPC controllers, and the idea is simple – at each sampling time, the model is updated from new measurements and state variable estimates. The manipulated variable are calculated over a finite prediction horizon with respect to some defined cost function, the manipulated variables for the subsequent prediction horizon are implemented and then the prediction horizon is shifted by one sampling time into the future for the previous steps to be repeated.

The choice of an appropriate observer can ultimately influence the performance of the controller; therefore, it is essential to use suitable state estimation algorithm or observer while bearing in mind the uniqueness of the underlying process and process model. In linear systems, the most commonly used state estimation technique is the Kalman Filter, Kalman (1960). By minimizing the mean square estimation error, Kalman Filters are capable of giving estimate of the state models of linear systems whose only source of uncertainty are states and measurements Gaussian noise. It is based on the assumption that the initial condition of the system and the measurement noise processes are independent of each other; thus, the measurements can be sequentially assimilated into the system. Another commonly used observer for state estimation in MPC strategies is the simple Luenberger observer – see Luenberger (1971), and Alessandri and Coletta (2001). The wide usage of Kalman Filters is centered on its optimality and low computational cost, as optimal estimate and corresponding error covariance is computed recursively via simple matrix multiplications.

However, it should be noted that if the underlying model is nonlinear, and the conditional probability of uncertain parameters is non-Gaussian, the Kalman Filter suffers severe performance limitations. For the sake of completeness, it is important to state that a straightforward extension of the Kalman Filter referred to as Extended Kalman filter (EKF) was developed to address this issue. The basic idea of the EKF is to linearize and approximate at each time step the nonlinear system as a time-varying system affine in the variables to be estimated, and then subsequently apply the Kalman Filter. For more review on EKF as well as other modifications of the Kalman Filters (such as Unscented Kalman Filters and Ensemble Kalman Filter), see sub-section 2.4.2.

2.4 History Matching

History matching is the process of reconditioning a working reservoir simulation model to available field data. Its primary purpose is to improve the predictive power of reservoir models as well as to reinforce the robustness of the development and production decisions they serve. The importance of history matching in reservoir engineering stems from the generally accepted fact that any model that can realistically predict unknown future production profile, should also be able to reproduce known historical production data. Thus, the fundamental idea behind history matching is simple – a reservoir simulation model that can capture the past behaviour of the physical reservoir is most likely to make robust and accurate predictions.

Mathematically, history matching is an ill-posed inverse problem; it therefore, has no unique solution. In other words, available field data may yield good matches or multiple realizations that match the same physical reservoir. These 'purportedly' history-matched models are likely to exhibit different future production behaviours; therefore, they can be employed as an important tool for reservoir uncertainty prediction and quantification. Again, since history matching involves finding reservoir model parameters from measured data, it is invariably a system identification problem. And since a number of combinations of different solution (reservoir parameter values) can equally yield good matches of the same physical reservoir, it can be said that reservoir model parameters are not uniquely identifiable.

Over the years, a number of history matching techniques have been developed, implemented and reported across the reservoir engineering academic literature. Generally, the problem is often approached by defining an objective function (which is usually the weighted squared difference between the predicted and the observed output) and minimizing the function over all possible parameter values, while obeying the constraints imposed by the reservoir model. However, the ill-posed nature of the mathematical problem inevitably leads to a number of combinations of different reservoir parameters that yield same minimum value of the objective function. To limit the solution space, the problem is often 'regularized' (i.e. made less ill-posed and more well-posed) by adding a data-independent term (prior knowledge) to the objective function; thereby allowing only models that are 'proximal' in some pre-defined sense to the underlying reservoir model to be selected. Thus, the process of history matching is notoriously demanding and time-consuming.

The solution techniques or approaches to history matching can be divided into variational, sequential, parameterization and metaheuristic. These approaches are reviewed in the following sub-sections, with the view of highlighting their advantages and underscoring the inherent limitations in them.

2.4.1 Variational Approach to History Matching

In the variational approach, the history matching problem is treated as an optimal control problem (the unknown model parameters are considered the control variable) where the objective function is minimized subject to prescribed model constraints. Usually, the problem is solved with the aid of a gradient-based optimization algorithm; where the gradients of the objective function with respect to the model parameters are computed via an adjoint or co-state formulation. Like in the production settings optimization problem (see section 2.2), the necessary conditions for optimality are essential for the computation of the gradients. The use of adjoint-based gradient methods in the history matching problem can be traced as far back as the 1970s. Slater and Durrer (1970) employed a gradient-based technique to solved history matching problems, and Chen et al. (1974) applied an optimal control approach (adjoint-based technique) in the characterization of a real reservoir. These works were closely followed by Chavent et al. (1975), Wasserman et al. (1975), Watson et al. (1980), Lee and Seinfeld (1987), and Yang et al. (1988). In the last decade, Li et al. (2003) and Oliver et al. (2008) applied the adjoint-based gradient method in the history matching of 3-dimensional multiphase reservoir models, as well as reducing the uncertainty in the estimates of reservoir parameters. The advantage of this method lies in its computational efficiency as the gradients are computed using the so-called adjoint model which allows the computation of all sensitivities (irrespective of the number of parameters) in two simulation runs – one forward and the other backward in time. However, as highlighted in section 2.1, the implementation of partial derivatives (gradients) of the system requires detailed knowledge of the simulation source code. Thus, the use of adjoint-based gradient methods is invasive with respect to the flow simulator; and therefore, the implementation can be challenging. Another gradient-based algorithm that has been employed in the history matching problem domain is the Gauss-Newton method. This basically calculates the first and approximates the second derivative of all measurement predictions with respect to parameters being estimated. Although similar to the adjoint-based gradient methods, the Gauss-Newton method differ in the sense that more than one adjoint model simulations are required for the computation of the sensitivities, Tarantola (1987) and Douma (2009).

2.4.2 Sequential Approach to History Matching

The sequential approach to history matching is based on the well-known Kalman Filters which have been widely deployed in linear models where the only source of uncertainty is Gaussian noise on the states and parameters. It is based on the assumption that the initial condition of the system and the measurement noise processes are independent of each other; thus, the measurements can be sequentially assimilated into the system, Kalman (1960). However, if the underlying model is nonlinear, and the conditional probability of uncertain parameters is non-Gaussian, the Kalman Filter suffers severe limitations. This is because linear models with Gaussian distribution of unknown states and parameters can be completely characterized by the first and second moments (i.e. the mean and the covariance matrix), whereas nonlinear models with non-Gaussian distribution of unknown parameters would normally be characterized by an infinite number of moments. To this end, a number of modifications of the Kalman Filters have been developed for nonlinear models. These include the Extended Kalman Filters (EKF), the Unscented Kalman Filters (UKF) and the Ensemble Kalman Filters (EnKF).

The EKF was presented in the study of Jazwinski (1970). In this technique, the estimate and corresponding error covariance are recursively computed via linearization of the underlying nonlinear model equations. Owing to the huge number of to-be-estimated states and parameters in a typical reservoir model, and the enormous computational requirement that is associated with computing the covariance matrices of the state variables and parameters each time new observations become available; the EKF is not an effective technique for large-scale nonlinear systems like reservoir models.

The first application of UKF was presented in Julier and Uhlmann (1997); and it is more capable of dealing with nonlinearities. It uses a deterministic sampling technique called *unscented transform* to select a minimal set of sample points (referred to as sigma points) around the mean, and these sigma points are subsequently propagated through nonlinear functions, from which the mean and covariance of the estimate are then computed. However, it requires two simulations for each to-be-estimated element in order to compute an estimate and corresponding error covariance, Zandvliet (2008).

The EnKF is much more effective for large-scale nonlinear systems. It was developed by Evensen (1994) for applications in oceanography, and is fundamentally a Monte-Carlo or stochastic (as against the deterministic EKF and UKF) approach of computing the error covariance through an ensemble of possible realizations. It appears that the first application of EnKF in history matching was by Lorentzen et al. (2003), where it was employed in the calibration of the parameters for a two-phase reservoir flow model; and enhanced well pressure behavior predictions were obtained by applying the full-scaled experimental data. This was later followed by Nævdal et al. (2005) – where the permeability field of a synthetic 2D model was estimated, Gu and Oliver (2005) – where it was used to estimate permeability and porosity field of the Production forecasting with UNcertainty Quantification (PUNQ–S3) model, Rommelse et al. (2006) – where the performance of the EnKF was compared with the so-called representer method, and Skjervheim et al. (2007) – where it was used in matching time-lapse (4D) seismic data from a North Sea oil field. A good review of EnKF in reservoir engineering is contained in Aanonsen et al. (2009), and rich information on current trends is available in Emerick and Reynolds (2012). Relative to the performance of other versions or modifications of the Kalman Filters for nonlinear models, the popularity of EnKF stems from its very simple conceptual formulation and easy-implementation property. The ensemble is a reflection of the uncertainty and probability distribution of the estimated variables. It is noted that the computational requirement associated with EnKF is directly proportional to the ensemble size; therefore, the size of the ensemble should be as small as possible, but not too small a size that would result in inaccurate or unstable results. Usually, the choice of an ensemble size of less than 100 realizations is quite common in history matching and other reservoir engineering applications.

2.4.3 Parameterization Approach to History Matching

Another approach to history matching is the parameterization methods. This involves all the techniques that seek to reduce or re-parameterize the huge number of reservoir unknown parameters. Thus, the aim is to express reservoir model parameters by a fewer number of new variables while preserving important geological variability. The parameterization techniques include zonation, pilot point, principal component analysis (PCA), discrete cosine transform (DCT) and the representer method.

In the zonation method, the reservoir is divided into a manageable number of zones where the properties are assumed to be uniform, and model parameters are adjusted for each zone from dynamic data measurements. It was introduced in the study of Jacquard and Jain (1965), and this was quickly followed by Jahns (1966). The gradzone method – Bissell (1994), and the adaptive multi-scale method – Grimstad and Mannseth (2000) draw inspiration from the zonation method. The pilot point method involves choosing locations (or pilot point) of the reservoir model where the parameters are to be adjusted and subsequently interpolated to neighboring points by kriging. It was proposed by de Marsily et al. (1984), and was applied in Cartes and de Marsily (1991), and RamaRao et al. (1995).

The PCA (also known as Karhunen-Loéve Expansion) are differentiable parameterization techniques that have been employed in history matching. In the standard PCA approach, the model grid-block parameter of interest (e.g. permeability and porosity) is expressed as a linear combination of some deterministic basis functions weighted by uncorrelated random coefficients. More details of this approach is available in Gavalas et al. (1976), Reynolds et al. (1996) and Oliver (1996) where it was used for reservoir model permeability and porosity parameterization. The kernel PCA method is basically an extension of the standard PCA. This technique preserves higher order statistics (unlike the standard PCA which preserves only second order moment or covariance matrix), and can therefore, be deployed in the parameterization of complex geological models with non-Gaussian distributed fields. Details of this approach is available in Sarma et al. (2007, 2008b).

The DCT parameterization approach to history matching draws inspiration from image processing. In this approach the permeability field is expressed as a linear combination of some predefined basis functions (equal to the number of permeability values) weighted by uncorrelated random coefficients. Thus, the permeability field is expanded into predefined basis functions that do not depend on the covariance matrix and do not need to be estimated from data as shown in Jafarpour and McLaughlin (2007a). Efficient history matching was achieved in Jafarpour and McLaughlin (2007b) by the application of a hybrid EnKF-DCT algorithm. The representer technique to history matching is inspired from meteorology and oceanography, Bennett et al. (1996) and Bennett (2002). This method allows for reduction of the number of unknown states/parameters to the number of measurements used in the inversion process, and this is achieved by expanding the parameter field into a finite set of basis functions called representers. The only unknowns are the expansion coefficients that need to be adjusted to match the available data. In this way the number of independent estimation parameters is reduced to the number of measurements used in the assimilation (as there is one representer defined per each measurement) while still providing a solution to a full inverse problem. It has been employed in Rommelse et al. (2006) and Przybysz-Jarnut (2010) for the estimation of the permeability in reservoir models.

2.4.4 Metaheuristic Approach to History Matching

Metaheuristics, which include evolutionary and swarm intelligence algorithms have become very popular in the history matching problem domain. Sen et al. (1995) applied and compared the duo of simulated annealing (SA) and genetic algorithm (GA), while Romero et al. (2000a, 2000b) employed GA in history matching problems of different complexities. Since then, other metaheuristic algorithms have been deployed in various history matching problems. Wang and Buckley (2006) and Hajizadeh et al. (2011) applied differential evolution (DE), Schulze–Riegert and Ghedan (2007) and Schulze–Riegert et al. (2009) employed evolutionary strategies (ES), while Mohamed et al. (2010) used a PCA-based model parameterization to apply particle swarm optimization (PSO) in the history matching of the Brugge field. Of course, it can be said that the popularity of metaheuristic algorithms in history matching stems from their simplicity, parallel implementation capabilities, and the fact that they do not require any gradient information from the optimization problem. The algorithms often use the objective function value to determine new search steps; and can therefore, be used in cases where gradient information are unavailable, or where traditional techniques fail due to significant nonlinearities or discontinuities in the search space.

2.4.5 Other Approaches to History Matching

It is important to note that there are other methods for matching observed and predicted data that have been mentioned in the reservoir engineering literature. These methods are not discussed in details, they are only mentioned for the sake of completeness; therefore, the interested reader is referred to the applicable references. The streamline simulator methods – Vasco and Datta-Gupta (1997), Wen et al. (1998), Vasco et al. (1999), Wu and Datta-Gupta

(2002) and Cheng et al. (2004) is a computationally efficient approach to history matching. It is basically based on rapid inversion of multiphase production data. Usually, the high permeability flow paths in the reservoir model (or parts of the reservoir model with greatest influence on flow to production wells) are identified; and the parameters (permeability and porosity) in those parts are adjusted to match production data. Other methods worth mentioning are the gradual deformation method (GDM) and the probability perturbation method (PPM) – both of which are iterative and stochastic algorithms. The GDM is an iterative geostatistical technique that perturbs a realization from a few parameters (referred to as deformation parameters), while preserving the spatial variability or geostatistical constraints. It was proposed in Roggero and Hu (1998), and was applied in Gallo and Ravalec-Dupin (2000), Roggero et al. (2002) and Hu and Jenni (2005). The PPM approach employs a training image to approximate the multiple point statistics of the facies distribution, while maintaining the prevailing geostatistical constraints, Caers (2003).

2.5 Summary

In this chapter, we undertook an in-depth review of the literature as it relates to the major problems considered in this thesis. We highlighted the strengths and weaknesses of various optimization, control and estimation techniques that have been applied in reservoir engineering and other related domains. In the well placement optimization problem, it was established that most algorithms that have been employed in the domain can be broadly categorized into local deterministic and global stochastic algorithms. We further underscored the fact that although the local deterministic algorithms such as gradient-based algorithms (where the gradients are computed via adjoint formulations) are computationally less demanding than the gradient-free global stochastic methods; they are usually difficult to implement because they often require full access to the flow simulator source code as well as detailed knowledge of the code. It was pointed out that the gradient-free stochastic algorithms possess easy–to–implement properties which arise from the fact that they are generally non-invasive with respect to the flow simulator. It was further highlighted that their inherent weakness (high computational cost of objective function evaluation) can be substantially overcome by deploying the implementation over multiple parallel processors. In production optimization and control, we established the fact that adjoint-based techniques appear to be the best option in the optimization of production settings. This has been significantly helped by the fact that most commercial and in-house reservoir simulators now have built-in adjoint functionalities for the purpose of effective production optimization tasks. However, owing to reservoir model uncertainties, it is virtually next to impossible for the physical reservoir to attain the optimal trajectory resulting from production optimization. In other words, the supposed optimal production profile is not always attainable in practice. To this end, a predictive control loop that is based on a simple data-driven model is coupled to the production optimization problem, for the sole purpose of tracking the optimal trajectory that results from the production optimization loop. The type of data-driven model upon which the predictive control strategy is based is of crucial importance. Certainly, the use of nonlinear models enhances accuracy; however, simple linear models that possess easy-to-implement controller-design properties are attractive from the control point of view. Striking the appropriate balance between accuracy and control is still lacking in reservoir engineering applications where predictive control strategies are deployed. Finally, we ended the chapter by reviewing the various approaches to history matching as reported in the literature.

CHAPTER 3

For a man's life from birth to death was a series of transition rites which brought him nearer and nearer to his ancestors – Chinua Achebe

RESERVOIR MODELLING

This chapter provides the fundamental mathematical concepts upon which the remaining chapters are built. We present the notations and the general laws governing the equations that describe fluid flow behaviour in porous media. We will also provide the reservoir model and constraints within our defined framework.

3.1 Flow In Porous Medium

The three main ingredients in modelling fluid flow in porous medium are mass conservation for each of the phases, an empirically determined constitutive equation relating the average mass flux of each phase to the corresponding fluid potential gradient, and the equation of state (thermodynamic or compressibility equation). The second equation which accounts for the conservation of momentum is governed by the Navier-Stokes equation, and is basically an extension of the Darcy's law. In other words, the governing equations describing flow in a porous medium are based on mass conservation, momentum conservation and the thermodynamic equation of state. For more details on the mathematics and modelling of flow in porous media, the interested reader is referred to Peaceman (1977, 1978), Aziz and Settari (1979), Ahmed (2001), Chen et al. (2006), Aarnes et al. (2007), Chen (2007) and Jansen (2012). This chapter is largely based on the aforementioned publications.

3.2 Single-Phase Flow Formulation

The mathematical formulation of flow in porous media is firmly based on a number of fundamental principles. These include:

- Mass balance (i.e. mass accumulated = mass inflow – mass outflow)
- Thermodynamic or compressibility equation (accounts for changes in fluid and rock properties as a result of changes in pressure)
- Darcy's law (momentum conservation)
- Initial and boundary conditions

Consider the porous medium domain **Ω** represented by the cube in Figure 3.1. Assuming that its faces are parallel to the coordinate axes, the centroid given by (x, y, z), and that the dimension in the *x*-, *y*- and *z*- coordinate directions are Δx, Δy and Δz respectively. If the spatial variable is given by $\mathbf{x} = (x, y, z)$, the time variable by t, the porosity of the medium is given by ϕ, the density of the fluid is represented by ρ, the Darcy velocity is given by $u = (u_x, u_y, u_z)$, and q is the external sources and sinks.

Since the mass flow per unit area per unit time (mass flux) is given by ρu_i (where *i* = *x*, *y*, *z*); the mass inflow across a surface at position $x - \Delta x/2$ per unit time is given by:

$$(\rho u_x)_{x-\Delta x/2,y,z}\,\Delta y \Delta z \tag{3.1}$$

and by extension, the mass outflow at a point $x + \Delta x/2$ which is directly opposite and in the same coordinate as $x - \Delta x/2$ is given by:

$$(\rho u_x)_{x+\Delta x/2,y,z}\,\Delta y \Delta z \tag{3.2}$$

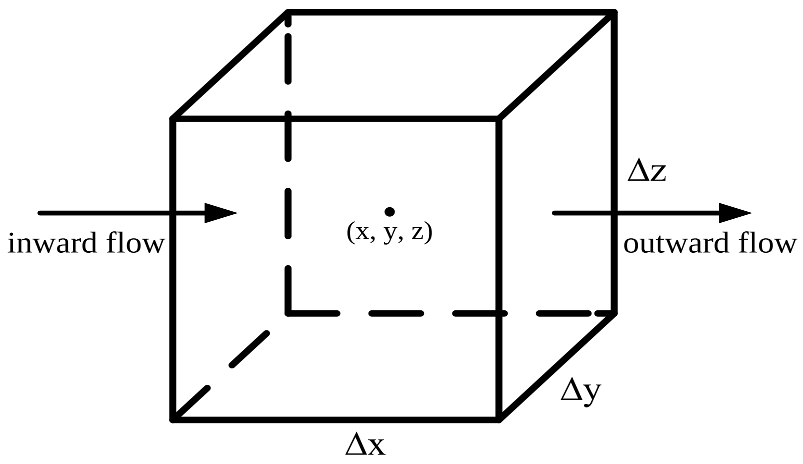


Figure 3.1: A porous medium domain Ω in 3-dimensional space courtesy Chen (2007)

By the same token, the mass inflow and outflow across the surfaces in the *y*- and *z*-coordinates are respectively given by:

$$(\rho u_y)_{x,y-\Delta y/2,z}\ \Delta x\Delta z \tag{3.3}$$

$$(\rho u_y)_{x,y+\Delta y/2,z}\ \Delta x\Delta z \tag{3.4}$$

$$(\rho u_z)_{x,y,z-\Delta z/2}\ \Delta x\Delta y \tag{3.5}$$

$$(\rho u_z)_{x,y,z+\Delta z/2}\ \Delta x\Delta y \tag{3.6}$$

Mass accumulation due to compressibility per unit time and sink are respectively given by:

$$\frac{\partial}{\partial t}(\phi\rho)\Delta x\Delta y\Delta z \tag{3.7}$$

$$-q\Delta x\Delta y\Delta z \tag{3.8}$$

For mass balance, the difference between mass inflow and outflow must equal the sum of mass accumulation in the volume; i.e.

$$\begin{aligned}&\left((\rho u_x)_{x-\Delta x/2,y,z} - (\rho u_x)_{x+\Delta x/2,y,z}\right)\Delta y\Delta z + \left((\rho u_y)_{x,y-\Delta x/2,z} - (\rho u_y)_{x,y+\Delta x/2,z}\right)\Delta x\Delta z\\ &+\left((\rho u_z)_{x,y,z-\Delta x/2} - (\rho u_z)_{x,y,z+\Delta x/2}\right)\Delta x\Delta y = \left(\frac{\partial(\phi\rho)}{\partial t} - q\right)\Delta x\Delta y\Delta z\end{aligned} \tag{3.9}$$

Now, dividing both sides of Eq. 3.9 by the bulk volume $(\Delta x\Delta y\Delta z)$, we obtain:

$$\begin{aligned}&-\left(\frac{(\rho u_x)_{x+\Delta x/2,y,z} - (\rho u_x)_{x-\Delta x/2,y,z}}{\Delta x}\right) - \left(\frac{(\rho u_y)_{x,y+\Delta x/2,z} - (\rho u_y)_{x,y-\Delta x/2,z}}{\Delta y}\right)\\ &-\left(\frac{(\rho u_z)_{x,y,z+\Delta x/2} - (\rho u_z)_{x,y,z-\Delta x/2}}{\Delta z}\right) = \left(\frac{\partial(\phi\rho)}{\partial t} - q\right)\end{aligned} \tag{3.10}$$

As $\Delta i \to 0,\ i = x, y, z;$ we obtain a continuity equation describing mass conservation:

$$\frac{\partial}{\partial t}(\phi\rho) = -\nabla.(\rho u) + q \tag{3.11}$$

where $\nabla.$ is the divergence operator, which is given by: $\nabla.u = \frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}$.

The conservation of momentum is governed by the Navier-Stokes equation, but is usually modeled for low velocity filtration through porous media by the semi-empirical Darcy's law; which states that the total volumetric flow rate *Q* of a fluid through a porous medium is dependent on the cross-sectional area of the medium *A*, the pressure gradient along the medium *p* and the fluid frictional property or viscosity μ.

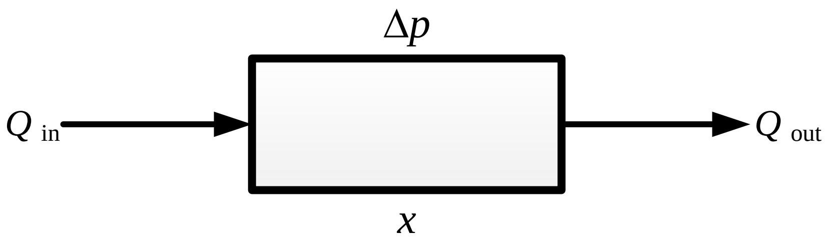


Figure 3.2: Illustration of Darcy's law

Accordingly, this can be expressed as $Q = -k \frac{A}{\mu} \frac{\partial p}{\partial x}$, where the constant *k* is referred to as the permeability of the medium. The permeability of the medium represents the ability (or otherwise) of the medium to allow the flow of fluid. It is analogous to the electric resistance in Ohm's law of electrical conduction and the heat conductivity tensor in Fourier's law of thermal conduction. Usually, *k* is a diagonal tensor, and depending on the geometry of the porous medium, *k* is given by:

1–D: $k = k_x$, 2–D: $k = \begin{pmatrix} k_x & 0 \\ 0 & k_y \end{pmatrix} = \text{diag}(k_x, k_y)$, 3–D: $k = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{pmatrix} = \text{diag}(k_x, k_y, k_z)$;

and for a 2-D and 3-D isotropic medium, $(k_x = k_y)$ and $(k_x = k_y = k_z)$ respectively.

Since $u = Q/A$, we can therefore express momentum conservation (Darcy's law) as:

$$u = -\frac{k}{\mu}(\nabla p - \rho g \nabla z) \tag{3.12}$$

where *g* is the magnitude of acceleration due to gravity, *z* is the depth, and ∇ is a gradient operator.

Note that whereas there is only one force property in both electrical (potential difference) and thermal (heat) conduction, there are two driving forces in porous media flow – gravity and the pressure gradient. However, if we assume that there are no gravitational effects, and that permeability is isotropic; then we need only the reservoir field pressure as our primary unknown, and this can be solved by substituting the Darcy's law (3.12) into the continuity equation (3.11). This leads to a linear equation in the form:

$$\frac{\partial}{\partial t}(\phi\rho) = \nabla . \left(\frac{\rho}{\mu} k \nabla p \right) + q \qquad (3.13)$$

3.3 Two-Phase Flow Formulation

It is important to underline that the working models used throughout this work is based on two-phase oil (*o*) and water (*w*) flow, where the oil is the non-wetting phase and water the wetting phase. In these flow situations, the macroscopic conservation laws that were derived for a single-phase fluid flow in porous media are augmented by empirical material-dependent constitutive relationships describing saturation and relative fluid permeability, which represents a reduction in the permeability of one phase due to its interference with the other phase. Importantly, we underline that the flow model formulation are based on the following underlying assumptions:

- the reservoir contains two-phase (oil and water) isothermal weakly compressible immiscible fluid
- there are no aquifer
- there are no capillary pressure effects
- there are no gravity effects in the formulation, although the effects of gravity is taken into account in the applications that follow in later chapters
- besides the sink and source terms, there are no flow across the reservoir geometry, i.e. Neumann boundary condition
- reservoir pressures are above the bubble point pressure of the oil phase; in other words, the reservoir is under-saturated and there is no mass transfer between the oil and gas phases

The mass and momentum conservation for the phases are given by:

$$\frac{\partial}{\partial t}(\phi\rho_\alpha S_\alpha) = -\nabla\cdot(\rho_\alpha u_\alpha) + q_\alpha, \qquad \alpha \in \{o, w\} \tag{3.14}$$

$$u_\alpha = -k\frac{k_{r\alpha}}{\mu_\alpha}\nabla p_\alpha, \qquad \alpha \in \{o, w\} \tag{3.15}$$

where t is time, $\rho_\alpha, S_\alpha, u_\alpha, \mu_\alpha$ and p_α are respectively density, saturation, velocity, viscosity and pressure of phase α; while $\phi, k, k_{r\alpha}, \nabla.$ and ∇ are porosity, permeability of the medium, relative permeability of the individual phase α, a divergence operator and a gradient operator respectively. Because of the assumed no-flow boundary condition across the reservoir geometry over which Eq. 3.14 is defined, q_α represents the sink/source terms of phase α.

Substituting the momentum conservation equation (3.15) into the mass conservation equation (3.14); we obtain two flow equations with four unknowns $(S_o, S_w, p_o$ and $p_w)$, viz:

$$\frac{\partial}{\partial t}(\phi\rho_o S_o) = \nabla\cdot\left(k\frac{k_{ro}}{\mu_o}\rho_o\nabla p_o\right) + q_o \tag{3.16}$$

$$\frac{\partial}{\partial t}(\phi\rho_w S_w) = \nabla\cdot\left(k\frac{k_{rw}}{\mu_w}\rho_w\nabla p_w\right) + q_w \tag{3.17}$$

Consequently, two additional equations are required for the complete description of the model. These equations are the closure equation, which requires that the oil and water phases jointly fill the void space; and the capillary pressure equation, which expresses the pressure due to interfacial forces across the interface between immiscible fluids.

$$S_o + S_w = 1 \tag{3.18}$$

$$p_{cow} = p_o - p_w \tag{3.19}$$

Recall that one of our key assumptions is the absence of capillary pressure effects; and on that evidence, (3.19) becomes $p_o = p_w$. Substituting $p_o = p_w = p$ and the closure equation (3.18) into the flow equations lead to:

$$\frac{\partial}{\partial t}(\phi\rho_o[1-S]) = \nabla\cdot\left(k\frac{k_{ro}}{\mu_o}\rho_o\nabla p\right) + q_o \tag{3.20}$$

$$\frac{\partial}{\partial t}\left(\phi\rho_w S\right) = \nabla\cdot\left(k\frac{k_{rw}}{\mu_w}\rho_w\nabla p\right) + q_w \tag{3.21}$$

where S (water saturation) and p (oil pressure) are dynamic state variables.

Note that flow equations parameters such as porosity, permeability, phase density and phase viscosities are generally dependent on the pressure (p); while the relative permeabilities are strongly dependent on saturation. Since the pressure dependency of permeability and phase viscosities are very weak (negligible); they are often ignored by treating the parameters as pressure independent; while the other pressure-dependent parameters (porosity and phase density) are expressed in the following isothermal relationships:

$$c_\alpha\left(p\right) = \frac{1}{\rho_\alpha}\frac{\partial\rho_\alpha}{\partial p}\bigg|_T, \qquad \alpha\in\{o,w\} \tag{3.22}$$

$$c_r\left(p\right) = \frac{1}{\phi}\frac{\partial\phi}{\partial p}\bigg|_T \tag{3.23}$$

where c_α and c_r are compressibility of the phase α and rock compressibility respectively.

Therefore, (3.20) – (3.23) results to the following nonlinear equations:

$$\phi\left(1-S\right)\left(c_o+c_r\right)\frac{\partial p}{\partial t} = \nabla\cdot\left(k\frac{k_{ro}}{\mu_o}\nabla p\right) + q_o \tag{3.24}$$

$$\phi S\left(c_w+c_r\right)\frac{\partial p}{\partial t} = \nabla\cdot\left(k\frac{k_{rw}}{\mu_w}\nabla p\right) + q_w \tag{3.25}$$

3.3.1 Relative Permeabilities

The concept of relative permeability is consequent upon the fact that the flow ability of one phase at any location is dependent on the prevailing environment at that location. In other words, the permeability of one phase is dependent on the saturation of the other phase at that location. Therefore, besides the permeability of the porous medium, relative permeability represents an additional resistance to flow of a phase – it is caused by the presence and interference of the other phase. It is important to underline that the relative permeabilities are highly dependent on the water saturation, and therefore accounts for a major source of nonlinearity in the two-phase flow model. This nonlinearity also means that the sum of the phase permeabilities is not necessarily equal to one. In this work, we employ the Corey model for relative permeability to describe the dependency between relative permeabilities on water saturation, and this is given by:

$$k_{rw} = k_{rw}^{0} S^{n_w} \tag{3.26}$$

$$k_{ro} = k_{ro}^{0} \left(1 - S\right)^{n_o} \tag{3.27}$$

with

$$S = \frac{S_w - S_{wc}}{\left(1 - S_{or} - S_{wc}\right)}, \quad 0 \le S \le 1 \tag{3.28}$$

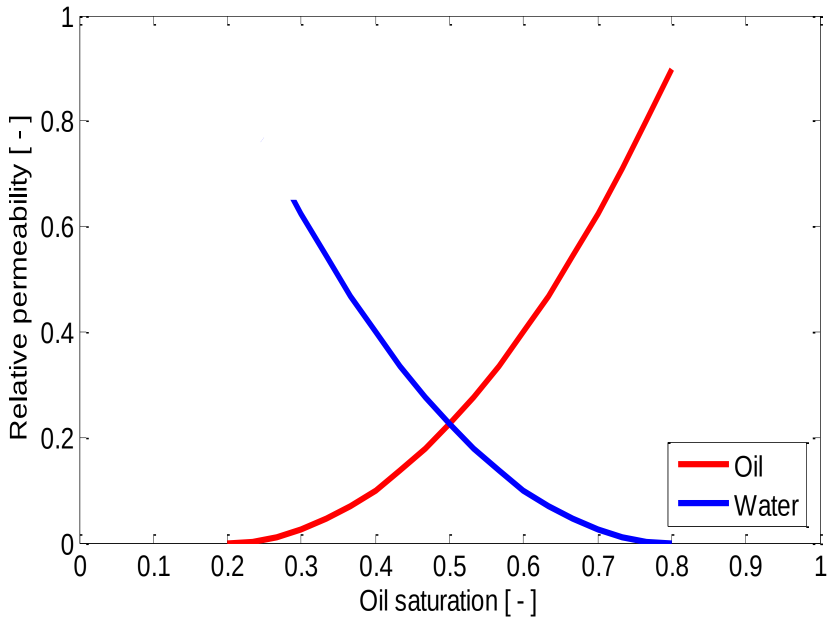


Figure 3.3: Relative permeability curve for oil and water

where k_{ro}^{0} and k_{rw}^{0} are respectively the end point relative permeabilities for oil and water, n_o and n_w are the Corey exponents, S_{wc} and S_{or} are respectively the connate water and residual oil saturations. The relative permeability curve used in the applications in this work is depicted in Figure 3.3.

3.3.2 Two-Phase Flow Equation Solution Methods

Equations 3.24 and 3.25 constitute a system of nonlinear partial differential equations, which are coupled through the saturation-dependent phase mobilities λ_α, (i.e. $\lambda_\alpha = k_{r\alpha}/\mu_\alpha$). These coupled nonlinear partial differential equations are impossible to solve analytically; therefore, we approximate solutions by some numerical methods. In many cases, this involves discretization of each equation and simultaneously solving for the relevant state variable or primary unknowns – which in this case are p and S.

3.3.3 Spatial Discretization and State-Space Representation

Spatial discretization is the first step in this numerical solution of the flow equations. Usually, the reservoir is divided into a finite number of grid-blocks (as illustrated in Figure 3.4) in which the geological properties are assumed to be homogeneous.

Since, the dynamic state of each grid-block i is given by the grid-block's pressure and saturation, we can define the reservoir state as a vector $\mathbf{x}$ such that:

$$\mathbf{x} = \left[\mathbf{p}^T \ \mathbf{S}^T\right]^T \tag{3.29}$$

with

$$\mathbf{p} = \left[p^{1,1} \cdots p^{n_x,n_y}\right]^T$$
$$\mathbf{S} = \left[S^{1,1} \cdots S^{n_x,n_y}\right]^T$$

where n_x and n_y are respectively the finite number of grid-block elements in the x- and y-directions.

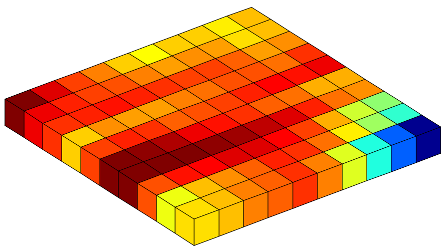


Figure 3.4: A spatially discretized 2-dimensional n_x x n_y heterogeneous reservoir model

In Jansen (2012), it is demonstrated that two-phase flow equations yield a continuous-time state-space equation of the form:

$$\mathbf{E}(\mathbf{x}(t))\dot{\mathbf{x}}(t) = \bar{\mathbf{A}}(\mathbf{x}(t))\mathbf{x}(t) + \bar{\mathbf{B}}(\mathbf{x}(t))\mathbf{u}(t) \tag{3.30}$$
$$\mathbf{x}(0) = \bar{\mathbf{x}}_0 \tag{3.31}$$

where $\mathbf{E}$ is a sparse state-dependent accumulation matrix that consists of four sub-matrices $\mathbf{V}_{wp}, \mathbf{V}_{ws}, \mathbf{V}_{op}$ and $\mathbf{V}_{os}$; and have entries that are dependent on grid-block dimensions, grid-block porosity, compressibility and saturations.

i.e. $$\mathbf{E} = \begin{bmatrix} \mathbf{V}_{wp} & \mathbf{V}_{ws} \\ \mathbf{V}_{op} & \mathbf{V}_{os} \end{bmatrix} \tag{3.32}$$

where
$$\begin{aligned} \mathbf{V}_{wp} &= V\left(c_w + c_r\right)\left[0\ldots0\ \phi^{i,j}S^{i,j}\ 0\ldots0\right] \\ \mathbf{V}_{op} &= V\left(c_o + c_r\right)\left[0\ldots0\ \phi^{i,j}(1-S^{i,j})\,0\ldots0\right] \\ \mathbf{V}_{ws} &= \left[0\ldots0\ \phi^{i,j}\ 0\ldots0\right] \\ \mathbf{V}_{os} &= -\left[0\ldots0\ \phi^{i,j}\ 0\ldots0\right] \end{aligned}$$

Note that V is the grid-block volume, and $\phi^{i,j}$ is the porosity in grid-block i, j corresponding to the x- and y- directions respectively.

The matrix $\bar{\mathbf{A}}$ contains transmissibility and fractional flow terms – both have entries which are dependent on the saturation (via relative permeability). While transmissibilities depend on grid-block dimension, permeability, relative permeability and viscosity; fractional flow terms are dependent on viscosity and relative permeability. Matrix $\bar{\mathbf{A}}$ is defined as:

$$\bar{\mathbf{A}} = -\begin{bmatrix} \mathbf{T}_w + \mathbf{F}_w\mathbf{J}_p & \mathbf{0} \\ \mathbf{T}_o + \mathbf{F}_o\mathbf{J}_p & \mathbf{0} \end{bmatrix} \tag{3.33}$$

where $\mathbf{T}_\alpha$ and $\mathbf{F}_\alpha$ are respectively the phase transmissibility and fractional flow ($\alpha \in \{o, w\}$), while $\mathbf{J}_p$ is a diagonal matrix containing well indices. The transmissibilities are given by:

$$\mathbf{T}_\alpha = \left[0\ldots -T_{\alpha_{i-1/2,j}} \ldots -T_{\alpha_{i,j-1/2}} \left(T_{\alpha_{i-1/2,j}} + T_{\alpha_{i,j-1/2}} + T_{\alpha_{i,j+1/2}} + T_{\alpha_{i+1/2,j}}\right) - T_{\alpha_{i,j+1/2}} \ldots -T_{\alpha_{i+1/2,j}} \ldots 0\right] \tag{3.34}$$

where
$$\begin{aligned} T_{\alpha_{i\pm1/2,j}} &= \frac{\Delta y}{\Delta x}\frac{z}{\mu_\alpha}\left(kk_{r\alpha}(S)\right)_{i\pm1/2,j}, \qquad \alpha \in \{o, w\} \\ T_{\alpha_{i,j\pm1/2}} &= \frac{\Delta y}{\Delta x}\frac{z}{\mu_\alpha}\left(kk_{r\alpha}(S)\right)_{i,j\pm1/2}, \qquad \alpha \in \{o, w\} \end{aligned}$$

and Δx and Δy are respectively the grid-block sizes along the x- and y- directions, and z is the height of the reservoir. Accordingly, the relative permeabilities are approximated based on upstream weighing which describes a convective behaviour, Aziz and Settari (1979); and the absolute permeability is computed by harmonic averages in accordance to:

$$k_{r\alpha}^{\,i\pm\frac{1}{2},\,j} = \begin{cases} k_{r\alpha}^{\,i\pm1,\,j} & \text{if} \quad p^{i\pm1,\,j} \geq p^{i,\,j} \\ k_{r\alpha}^{\,i,\,j} & \phantom{\text{if}} \quad p^{i\pm1,\,j} < p^{i,\,j} \end{cases}, \quad \alpha \in \{o, w\} \tag{3.35}$$

$$k = \frac{2}{\dfrac{1}{k_{i,j}} + \dfrac{1}{k_{i\pm1,j}}} \tag{3.36}$$

The sub-matrices $\mathbf{F}_o$ and $\mathbf{F}_w$ are diagonal matrices containing the fractional flows f_o for oil and f_w for water at diagonal elements corresponding to grid-blocks with prescribed bottom hole pressure and flow rates.

$$f_\alpha = \frac{\lambda_\alpha}{\lambda_o + \lambda_w}, \qquad \alpha \in \{o, w\} \tag{3.37}$$

where λ_α is the phase mobility, which is defined as $\lambda_\alpha = k_{r\alpha} / \mu_\alpha$ for each phase.

Matrix $\mathbf{J}_p$ is a diagonal matrix containing well indices j_p placed at diagonal elements corresponding to the grid-blocks with prescribed bottom hole pressure such that

$$j_p = \left[\frac{k_{rw}(S)}{\mu_w} + \frac{k_{ro}(S)}{\mu_o} \right] \cdot j_{pc} \tag{3.38}$$

where j_{pc} is a constant part of the well index, and is mathematically calculated by:

$$j_{pc} = \frac{2\pi k \Delta z}{\ln\left(0.14 \dfrac{r_e}{r_{well}} \right) + \mathrm{S}} \tag{3.39}$$

where z is the height of the reservoir or height of the grid-block model, r_e is the well external radius which is given by $r_e = \sqrt{\Delta x^2 + \Delta y^2}$, r_{well} is the well bore radius, and S is the well skin factor which is generally assumed to be zero, Peaceman (1978). Multiplying $\mathbf{F}_w$ and $\mathbf{J}_p$ yields a diagonal matrix with non-zero elements corresponding to grid-blocks with defined bottom hole pressure; and these non-zero elements of the diagonal matrix $\mathbf{F}_w.\mathbf{J}_p$ are:

$$\frac{k_{rw}(S)}{\mu_w} \cdot j_{pc} \qquad \text{(in production wells)} \tag{3.40}$$

$$\left[\frac{k_{rw}(S)}{\mu_w} + \frac{k_{ro}(S)}{\mu_o} \right] \cdot j_{pc} \qquad \text{(in injection wells)} \tag{3.41}$$

By the same token, $\mathbf{F}_o.\mathbf{J}_p$ is also a diagonal matrix with non-zero elements corresponding to grid-blocks with prescribed bottom hole pressure, and the non-zero elements are generally given by:

$$\frac{k_{ro}(S)}{\mu_o} \cdot j_{pc} \qquad \text{(in all production wells)} \tag{3.42}$$

The matrix $\bar{\mathbf{B}}$ is defined as:

$$\bar{\mathbf{B}} = \begin{bmatrix} \mathbf{F}_w \\ \mathbf{F}_o \end{bmatrix} \cdot \left(\mathbf{I}_q + \mathbf{J}_p{}^* \right) \tag{3.43}$$

where matrix $\mathbf{I}_q$ is a partial identity matrix – which is a diagonal matrix with unity elements in all grid-blocks with prescribed flow rates, and $\mathbf{J}_p{}^*$ is a block matrix with zeros and $\mathbf{J}_p$ as sub-matrices. Multiplying $\mathbf{F}_w$ and $\mathbf{I}_q$ yields a diagonal matrix with non-zero elements in grid-blocks that are defined by flow rates $\mathrm{q}_\alpha \left(\alpha \in \{o, w\} \right)$.

The input vector $\mathbf{u}$ represents the non-zero elements of vector $\mathbf{q}_t$ – which is the vector of the total flow rates for all grid-blocks. The continuous-time state-space equation (3.30) can be represented in a partitioned form as follows:

$$\begin{bmatrix} \mathbf{V}_{wp} & \mathbf{V}_{ws} \\ \mathbf{V}_{op} & \mathbf{V}_{os} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{s}} \end{bmatrix} + \begin{bmatrix} \mathbf{T}_w + \mathbf{F}_w \mathbf{J}_p & \mathbf{0} \\ \mathbf{T}_o + \mathbf{F}_o \mathbf{J}_p & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{s} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_w \\ \mathbf{F}_o \end{bmatrix} \cdot (\mathbf{I}_q + \mathbf{J}_p{}^*) \mathbf{q}_t \tag{3.44}$$

It is noted that the accumulation matrix **E** is invertible as long as the fluids are compressible and the porosity in all the grid-blocks is non-zero. We can therefore re-write (3.30) and (3.44) in the following state-space format:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(\mathbf{x}(t))\mathbf{x}(t) + \mathbf{B}(\mathbf{x}(t))\mathbf{u}(t) \tag{3.45}$$

$$\mathbf{x}(0) = \bar{\mathbf{x}}_0 \tag{3.46}$$

where

$$\mathbf{A} := \mathbf{E}^{-1} \bar{\mathbf{A}} = - \begin{bmatrix} \mathbf{V}_{wp} & \mathbf{V}_{ws} \\ \mathbf{V}_{op} & \mathbf{V}_{os} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{T}_w + \mathbf{F}_w \mathbf{J}_p & \mathbf{0} \\ \mathbf{T}_o + \mathbf{F}_o \mathbf{J}_p & \mathbf{0} \end{bmatrix}, \ \mathbf{B} := \mathbf{E}^{-1} \bar{\mathbf{B}} = \begin{bmatrix} \mathbf{V}_{wp} & \mathbf{V}_{ws} \\ \mathbf{V}_{op} & \mathbf{V}_{os} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{F}_w \\ \mathbf{F}_o \end{bmatrix} \cdot (\mathbf{I}_q + \mathbf{J}_p{}^*).$$

For more on the mathematical formulation of reservoir equations in control format, the interested reader is referred to Jansen (2012), from which this sub-section was adapted.

3.3.4 Temporal Discretization

After spatial discretization, performing a reservoir simulation requires discretization of the time derivative $\dot{\mathbf{x}}(t)$. Since the initial condition value is known, the time derivative $\dot{\mathbf{x}}(t)$ is approximated by a first-order Euler scheme, viz:

$$\dot{\mathbf{x}}(t) \approx \frac{\mathbf{x}(t+\Delta t)-\mathbf{x}(t)}{\Delta t} \tag{3.47}$$

In so doing, it is important to select appropriate instances at which the time-dependent inputs and parameters are evaluated, as the choice can significantly influence the computational efficiency as well as the stability. Numerical temporal discretization schemes may be fully explicit, fully implicit, IMplicit Pressure Explicit Saturation (IMPES) or Adaptive Implicit Method (AIM).

If the inputs and parameters are evaluated at the current time instance *k*, the discretization is referred to as fully explicit; thus, the state vector at the next time-step $\mathbf{x}_{k+1}$ can be obtained as an explicit expression in terms of state vector at current time-step $\mathbf{x}_k$. Because there is no need to solve any system of equations, this discretization approach often results into fast computation of variables. However, stability issues arising from time-step restrictions can be problematic; and this can considerably increase the computational requirements. The explicit temporal discretization leads to the nonlinear discrete-time state-space equation, viz:

$$\mathbf{x}_{k+1} = \mathbf{A}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{B}(\mathbf{x}_k)\mathbf{u}_k \tag{3.48}$$

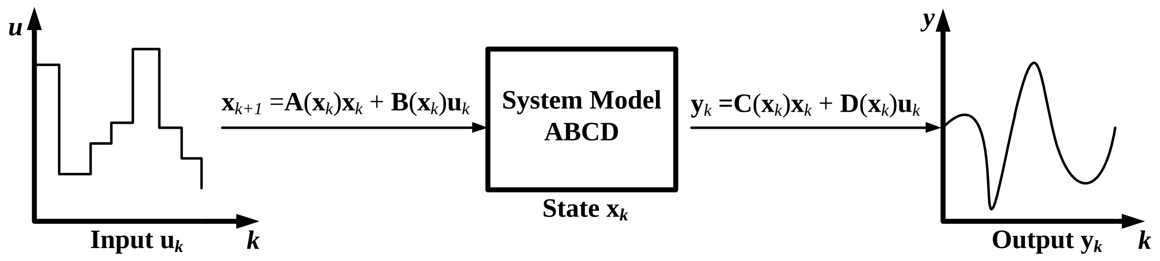
$$\mathbf{y}_k = \mathbf{C}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{D}(\mathbf{x}_k)\mathbf{u}_k \tag{3.49}$$

For a reservoir model of *N* grid-blocks, with *m* inputs, *p* outputs and *n* state variables, the vectors $\mathbf{x}$, $\mathbf{u}$ and $\mathbf{y}$ are defined as follows:

$$\mathbf{x} := \begin{bmatrix} \mathbf{p} \\ \mathbf{s} \end{bmatrix} \in \mathbb{R}^{n=2N}, \quad \mathbf{u} := \begin{bmatrix} \tilde{\mathbf{q}}_{well} \\ \tilde{\mathbf{p}}_{well} \end{bmatrix} \in \mathbb{R}^{m} \quad \text{and} \quad \mathbf{y} := \begin{bmatrix} \bar{\mathbf{p}}_{well} \\ \bar{\mathbf{q}}_{well,w} \\ \bar{\mathbf{q}}_{well,o} \end{bmatrix} \in \mathbb{R}^{p}.$$

Note that the state vector $\mathbf{x}_k \in \mathbb{R}^{n=2N}$ represents the state variables i.e. pressure and water saturation values in all the grid-blocks, the input vector $\mathbf{u}_k \in \mathbb{R}^{m}$ contains the designated flow rates and bottom-hole pressure in the wells, and the output vector $\mathbf{y}_k \in \mathbb{R}^p$ include measured flow rates and bottom-hole pressure in each well.

The state-space equation presented in (3.48) and (3.49) can be represented schematically as shown in Figure 3.5



u
$\mathbf{x}_{k+1} = \mathbf{A}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{B}(\mathbf{x}_k)\mathbf{u}_k$
System Model ABCD
State $\mathbf{x}_k$
$\mathbf{y}_k = \mathbf{C}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{D}(\mathbf{x}_k)\mathbf{u}_k$
y
Input $\mathbf{u}_k$
k
Output $\mathbf{y}_k$
k

Figure 3.5: A schematic representation of reservoir model state space equation

In systems and control literature, matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is often referred to as the '*system matrix*' or '*dynamics matrix*' as it contains the dynamic properties of the system; matrix $\mathbf{B} \in \mathbb{R}^{n\times m}$ is the '*input matrix*' as it maps the inputs to the states; matrix $\mathbf{C} \in \mathbb{R}^{p\times n}$ is the '*output matrix*' since it maps the states to the output, and matrix $\mathbf{D} \in \mathbb{R}^{p\times m}$ is the '*feedthrough matrix*'.

Under the assumption of no capillary pressure (as we have earlier underlined), Eq. 3.48 can be represented in the form:

$$\underbrace{\begin{bmatrix} \mathbf{p}_{k+1} \\ \mathbf{s}_{k+1} \end{bmatrix}}_{\mathbf{x}_{k+1}} = \underbrace{\begin{bmatrix} \mathbf{A}_{11}(\mathbf{s}_k) & \mathbf{0} \\ \mathbf{A}_{21}(\mathbf{s}_k) & \mathbf{0} \end{bmatrix}}_{\mathbf{A}(\mathbf{x}_k)} \underbrace{\begin{bmatrix} \mathbf{p}_k \\ \mathbf{s}_k \end{bmatrix}}_{\mathbf{x}_k} + \underbrace{\begin{bmatrix} \mathbf{B}_1(\mathbf{s}_k) \\ \mathbf{B}_2(\mathbf{s}_k) \end{bmatrix}}_{\mathbf{B}(\mathbf{x}_k)} \mathbf{u}_k \tag{3.50}$$

showing that the reservoir dynamic states (grid-blocks pressures and saturations) are driven by the pressures of the previous time step. Again, because of the saturation dependency of relative permeability in the system matrix $\mathbf{A}_{21}$, the saturation part of the grid-block states are also driven by the saturations of the previous time step.

For the fully implicit scheme, the dependent variables and inputs are evaluated at the next time-step i.e. at time k+1. The computation of the variables at the next time-step requires that a system of N nonlinear equations (where N denotes the total number of grid-blocks) is solve simultaneously; and this is often carried out through some iterative procedure such as the Newton–Raphson method. Fully implicit discretization scheme are usually robust and unconditionally stable. In other words, they yield a stable solution for large time-steps; the only restrictions on time-step size are those necessary to ensure convergence of the iteration scheme employed. However, because of the huge number of nonlinear system of equations (equal to the number of reservoir grid-blocks) involved in this approach, they are usually computationally expensive. The fully implicit scheme results to a coupled system of nonlinear equations of the format:

$$\mathbf{g}_k\left(\mathbf{u}_k, \mathbf{x}_{k-1}, \mathbf{x}_k\right) = \mathbf{0} \tag{3.51}$$

$$\mathbf{h}_k\left(\mathbf{u}_k, \mathbf{x}_k, \mathbf{y}_k\right) = \mathbf{0} \tag{3.52}$$

where $\mathbf{g}$ and $\mathbf{h}$ are nonlinear vector-valued functions.

The IMplicit Pressure Explicit Saturation (IMPES) scheme employs a splitting approach which capitalizes on the physics or nature of the coupled systems of flow equations. Under this scheme, the coupled system of partial differential equations is decoupled into the two fundamentally different equations – pressure and saturation equations; and each of the equation is solved using different discretization methods. The pressures are subsequently determined by solving the pressure equations implicitly, while the saturations are determined explicitly by solving the material balance equations. In other words, after solving for pressures, the two-phase saturations are updated explicitly by computing Darcy’s velocity from the pressure distribution obtained earlier. Now, it is important to note that since the time scales of the dynamic behavior of the two-phase flow model differ remarkably (the changes in pressure are less rapid than the changes in saturation); it is therefore, logical to assign different time-sizes to the decoupled equations. While the implicit pressure update can deal with large time-step sizes, there is a time-step size restriction on the explicit saturation updates so as to guarantee stability. It is noted that the instability of the IMPES approach is as a result of the explicit treatment of the capillary pressure and the decoupling between the pressure equation and the saturation equation, Kou and Sun (2010). This means that the IMPES approach is conditionally stable; hence, its application in highly heterogeneous permeable media (where capillary pressure effects play a significant role in fluid flow path) must be restricted to very small time-step sizes. Because saturation dependent parameters such as relative permeabilities and capillary pressures are often assumed to be constants at each time-step, the formulation and implementation of IMPES are relatively easy; its low memory and CPU time requirements also makes it quite attractive, Markovinovic (2009).

Another popular discretization scheme is the approach known as adaptive implicit method (AIM). Under this scheme, the time-dependent variables in some grid blocks are solved fully implicitly while the rest are solved using IMPES. Thus, this scheme assigns different levels of implicitness to the grid-blocks, and these levels are appropriately adjusted (as required) in space and time to maintain stability. The method therefore gives robustness in problematic areas with large changes in pressure and saturations (like near a wellbore), while at the same time giving high computational efficiency away from problem regions, Aarnes et al. (2007).

3.4 Model Nonlinearity

The nonlinearity in the reservoir flow model represented in (3.48) stems from the system matrices which have coefficients that are functions of the saturations. As we pointed out earlier, the system matrix **A** contains transmissibilities which are functions of the water saturation through their relationship with relative permeabilities. Again, it is important to point out that the transmissibilities are also functions of pressure due to the upstream weighting approach used for the computation of relative permeabilities as indicated in (3.35). Any change in neighboring grid-blocks pressures would inadvertently lead to change in the upstream relative permeability. Because the input matrix **B** contains entries that are based on the fractional flows of oil and water, it constitutes another source of nonlinearity. The fractional flows of oil f_o and water f_w which are required for the computation of matrix **B** are functions of the saturation through the oil and water mobilities as shown in (3.37).

3.5 Model Uncertainty

The reservoir model described so far in this chapter has been modelled under a number of simplifying assumptions that we have noted. Therefore, they inherently contain some degrees of uncertainty. Loosely speaking, uncertainty in reservoir models can be categorized into those that are introduced as a result of modelling simplifications, and those that are introduced as a result of limited knowledge of reservoir geology or sparsity of data. Below are some of the sources of uncertainty in numerical and computational reservoir models.

3.5.1 Assumptions and Simplifications

The many simplifying assumptions used in deriving the flow model are a major source of uncertainty. Although the reservoir flow model equations are generally based on the physics of multiphase flow, the underlying governing equations are not without some assumptions as clearly spelt out in section 3.3 of this chapter. For reasons bothering on computational resource affordability, the black-oil formulation is often used instead of a formulation that takes care of all the components that are present in the reservoir. In the same vein, the semi-empirical Darcy's law which only approximates the true physics behind the multiphase flow is used. Also, gravity and capillary pressures effects are often neglected in order to simplify the model. After spatial discretization, the fact that the states (pressures and saturations) and the geological properties in each of the coarse-sized (approximately in the order of 100m×100m×10m) grid-block are assumed to be homogenous brings with it an unknown degree of model uncertainty. And besides that, the reservoir geometry is also uncertain; thus, the boundary conditions applied in the discretized model may be highly inaccurate. These uncertainties would one way or the other influence the reliability of the underlying numerical reservoir model.

3.5.2 Limited and Sparse Data

One major source of reservoir model uncertainty is model parameters. These parameters manifest as geological properties (grid-block permeability and porosity) and fluid properties (relative permeability, density and viscosity). Some of these parameters are based on laboratory experiments on core samples taken from the field, while others are often inferred from sparse measurements that are taken at the wells or at locations that are in the proximity of the wells. Thus, the parameter values in the formation are often unknown and have to be determined by some kind of extrapolation. Also, the initial conditions of the reservoir dynamic states (pressures and saturations) are usually uncertain as they are often computed from very limited set of data gathered from wells and locations in close vicinity of wells. This means that the contact depths of the fluids and indeed the initial oil in place are to a large extent very unknown and uncertain. Finally, it is important to note that the presence of an active aquifer can be a source of uncertainty in the reservoir model. Though the presence of such active aquifer can be desirable – as they can slow down reservoir pressure decline, there is no gainsaying that they can have a significant impact on the predictions purposes the models serve. Generally speaking, all these uncertainty types are not explicitly taken into account in the reservoir model, and are therefore difficult to assess.

3.6 Model Limitations

In the previous section, the inherent uncertainty in reservoir models were described; we further highlighted the various sources of the uncertainty. The presence of these uncertainties inevitably translates to the fact that reservoir models are only a (very) crude approximation of the real physical reservoir. In other words, they have limitations which can adversely affect their prediction capabilities. For reasons bothering on these limitations, multiple reservoir models are often employed when making future production predictions. The spread in the predictions arising from the different models of the same reservoir, together with their probabilities can be used to assess the impact of model uncertainty within a statistical framework. However, this spread in the probabilities of the different models of the same physical reservoir can be very large and cumbersome; invariably, this could constitute a major impediment or limitation in the deployment of reservoir models as a tool for field development decisions. In this regard, a large spread would be a thing of concern – it implies significant financial risk in the development and production of the field. Nevertheless, reservoir models are widely used in the E&P industry as an essential tool during the development, production and re-development exercises of oil and gas fields. There are quite a number of reservoir model simulators which basically implement equations of the form (3.48). Some of these simulators are commercially available – **ECLIPSE**® (Schlumberger), **VIP**® (Halliburton), **IPM**® (Petroleum Experts); others are proprietary – **CHEARS**® (Chevron), **MoReS**® (Shell), **PSim**® (ConocoPhillips); and some are open source – **MRST**® (SINTEF). In this thesis, we employed **MRST**® as well as **ECLIPSE**® in our implementations.

3.7 Summary

In this chapter, we presented the basic concepts that describe flow in porous media, as well as the notations that would be employed in the remainder of this thesis. The equations governing the flow of each phase in a porous medium are a combination of a mass balance, momentum conservation and compressibility equations as well as boundary and initial conditions. The resulting reservoir model is a system of coupled nonlinear partial differential equations (with large number of dynamic states and physical parameters) that cannot be solved analytically. Numerical solution of these coupled equations often involve spatial and temporal discretization, and this leads to a large number of ordinary differential equations which can be written in state-space form. The states in the resulting state-space equation are the grid-block pressures and saturations, the inputs are the production settings of the wells, while the physical parameters include fluid and geological properties such as grid-block permeabilities and porosities, density, viscosity, and relative permeabilities. We noted that the reservoir models contain a significant quantity of uncertainty arising from various sources, and that these uncertainties invariably translate to limitations in the predictive power and capability of the models. Often, the effect or impact of this uncertainty can be assessed by considering multiple realizations of the physical reservoir and analyzing their probability within a statistical framework.

CHAPTER 4

When a man is at peace with the gods and his ancestors, his harvest will be good or bad according to the strength of his arm – Chinua Achebe

WELL PLACEMENT OPTIMIZATION

The focus in this chapter is the optimization of well locations in reservoir models of varying complexities. Well placement optimization often entails determining the optimal number, type, location (and possibly the drilling sequence) of wells for a hydrocarbon reservoir. It is arguably the most important decision-input of field development planning. We will describe in detail, three metaheuristic algorithms viz: differential evolution (DE), particle swarm optimization (PSO) and hybrid particle swarm differential evolution (HPSDE) – which is a hybrid of DE and PSO; and we will discuss the application of these algorithms in the well placement optimization problem.

4.1 Well Placement Problem Formulation

For any given hydrocarbon reservoir of the form of Eq. 3.45, determining the optimal well configuration that maximizes the recovery factor over a time interval [0, *T*] can be posed as an optimization problem. In every practical sense, the maximization of the recovery factor (objective function) for a water-flooded reservoir is equivalent to any of the following:

1. maximizing the cumulative volumes of hydrocarbon produced at terminal time *T*
2. maximizing the water saturation of the reservoir at terminal time *T* or
3. minimizing the volume of hydrocarbon in place at terminal time *T*

However, the objective for most E&P companies is to maximize the economic value of their asset. The commonly used economic criterion for this purpose is the net present value (NPV) as employed by Beckner and Song (1995), Montes et al. (2001), Yeten (2003), Aitokhuehi et al. (2004), Bangerth et al. (2006), van Essen et al. (2006), Sarma and Chen (2008), Zandvliet et al. (2008), Onwunalu and Durlofsky (2010), Bouzarkouna et al. (2011), Ciaurri et al.

(2011) and Dong et al. (2011). Therefore, NPV is designated as objective function in all the problems considered in this chapter; and for all potential well configuration (solutions), the NPV is computed from the fluid production profiles generated as a result of simulation run associated with corresponding well placements. In other words, the NPV is a measure of the cash flow (CF) generated from sale of produced volumes of oil.

Following a slight modification of the economic model described in Onwunalu and Durlofsky (2010), we define the NPV as the total oil revenues minus the capital expenditure (CAPEX) and the operation cost (OPEX), in combination with a discount factor d – which represents the time value of money (interest rate or inflation). This is represented mathematically as:

$$\mathrm{NPV} = \sum_{t=1}^{T} \frac{\mathrm{CF}_{(t)}}{(1+\mathrm{d})^{t}} - \mathrm{CAPEX} \tag{4.1}$$

where T is the terminal time or total production years, d is the discount factor, and $\mathrm{CF}_{(t)}$ represents the cash flow at time t. The cash flow at any time is given by:

$$\mathrm{CF}_{(t)} = \mathrm{REV}_{(t)} - \mathrm{OPEX}_{(t)} \tag{4.2}$$

where $\mathrm{REV}_{(t)}$ is the revenue accrued from sale of products at time t; and $\mathrm{OPEX}_{(t)}$ represents the operating expenditure (or cost of production) at time t. Both quantities are measured in US dollars ($).

For a two-phase (oil and water) flow reservoir model, the values of $\mathrm{REV}_{(t)}$ and $\mathrm{OPEX}_{(t)}$ at any time (t) are respectively given by:

$$\mathrm{REV}_{(t)} = p_{(t)}^{\mathrm{oil}}\, \upsilon_{(t)}^{\mathrm{oil}} \tag{4.3}$$

$$\mathrm{OPEX}_{(t)} = p_{(t)}^{\mathrm{w,p}}\, \upsilon_{(t)}^{\mathrm{w,p}} + p_{(t)}^{\mathrm{w,i}}\, \upsilon_{(t)}^{\mathrm{w,i}} \tag{4.4}$$

where $p_{(t)}^{\mathrm{oil}}$ is the price of oil at time t, $p_{(t)}^{\mathrm{w,p}}$ is the cost of producing water in the production wells, and $p_{(t)}^{\mathrm{w,i}}$ is the cost of injecting water in the injection wells at time t – all three quantities are measured in dollars per barrel. On the other hand, $\upsilon_{(t)}^{\mathrm{oil}}$ (measured in barrels) is the total volume of oil produced at time t, while $\upsilon_{(t)}^{\mathrm{w,p}}$ and $\upsilon_{(t)}^{\mathrm{w,i}}$ (both measured in barrels) represents the total volumes of water produced (from production wells) and injected (in injection wells) respectively.

The CAPEX represents the total cost to drill and complete all wells; it is noted that as far as production is concerned, CAPEX is incurred at time $t = 0$. It is computed as follows:

$$\mathrm{CAPEX} = \sum_{\mathrm{w}=1}^{n^{\mathrm{w}}} \left(\mathrm{C}_{\mathrm{w}}^{\mathrm{top}} + \mathrm{L}_{\mathrm{w}}^{\mathrm{main}} \times \mathrm{C}^{\mathrm{drill}} \right) \tag{4.5}$$

where n^{w} is the total number of wells, $\mathrm{C}_{\mathrm{w}}^{\mathrm{top}}$ is the cost of drilling the main bore to the top of the reservoir (in \$), $\mathrm{L}_{\mathrm{w}}^{\mathrm{main}}$ is the length of the main bore (in meters) and $\mathrm{C}^{\mathrm{drill}}$ is the cost of drilling within the reservoir (in dollar per meter).

There have been a number of publications which have attempted to solve well placement optimization problem akin to that described in (4.1). Broadly speaking, most of the solution techniques employed in this problem domain can be categorized either into the gradient-based local optimization methods, where the derivatives of the objective function are computed using some numerical finite difference schemes or the adjoint formulations, or derivative-free stochastic optimization techniques, which often involves the use of heuristics or metaheuristic algorithms. A review of these techniques is highlighted in section 2.1 of this thesis. With the benefit of this review, it follows that the use of derivative-free stochastic or metaheuristic algorithms are quite popular in well optimization problem domain. This notwithstanding, a computational efficient metaheuristic algorithm that require limited number of simulations (objective function evaluation) is still lacking. Based on experimental results, we propose a hybrid algorithm of differential evolution and particle swarm optimization referred to as hybrid particle swarm differential evolution for well placement optimization problems.

4.2 Differential Evolution (DE)

In Storn and Price (1995), an encoded floating point population-based metaheuristic algorithm was introduced as a computational intelligence paradigm for global optimization; and the new algorithm was named differential evolution (DE). It derived its name from a special kind of differential operator which they invoked when creating new offspring of its population, Das et al. (2008). Being an evolutionary algorithm, DE is based on the Darwin's principle of “survival of the fittest”, a strategy in which the individuals in a population evolve by improving their fitness value through the probabilistic operations of mutation, recombination and selection. The individuals are evaluated with respect to their fitness against a defined objective function, and those with superior fitness are selected to compose the population of the next generation.

Like the well-known genetic algorithm (GA), DE is based on the theory of natural selection and in both algorithms, the selection operator is the sole mechanism for choosing the best individuals from the population in every generation (or iteration); thus, many researchers have reported DE as an improved version of GA. However, it is important to note that there are salient differences between both algorithms. While GA relies either on binary or real-valued (continuous) strings, DE operates directly on floating point vectors; whereas GA maintain a genetic link from one generation to another, DE is an abstraction of evolution at individual behavioral level; and most importantly, GA relies mainly on the crossover operator to explore the search space, while a special form of mutation operator effects the working of DE. In other words, crossover and mutation mechanisms are the dominant operator in GA and DE algorithms, respectively. Over the past years, DE has been shown to be a simple but versatile metaheuristic algorithm for real-parameter optimization, Storn and Price (1997), Rogalsky et al. (2000), Das et al. (2008); it is arguably one of the main advancements in computational intelligence research domain; and the spurt in interest in this subject is evident from the wide array of application areas (science, engineering, statistics, economics and finance) in which it has been deployed.

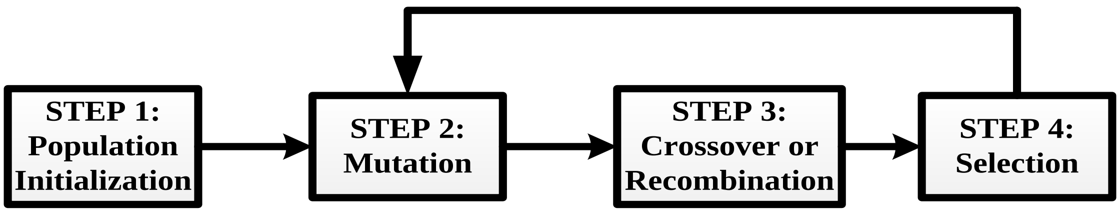

STEP 1: Population Initialization
STEP 2: Mutation
STEP 3: Crossover or Recombination
STEP 4: Selection

Figure 4.1: Basic DE algorithm procedure

Since inception, several DE strategies have evolved, and a comprehensive naming notation to classify these strategies is presented in Storn and Price (1997). The standard nomenclature of the various DE strategies is consistent with the DE/*a*/*b*/*c* format; where *a* represents a string denoting the target of the mutation operation, *b* defines the number of difference vectors used in the mutation, and *c* stands for the type of crossover employed. Based on the standard and notations defined above, the most widely used DE strategy is the DE/rand/1/bin. This strategy indicates that the mutation target is randomly selected from the population, and the mutation is performed using a single difference vector, as well as a uniform binomial crossover. The basic DE algorithm consists of four distinct events which are as represented in Figure 4.1. The first step is the initialization of a population of candidate solutions N_P at iteration k = 1, and this is given by:

$$N_p(k) = \left[X_1(k), X_2(k), ..., X_{Np}(k) \right] \tag{4.6}$$

where each candidate solution $X_i(k)$, is a D-dimensional vector containing as many real-valued parameters as the problem dimension D. Each of the candidate solution is given by:

$$X_i(k) = \left[x_{i,j}(k), x_{i,j}(k), ..., x_{i,D}(k) \right] \tag{4.7}$$

where $i = 1, 2, ..., N_p$ and $j = 1, 2, ..., D.$

Typically, each decision parameter in every candidate solution of the initial population is assigned a randomly chosen value from a pre-defined feasible numerical bound. In other words, the j-th component of the i-th population member at the initialization step is given by:

$$x_{i,j}(1) = x_j^L + \text{rand } (0,1) \cdot \left(x_j^U - x_j^L \right) \tag{4.8}$$

where x_j^L and x_j^U are respectively the lower and upper bound of the j-th parameter, and rand (0,1) is a uniformly distributed random number between 0 and 1.

Once the population has been initialized, the corresponding fitness value is evaluated and stored in memory for future reference. In each generation, a mutant vector is created for each i-th population member by randomly choosing three parameter vectors from the current population. A scalar number F is used to scale the difference of any two of the three random-chosen vectors; and the scaled difference is added to the third one. We can express the mutation process of the j-th component of each vector as follows:

$$v_i(k) = x_{r1,j}(k) + F \cdot \left(x_{r2,j}(k) - x_{r3,j}(k) \right) \tag{4.9}$$

where $F \in [0,1+]$ is a user-defined constant known as the scaling (or mutation) factor, and vector indices $r1$, $r2$ and $r3$ are randomly chosen with $r1$, $r2$ and $r3 \in \{1, 2, ..., N_p\}$.

Note that $r1 \neq r2 \neq r3 \neq i,$ and that x_{r1}, x_{r2} and x_{r3} are selected anew for each parent vector in every generation. The magnitude and direction of the mutation step is defined by the difference between two of the three random-chosen population vectors; and this makes the mutation operation to exhibit a self-adaptive behavior, such that the average mutation length decreases as the population converges, Storn and Price (1997). The third step is the crossover or recombination scheme. The main purpose of this process is to increase the potential diversity of the population by mixing the parameters of the mutant vector with the target vector according to a selected probability distribution. There are mainly two kinds of crossover schemes – binomial and exponential. The result of the crossover step at iteration *k* is the birth of a trial vector which is defined as:

$$U_i(k) = \left[u_{i,j}(k), u_{i,j}(k), ..., u_{i,D}(k) \right] \tag{4.10}$$

For a maximization problem, the binomial crossover scheme is performed on each of the *D*-dimensional variables according to the equation:

$$u_{i,j}(k) = \begin{cases} v_{i,j}(k), & \text{if } \operatorname{rand}(0,1) \leq CR \\ x_{i,j}(k), & \text{else} \end{cases} \tag{4.11}$$

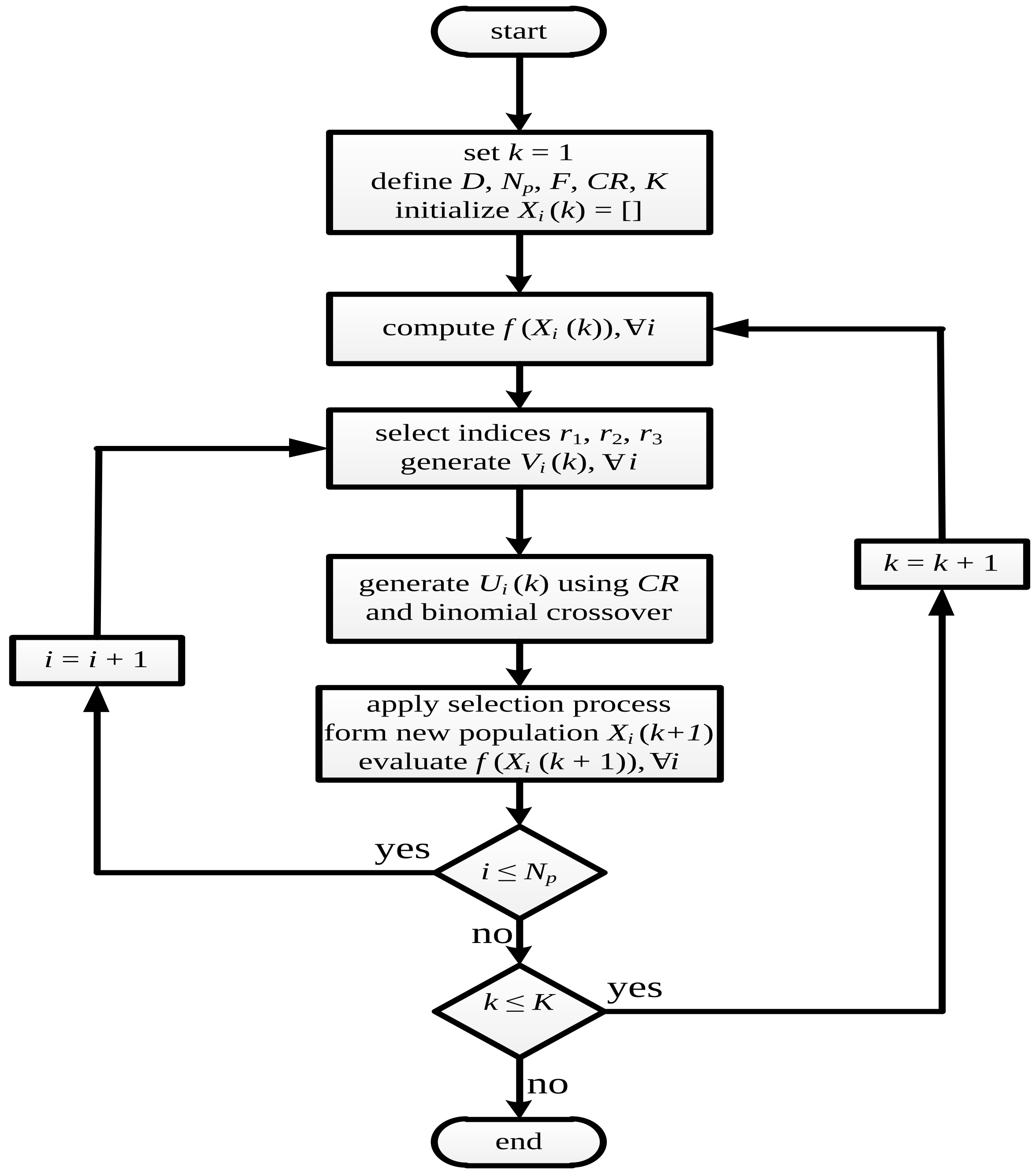
where *CR* is a user-defined crossover rate which is usually in the range [0,1].

The crossover rate controls the diversity of the population and aids the algorithm to avoid getting stuck in local optima. At the end of the iteration, the selection operator is applied to determine which one of the target and the trial vectors would survive in the next iteration, i.e. at iteration $k = k + 1$. This operator compares the fitness of the trial vectors against the corresponding target vectors and selects the better solution according to the equation:

$$X_i(k+1) = \begin{cases} U_i(k), & \text{if } f(U_i(k)) \geq f(X_i(k)) \\ X_i(k), & \text{else} \end{cases} \tag{4.12}$$

where $f(x)$ is a fitness value.

Thus, if the new trial vector yields a better fitness value, it automatically replaces its target in the next iteration; otherwise the target vector is retained in the population. In other words, the population must either get better (with respect to the fitness value) or remain constant, it never deteriorates. Figure 4.2 is a simple flowchart that illustrates the DE algorithm; its control parameters are mutation factor *F*, crossover rate *CR* and population size N_p.



start
set $k = 1$
define D, N_p, F, CR, K
initialize $X_i(k) = []$
compute $f(X_i(k)), \forall i$
select indices r_1, r_2, r_3
generate $V_i(k), \forall i$
generate $U_i(k)$ using CR
and binomial crossover
apply selection process
form new population $X_i(k+1)$
evaluate $f(X_i(k+1)), \forall i$
$i = i + 1$
$k = k + 1$
yes
$i \le N_p$
no
$k \le K$
yes
no
end

Figure 4.2: Flowchart showing the DE algorithm.

4.2.1 Treatment of Infeasible Solutions

Evolutionary algorithms such as DE were originally proposed to solve unconstrained optimization problems. The application of these algorithms on boundary-constrained problems such as well placement optimization problem may result in solutions that violate the physical boundary of the search space. All such bound offending values are termed infeasible solutions; and a comprehensive review of methods for preserving feasibility of solutions is available in the study of Michalewicz and Schoenauer (1996). In this work however, we employ the “out-of-bound value” technique, Lampinen (2002) and Davendra et al. (2009). This involves the use of specialized operators to create and retain candidate solutions that are feasible. Accordingly, to ensure that bound offending values are reset to boundary values, the following equation is implemented:

$$x_{i,j}(k) = \begin{cases} x_{i,j}^{\min} & \text{if } x_{i,j}(k) \le x_{i,j}^{\min} \\ x_{i,j}^{\max} & \text{if } x_{i,j}(k) \ge x_{i,j}^{\max} \\ x_{i,j}(k) & \text{otherwise} \end{cases} \tag{4.13}$$

where $x_{j,i}^{\min}$ and $x_{j,i}^{\max}$ are respectively the minimum and maximum bound of the j-th component of the i-th population member.

4.2.2 Implementation of DE in Well Placement Problem

We describe the implementation of the DE algorithm for a well placement optimization problem. Algorithm 4.1 presents the steps in the DE /1/rand/bin strategy for a maximization problem. It is adapted and modified from Storn and Price (1997), and implemented in **MATLAB**®. The first step signifies the beginning of the algorithm; and step 2 assigns values to DE parameters – N_p, F, CR, D and $K.$ In step 3, the population vector is initialized such that each component $x_{i,j}(k), \forall i \in \{1, 2, ..., N_p\}, \forall j \in \{1, 2, ..., D\}$ are made of random elements drawn from predefined lower (L) and upper (U) bounds in accordance to Equations 4.7 and 4.8. Step 4 computes the objective function of the initialized population, and the evaluated objective function is saved in step 6 for future reference. Steps 9–11 compute a mutant vector $v_i(k)$ in accordance with Eq. 4.9. Step 12 is used in order to generate a trial vector U_i in accordance with Eq. 4.11. Following the birth of the trial vector, if any elements of the ‘newly born’ vector are outside the feasible region of the search space, step 13 is activated to modify and adjust the trial vector within the feasible region. Step 14 evaluates the objective function of the trial vectors. Steps 18–25 describe the selection process; the objective function of the trial vector is compared against the target vector in order to determine the population of the next iteration index. The algorithm terminates when the maximum iterations K is reached.

Algorithm 4.1 DE Algorithm

```
 1: Set iteration index $k = 1$
 2: Define $D, N_p, F = 0.5, CR = 0.1, K$
 3: Initialize $X_i(k) : x_{i,j}(k) \sim \text{rand}\,(L_j, U_j) \forall j, \forall i$
 4: Compute objective function, $f(X_i(k)), \forall i$
 5: while $k \le K$ do
 6:     Save, $f(X_i(k)), \forall i$
 7:     $i = 1$
 8:     while $i \le N_p$ do
 9:         Select $r_1, r_2, r_3 \in \{1, 2, ..., N_p\}, r_1 \ne r_2 \ne r_3 \ne i$
10:         Randomly select $j \in \{1, 2, ..., D\}$
11:         Compute mutant vector $v_i(k), \forall i$
12:         Apply (4.11) to generate trial vector $U_i(k), \forall i$
13:         Apply (4.13) if necessary
14:         Compute objective function, $f(U_i(k)), \forall i$
15:         $i = i + 1$
16:     end while
17:     $i = 1$
18:     while $i \le N_p$ do
19:         if $f(U_i(k)) > f(X_i(k))$ then
20:             $X_i(k+1) = U_i(k)$
21:         else
22:             $X_i(k+1) = X_i(k)$
23:         end if
24:         $i = i + 1$
25:     end while
26:     $k = k + 1$
27: end while
```

4.3 Particle Swarm Optimization (PSO)

It can be said that the basic idea behind the PSO algorithm is the simulation of the social behavior metaphor of bird flocks and fish schools. It is another population-based biologically-inspired stochastic algorithm which has been widely and efficiently deployed in non-linear optimizations of varying complexities, and across diverse engineering and computational science disciplines. Introduced in 1995 in Eberhart and Kennedy (1995) and Kennedy and Eberhart (1995); its popularity has gained momentum because of its low memory requirement, high computational efficiency and easy-to-implement properties. Being a population based algorithm, the individuals of the population are referred to as particles, and a collection of particle at any given iteration is called the swarm. The particles are flown through the search space, with each particle representing a possible or potential solution of the optimization problem. In any given iteration, a particle's fitness is based on a performance function related to the optimization problem; or in other words, the position of each particle is continually adjusted according to its relative fitness and position to other particles that make up the swarm.

The movement of the particles across the search space is influenced by two factors – information from iteration-to-iteration, and information from particle-to-particle interactions. Based on iteration-to-iteration information, the particle stores in its memory the best solution attained so far, and it experiences an attraction towards this solution (*pbest*) as it traverses across the problem search space. On the other hand, the outcome of the particle-to-particle information is that each particle stores in its memory, the best solution (*gbest*) attained by any particle in the swarm, and experiences an attraction towards this solution. These factors are respectively referred to as the cognitive and social components of the algorithm. At the end of each iteration, the *pbest* and *gbest* are updated for each particle, and this update process continues iteratively until the desired result is converged upon, or it is determined that an acceptable solution cannot be found within available computational limit, or the predefined maximum number of iterations have been attained.

At iteration *k*, if the position of the *i*-th particle of the swarm across the search space is represented by a *D*-dimensional vector $\mathbf{x}_i(k)$, and the velocity of this particle is given by vector if $\mathbf{v}_i(k)$, the best position found in the search space by particle *i* up to iteration index *k* is represented by another vector $\mathbf{y}_i(k)$, and if the best position found by any of the particles in the neighborhood of particle *i* up to iteration index k is represented by yet another vector $\mathbf{y}^*(k)$; then we can mathematically represent all four vectors as:

$$\mathbf{x}_i\left(k\right)=\left[x_{i,1}(k),x_{i,2}(k),...,x_{i,D}(k)\right] \tag{4.15}$$

$$\mathbf{v}_i\left(k\right)=\left[v_{i,1}(k),v_{i,2}(k),...,v_{i,D}(k)\right] \tag{4.16}$$

$$\mathbf{y}_i\left(k\right)=\left[y_{i,1}(k),y_{i,2}(k),...,y_{i,D}(k)\right] \tag{4.17}$$

$$\mathbf{y}^*(k) = \left[y^*_{i,1}(k), y^*_{i,2}(k), ..., y^*_{i,D}(k) \right] \tag{4.18}$$

At the next iteration index, the position and velocity vectors are updated accordingly, and the new position of particle i can be computed with respect to its previous position $\mathbf{x}_i(k)$, by adding an updated velocity vector to the previous position vector:

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{v}_i(k+1) \tag{4.19}$$

The elements of the updated velocity vector $\mathbf{v}_i(k+1)$ are given by:

$$\begin{aligned} v_{i,j}(k+1) = v_{i,j}(k) &+ c_1 r_{1,j}\left(y_{i,j}(k) - x_{i,j}(k)\right) \\ &+ c_2 r_{2,j}\left(y^*_j(k) - x_{i,j}(k)\right) \end{aligned} \tag{4.20}$$

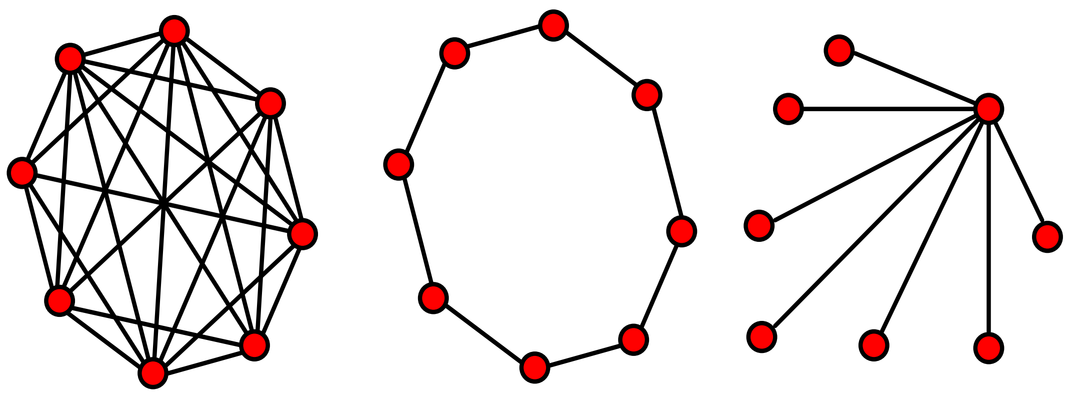
where $j = 1, 2, ..., D$ represents the components or dimension of the search space, c_1 and c_2 are constants respectively called cognitive and social scaling parameters, and $r_{1,j}$ and $r_{2,j}$ are random numbers drawn from a uniform distribution between 0 and 1.

Equations 4.19 and 4.20 represent the classical version of the PSO algorithm as reported in Deep and Bansal (2009). The concept of an inertia weight (ω) was developed to better control the exploration and exploitation abilities of the PSO algorithm. It was incorporated into the algorithm, and was first reported in the literature by Shi and Eberhart (1998). The resulting velocity update equation is given by:

$$\begin{aligned} v_{i,j}(k+1) = \underbrace{\omega v_{i,j}(k)}_{\text{inertia term}} &+ \underbrace{c_1 r_{1,j}\left(y_{i,j}(k) - x_{i,j}(k)\right)}_{\text{cognitive term}} \\ &+ \underbrace{c_2 r_{2,j}\left(y^*_j(k) - x_{i,j}(k)\right)}_{\text{social scaling term}} \end{aligned} \tag{4.21}$$

Equations 4.19 and 4.21 define the standard version of the PSO algorithm. A close look Eq. 4.21 shows it is the sum of three components namely – the inertia, the cognitive and the social scaling components. The inertia component (in apparent reference to its relationship with the inertia weight ω), defines the particle's momentum and it causes the particle to continue in the direction in which it is moving at iteration index k in accordance to the second law of motion. The cognitive component (in apparent reference to its relationship with the cognitive parameter c_1) captures the particle's memory with respect to its previously attained best position; it provides a velocity component in this direction, and is responsible for local search. The third term, which is called the social component (in apparent reference to its relationship with the social scaling parameter c_2), represents information stored in memory about the best position of any particle in the neighborhood of particle *i*, and causes movement towards this particle. This component is responsible for global search. Thus, the position of each particle at every instance is determined by its momentum, its memory, and the collective experience of other particles in the swarm. Of these three components, it appears the social component have the greatest influence on the overall performance of the PSO algorithm. This is because an individual particle (on its own) has little or no power to solve any problem whatsoever; problem-solving can only take place when the particles in the swarm interact. In other words, the effectiveness or otherwise of problem-solving by PSO is a population-wide phenomenon, emerging from the individual behaviors of the particles through their interactions with one another, and in accordance to some sort of communication structure called neighborhood topologies.

The topology typically consists of bi-directional edges connecting pairs of particles, so that if a particle *j* is in the neighborhood of another particle *i*, then particle *i* is also in *j*'s neighborhood. Each particle communicates with some other particles and is affected by the best point found by any member of its topological neighborhood. Several types of PSO neighborhood topologies have been reported; however, it is noted that PSO algorithms with small neighborhoods perform better on complex problems while PSO algorithms with large neighborhoods perform better for simple problems, Kennedy (1999).



(a) Star topology (b) Ring or Circular topology (c) Wheel topology

Figure 4.3: Examples of PSO neighborhood topologies, Das et al. (2008)

The basis of the k-best topology as proposed in Kennedy (1999) is to connect every particle to the k nearest particles in its topological space. Generally speaking, there are as many neighborhoods as there are particles in a given swarm. This is so because each of the particles can form its own neighborhood in its own right.

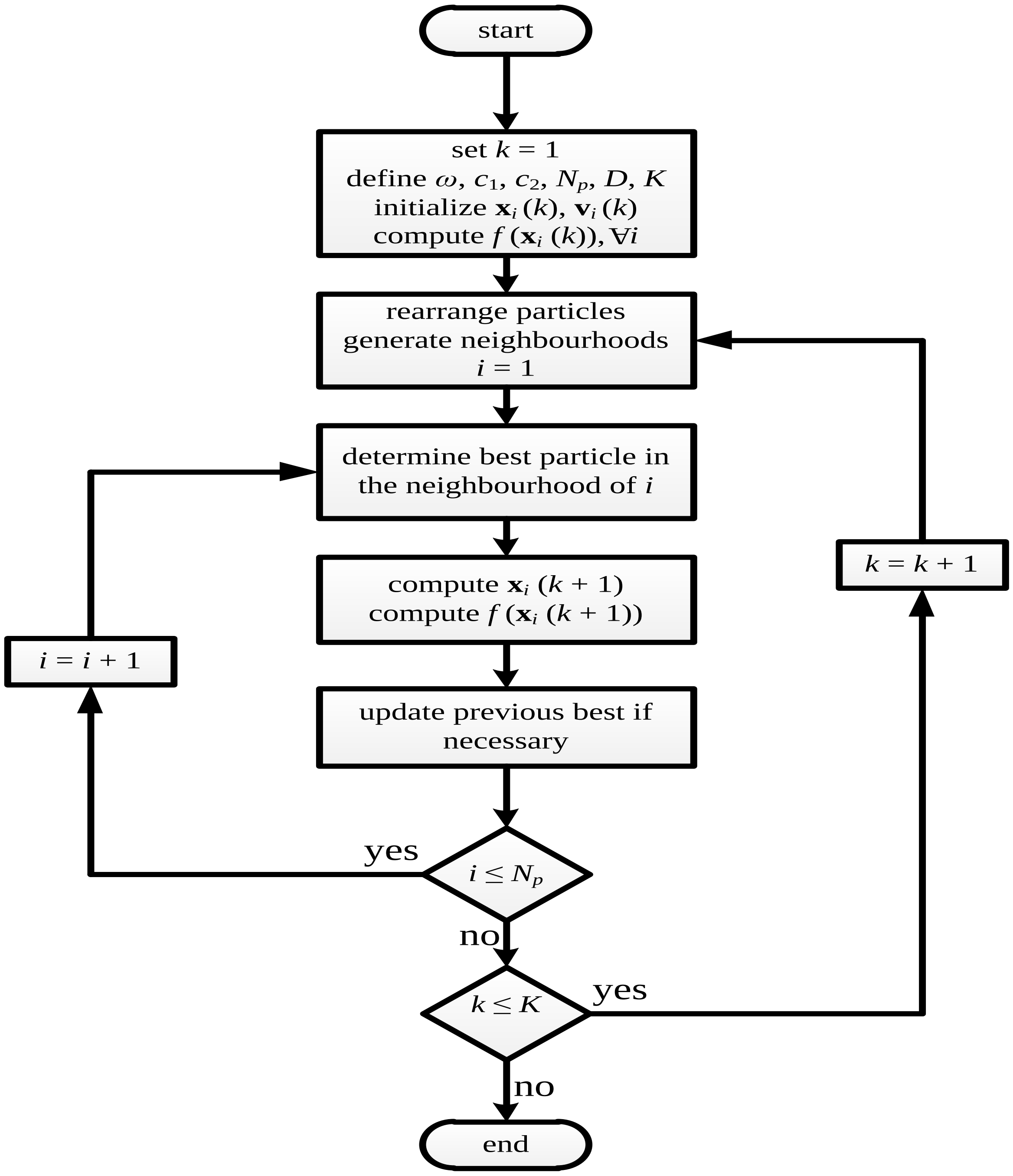

start
set $k = 1$
define ω, c_1, c_2, N_p, D, K
initialize $\mathbf{x}_i(k)$, $\mathbf{v}_i(k)$
compute $f(\mathbf{x}_i(k)), \forall i$
rearrange particles
generate neighbourhoods
$i = 1$
determine best particle in the neighbourhood of i
compute $\mathbf{x}_i(k+1)$
compute $f(\mathbf{x}_i(k+1))$
update previous best if necessary
$i \leq N_p$
yes
$i = i + 1$
no
$k \leq K$
yes
$k = k + 1$
no
end

Figure 4.4: Flowchart showing the PSO algorithm

However, when k = 1, the neighborhoods of individual particles are the same for all particles. In this special case, we have one neighborhood which has a star topology where each particle has a direct link to every other particle in the neighborhood as shown in Figure 4.3a. Although they have fast convergence properties, PSO algorithms using this topology are susceptible to premature convergence, and are generally referred to as "*global best*" or "*gbest*" algorithms, Eberhart and Kennedy (1995), Kennedy (1999) and Engelbrecht (2005). With k = 2, this becomes the circle (or ring) topology where each particle is directly linked to two adjacent particles in its topological space as shown in Figure 4.3b. There are diverse neighborhood topologies that have been reported in the PSO literature – this includes the wheel topology, which effectively isolates the particles from one another, as information is communicated to other particles through a focal (or central) particle as shown in Figure 4.3c. We note that besides the star topology, PSO algorithms using other topologies are referred to as "*local best*" or "*lbest*" algorithms, Engelbrecht (2005). The Figure 4.4 depicts a flowchart illustrating the PSO algorithm.

4.3.1 Treatment of Infeasible Solutions

Like in DE, one of the challenges in implementing PSO in boundary-constrained optimization problems is the issues that arise from infeasible solutions. Direct use of Equations 4.19 and 4.21 on boundary-constrained problems may result in 'solutions' that violate the physical boundary. To handle this issue, we employ the "absorb" technique – Clerc (2006) and Onwunalu and Durlofsky (2010) – by applying the following equations.

$$x_{i,j}(k+1) = \begin{Bmatrix} l_j & \text{if} & x_{i,j}(k+1) < l_j \\ u_j & \text{if} & x_{i,j}(k+1) > u_j \end{Bmatrix} \tag{4.22}$$

$$v_{i,j}(k+1) = \begin{Bmatrix} 0 & \text{if} & x_{i,j}(k+1) < l_j \\ 0 & \text{if} & x_{i,j}(k+1) > u_j \end{Bmatrix} \tag{4.23}$$

where l_j and u_j are the lower and upper bounds of the j-th component of the search space.

While Eq. 4.22 has the effect of moving infeasible solutions to the nearest boundary by setting all variables outside the feasible region to the nearest bound; Eq. 4.23 has the effect of halting the affected particles by setting their velocities to zero. In any case, the "absorb" technique works in a very similar fashion with the "out-of-bound value" technique that we employ in the DE algorithm.

4.3.2 Implementation of PSO in Well Placement Problem

The implementation of the PSO algorithm in well placement optimization problem is given below; it is adopted from Onwunalu and Durlofsky (2010). Algorithm 4.2 presents the steps as implemented in **MATLAB**®. Like Algorithm 4.1; the PSO algorithm presented here is for a

Algorithm 4.2 PSO Algorithm

```
 1: Set iteration index $k = 1$
 2: Define $N_p, \omega = 0.721, c_1 = c_2 = 1.193, K$
 3: Initialize $\mathbf{x}_i(k) : x_{i,j}(k) \sim \mathrm{U}(l_j, u_j) \forall j, \forall i$
 4: Initialize $\mathbf{v}_i(k) : v_{i,j}(k) \sim \mathrm{U}(0,1) \forall j, \forall i$
 5: Compute objective function, $f(\mathbf{x}_i(k)), \forall i$
 6: $\mathbf{y}_i(k) = \mathbf{x}_i(k), \forall i$
 7: while $k \le K$ do
 8:     Permute the particle indices
 9:     Generate neighborhood for each particle
10:     $i = 1$
11:     while $i \le N_p$ do
12:         Determine best particle in neighborhood of particle $i$
13:         Compute $\mathbf{v}_i(k+1)$ using Eq. 4.21
14:         $j = 1$
15:         while $j \le D$ do
16:             $x_{i,j}(k+1) = x_{i,j}(k) + v_{i,j}(k+1)$
17:             Apply Eqs. 4.22 and 4.23 if necessary
18:             $j = j + 1$
19:         end while
20:         Compute objective function, $f(\mathbf{x}_i(k+1)), \forall i$
21:         $i = i + 1$
22:     end while
23:     $i = 1$
24:     while $i \le N_p$ do
25:         if $f(\mathbf{x}_i(k+1)) > f(\mathbf{y}_i(k))$ then
26:             $\mathbf{y}_i(k+1) = \mathbf{x}_i(k+1)$
27:         else
28:             $\mathbf{y}_i(k+1) = \mathbf{y}_i(k)$
29:         end if
30:         $i = i + 1$
31:     end while
32:     $k = k + 1$
33: end while
```

maximization problem. Step 1 signifies the beginning of the algorithm. Step 2 initializes the values of the PSO parameters ω, c_1, c_2, N_p and K. Step 3 initializes each component of the particle position, $x_{i,j}(k)$ with random elements drawn from a uniform distribution U, such that U, $\forall j \in \{1, 2, ..., D\}, \forall i \in \{1, 2, ..., N_p\}$. Step 4 initializes the components of the velocity vector, $v_{i,j}(k)$, in a similar fashion as step 3. Step 5 computes the objective function of all particles. Step 6 updates the previous best position for each particle. The particle indices (positions of particles in the array of particles) are permuted in step 8, and the neighborhoods for each particle are generated in step 9. Step 12 determines the best particle in the neighborhood of particle *i*. The elements of the updated velocity vector of new particle $\mathbf{v}_{i,j}(k+1)$, are computed in accordance with (4.21) in step 13. Steps 15–19 update all components of the position of particle *i*. In step 17, infeasible solutions are modified using (4.22) and (4.23). Step 20 evaluates the objective function $f(\mathbf{x}_i(k+1))$ based on the new particle position. Steps 24–31 update the previous best positions for each particle, $\mathbf{y}_i(k)$, if the new objective function value, $f(\mathbf{x}_i(k+1))$, is better than that at the previous best position, $f(\mathbf{y}_i(k))$. The algorithm terminates at K – the maximum number of iterations.

4.4 Hybridization of Metaheuristic Algorithms and Hybrid Particle Swarm Differential Evolution (HPSDE)

Though classified as global optimization techniques, DE and PSO have their own drawbacks. They are susceptible to premature convergence which can lead to potential solutions being trapped in local optima. This disadvantage is much more pronounced in domains where the search space is nonlinear, non-continuous and non-smooth, as often the case in many reservoir engineering applications. To overcome these drawbacks, researchers have over the years employed various hybridization techniques to create hybrid metaheuristic algorithms that are more robust and effective in problem solving. In a general sense, hybridization is simply an attempt at combining the good traits of participating algorithms or concepts, with the ultimate view of improving the efficiency and capabilities of the newly created 'hybrid' algorithm. We justify the use of hybridization as a direct consequence of the so called "*no free lunch*" theorem. In Wolpert and Macready (1997), it was established that any elevated performance over one class of problem by any algorithm, is offset by performance over another class of problem. This underlines the fact that no single optimization technique can solve all optimization problems optimally.

Generally speaking, most of the hybrid metaheuristics that have been published in the literature can be loosely grouped into three categories – those created by combining one metaheuristic algorithm with another metaheuristic algorithm; those developed by combining a standard metaheuristic algorithm with mathematical operators; and those developed by incorporating evolutionary operators (selection, mutation and crossover) into non-evolutionary metaheuristic algorithms.

Among the three evolutionary operators, mutation appears to be the most commonly applied operator in the hybridization of non-evolutionary metaheuristic algorithms such as PSO. The purpose of applying mutation to PSO is to increase the diversity of the population and enable the PSO to escape local optima, Blackwell and Bentley (2002), Krint et al. (2002), Lovbjerg and Krink (2002), Miranda and Fonseca (2002), Higashi and Iba (2003) and Ratnawera et al. (2004). In Juang (2004), mutation alongside crossover and elitism is incorporated into PSO; and the resulting algorithm outperformed both PSO and GA in recurrent network design problem. The selection operator (which entails copying the particles with the best performance into the next generation) was applied to a PSO algorithm in Angeline (1998), and this led to a continuous retention of the best performing particles; and Lovbjerg and Rasmussen (2001) showed that incorporating the crossover operation in PSO algorithms effect information-swap between individual particles. A quadratic approximation (QA) operator was used to hybridize a binary GA and PSO in Deep and Das (2008) and Deep and Bansal (2009) respectively. In both cases, the QA operator is used to update a part of the population while the remaining of the population is updated by either GA or PSO as the case may be. In Poli et al. (2005a, 2005b), a hybrid PSO based on genetic programming (GP) was proposed. The GP is used to evolve new laws for the control of particles' movement for specific classes of problems. Ant colony optimization (ACO) was combined with PSO by Hendtlass and Randall (2001); while DE and PSO are combined by Hendtlass (2001). The particles in the swarm drift according to position update equation, but occasionally DE is applied to replace poorly performing particles while retaining their velocity. The DEPSO algorithm in Zhang and Xie (2003) involves the use of DE and PSO operators in alternate iterations. The hybrid achieved better results than PSO in problems with high dimensionality.

From the foregoing, it is evident that a wide array of hybridized metaheuristic algorithms have been designed and implemented for the purpose of improving the performance and problem-solving capabilities of the participating algorithms. A comprehensive (but not exhaustive) review is available in Banks et al. (2008).

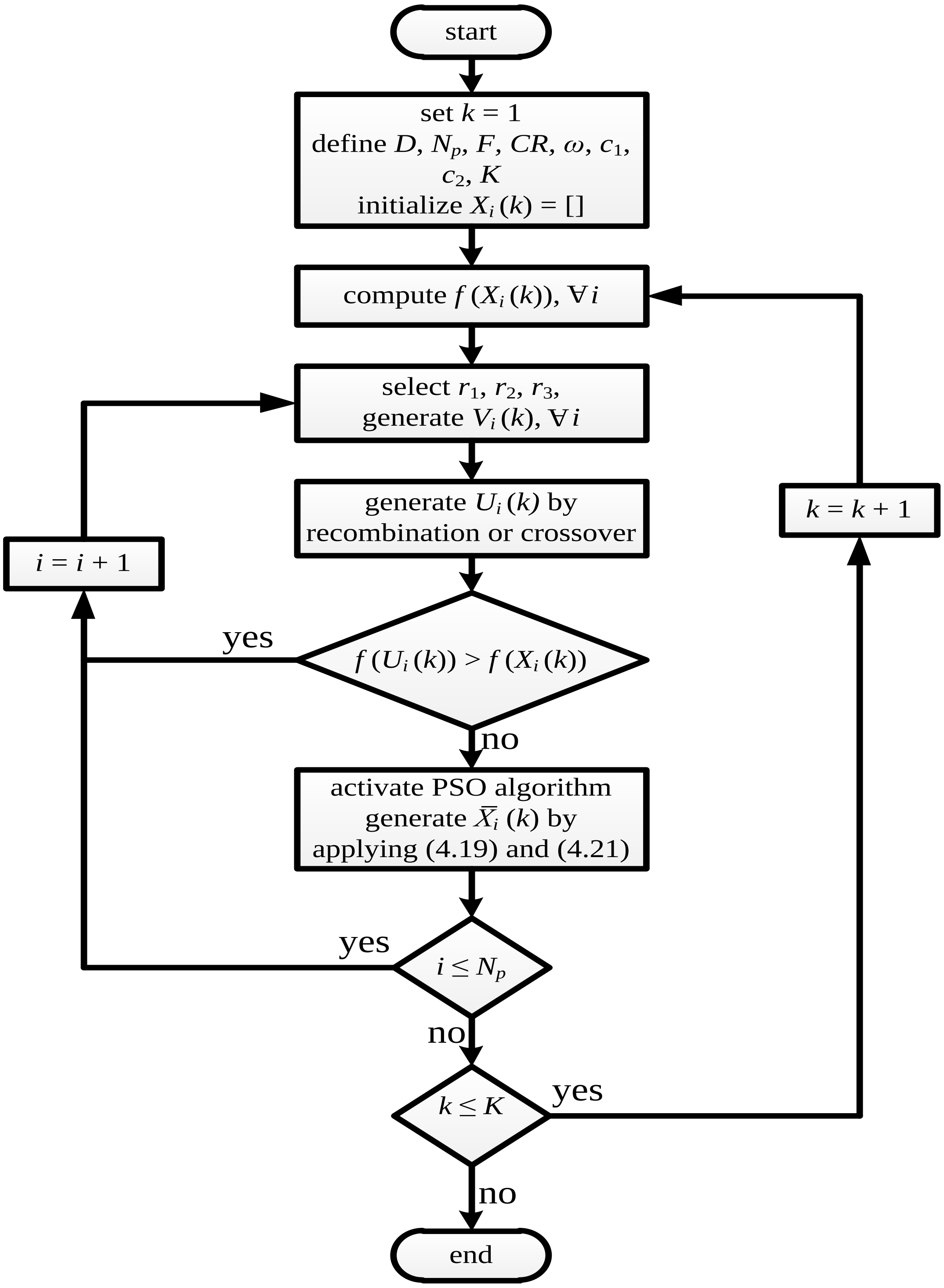

start
set $k = 1$
define D, N_p, F, CR, ω, c_1, c_2, K
initialize $X_i(k) = []$
compute $f(X_i(k))$, $\forall i$
select r_1, r_2, r_3,
generate $V_i(k)$, $\forall i$
generate $U_i(k)$ by recombination or crossover
$k = k + 1$
$i = i + 1$
yes
$f(U_i(k)) > f(X_i(k))$
no
activate PSO algorithm
generate $\bar{X}_i(k)$ by applying (4.19) and (4.21)
yes
$i \leq N_p$
no
$k \leq K$
yes
no
end

Figure 4.5: Flowchart illustrating the HPSDE algorithm

In this work however, we employ a hybrid of DE and PSO referred to as hybrid particle swarm differential evolution (HPSDE). This is a modified version of the algorithm proposed by Zhang and Xie (2003) and Thangaraj et al. (2011). It starts off as a standard DE algorithm up to the point where the trial vectors are generated. If the fitness of the trial vector is better than the corresponding target vector, then it is included in the population; otherwise, the algorithm activates the PSO phase and generates a new candidate solution using the position and velocity update equations. The method is repeated iteratively with the hope of finding better solutions or until the maximum number of iteration is reached. The inclusion of PSO loop creates a perturbation in the population; this in turn helps in maintaining diversity of the population and producing better solutions, Thangaraj et al. (2011). The HPSDE flowchart is depicted in Figure 4.5.

4.4.1 Treatment of Infeasible Solutions

Like in DE and PSO algorithms, a major challenge with the implementation of HPSDE algorithm and indeed most metaheuristic algorithms in boundary-constrained optimization problems is the issue arising from infeasible solutions. This is an inevitable consequence of the stochastic nature of these algorithms. Since HPSDE is a hybrid of DE and PSO algorithms, solutions that violate the real physical search space are treated by calling Eq. 4.13, or Equations 4.22 and 4.23 as the case may be. Depending on the stage of the algorithm at which the bound-offending solution occurs; Eq. 4.13 is utilized if it occurs at the DE stage, or Equations 4.22 and 4.23 if we have to deal with it in the PSO stage of the algorithm.

4.4.2 Implementation of HPSDE in Well Placement Problem

The HPSDE algorithm is a hybrid of DE and PSO. It begins with DE algorithm up to the point where the trial vectors are generated. The fitness of the trial vector is compared with that of the corresponding target vector to determine if it is included in the population of the next iteration or if it is updated using a global best PSO algorithm. By so doing, we combine the global information obtained via PSO algorithm into DE algorithm, thereby maintaining a fair balance between the exploration and exploitation factors of the algorithms. Algorithm 4.3 presents the steps in HPSDE for a maximization problem. It is adapted and modified from Thangaraj et al. (2011), and implemented in **MATLAB**®. The algorithm begins in step 1 with the initialization of the first iteration index $k=1$. Step 2 assigns values to DE and PSO parameters. Step 3 initializes a population of vectors $X_i(k)$ such that each component $x_{i,j}(k), \forall i \in \{1,2,...,N_p\}, \forall j \in \{1,2,...,D\}$ are made of random elements drawn from the pred–

Algorithm 4.3 HPSDE Algorithm

```
 1: Set iteration index $k = 1$
 2: Define $D, N_p, F, K, CR, \omega, c_1, c_2$
 3: Initialize $X_i(k) : x_{i,j}(k) \sim$ rand $(L_j, U_j) \forall j, \forall i$
 4: Compute objective function, $f(X_i(k)), \forall i$
 5: **while** $k \le K$ **do**
 6:     Save, $f(X_i(k)), \forall i$
 7:     $i = 1$
 8:     **while** $i \le N_p$ **do**
 9:         Select $r_1, r_2, r_3 \in \{1,2,...,N_p\}, r_1 \ne r_2 \ne r_3 \ne i$
10:         Randomly select $j \in \{1,2,...,D\}$
11:         **while** $j \le D$ **do**
12:             Compute mutant vector $v_i(k), \forall i$
13:         **end while**
14:         Apply Eq. 4.11 to generate trial vector $U_i(k), \forall i$
15:         Apply Eq. 4.13 if necessary
16:         Compute objective function, $f(U_i(k)), \forall i$
17:         $i = i + 1$
18:     **end while**
19:     $i = 1$
20:     **while** $i \le N_p$ **do**
21:         **if** $f(U_i(k)) > f(X_i(k))$ **then**
22:             $X_i(k+1) = U_i(k)$
23:         **else** activate PSO algorithm
24:             $j = 1$
25:             **while** $j \le D$
26:                 $\mathbf{v}_i(k+1) = \omega v_i(k) + c_1 r_1(pbest_i(k) - X_i(k)) + c_2 r_2(gbest_i(k) - X_i(k))$
27:                 $\bar{X}_i(k) = X_i(k) + \mathbf{v}_i(k+1)$
28:             **end while**
29:         **end if**
30:         **if** $f(\bar{X}_i(k)) > f(X_i(k))$ **then**
31:             $X_i(k+1) = \bar{X}_i(k)$
32:         **else**
33:             $X_i(k+1) = X_i(k)$
34:         **end if**
35:         $i = i + 1$
36:     **end while**
37:     $k = k + 1$
38: **end while**
```

efined lower (*L*) and upper (*U*) bounds in accordance to Equations 4.7 and 4.8. Step 4 computes the objective function of the initialized population, and the computed objective function is saved in step 6 for future reference. In steps 9–12, we compute a mutant vector $v_i(k)$ in accordance with Eq. 4.9; while a trial vector U_i is generated in step 14 in accordance with Eq. 4.11. If any element of the trial vector is outside the feasible region of the search space, step 15 is activated in order to modify and adjust the trial vector within the feasible region. In step 16, the objective function of the trial vectors is evaluated. Steps 21 and 22 compare the objective function of the trial vector with that of the corresponding target vector in order to determine the population of the next iteration. The fitness value of the trial vector must be greater than the fitness value of the target vector in order to make it to the next iteration index; otherwise, the algorithm uses the PSO velocity and position update equation to generate new candidate solution as illustrated in steps 23–27. In steps 30–33, the objective function of the newly generated vectors $\bar{X}_i(k)$ is evaluated and compared with the fitness of corresponding target vectors to determine the population of the next iteration. The algorithm continues iteratively until it terminates when the maximum number of iterations *K* is attained.

4.5 Objective Function Evaluation and Applications

The objective function employed in the well placement applications in this work is the net present value (NPV) as presented earlier in the problem formulation section, and as mathematically stated in Eq. 4.1. The values of the parameters for the NPV computation are given in Table 4.1; and we assume that all time-dependent parameters such as d, $p_{(t)}^{\mathrm{oil}}$, $p_{(t)}^{\mathrm{w,p}}$ and $p_{(t)}^{\mathrm{w,i}}$ are constant over the operational interval [0, *T*].

Table 4.1: NPV Computation Parameters		
Parameters	**Symbol**	**Value**
Price of oil	p^{oil}	\$50 / bbl
Water production cost	$p^{\mathrm{w,p}}$	\$10 / bbl
Water injection cost	$p^{\mathrm{w,i}}$	\$5 / bbl
Discount factor	d	0.1
Drilling cost per meter	$\mathrm{C}^{\mathrm{drill}}$	$\$5.3\times10^4$/m
Drilling cost to reservoir top	$\mathrm{C}_{\mathrm{w}}^{\mathrm{top}}$	$\$50\times10^6$

Table 4.1: NPV computation parameters

In the first application, five optimization runs of the algorithms are performed, while thirty optimization runs of the algorithms are performed in the second and third applications. The results are subsequently averaged over the number of runs corresponding to each application so as to reflect the relative performance of each algorithm. The choice of five runs for the first application is based on suggestion in Vasiljevic and Golobic (1996) and Ciaurri et al. (2011); while the choice of thirty runs in the second and third applications rests on the need to carry out a more systematic and reliable performance evaluation of the algorithms using well-established statistical indices.

It is noted that the importance of comparing the average performance (over multiple optimization runs) of these algorithms stems from their non-deterministic nature; and the need to reduce the effects caused by different distribution of the initial solutions, as well as the randomness resulting from the probabilistic operators in the algorithms. To further reinforce fairness in the comparisons; the control parameters used in each of the three algorithms are the same in all the problems considered. By and large, these factors afford us the level-playing platform to draw a more general conclusion with respect to the performance of each of the algorithms.

Also, it is noted that all applications in this chapter are model-based, and the simulations are performed using **MRST®**. Since the applications are model-based, we note the inevitable presence of geological uncertainty – a direct consequence of the fact that reservoir models are a 'crude' approximation of real physical petroleum reservoirs – as highlighted in chapter 3. To address this mismatch between the physical reservoir and the reservoir model, we employ a robust optimization strategy in the first two applications.

By robust optimization, the optimization procedure is carried out over a set of realizations which explicitly accounts for the geological uncertainty in the models – see van Essen et al (2006). The robust optimization objective adopted is the *max-mean objective* – which basically seeks to maximize the average performance measure associated with each of the realizations. It is given by:

$$\langle \mathrm{NPV} \rangle = \left(\frac{1}{R} \sum_{i=1}^{R} \mathrm{NPV}_i \right) \tag{4.24}$$

where $\langle \mathrm{NPV} \rangle$ is the expected NPV, and R is the number of geological realization.

4.5.1 Application 1: Placement of a Single Producer

In the first application, we consider optimizing the placement of a single producer well in a 2D reservoir model with $45 \times 45 \times 1$ grid-blocks of dimensions $10\text{m} \times 10\text{m} \times 10\text{m}$ as shown in Figure 4.6. In order to address the mismatch between the model and the physical reservoir; we incorporate geological uncertainty by considering 10 realizations of the model. Usually, it is common to decide upon a few sources of uncertainty that presumably have the largest impact on the model; in this regard, we choose the permeability distribution in the grid-blocks of the reservoir model. Thus, the realizations are generated based on varying permeability distribution, the remaining nine realizations of the reservoir model are depicted in Figure 4.7.

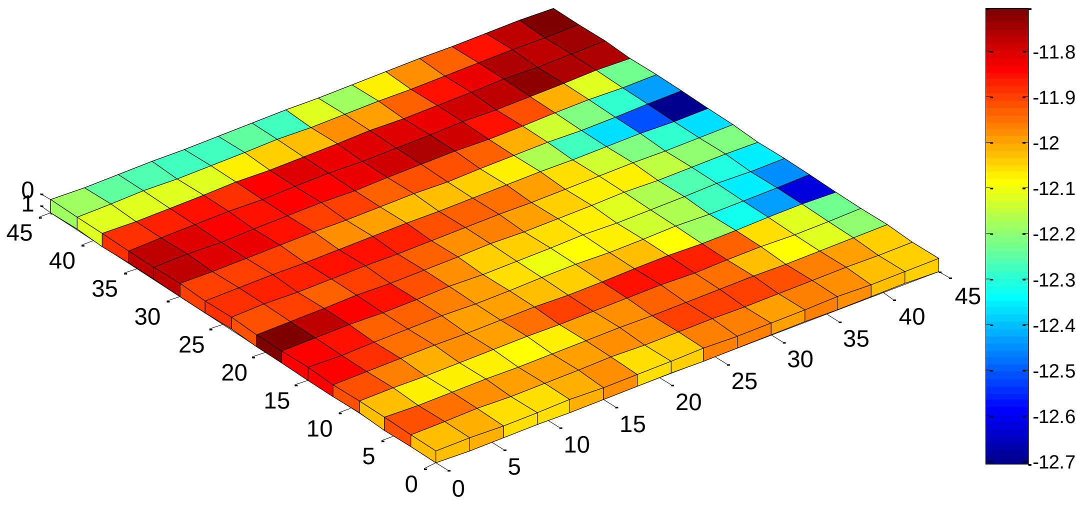


Figure 4.6: Reservoir model of 45×45×1 grid-blocks used in Application 1.

The system contains oil and connate water; the initial pressure and saturation (connate water saturation) are $350 \times 10^5\text{Pa}$ and 0.2 respectively; and both are uniform throughout the reservoir model. There are no aquifers, and no water injection; thus, only oil is produced. The remaining system properties are given in Table 4.2. The reservoir is simulated for 10 years; with the single producer placed in grid block position corresponding to the results obtained from five runs of each of the algorithms, and constrained to operate at a bottom hole pressure (BHP) of $65 \times 10^5\text{Pa}$. The cumulative volume of oil produced is used to compute the NPV (by applying Eq. 4.1) corresponding to each optimization run of the algorithms, and the computed NPVs are averaged over the number of runs to reflect the relative performance of each algorithm. In the same way, the average NPV achieved by each of the algorithm is computed for the remaining realizations; and using Eq. 4.24, $\langle \text{NPV} \rangle$ of the reservoir model is computed for each of the algorithm.

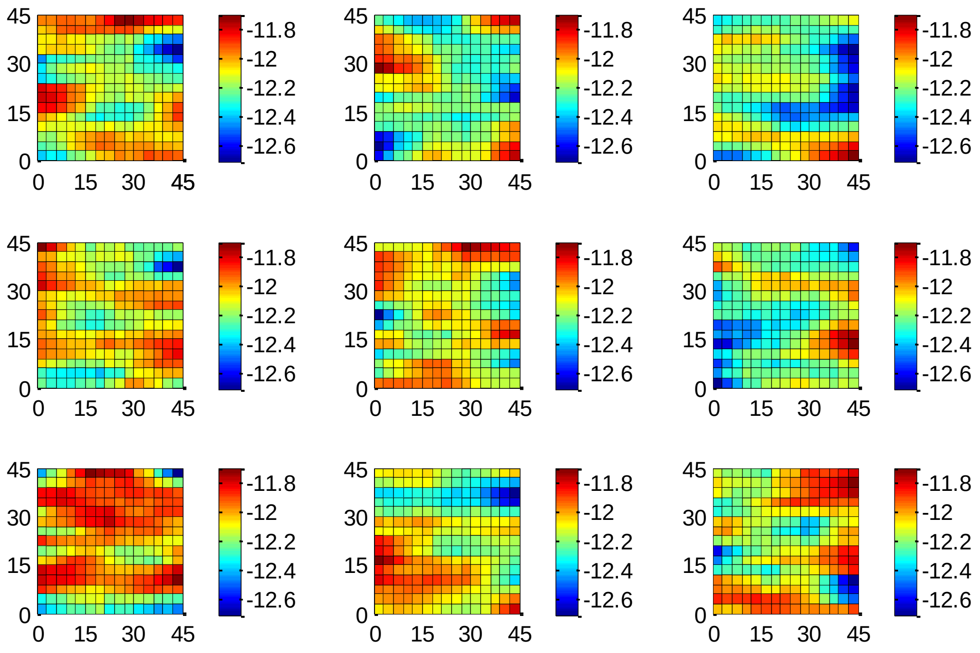


Figure 4.7: Remaining realizations of Application 1 model – realizations are based on varying permeability

Table 4.2 Systems properties		
Properties	**Symbol**	**Value**
Porosity	ϕ	0.3
Oil Viscosity	μ_o	10^{-3} Pas
Oil Compressibility	c_o	10^{-10} Pa^{-1}
Rock Compressibility	c_r	1.8×10^{-10} Pa^{-1}

Table 4.2: System Properties and their values

We perform a sensitivity analyses in order to examine the effect of different population size and iteration number combination on the performance of the three algorithms. To this end, we consider six population size and iteration combinations – (5,10), (10,10), (20,10), (10,20), (10,40) and (10,80) respectively. Note that in three of the six population size and iteration combinations, the number of iteration is held constant (while the population size is varied); and the population size is held constant (while the maximum number of iteration is varied) in the remaining three combinations. For each of the population size and iteration combination, we perform five optimization runs of the algorithms, and compute the average NPV relating to each algorithm. Figure 4.8 shows the corresponding $\langle \text{NPV} \rangle$ for each algorithm against the total number of simulations per realization, which is given by $N_p \times K$. It is obvious in all six population size and iteration number combinations that HPSDE (red line) algorithm outperf–ormed both DE (blue line) and PSO (green line) algorithms.

The advantage of HPSDE algorithm over DE and PSO algorithms was much pronounced in cases where the population size and iteration combination were low. For example, at the respective population size and iteration combination of 5 and 10; an $\langle \text{NPV} \rangle$ of $\$175.8 \times 10^6$ was attained for HPSDE. This represents a 24% increase in the $\langle \text{NPV} \rangle$ of $\$141.7 \times 10^6$, and a 52% increase in the $\langle \text{NPV} \rangle$ of $\$115 \times 10^6$ achieved by PSO and DE algorithms respectively. The comparative advantage of the performance of HPSDE over PSO and DE algorithms becomes less remarkable as the total number of simulations per realization (given by $N_p \times K$) increases from 50 to 800. For a population size of 10, and maximum iteration of 80; HPSDE algorithm achieved an $\langle \text{NPV} \rangle$ of $\$303.7 \times 10^6$, whereas PSO and DE algorithms attained a near-convergence $\langle \text{NPV} \rangle$ of $\$261 \times 10^6$ and $\$259.9 \times 10^6$ respectively. This performance of PSO and DE algorithms is 14% less than the performance of HPSDE.

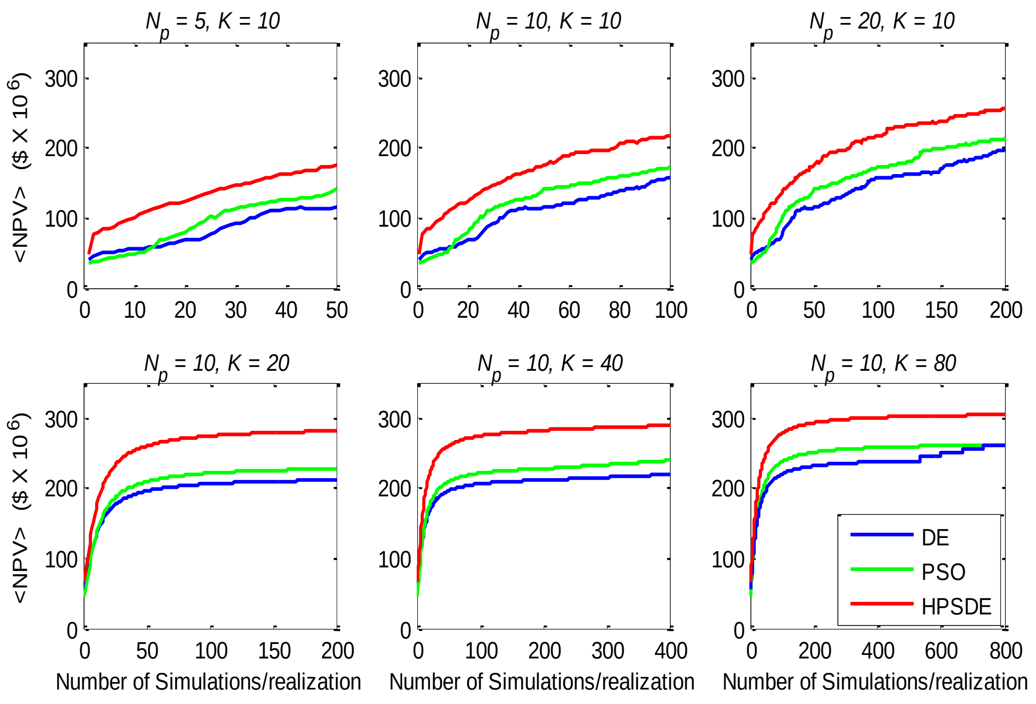


Figure 4.8: <NPV> of DE, PSO and HPSDE versus number of simulations per realization for different population size and maximum iteration number combinations

Another interesting observation from the view points of the DE and PSO algorithms is that the performance of one algorithm over the other in this problem appears to be dependent on the total number of simulations. In all six population size and iteration number combinations, the DE algorithm attained higher $\langle \text{NPV} \rangle$ than the PSO algorithm at very low simulation per realization. For example, in the first population size and iteration combination (i.e. $N_p = 5, K = 10$), DE outperforms PSO algorithm from the start till 15 simulations per realization. Beyond this 'threshold' number of simulations, PSO achieved better $\langle \text{NPV} \rangle$ than DE. This pattern is observed in all six population size and iteration number combination. However, the 'threshold' number of simulation after which PSO outperforms DE varies from one population size and iteration number combination to the other. In the last population size and iteration combination (i.e. $N_p = 10, K = 80$), although the PSO outperformed the DE from simulation per realization of 22, both algorithms achieved near-convergent $\langle \text{NPV} \rangle$ of $\$261 \times 10^6$ and $\$259.9 \times 10^6$ from a total simulation per realization of 744 to 800. In any case, however, the HPSDE algorithm outperforms both the DE and PSO algorithms.

4.5.2 Application 2: Placement of a Producer and an Injector

In this application, we consider optimizing the placement of a producer and an injector in a 2D reservoir model where the oil is to be replaced by water in a simple waterflooding production operation. The reservoir model has $50 \times 50 \times 1$ grid-blocks, each grid-block has dimensions $10\text{m} \times 10\text{m} \times 10\text{m}$ as shown in Figure 4.9, and the residual oil and connate water saturation is 0.2.

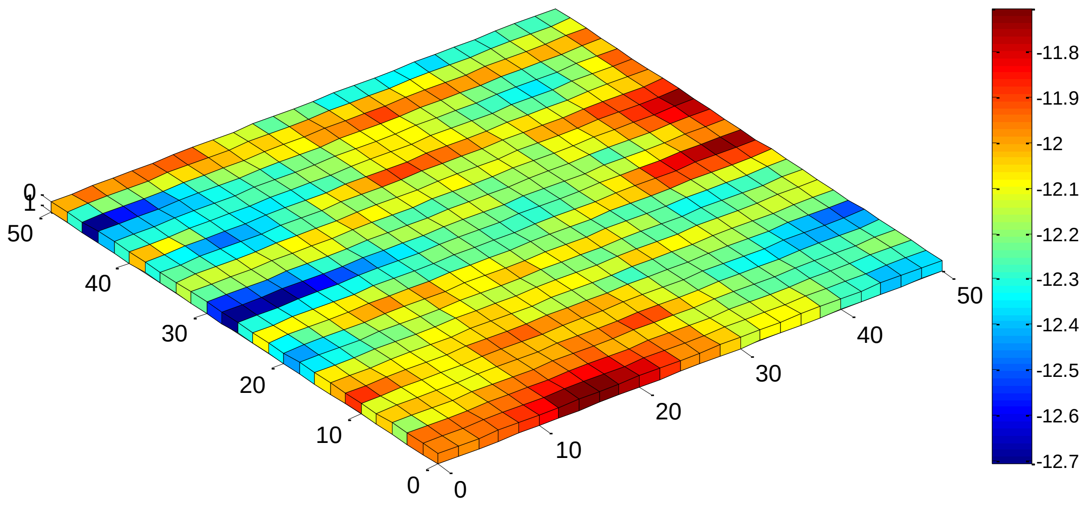


Figure 4.9: Reservoir model of 50×50×1 grid-blocks used in Application 2

Unlike in application 1 (which did not require relative permeability), we employ the Corey model for relative permeability, with Corey exponents $n_o = n_w = 2.45$ and relative permeability endpoints for oil and water 0.9 and 0.65 respectively. The relative permeability curve used is as depicted in Chapter 3 – see Figure 3.3; the water viscosity (μ_w) is 10^{-3} Pa^{-1}, and the remaining system properties are same as given in Table 4.2. Thirty optimization runs of the algorithms are performed in this problem, and the solutions from each of the run is applied on five realizations of the model – this ensures that results are robust against geological uncertainty. Figure 4.10 depicts the remaining realizations of the reservoir.

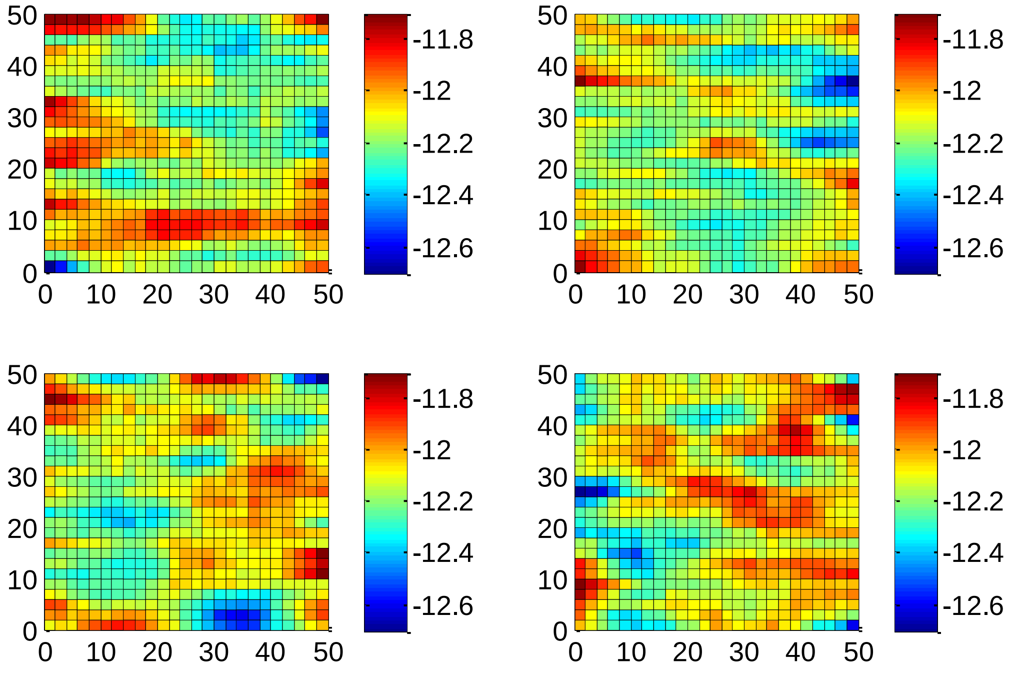


Figure 4.10: Remaining realizations of Application 2 model – realizations are based on varying permeability

In mathematical probability, we often encounter occupancy problems where the task is to find the total number of possible placement of *m* different balls into *n* bins (read: placement of two non-identical wells in 2500 possible grid-blocks). However in this case, we are not interested in the total number of possible outcomes; we are only interested in the outcome that yields the highest NPV in each realization of the reservoir model.

For each of the wells, there are three optimization variables $\{x, y, I\}$, which results in a total of six variables. The Cartesian coordinates of each well location is represented by the variables *x* and *y* while the variable $I \in (0,1)$ is a binary indicator that represents the well type (i.e. $I = 0$ designates a production well, and $I = 1$ designates an injection well). Such binary indicator was employed in Yeten (2003); and we have adopted it in this work because of simplicity and ease of implementation.

Using a population size of 25 and maximum iterations of 120; thirty optimization runs of DE, PSO and HPSDE algorithms are performed to determine the placement of the injector and producer. It is instructive to note that the choice of this large population and iteration size combination is informed on the basis of the inference made from the sensitivity analyses performed in the first application – where it was observed that the comparative advantage of HPSDE over PSO and DE algorithms was less remarkable as $N_p \times K$ increases.

Subsequently, both wells are placed at locations corresponding to the solutions from each run of the algorithms; and the system is simulated for ten years, with the producer constrained to operate at a BHP of 65×10^5Pa, and the injector operating at a BHP of 140×10^5Pa. The initial pressure map of from a model realization with best performance of the algorithms is shown in Figure 4.11.

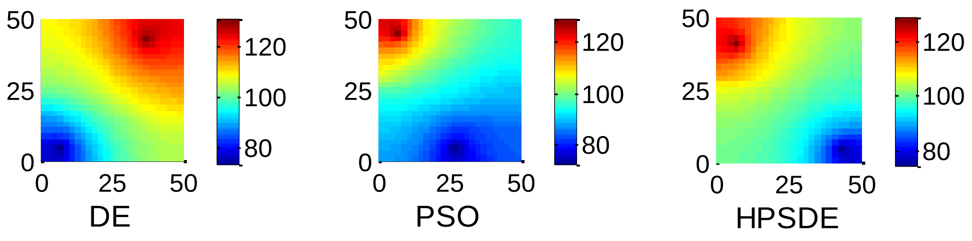


Figure 4.11: Initial pressure map from realization with best performance of DE, PSO and HPSDE runs

The volumes produced are used to compute the NPV in accordance with Eq. 4.1; and the computed NPVs are averaged over the number of runs to determine the average NPV associated with each algorithm. Accordingly, this is repeated for each of the five realizations of the model; and the $\langle \text{NPV} \rangle$ for the reservoir model corresponding to each algorithm is computed using Eq. 4.24. The results are plotted and shown in Figure 4.12.

The DE and the PSO algorithms yielded $\langle \text{NPV} \rangle$ of $\$544.2\times10^6$ and $\$534.7\times10^6$ respectively, and both performance values were below the $\langle \text{NPV} \rangle$ of $\$575.4\times10^6$ that was achieved using the HPSDE algorithm. In fact, the performance of HPSDE is 5.7% higher than DE; and 7.6% higher than PSO. Again, it was observed that DE achieved better results than PSO when the number of simulation was below a threshold value. This is consistent with the pattern observed in the first application. Beyond this threshold number of simulations (118 in this case), PSO outperforms DE until after 2058 simulations, when both algorithms begin to converge to same-value ⟨NPV⟩.

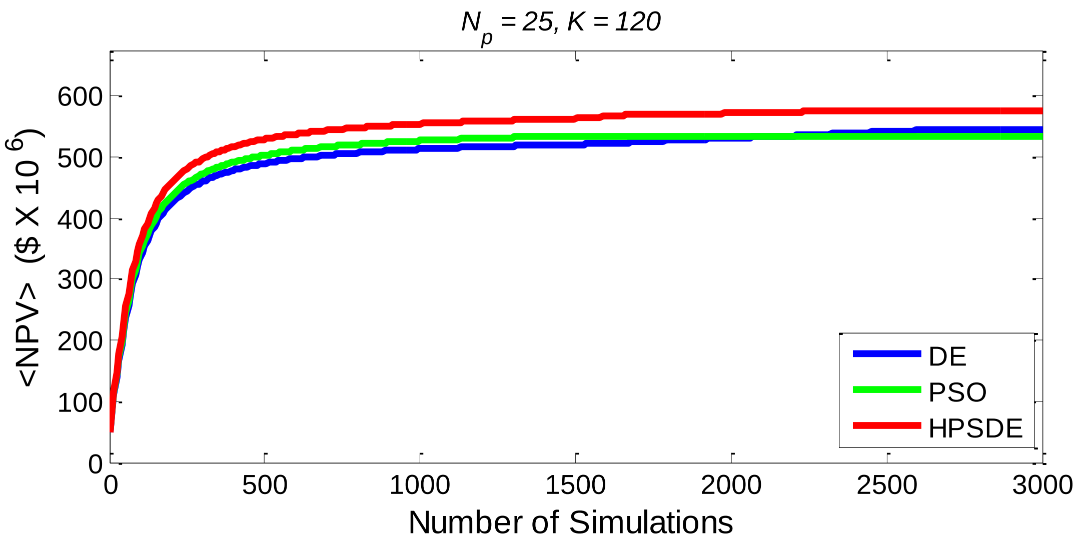


Figure 4.12: <NPV> of DE, PSO and HPSDE algorithms versus number of simulations for application 2

We also note that DE attained nominally better ⟨NPV⟩ than PSO after 2452 simulations. In any case however, HPSDE achieved better ⟨NPV⟩ values than both DE and PSO algorithms. The water saturation maps of the reservoir from the best run of the algorithms are shown in Figure 4.13.

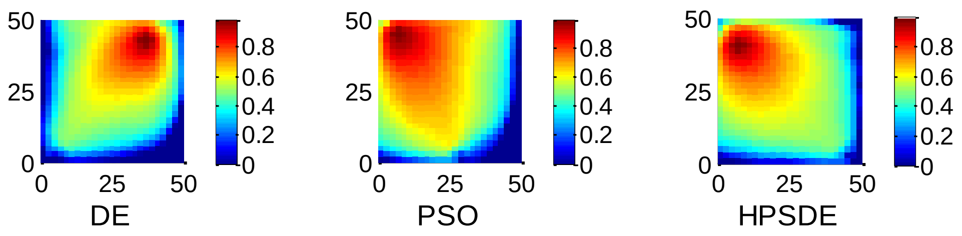


Figure 4.13: Water saturation map from realization with best performance of DE, PSO and HPSDE runs

Now, since this application involves five realizations of the reservoir model, there is a total of 450 NPVs resulting from the optimization runs of all three algorithms under consideration. With the aid of statistical indices such as best performance, worse performance, mean performance and standard deviation; these performance measures are further analyzed based on the underlying algorithm from which they emanated, and the result of this analysis is presented in Table 4.3. The HPSDE algorithm achieved both the best and worst NPVs and these were attained in the twenty-seventh and eighth run of the algorithm respectively. The fact that the NPV resulting from the eighth run of HPSDE in one of the realizations of the model was lower than all the NPVs resulting from each of the thirty runs of both DE and PSO algorithms reinforces the importance and need of multiple runs.

	Statistical Indices ($\$\times 10^6$)			
Algorithms	**Best NPV**	**Worse NPV**	**Mean NPV**	**STD**
DE	594.6281	354.3367	503.9902	66.7238
PSO	591.1008	355.9117	509.9053	70.9002
HPSDE	618.8412	351.6558	521.8678	64.7480

Table 4.3: Statistical analysis of results from 30 optimization runs of each algorithm

Of interest again is the fact that the HPSDE algorithm yielded the lowest standard deviation over the entire number of runs. Invariably, this means that the data points in HPSDE algorithm tends to be close to the arithmetic mean of its distribution than those of DE and PSO. Thus, it can be inferred that for all three algorithms under consideration, the probability of attaining near-average result is higher when HPSDE is employed in this problem domain.

4.5.3 Application 3: Placement of 9 Wells in 3D Reservoir

A 3D reservoir model is considered in this application. It contains $50\times50\times8$ grid blocks of dimensions $10\text{m}\times10\text{m}\times10\text{m}$. The fluid and geological properties of this model are same as those in application 2; and Figure 4.14 shows the permeability distribution of the model. The task is to determine the optimal type and placement of 9 well for a waterflooding operation. In this example, we ignore the effects of geological uncertainty; therefore, only one realization of the model is employed, and the performance measure is the simple NPV resulting from the fluid profile generated.

Like in application 2, each of the nine wells has three optimization variables $\{x, y, I\}$, and these results in a total of 27 variables. The variables x and y represent the Cartesian coordinates of each well location; and the variable $I \in (0,1)$ is a binary indicator that represents the well type (producer or injector). Using a population size of 60 and maximum iterations of 150; thirty runs of DE, PSO and HPSDE algorithms are used to determine the placement of the wells. The system is simulated for 10 years, with the producers constrained to operate at a BHP of $65\times10^5\,\text{Pa}$, and the injectors at $100\times10^5\,\text{Pa}$.

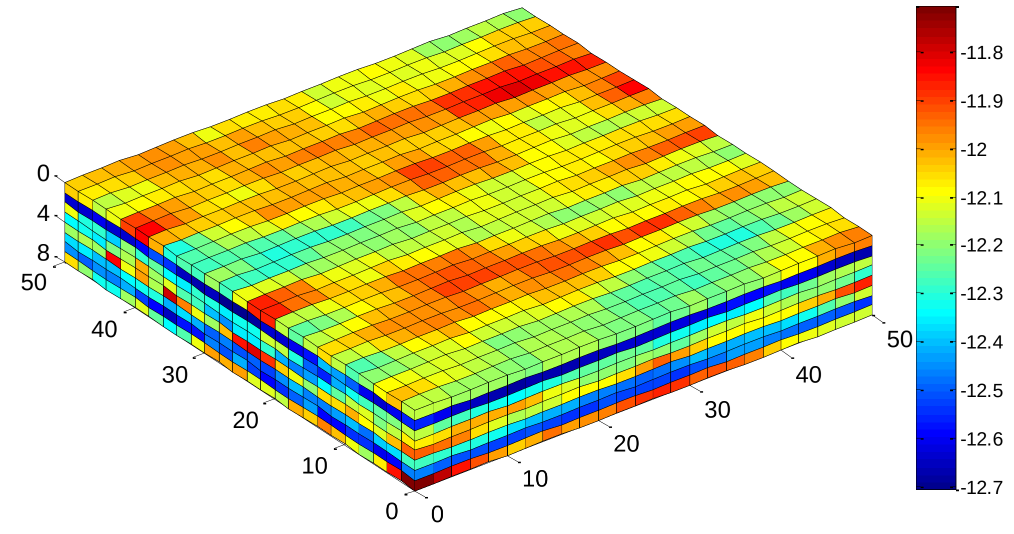


Figure 4.14: Permeability field for application 3

Interestingly, it was observed that there were more producers than injectors in the best optimization runs of the algorithms. In this regard, the well split of the producers to injectors in the best optimization run for all three algorithms is 6:3; and these best runs were achieved in the nineteenth, fourth and twenty-second run of DE, PSO and HPSDE algorithms respectively. The 2D and 3D pressure and water saturation maps from well location obtained from best runs of the algorithms are shown in Figures 4.15 and 4.16 respectively.

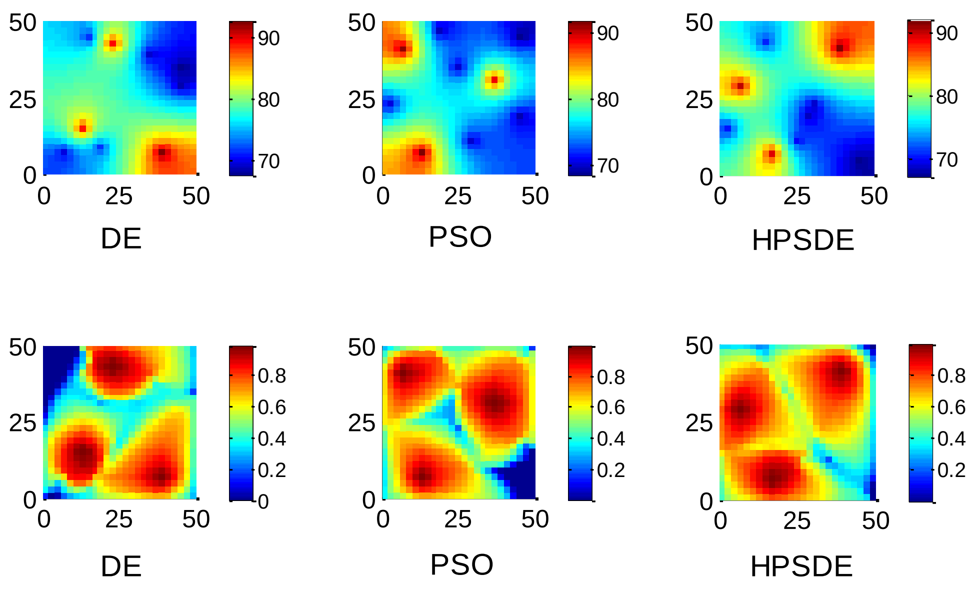


Figure 4.15: 2D maps of initial pressure (top row) and water saturation (bottom row) from best optimization runs of DE, PSO and HPSDE

The generated fluid profiles are used to compute the NPV corresponding to each optimization run of the algorithms by applying Eq. 4.1. Subsequently, the computed NPVs are averaged over the number of optimization runs to determine the average performance or NPV associated with each of the algorithm. The results are plotted and shown in Figure 4.17.

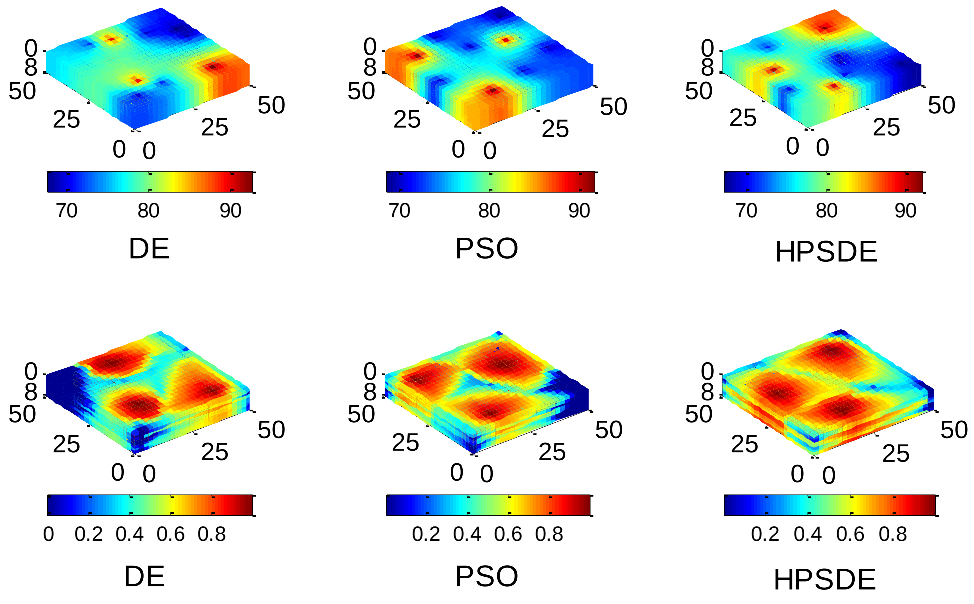


Figure 4.16: 3D maps of initial pressure (top row) and water saturation (bottom row) from best optimization runs of DE, PSO and HPSDE

At the end of the total number of simulations, the performance of DE algorithm was higher than that of PSO algorithm. It achieved a maximum NPV of $\$47.87\times10^9$; which represents a 2.6% increase in the NPV of $\$46.67\times10^9$ attained by PSO algorithm. And with an NPV of

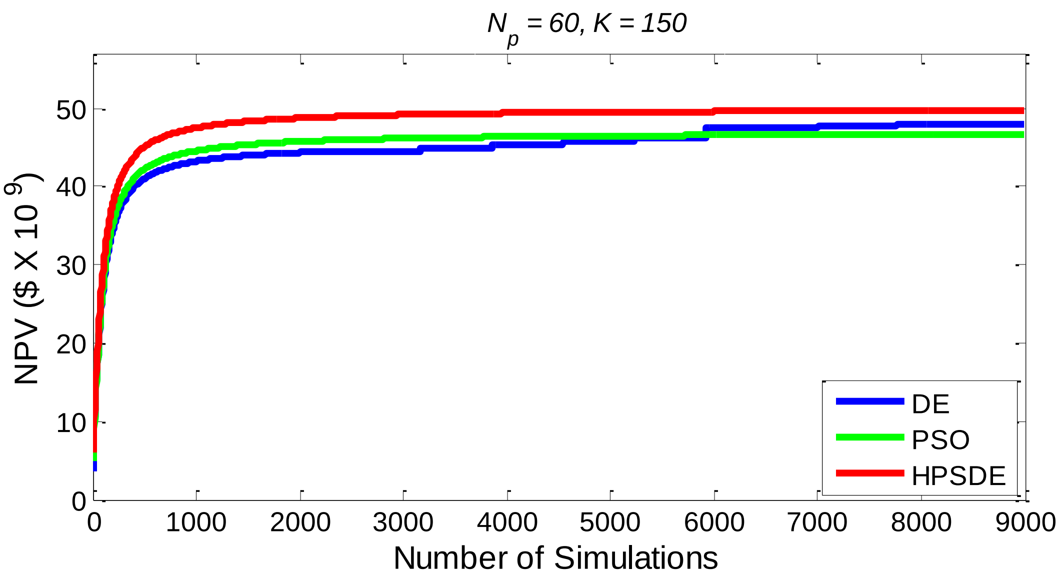


Figure 4.17: NPV of DE, PSO and HPSDE algorithms for application 3

$\$49.71 \times 10^9$, the result achieved by HPSDE represents a 3.8% rise in the performance of DE, and a 6.5% increase in the performance of PSO. Like in application 2, there were periods where DE and PSO converge to near and same NPV measure; in this application, this trend was observed between 5210 and 5903 simulations. Besides, it was also noted that DE outperformed PSO algorithm at very low and very high numbers of simulation; the better performance of DE over PSO at low number of simulation was consistent with the pattern observed in applications 1 and 2. Although the exact reason for this is unknown at this time, we note that the overall performance of PSO was better than the overall performance of DE over reasonable number of simulations. In all cases however, both algorithms did not achieve better NPV than the HPSDE algorithm. This result is also consistent with the results from the first and second applications where *max-mean objective* robust optimization was performed.

	Statistical Indices ($\$\times 10^9$)			
Algorithms	**Best NPV**	**Worse NPV**	**Mean NPV**	**STD**
DE	48.3368	26.3855	45.0748	3.8736
PSO	49.8117	29.0037	45.4302	3.9884
HPSDE	49.7912	29.1131	48.4138	3.4919

Table 4.4: Statistical analysis of results from 30 optimization runs of each algorithm

Table 4.4 shows the result of the statistical analyses of the NPV accrued from the solution of the thirty optimization runs of each of the algorithms. Although the best NPV (attained via the fourth run of PSO) was marginally better than the best NPV attained by HPSDE, the mean NPV attained by the latter was much better than the mean NPV of the other algorithms. It is noted that the better NPV attained by the fourth run of the PSO algorithm over the best HPSDE and DE runs underscores the need, and reinforces the importance for multiple runs of the algorithms – perhaps in the order of tens, as thirty was carried out in this example.

This is imperative because the distribution of the initial solutions (initialization of candidate solutions) in stochastic algorithms often affects the performance of the algorithms. It is further noted that even the same initial solutions are used across the different algorithms (DE, PSO and HPSDE); the randomness resulting from the stochastic operations in the algorithms culminates to a 'non-zero-sum' effect on the performance of each algorithm. To this end, the relative performance of stochastic algorithms cannot be fairly compared on the basis of results arising from a single optimization run of the algorithms. It is only fair to compare and draw conclusions on the basis of results averaged over multiple runs of the algorithms. Thus, despite the better NPV accrued from the fourth run of the PSO algorithm, we conclude that HPSDE outperforms both PSO and DE; and this conclusion is firmly based on the fact that the average NPV accrued by HPSDE over thirty optimization runs was better than the mean NPV for DE and PSO over the number of runs. Interestingly, the standard deviation associated with the HPSDE algorithm was lower than those of DE and PSO algorithms; this is consistent with the pattern observed in Application 2.

Furthermore, we compared the performance of these stochastic algorithms against the computed NPV for a number of specific well patterns. In this regard, the inverted five-spot, the inverted seven-spot and the inverted nine-spot arrangements were considered. These specific well arrangements yielded the NPV of $\$32.27\times10^9$, $\$38.98\times10^9$ and $\$41.90\times10^9$ respectively. In other words, the three stochastic algorithms attained higher NPVs than the specific well patterns considered. Specifically, the performance of the HPSDE algorithm represents an increment of 54%, 27% and 19% in the NPV attained by the inverted five-spot, the inverted seven-spot and the inverted nine-spot patterns respectively.

However, we must note with caution that the above comparison of the performance of the stochastic algorithms on one hand, against the performance of the inverted five-spot and inverted seven-spot arrangements on the other hand, is unfair. The unfairness stems from the overriding discrepancy in the economic constraints in both scenarios. This discrepancy is as a result of the CAPEX variable in the objective function (Eq. 4.1) – the difference in the value of this economic constraint for a five-well system, a seven-well system and a nine-well system is simply not negligible. As such, for this reservoir model, it is only fair to compare the performance of the stochastic algorithms against the performance of specific well pattern of same economic constraint. Thus, it can be said that the comparison of the stochastic algorithms against the inverted nine-well arrangement is the only fair comparison. Since the inverted nine-well pattern yielded an NPV of $\$41.90\times10^9$; it reinforces the belief that metaheuristic algorithms are able to provide better results than specific well pattern arrangement of same economic constraints (i.e. same number of wells).

Subsequently, the result arising from HPSDE algorithm is compared with results emanating from more established optimization techniques such as linear programming (LP) and binary genetic algorithm (bGA), subsequently referred to as genetic algorithm (GA). Though LP is a mathematical optimization method used for determining the optimum of linear objective functions that are subject to linear equality and/or linear inequality constraints; the GA is a population-based evolutionary algorithm proposed in 1975 by John Holland, it is perhaps the most popular and most used evolutionary algorithm in the well optimization problem as presented in section 2.1 of this thesis.

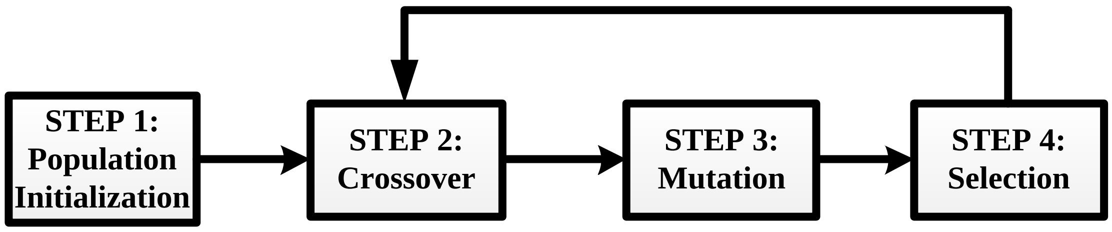


Figure 4.18: Basic GA procedure

The appeal for GAs across diverse problem domains stems from their simplicity and easy-to-implement properties, Farshi (2008). The implementation of the algorithm involves the steps shown in Figure 4.18; thus, this procedure basically differs from the DE procedure illustrated in Figure 4.1 by virtue of the relative positions of steps 2 and 3 above. For this reason, many researchers have reported DE as a deviant and improved version of GA, Konar (2005), Das et al. (2008). Like other population-based stochastic algorithm, GA begins with the initialization of a population of candidate solutions or phenotypes; these proposed solutions are usually parameterized variables encoded as binary strings referred to as chromosomes. Next, the fitness values of these potential solutions are evaluated with the view of selecting better-fit candidates as “parents” for reproduction or recombination – a process which involves the swapping of genetic material between the reproducing parents in accordance to a pre-defined crossover rate. Following the reproduction stage is a mutation process in which an individual (or individuals) of the population is randomly altered; and in so doing, prevents the algorithm from premature convergence by exploring new regions in the problem search space. This ultimately leads to the birth of new offspring chromosomes that are genetically different from the parent chromosomes. The algorithm continues iteratively until the maximum number of iteration is reached; a flowchart depicting the algorithmic process is shown in Figure 4.19.

Note that the control parameters of this algorithm are the population size (N_p), the crossover rate (*CR*) and the mutation rate (*MR*). In this application, 0.6 and 0.017 (the inverse of the population size) were employed for crossover rate and mutation rates respectively; and the choice of these values for *CR* and *MR* is informed from previous applications in Goldberg et al. (1991), Guyaguler (2002), Podnar and Kapov (2003) and Brain and Addicoat (2010).

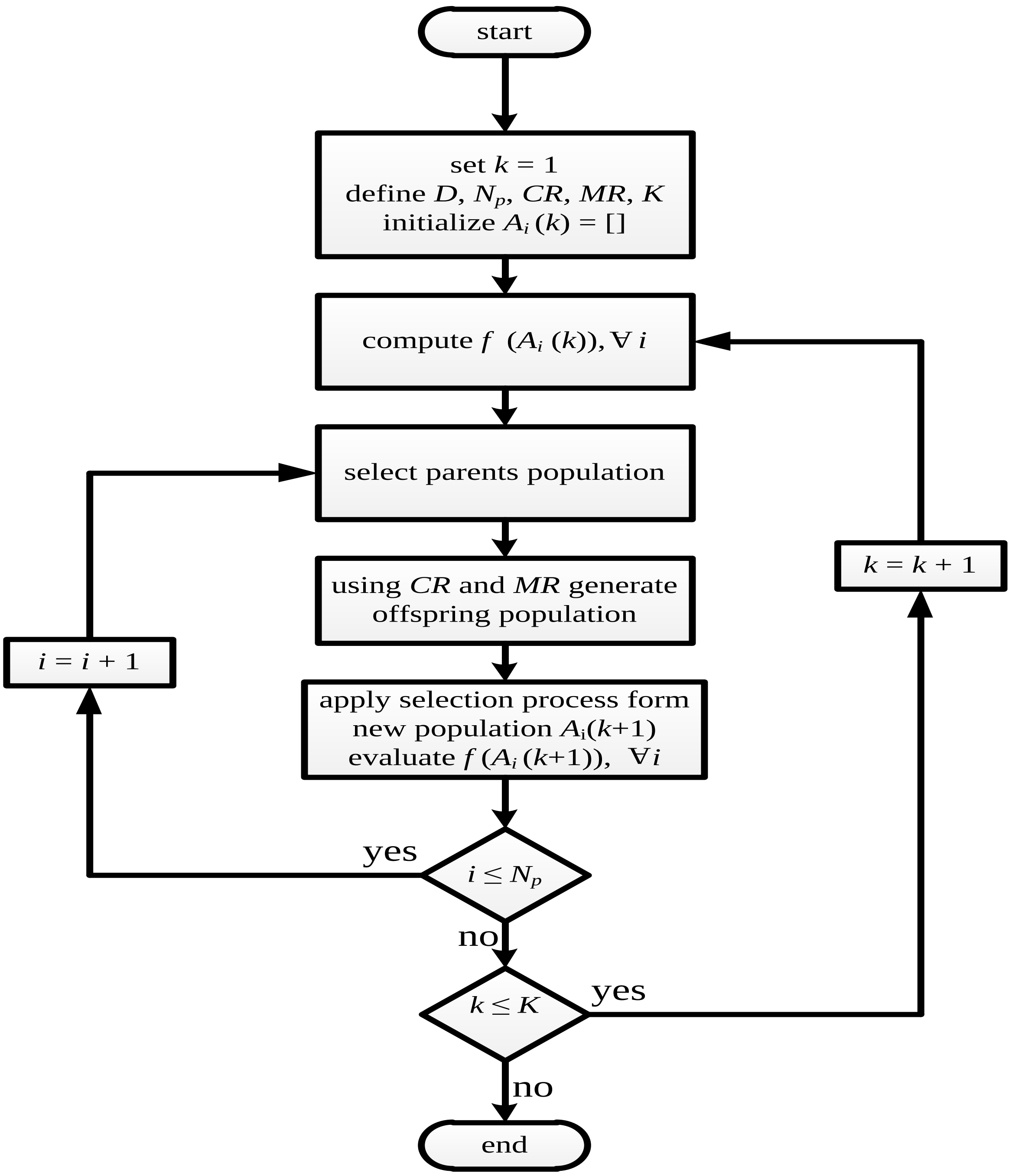

start
set $k = 1$
define D, N_p, CR, MR, K
initialize $A_i(k) = []$
compute $f\ (A_i\ (k)), \forall\ i$
select parents population
using CR and MR generate offspring population
apply selection process form new population $A_i(k+1)$
evaluate $f\ (A_i\ (k+1)),\ \forall i$
$i \le N_p$
yes
no
$i = i + 1$
$k \le K$
yes
no
$k = k + 1$
end

Figure 4.19: Flowchart showing the genetic algorithm (GA)

In the application of LP in this problem, we espoused a technique modified from Guyaguler and Gumrah (1999). We denote $W = \{w_1, w_2, ..., w_9\}$ as a vector of nine wells to be placed on a reservoir model with square grid-blocks of index defined by the set $\mathbb{R} = \{1, 2, ..., 2500\}$, the Cartesian coordinate of each placed well is defined by $x_i, y_i \in \mathbb{R}$ and $o_i \in \{0,1\}$ is a binary indicator that defines the well type – 0 for production wells and 1 for injection wells. The LP problem was solved using LPLOG solver, and the resulting solution used as indices for the placement for the nine wells in the reservoir system. The wells are constrained to operate at BHPs of $65\times10^5\,\text{Pa}$ and $100\times10^5\,\text{Pa}$ for producers and injectors respectively; and the system simulated for a period of 10 years. Using fluid profile generated, the NPV was computed in accordance to (4.1); in this regard, an NPV of $\$37.91\times10^9$ was computed.

Using an initial population size of 60 and a maximum iteration of 150 generations; thirty optimization runs of the GA algorithm were performed in **MATLAB**® and the results were used for placing the wells in the reservoir. Like in the previous metaheuristic algorithms, the wells are constrained to operate at BHPs of $65\times10^5\,\text{Pa}$ and $100\times10^5\,\text{Pa}$ for producers and injectors respectively. The system is simulated and the fluid profiles are used to compute the NPV corresponding to each optimization run of the GA by applying Eq. 4.1; and when the computed NPVs accruing from the thirty runs were averaged, the resulting NPV was $\$46.33\times10^9$. The results accruing from the application of LP and GA are tabulated and presented in Table 4.5 below.

	Statistical Indices ($\$\times 10^9$)			
Algorithms	**Best NPV**	**Worse NPV**	**Mean NPV**	**STD**
LP	37.9118	37.9118	37.9118	–
GA	46.9269	28.0251	44.8917	4.3601

Table 4.5: Results of LP and GA algorithms

Thus, the result achieved from the application of GA was comparable to the NPV of $\$46.67\times10^9$ that was achieved by the PSO algorithm; however, this performance represents a 3.2% and a 6.8% reduction in the performance attained by DE and HPSDE algorithms respectively. Interestingly, the NPV of $\$37.91\times10^9$ that was accrued from the application of LP fell short of the performance of all the stochastic algorithms considered in this application. As a matter of fact, it performed lower than the specific well arrangement of equal well (the inverted nine-spot arrangement); and this shows that LP is not a suitable optimization technique for this problem type. Also, it is important to note the fact that the search space in this problem (objective function) is non-smooth and nonlinear; to a large extent, this may be the reason behind the relative poor performance of LP in this problem. It is noted that LPs are often deployed in problems with linear objective functions that are subject to linear equality and inequality constraints.

Now, owing to the fact that HPSDE involves the hybridization of two global optimization algorithms – DE and PSO; we further compare its performance with a hybrid algorithm that combines a global optimization algorithm and a local search algorithm. In this regard, the PSO algorithm was combined with tabu search (TS) to form a hybrid algorithm referred to as PSOTS. The TS algorithm was proposed and formalized as an extension of local search technique by Fred Glover in Glover (1986) and Glover (1989) respectively. The algorithm originated from the simulation of human intelligence progress; the most salient feature of this local search technique is its ability to self-mark the searched local minima and avoid them as much as it is possible. The populations with good fitness are marked in the tabu list to prevent cycling or re-visitation, and the algorithm guarantees diversity by implementing a flexible tabu list and a well-planed tabu strategy, Wang et al. (2007). In order words, it avoids local minima entrapment through a strategy that prohibits certain previously search directions. Usually, an aspiration criterion is incorporated into the algorithm so as to override promising solutions that may have been excluded or consigned to the tabu list. A simple and commonly used aspiration criterion is to adopt solutions with tabu status which are better than the "best so far" solution. That is to say that if a candidate has a higher fitness value than the "best so far" solution, its tabu status is ignored and adopted as the current solution. The hybridization of this algorithm with PSO (to form PSOTS), and its implementation in this problem was adapted from the study of Talbi and Belarbi (2011).

The basic idea of the PSOTS is to update the PSO with TS at any iteration where the updated particle position objective function value, $f(\mathbf{x}_i(k+1)),$ is less than the previous best position, $f(\mathbf{y}_i(k)),$ of the particle. The algorithm is as illustrated in Algorithm 4.4; it starts with the PSO algorithm and continues until the point where the objective function value of the updated particle position is compared with that of the previous position. At that stage, a number of candidates of the current solution are determined. If a candidate on the tabu list is

Algorithm 4.4 PSOTS Algorithm

```
1: Set iteration index $k = 1$
2: Define $N_p, \omega = 0.721, c_1 = c_2 = 1.193, K$
3: Initialize $\mathbf{x}_i(k) : x_{i,j}(k) \sim \text{U}(l_j, u_j) \forall j, \forall i$
4: Initialize $\mathbf{v}_i(k) : v_{i,j}(k) \sim \text{U}(0,1) \forall j, \forall i$
5: Compute objective function, $f(\mathbf{x}_i(k)), \forall i$
6: $\mathbf{y}_i(k) = \mathbf{x}_i(k), \forall i$
7: while $k \le K$ do
8:     Permute the particle indices
9:     Generate neighborhood for each particle
10:    $i = 1$
11:    while $i \le N_p$ do
12:        Determine best particle in neighborhood of particle $i$
13:        Compute $\mathbf{v}_i(k+1)$ using Eq. 4.21
14:        $j = 1$
15:        while $j \le D$ do
16:            $x_{i,j}(k+1) = x_{i,j}(k) + v_{i,j}(k+1)$
17:            Apply Eqs. 4.22 and 4.23 if necessary
18:            $j = j + 1$
19:        end while
20:        Compute objective function, $f(\mathbf{x}_i(k+1)), \forall i$
21:        $i = i + 1$
22:    end while
23:    $i = 1$
24:    while $i \le N_p$ do
25:        if $f(\mathbf{x}_i(k+1)) > f(\mathbf{y}_i(k))$ then
26:            $\mathbf{y}_i(k+1) = \mathbf{x}_i(k+1)$
27:        else
28:            initialize the tabu list
29:            initialize the aspiration criterion
30:            while $i \le N_p$ do
31:                generate solutions randomly
32:                evaluate each neighborhood
33:                choose best neighbor
34:                update the tabu list and aspiration criterion
35:            end while
36:        end if
37:        $i = i + 1$
38:    end while
39:    $k = k + 1$
40: end while
```

better than the prevailing best solution at that time, the tabu attribute of the candidate on the tabu list is neglected. Therefore, the prevailing best solution is substituted by the candidate on the tabu list, and consequently, the tabu list is updated accordingly. If the candidate mentioned above does not exist, the original PSO candidate solution is preferentially selected while ignoring its strengths and weaknesses to the current solution and modifying the tabu list at the same time. This process is repeated iteratively until the end criteria are met.

Like in the previous stochastic algorithms, multiple runs of this hybrid algorithm is performed using an initial population size of 60 and a maximum iteration of 150 generations; the results were used to place reservoir wells constrained at BHPs of 65×10^5 Pa and 100×10^5 Pa for producers and injectors respectively. The system is simulated and the fluid profiles are used to compute the NPV corresponding to each optimization run of the PSOTS by applying Eq. 4.1; and when the computed NPVs accruing from the thirty runs were averaged, the resulting average NPV was $\$48.42\times10^9$. Figure 4.20 shows how GA and PSOTS compare against HPSDE; and importantly, the result also highlights the effect of tabu search on the convergence of PSO algorithm.

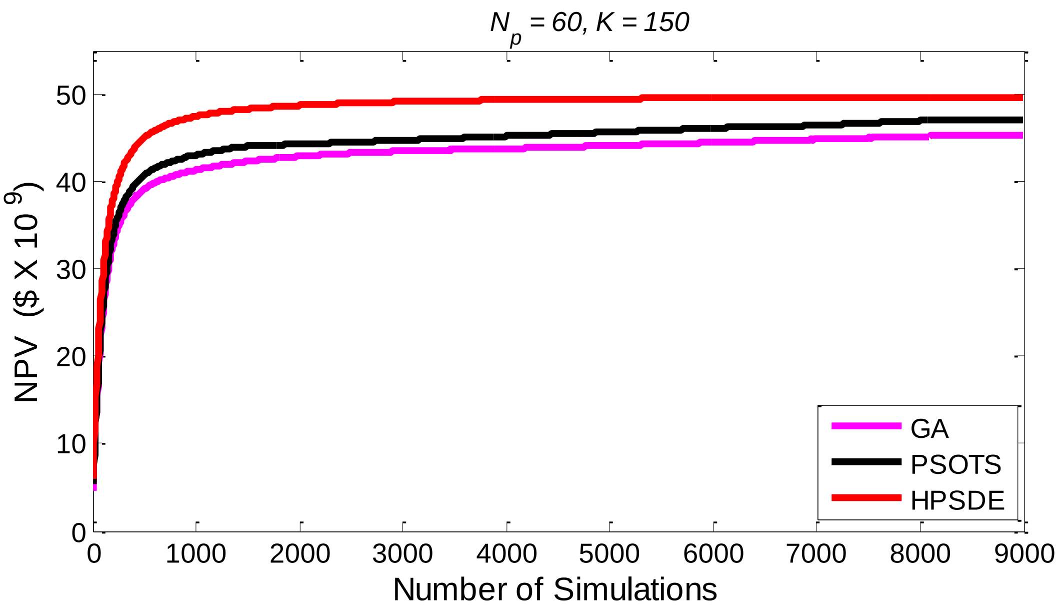


Figure 4.20: NPV of GA, PSOTS and HPSDE algorithms for application 3

It was observed in our earlier analysis that the PSO algorithm converged to a steady NPV from 5113 simulations (see Figure 4.17), however, the hybrid PSO (PSOTS) algorithm converged at about 7902 simulations. In other words, the propensity of the PSO algorithm towards premature convergence was very much reduced by the hybridization of the algorithm with TS. However, this hybrid algorithm did not attain a better NPV than the HPSDE. The average performance of the PSOTS represents a reduction of $\$1.29\times10^{9}$ in the NPV attained by HPSDE algorithm. The relative performance of all five stochastic algorithms employed in this problem are plotted and shown in Figure 4.21. But for the HPSDE, the hybrid PSOTS outperformed the rest of the algorithms (DE, PSO and GA in that order). In cases where the HPSDE was compared with DE and PSO algorithms, the performance of PSO fell behind the other algorithms; we believe that the reason behind this is perhaps the inclination or natural tendency of PSO towards premature convergence. This often makes the algorithm susceptible to missing better solutions by converging or stagnating at local minima. A thorough analysis of the result arising from the PSOTS algorithm underscores the fact that this issue of premature convergence was to an extent addressed by the hybridization of the PSO algorithm with a local tabu search algorithm.

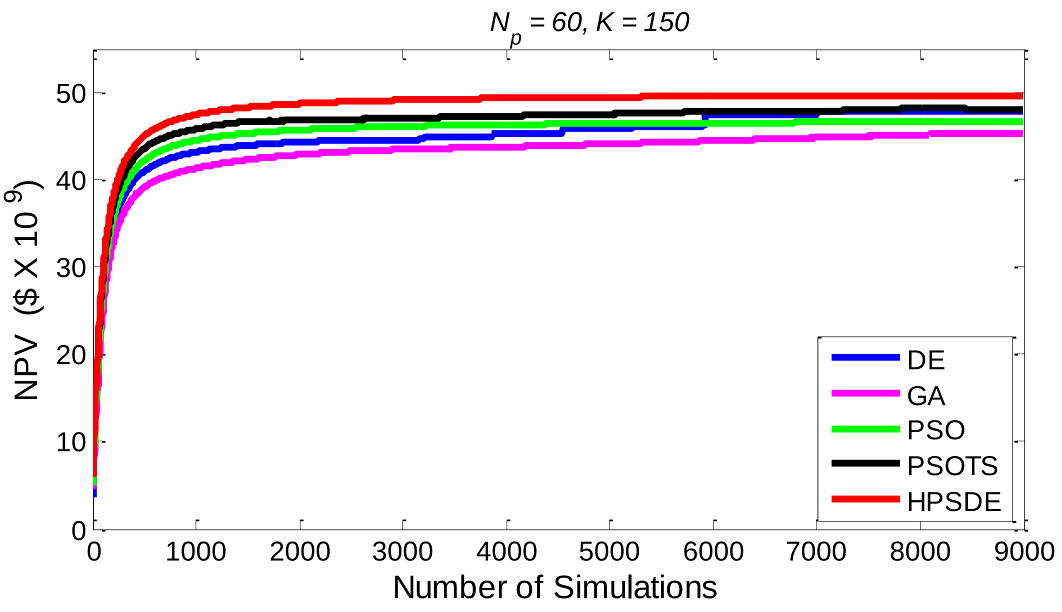


Figure 4.21: Comparison of the NPV of DE, GA, PSO, PSOTS and HPSDE algorithms for application 3

As a result of these observations, we infer that that the number of function evaluation required to attain optimal result in PSO algorithm was lower than those of DE and HPSDE algorithms. In other words, the PSO algorithm requires the fewest enumeration of the entire search space of the problem as against the DE and HPSDE algorithms. This inference corroborates findings of previous studies reported in the literature. Thus, PSO algorithms lack exploitation abilities, they possess the ability to quickly go down to good, promising regions of the search space; however, they usually lack the ability to refine solutions, Li et al. (2010).

4.6 Benchmark Tests and Computational Complexity

In addition to their easy-to-implement properties, biologically inspired algorithms are generally popular because of their efficiency at finding approximate solutions in optimization problems. However, it is instructive to note that the performance of these algorithms across optimization problems of different complexities exhibit varying degrees of limitations in accordance to the *no free lunch* theorem. Thus, the effectiveness or otherwise of HPSDE over DE and PSO in the well placement problems as considered in this chapter are not sufficient conditions for making far-reaching conclusions. In this section, all three metaheuristic algorithms are subjected to benchmark problems of different complexities. The complexity of the test problems depends on the number and distribution of local optimums as well as the number of variables. In this regard, the algorithms are tested using six benchmark problems including, namely: Ackley Problem, De Jong F1 Function, Griewangk Problem, Rastringin Problem, Rosenbrock Problem and Schaffer F6 Function.

Besides the fact that these six test functions exhibits different degrees of complexities, we justify their selection in the light of the fact that they reflect opposite sides of fundamental complexity factors, such as modality, separability and scalability. Modality refers to the number of local optima a problem possesses. Most real life applications are multimodal (i.e. contains more than one local optima) as against unimodal or convex problems which have one optimum only. A function of variables p is said to be a separable problem if it can be expressed as a sum of y functions of one variable. In other words, separability entails that the optimization problem can be re-partitioned into sub-problems of lower dimensionality and therefore, is considerably easier to solve. Scalability refers to the ability to suitably and efficiently apply the algorithm to problems of larger dimensionality. Thus, scalability is the property of the problem which determines its behavior in different dimensions.

With respect to the benchmark problems considered here, De Jong F1 Function is unimodal, while Griewangk's Problem is multimodal; Rastringin's Problem is separable, while the Rosenbrock Problem is non-separable, and Ackley's Problem is scalable while the Schaffer F6 Function is non-scalable. Note that these benchmark problems are often characterized by more than one complexity factor. Take for example, besides the Sphere Function, the remaining five benchmark problems are multimodal; again, besides the Rastringin's Problem, all other problems are inseparable. A general description of the benchmark problems employed in this complexity analyses is presented below.

1. De Jong's F1 Function (also known as the Sphere Function) is a smooth, symmetrical, unimodal and strongly convex function. This function is not a complex problem because it has only one solution; its usefulness stems from the fact that it is a simple test function that allows one to check that the algorithms are in good working condition and that there are no coding mistakes. It is given by $\min_x f(x) = \sum_{i=1}^{n} x_i^2$ and the test area is often restricted to the hypercube $-5.12 \le x_i \le 5.12,\ i = 1, 2, ..., n.$

2. The Rosenbrock Problem is characterized by a very narrow and sharp ridge which rotates around a parabola; hence, it is considered a difficult problem. Algorithms with weak exploration capabilities (i.e. not able to search new and better regions of the search space) suffer severe limitation when deployed to this problem. Mathematically, the problem is given by $\min_x f(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right]$ and the test area is usually restricted to $-2.048 \le x_i \le 2.048,\ i = 1, 2, ..., n.$

3. With a large search space and high number of local minima, the Rastrigin's function is multimodal and fairly complex. It has a linearithmic complexity, and the surface of the function is determined by the amplitude (A) and the modulation frequency (ω). With A=10 (as is the case in this work) the selected domain is dominated by the modulation, and the local minima are located at a rectangular grid with size 1. The fitness values of the local minima increases with increasing distance to the global minimum; the function is represented as $\min_x f(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) \right]$ and like the Sphere Function, the test area is restricted to $-5.12 \le x_i \le 5.12,\ i = 1, 2, ..., n.$

4. Like the Rastrigin's problem, the Griewangk problem is a multimodal function which has a linearithmic complexity of $O(n \ln(n)),$ where n is the number of the function's parameters. The function is given by $\min_x f(x) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$ and its test area is restricted to the hypercube $-600 \le x_i \le 600,\ i = 1, 2, ..., n.$ The terms in the

summation produce a parabolic solution space, while the local optima (created by the cosine function) are above the parabola. The dimensions of the search range increase on the basis of the product, which results in the decrease of the local minima. It is also noted that the function gets flatter the more the search range is increased. Thus, most algorithms have difficulties to converge close to the minima of this function; and this is because the probability of making progress rapidly decreases as the minima is approached, Diglakis and Margaritis (2000).

5. The Ackley's Problem is a widely used multimodal test function which is defined by $\min_x f(x) = -20\exp\left(-0.2\sqrt{n^{-1}\sum_{i=1}^{n} x_i^{\,2}}\right) - \exp\left(n^{-1}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e.$ The presence of an exponential term in this function creates numerous local optima that covers its surface; it is noted in Domingo et al. (2005) that algorithms with strong exploratory and exploitative properties yield good results when tested on Ackley's Problem. The test area of this benchmark test is within the hypercube $-30 \le x_i \le 30,\ i = 1, 2, ..., n.$

6. Schaffer's F6 Function is given as $\min_x f(x, y) = 0.5 - \dfrac{(\sin\sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2}$, and its test area is often restricted to the hypercube $-100 \le x_i \le 100,\ i = 1, 2, ..., n.$ The main difficulty of the Schaffer's F6 test function is that the size of the potential optimum that need to be overcome to get to a minimum increases the closer one gets to the global minimum, Pohlheim (2006).

Having described the benchmark test functions and the reason behind the choice of the selected functions, we carry out benchmark tests with the view to ascertain the comparative performance of the algorithms. In our analyses of the efficiency of the algorithms, we use well established statistical quality indicators (or indices) such as best values, mean values and standard deviation of the results obtained. To this end, 25 independent runs were performed with randomly initialized populations of all three algorithms, and a common termination criterion of 5000 function evaluations is set for the algorithms. The termination criterion set out above (i.e. same function evaluation) serves as a level playing ground for all three algorithms – it restricts the window in which our inferences are made; and the choice of 25 runs is based on established rule of thumb as highlighted in Mersmann et al. (2010).

A summary of the best and mean function values of the experimental results are shown in Tables 4.6 and 4.7 respectively; whereas Table 4.8 shows the standard deviation – a statistical indicator which represents the extent of dispersion or variation of the data points from the arithmetic mean.

	Benchmark Tests Functions					
Algorithms	**De Jong's F1 Function**	**Rosenbrock's Problem**	**Rastrigin's Problem**	**Griewangk's Problem**	**Ackley's Problem**	**Schaffer's F6 Function**
DE	0.193E+00	–2.899E+01	–3.114E+01	–2.750E+01	3.66E+02	1.40E+01
PSO	0.190E+00	–2.705E+01	**–4.192E+01**	–2.767E+01	**1.03E+02**	1.31E+01
HPSDE	**0.043E+00**	**–3.131E+01**	–3.511E+01	**–2.899E+01**	1.71E+02	**1.22E+01**

Table 4.6: Best values of 25 runs after 5000 function evaluations of algorithms on benchmark problems

	Benchmark Tests Functions					
Algorithms	**De Jong's F1 Function**	**Rosenbrock's Problem**	**Rastrigin's Problem**	**Griewangk's Problem**	**Ackley's Problem**	**Schaffer's F6 Function**
DE	2.852E+00	–2.578E+01	–2.834E+01	–2.020E+01	5.78E+02	1.46E+01
PSO	2.069E+00	–2.523E+01	–3.125E+01	–2.441E+01	**1.09E+02**	1.45E+01
HPSDE	**1.813E+00**	**–2.678E+01**	**–3.381E+01**	**–2.583E+01**	1.93E+02	**1.31E+01**

Table 4.7: Mean values of 25 runs after 5000 function evaluations of algorithms on benchmark problems

	Benchmark Tests Functions					
Algorithms	**De Jong's F1 Function**	**Rosenbrock's Problem**	**Rastrigin's Problem**	**Griewangk's Problem**	**Ackley's Problem**	**Schaffer's F6 Function**
DE	2.254E+02	2.416E-01	1.910E-02	1.211E+01	2.506E-02	2.76E+01
PSO	3.734E+01	2.030E-01	1.393E-02	1.541E+01	1.753E-02	6.10E+01
HPSDE	**2.223E+01**	**1.383E-01**	**1.093E-02**	**1.150E+01**	**1.024E-02**	**1.15E+01**

Table 4.8: Standard deviation after 5000 function evaluations of the algorithms on benchmark problems

On the strength of the results of the benchmark test functions given in Tables 4.6 and 4.7; we can infer that on the average, HPSDE algorithm outperformed both DE and PSO algorithms in all of the benchmark test function but for Ackley's Problem in which the PSO algorithm yielded better results for both the best function value as well as the mean function value. It is also noted that although the PSO algorithm yielded the best function value in the Rastrigin's Problem; the mean value attained by HPSDE over the 25 optimization runs was better than those of the other algorithms in the same benchmark test (Rastrigin's Problem). Interestingly, the lowest standard deviation in all six benchmark functions were those associated with experimental results emanating from HPSDE algorithm. In statistical theory and indeed in the theory of probability, low standard deviation indicates that the data points tend to be in or around the proximity of the arithmetic mean of the distribution; whereas a high standard deviation indicates that the data points are well spread out over a large range of values. Consequently, it is safe to infer that for fewer number of optimization runs; the probability of attaining near-average optimal results is higher when HPSDE algorithm is employed than the other two algorithms. This corroborates the trends in the standard deviation resulting from the statistical analyses of the NPVs accruing from the fluid profile generated from 30 optimization runs of the algorithms as was carried out in applications 2 and 3 (sub-sections 4.5.2 and 4.5.3) – see Tables 4.3 and 4.4 respectively.

Furthermore, we compare the computation complexity of the three algorithms. As a valuable and qualitative insight into algorithmic efficiency, the objective of computational complexity is to determine the feasibility of an algorithm by estimating an upper bound on its usability. It also provides an avenue for relative comparison of algorithms in order to decide algorithmic suitability for any given problem. In this regards, algorithmic efficiency or computational complexity is measured in terms of time complexity – a measure of the amount of time required to execute the algorithm and its space complexity – which is a measure of the number of memory cells or nodes it requires for its execution. The selection and deployment of algorithms for any given problem often involve some kind of time-space-tradeoff; this is because most computational problems cannot be solved with short computing time and low memory space, Ziegler (2002).

Generally speaking, the better the time complexity of an algorithm, the faster the algorithm is in practice; and the better the space complexity, the lower the risk of running out of memory cells. Over the years, the big O notation has been a convenient way of expressing the computational complexity of problem-solving algorithms. This notation provides a simple but qualitative insight into how changes in the input of the algorithm N affect the algorithmic performance as N grows larger. In other words, it provides the window for understanding how the performance of an algorithm responds to changes in problem input size.

In Zielinski et al. (2006), it was demonstrated that the control parameters of DE algorithm (i.e. population size N_p, crossover rate CR and mutation factor F) have a direct bearing on the computational complexity of the algorithm. Since each iteration of the algorithm involves a loop of N_p conducted over another loop *D*; and the mutation and crossover operations are performed at the component level for each DE vector, it therefore, follows that the number of fundamental operations in the algorithm (DE/rand/1/bin) is proportional to the total number of loops conducted until the maximum number of iterations *K* is reached. Thus, the runtime complexity of the algorithm is given by $O(N_p \times D \times K).$ For this algorithm, the space requirements is in the order of $O(N_p \times K) = O(E),$ where E is the number of fitness evaluations required by the algorithm in a given problem. The space complexity of DE is low when compared to other popular metaheuristic algorithms; perhaps this explains why DE has been extensively employed in large scale optimization problems across a broad range of disciplines, Das and Suganthan (2011).

For the PSO algorithm, the runtime complexity is given by $O(N_p \times M \times K),$ where $N_p,$ M and *K* are population size, number of neighborhood and maximum number of iteration respectively. According to Liu et al. (2011), the worst case scenario in this algorithm occurs when the number of sub-swarms (or neighborhood) remains unchanged, and the number of iteration reaches the designated maximum iteration number. In that situation, the runtime complexity is given by $O(N_p \times M \times K);$ however, if the number of sub-swarms is reduced after some iterations, the runtime complexity reduces to $O\left(\sum_{i=1}^{K} L \times N_p\right),$ where $1 \leq L \leq M$. The space complexity of the PSO algorithm is in the order of $O(N_p \times K),$ and this complexity increases to the order of $O(Q \times N_p \times K)$ when Q number of particles $(Q < N_p)$ overlap on the same node, as pointed out in Gheitanchi et al. (2008).

The time and space complexities of HPSDE algorithm are in the order of $O(N_p \times D \times M \times K)$ and $O(N_p \times K)$ respectively. The flowchart of the algorithm as depicted in Figure 4.5 shows that in every iteration, there is an extra function evaluation that is absent in both the DE and PSO algorithms. The resources required to carry out this additional function evaluation in each iteration of the algorithm constitutes an increased computational burden vis-à-vis the resources required in each iteration of DE and PSO. In order words, although the use of hybrid algorithms may be desirable from a performance point of view; it however, exacerbates computational efficiency by virtue of the fact that it increases runtime computational complexity, and to some extent, the space complexity. Therefore, it is needful to develop simple adaptation rules for algorithmic control parameters so as to improve performance without imposing considerable computational burden when hybrid algorithms such as HPSDE are deployed.

Finally, we end this section by analyzing the ratio of the function evaluations required by each of the algorithms with respect to full enumeration of the search space in each of the well placement optimization scenario considered in this chapter. Exhaustive enumeration of the search space is often undesirable and can be computationally prohibitive; particularly in large scale engineering problems where they can easily become intractable, regardless of the computational resource availability.

Using brute-force or exhaustive search as a baseline, we compared the number of function evaluations performed by DE, PSO and HPSDE algorithms; with the view of understanding the relative advantage of one algorithm over the other in this problem domain. In the first application considered in this chapter, the problem involved optimal placement of a single producer in a 2–D reservoir model with $45 \times 45 \times 1$ grid-blocks. Theoretically, there are 2025 (or ${}^{2025}\boldsymbol{C}_1$) different possible placement of the single well; and this means that a full enumeration of the search space (i.e. sampling the entire search space) would require 2025 function evaluations. In this problem, however, the DE, PSO and HPSDE algorithms began to converge to near-optimum solutions after 740, 591 and 772 iterations respectively. Note that there is an extra function evaluation in each iteration of the HPSDE algorithm; thus, the minimum numbers of function evaluations required in this problem scenario (Application 1) are 740, 591 and 1544 for DE, PSO and HPSDE algorithms respectively. Therefore, using the theoretical maximum required function evaluation as a baseline, the ratio of the actual objective function evaluations for these algorithms are 0.3654, 0.2919 and 0.7625 for DE, PSO and HPSDE respectively.

In the second application considered, the defined problem statement was to optimize the placement of two wells in a 2–D reservoir model of $50 \times 50 \times 1$ grid-block. Theoretically, there are 3123750 (or ${}^{2500}\boldsymbol{C}_2$) possible placement for both wells; in other words, the computational cost of an exhaustive enumeration of the search space would entail over three million function evaluations. However, the DE, PSO and HPSDE algorithms attained near-optimal solutions after 2632, 1459 and 5726 objective function evaluations respectively. Indeed, these numbers are much fewer than the theoretically required 3123750 objective function evaluations required for full enumeration of the search space; in fact, the numbers represent a ratio (with respect to the brute force baseline) of 8.4258×10^{-4}, 4.6707×10^{-4} and 1.8331×10^{-3} for DE, PSO and HPSDE algorithms respectively.

For the nine wells placement problem considered in application 3, a full enumeration of the search space requires 1.036182146×10^{25} (or ${}^{2500}\boldsymbol{C}_9$) function evaluations; whereas DE, PSO and HPSDE algorithms converged to near-optimal solutions after 7010, 5097 and 17126 objective function evaluations respectively. This application clearly underscores the desirability of metaheuristic algorithms in this problem domain; it highlights their ability to yield approximate solution without exhaustive enumeration of the search space. It further highlights the fact that full enumeration of the search space in the well placement optimization problem becomes intractable as the number of decision variables (number of wells) increases; thus brute-force algorithms would suffer severe performance limitations in this domain. For this problem, the ratio of the actual number of objective function evaluations needed to reach near-optimum solutions to full enumeration of the search space pales into insignificance; they are 6.7652×10^{-22}, 4.9190×10^{-22} and 1.6528×10^{-21} for DE, PSO and HPSDE algorithms respectively. The computed results for the ratio of the number of actual objective function evaluations required to reach near-optimum solutions to the theoretical maximum number of objective function evaluations for all three problem scenarios are tabulated and presented in Table 4.9 below.

Expectedly, the algorithms attained near-optimal solutions with fewer function evaluations as against the prohibitive number of function evaluations required for full enumeration of the search space; and the comparative advantage of the algorithms in terms of the ratio of the number of actual objective function evaluations required to reach near-optimum solutions to the theoretical maximum number of objective function evaluations become much pronounced as the decision variables of the underlying optimization problem increases. Interestingly, however, of the three metaheuristic algorithms; the PSO consistently converged to near-optimum solution with the fewest number of objective function evaluations in all the applications. This evidence is also reflected in the fact that the computed ratio of the number of actual objective function evaluations required to reach near-optimum solutions to the theoretical maximum number of objective function evaluations for the PSO algorithm was lowest amongst the algorithms. On the other hand, the computed ratio of the number of actual objective function evaluations required to reach near-optimum solutions to the theoretical maximum number of objective function evaluations for the HPSDE algorithm was the highest in all the applications.

Algorithms	**App. 1**	**App. 2 ($\times 10^{-4}$)**	**App. 3 ($\times 10^{-22}$)**
DE	0.3654	8.4258	6.7652
PSO	0.2919	4.6707	4.9190
HPSDE	0.7625	18.3310	16.5280

Table 4.9: Ratio of number of actual FEs required to reach near-optimum solutions to brute force FEs

In fact, one of the salient points that can be deduced from Table 4.9 is that the computed ratios of the actual objective function evaluation performed by the PSO algorithm to the full enumeration of the search space represent a respective reduction of 20.1% and 61.7% in the computed ratio of actual function evaluation to full enumeration of the search space for DE and HPSDE in application 1, a respective reduction of 44.5% and 74.5% in the computed ratio of actual function evaluation to full enumeration of the search space for DE and HPSDE in application 2, and a respective reduction of 27.3% and 70.2% in the computed ratio of actual function evaluation to full enumeration of the search space for DE and HPSDE in application 3. The ability of PSO algorithm to converge quickly to near-optimal solutions as highlighted above is consistent with previous findings in the literature; see Li et al. (2010).

Hence, none of the algorithms requires exhaustive enumeration of the search space; however, of all three algorithms, HPSDE requires the highest number of function evaluation while the PSO requires the fewest. The comparative higher number of function evaluation associated with HPSDE is perhaps as a result of the extra function evaluation that is present in every iteration of the algorithm as evident in Figure 4.5. Since full enumeration of the search space is prohibitive and undesirable; algorithms that require fewer objective function evaluations are most desirable. Thus, on the strength of the ratio of actual function evaluations required to achieve near-optimal solutions to full enumeration of the search space; it could be said that PSO outperformed both DE and HPSDE algorithms. Although the HPSDE algorithm achieved better results (NPV) than DE and PSO in the well placement problem; it could be said that its performance in this problem domain came at a price.

By and large, since the number of function evaluation performed by HPSDE is within computational resource affordability; its performance in the well optimization problem outweighs the undesirability associated with its higher ratio of actual function evaluations to full enumeration of the search space as against the PSO. Ultimately, it is reasonable to give up the desirability of fewer search space enumeration associated with PSO in exchange for the high performance measure or NPV associated with HPSDE since such solutions are attained within non-prohibitive polynomial time.

4.7 Discussion

In the well placement optimization applications considered in this work, HPSDE algorithm yielded higher NPV than DE and PSO algorithms. While we note that these findings and results are interesting and potentially useful, we acknowledge that there are issues or limitations that need to be addressed. Chief among these limitations is the issue of control parameters tuning. For instance, in the second and third examples, there are instances where DE outperformed PSO, and vice versa. It is important to understand how these behaviors are influenced by relevant control parameters of the algorithm. We note that DE parameters (F = 0.5, CR = 0.1) used in this work were adapted from Storn and Price (1997), and the PSO parameters $(c_1 = c_2 = 1.193, \omega = 0.721)$ were adapted from Onwunalu and Durlofsky (2010). For reasons bothering on fair comparison of results, all three algorithms were used without parameter tuning of any kind. Although the population size and the maximum number of iteration are largely dependent on the complexity of the underlying well optimization problem, as well as the number of optimization variables; we believe that effective parameter tuning (which will be computationally expensive, as it will require extra function evaluations) would further enhance the performance of the algorithms. This is so because generalized adjustment of metaheuristic control parameters cannot be achieved from theoretical analyses on the algorithms alone. Thus, an effective mechanism for control parameter tuning would depend on the demands of the underlying optimization problem as well as the experience and background knowledge of the user.

A closely related limitation is the issue of usability in practical field optimization problem. Indeed, a hybridized metaheuristic optimization algorithm such as HPSDE is potentially a viable and promising alternative in reservoir engineering optimization problems; however, issues of usability have to be addressed before it can be deployed for practical use in the industry. To some degree, the usability limitation is intertwined with parameter tuning. We note that usability of metaheuristic algorithms would be greatly enhanced if the issue of parameter tuning is sorted out at the design-end, and not at the user-end of the algorithms. This is so because it is generally unrealistic for industrial end-users to waste expensive function evaluations in correcting the weakness of the design phase of an algorithm. We also note that the performance of HPSDE algorithm, and indeed other hybridized stochastic algorithms; could be further improved by incorporating into the algorithms, prior knowledge and relevant information about the optimization problem.

4.8 Summary

In this chapter, the importance of well optimization was highlighted; we showed that it is a field development decision input that can ultimately determine a reservoir's production profile, and therefore, the recoverability of the reservoir. For all intents and purposes, the recoverability is a direct measure of the economic value of the portfolio or NPV of the asset. In this work, we employed three metaheuristic algorithms in this problem domain; one of the algorithms (HPSDE) is a 'hybrid' of the other two algorithms – DE and PSO. With NPV as performance measure, we considered three examples involving the placement of one, two and nine vertical wells in reservoir models of varying complexities.

Based on suggestions from Vasiljevic and Golobic (1996) and Ciaurri et al. (2011), five runs of each of the algorithms are performed in the first application; owing to the need to carry out a more detailed, reliable and systematic analyses of the algorithms, thirty optimization runs of the algorithms were performed in the second and third applications. In all cases, the results were averaged over the number of optimization runs so as to determine the relative strength of each algorithm. The HPSDE algorithm consistently outperformed DE and PSO algorithms. In two of the examples, we factored in geological uncertainty by addressing the discrepancies between physical reservoir and reservoir model. To this end, we performed a *max-mean objective* robust optimization of the performance measure; and HPSDE yielded better results than DE and PSO. In the third example, we compared the performance of the metaheuristic algorithms with the NPVs attained via different specific well pattern arrangements; and the stochastic algorithms yielded higher NPV than the specific well pattern arrangements. We also showed that the performance of DE and PSO was dependent on the total number of simulations – in other words, there was a variation in performance in the early, mid and later stages of simulation. DE attained higher NPV than PSO at very low and very high number of total simulation. However, in all examples considered, the overall performance of PSO was better than that of DE. More importantly, we note that HPSDE outperformed both algorithms in all cases. Besides, the performance of the HPSDE algorithm was compared with the performance achieved by more established optimization techniques such as LP and GA; and HPSDE outperformed both algorithms. Although the result attained by GA was comparable to that of PSO and DE, the result emanating from LP fell way behind those attained by the stochastic algorithms. Because HPSDE algorithm was created as a result of the hybridization of two global stochastic algorithms – DE and PSO; we compared its performance with results from another hybrid algorithm created by the hybridization of a global algorithm (PSO) and a local search algorithm (TS). In this regards, HPSDE was compared with PSOTS algorithm; and the result showed that the performance of PSOTS was 2.6% less than the performance of HPSDE. Interestingly, the PSOTS algorithm outperformed the remaining metaheuristic algorithms in the order DE, PSO and GA respectively.

Furthermore, with the aid of statistical tools, we used a collection of six benchmark tests of varying complexity to gain further insight into the relative performance of the algorithms. Based on the analyses of the results from the benchmark tests, the effectiveness of HPSDE algorithm over DE and PSO as demonstrated in the well placement optimization problems was reinforced. Also, due to the fact that computational complexity of population-based stochastic algorithms are critical to understanding relative algorithmic efficiency, the runtime and space complexity of the algorithms were analyzed by evaluating fundamental arithmetic and logical operations performed by the algorithms. By and large, these tools afforded us the ability to draw conclusions on the relative performance of the algorithms.

Despite the limitations that were highlighted in terms of usability of metaheuristic algorithms in the industry; this work demonstrates the potential benefit of hybridized metaheuristic algorithms over more established stochastic techniques in reservoir engineering applications. Besides the fact that these findings are promising, the applicability of HPSDE algorithm in well placement optimization problem shows that hybridization could be key to unlocking some of the challenging optimization problems in field development planning and reservoir engineering in general.

CHAPTER 5

But let the slave who sees another cast into a shallow grave know that he will be buried in the same way when his day comes – Chinua Achebe

PRODUCTION OPTIMIZATION AND CONTROL

In this chapter, the focus is on optimization and control of the production settings of a waterflooded reservoir. The working model is a two-phase immiscible reservoir flow model, and the implementation of the optimization and control strategies is such that the controller algorithm is embedded in the optimization algorithm. In other words, the optimization loop is fundamental in substance than the control loop – which is essentially a means for achieving as close as possible, the optimal trajectory resulting from the optimization loop. Although it has been mentioned that the working model is a two-phase immiscible reservoir flow model, it is important to note that the physics of the working models employed in the optimization and control loops are significantly different. Whereas a physics-based white-box reservoir model is employed for the optimization strategy, the embedded control strategy is carried out with the aid of a data-based black-box model. In the optimization loop, we espouse a technique presented in Sarma et al. (2005) and Jansen (2012) by employing a gradient-based algorithm in which the derivatives of the objective function are computed using an adjoint formulation. However, the control loop is based on a model predictive control (MPC) algorithm that employs linearized data-driven nonlinear models.

5.1 Well Flooding Optimization Formulation

Given the solution to the well placement problems presented in the previous chapter, we attempt in this chapter to find the dynamic production settings that maximizes the recovery factor of a water-flooded reservoir over a time interval [*0*, *T*]. For the avoidance of doubt, it is important to note that the applications considered in this chapter are direct follow-ups to sub-sections 4.5.2 and 4.5.3 of Chapter 4.

In every practical sense, the maximization of the recovery factor of a waterflooded petroleum reservoir is equivalent to any of the following:

- maximizing the cumulative volumes of hydrocarbon produced at terminal time *T*
- maximizing the water saturation of the reservoir at terminal time *T* or
- minimizing the volume of hydrocarbon in place at terminal time *T*

However, for conformity reasons, we choose to express the objective function in terms of an economic criterion such as net present value (NPV) of the asset. In this case, the NPV is a measure of the cash flow (CF) generated from produced volumes of hydrocarbon.

Mathematically, the NPV is defined as the total oil revenues minus the total cost of production, in combination with a discount factor – i.e. the time value of money. This can be represented in the equation below:

$$\mathrm{NPV} = \sum_{t=1}^{T} \frac{\mathrm{CF}_{(t)}}{\left(1+\mathrm{d}\right)^{t}} \tag{5.1}$$

where *T* is the terminal time or total production period (in days), d is the discount factor and $\mathrm{CF}_{(t)}$ represents the cash flow at time *t*. The cash flow at any time *t*, is given by:

$$\mathrm{CF}_{(t)} = \mathrm{REV}_{(t)} - \mathrm{OPEX}_{(t)} \tag{5.2}$$

where $\mathrm{REV}_{(t)}$ is the revenue accrued from sale of products at time *t*; and $\mathrm{OPEX}_{(t)}$ represents the operating expenditure at time *t*. Both quantities are usually measured in US dollars.

For a two-phase (oil and water) flow reservoir model, the values of $\mathrm{REV}_{(t)}$ and $\mathrm{OPEX}_{(t)}$ at any time (*t*) are respectively given by:

$$\mathrm{REV}_{(t)} = p_{(t)}^{\mathrm{oil}} \,\upsilon_{(t)}^{\mathrm{oil}} \tag{5.3}$$

$$\mathrm{OPEX}_{(t)} = p_{(t)}^{\mathrm{w,p}} \,\upsilon_{(t)}^{\mathrm{w,p}} + p_{(t)}^{\mathrm{w,i}} \,\upsilon_{(t)}^{\mathrm{w,i}} \tag{5.4}$$

where $p_{(t)}^{\mathrm{oil}}$ is the price of oil at time *t*, $p_{(t)}^{\mathrm{w,p}}$ is the cost of producing water in the production wells, and $p_{(t)}^{\mathrm{w,i}}$ is the cost of injecting water in the injection wells at time *t* – all three quantities are measured in dollars per m^3. On the other hand, $\upsilon_{(t)}^{\text{oil}}$ is the total volume of oil produced at time *t*, while $\upsilon_{(t)}^{\text{w,p}}$ and $\upsilon_{(t)}^{\text{w,i}}$ (all measured in m^3) represents the total volumes of water produced (from production wells) and injected (in injection wells) respectively.

Equations (5.1) to (5.4) can be cast in the form of the optimization problem below:

$$\underset{\text{NPV}}{J(\mathbf{u})} = \int_0^T \underbrace{\left[\underbrace{\sum_1^{\aleph^{\text{prod}}} p_{(t)}^{\text{oil}} \upsilon_{(t)}^{\text{oil}}}_{\text{REV}} - \underbrace{\left(\underbrace{\sum_1^{\aleph^{\text{prod}}} p_{(t)}^{\text{w,p}} \upsilon_{(t)}^{\text{w,p}}}_{\text{production cost}} + \underbrace{\sum_1^{\aleph^{\text{inj}}} p_{(t)}^{\text{w,i}} \upsilon_{(t)}^{\text{w,i}}}_{\text{injection cost}} \right)}_{\text{OPEX}} \right]}_{\text{CF}} \frac{1}{(1+\text{d})^t} dt \tag{5.5}$$

where $\aleph^{\text{prod}}$ and $\aleph^{\text{inj}}$ represent the total number of production wells and injection wells respectively, and **u** is a vector containing all the manipulated or input variables.

Using a discrete-time formulation, the performance measure *J* (**u**) in Eq. 5.5 can be cast into the following optimization problem:

$$J\left(\mathbf{u}_{1:K}, \mathbf{y}_{1:K}(\mathbf{u}_{1:K})\right) = \sum_{k=1}^{K} J_k\left(\mathbf{u}_k, \mathbf{y}_k\right) \tag{5.6}$$

where **y** is a vector containing the outputs, and J_k is the contribution (i.e. REV, OPEX) to J in each discrete time-step *k*.

In (5.6), it is important to highlight that the elements of the manipulated vector $\mathbf{u}_k$ are often subject to operational constraints. A bound constraint of the form $\mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max}$, underlines the fact that the inputs must come from within a specified feasible or admissible bound; there may be additional constraints in the form of maximum and minimum production rates (within the bounds of the handling capacity of the surface facility), water-cut thresholds, etc.

Following a slight modification of the formulation presented in the works of Sarma (2005) and Jansen (2012), the optimization problem cast in Eq. 5.6 can be re-formulated into the following optimal control problem:

$$\max_{\mathbf{u}_{1:K}} J\left(\mathbf{u}_{1:K}, \mathbf{y}_{1:K}(\mathbf{u}_{1:K})\right) \tag{5.7}$$

$$\text{over } \mathcal{U} = \left\{\mathbf{u}_k \in \mathbb{R}^m : \mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max}\right\}$$

subject to: $\mathbf{g}_k\left(\mathbf{u}_k, \mathbf{x}_{k-1}, \mathbf{x}_k\right) = \mathbf{0}, \forall k \in (1, 2, \ldots, K)$ (system equation)

$\mathbf{x}(0) = \bar{\mathbf{x}}_0$ (initial condition)

$\mathbf{u}_k \in \mathcal{U}\, \forall k, k \in (1, 2, \ldots, K)$ (allowable input)

$\mathbf{h}_k\left(\mathbf{u}_k, \mathbf{x}_k, \mathbf{y}_k\right) = \mathbf{0}$ (output equation)

$\mathbf{c}_k\left(\mathbf{u}_k, \mathbf{y}_k\right) = \mathbf{0}$ (equality constraint)

$\mathbf{d}_k\left(\mathbf{u}_k, \mathbf{y}_k\right) \le \mathbf{0}$ (inequality constraint)

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^p$, $\mathbf{c} \in \mathbb{R}^q$ and $\mathbf{d} \in \mathbb{R}^r$.

The interpretation of the optimization problem in Eq. 5.7 is self-explanatory; however, for the sake of clarity it is worth explaining. The unknown we seek to find is the manipulated variable $\mathbf{u}_{1:K}$ and this control input must be from the allowable input within the definitive bound $\mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max}$. Once this variable is obtained, it determines the resulting output from which we can subsequently determine a value for the objective function *J*. In other words, the problem is finding the control $\mathbf{u}_{1:K}$ which satisfies all prevailing constraints, and leads to the highest possible value of the objective function *J*.

A number of techniques have been employed in attempts at solving well production settings optimization problems cast in the format of Eq. 5.7; and of these techniques, the gradient-based methods have generally proved computationally friendly. Broadly speaking, gradient-based algorithms are iterative algorithms which often require derivative information of the objective function with respect to defined control variables. Essentially, there are three main techniques for computing gradients – the finite differences, forward sensitivity equation, and backward adjoint formulation. The finite difference approach has the drawback of requiring far too many objective function evaluations. In fact, a minimum of $2m+1$ objective function evaluations are required for m decision variables; and since objective function evaluation in reservoir engineering problems often entails reservoir simulation run; the applicability of finite difference approach in large-scale systems like reservoir models suffer severe limitations. With the forward sensitive equation approach, one simulation run of the model is required in addition to m sensitivity models. Again, it is also important to underscore that many objective function evaluation is necessary in order to obtain the sensitivities of the control and state variables; and these sensitivities are subsequently used for the computation of the sensitivities to the outputs by applying the chain rule, Ringset et al. (2011). The obvious drawback of this approach in large-scale systems is the computational memory required for the storage of the huge sensitivity information that is required for the computation of the objective function derivatives. The computation of gradients via adjoint formulation is fundamentally based on the introduction of extra variables (to satisfy some special condition known as the adjoint equations) into the underlying optimization problem, and using these variables to circumvent certain parts of the derivative information calculations. In other words, by solving the adjoints equations, we circumvent the onerous computational task of computing the sensitivities of both the control and state variables. Regardless of the number of decision variables, the use of the adjoint formulation method for computation of derivatives requires only two simulation runs. Importantly, optimal control theory offers a very reliable and efficient method of solving the adjoint equation as demonstrated in Luenberger (1979). In other words, this method reduces the computation cost of the objective function gradient information computation; it is therefore efficient, and can be applied in large-scale nonlinear systems (as evident in its application in metrology and oceanography) such as reservoir models.

In reservoir engineering, the use of gradient-based adjoint formulation in production optimization dates back to the 1980s; a non-exhaustive review of its application in this field is presented in section 2.2. In the past decade, the implementation of this technique in the oil and gas industry has gained considerable popularity – this is evident in the fact that most reservoir flow simulators now possess adjoint functionality for the computation of the derivatives of the objective function with respect to control variables.

5.1.1 Necessary Conditions for Optimality

One of the issues associated with optimal control problems is the fundamental question of characterizing an optimal solution. Put differently, how and when do we adjudge or designate a particular control vector as optimal? In seeking to answer this question, we exploit the fact that both classical and modern optimal control theories are natural extension of the calculus of variation. In many respects, optimal control and the maximization of a function of a single variable as contained in the study of calculus share a lot of similarities among which include the fact that the conditions for optimization are derived by considering the effect of small changes in the control near the optimal point. Therefore, to characterize a control vector $\mathbf{u}$ at time-step k as optimal, there is need to trace the effect of an arbitrary change in $\mathbf{u}_k$ on the objective function, and importantly, there is a requirement that the resulting performance index J is non-improving – see Luenberger (1979).

However, in this problem, the computation of the derivatives of the objective function J with respect to the manipulated variable $\mathbf{u}_{1:K}$ is a difficult task. As pointed out in Jansen (2012), the difficulty arises because of the indirect dependence of the variation in the objective function (δJ) on the variation of elements of the input vector. This is the case because a variation in any element i of the input at any arbitrary time-step (δu_{ik}), does not only influence the outcome of J at k, it also influences the states $\mathbf{x}_{k:K}$, the outputs $\mathbf{y}_{k:K}$ and consequently, J at subsequent time-steps.

Thus, the variation has a "knock-on" effect which only makes sense if it is computed by the chain rule of differentiation as follows:

$$\frac{dJ}{d\mathbf{u}_k} = \left[\frac{\partial J_k}{\partial \mathbf{u}_k} + \sum_{j=k}^{K} \frac{\partial J_j}{\partial \mathbf{y}_j} \left(\frac{\partial \mathbf{y}_j}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_j} \frac{\partial \mathbf{x}_j}{\partial \mathbf{u}_k} \right) \right] \tag{5.8}$$

Since the output equation $\mathbf{h}_k(\mathbf{u}_k, \mathbf{x}_k, \mathbf{y}_k) = \mathbf{0}$ generally forms a system of explicit nonlinear algebraic equations, output terms such as $\partial \mathbf{y}_j / \partial \mathbf{u}_k$ and $\partial \mathbf{y}_j / \partial \mathbf{x}_j$ in Eq. 5.8 can be computed directly. Also, the terms $\partial J_k / \partial \mathbf{u}_k$ and $\partial J_j / \partial \mathbf{y}_j$ can also be computed without problems. However, computing the term $\partial \mathbf{x}_j / \partial \mathbf{u}_k$ in Eq. 5.8 causes a lot of difficulties because of the need to solve the recursive system of discrete-time differential equations and connect the state vector $\mathbf{x}_j$ (where $j = k, k+1, \ldots, K$) to the input $\mathbf{u}_k$.

Consequently, there is need to employ an indirect approach in order to overcome the difficulties arising from the complex temporal dependence of elements in $\partial \mathbf{x}_j / \partial \mathbf{u}_k$. The 'trick' is to modify the objective function by 'adjoining' additional terms (constraints) which has a net zero-sum effect on the modified objective function. In this regard, the initial condition of the system $\mathbf{x}(0) = \bar{\mathbf{x}}_0$, the systems equation $\mathbf{g}_k(\mathbf{u}_k, \mathbf{x}_{k-1}, \mathbf{x}_k)$, the output equation $\mathbf{h}_k(\mathbf{u}_k, \mathbf{x}_k, \mathbf{y}_k)$ and the equality constraint $\mathbf{c}_k(\mathbf{u}_k, \mathbf{y}_k)$ are 'adjoined' to the objective function using distinct vectors of Lagrange multipliers $\boldsymbol{\lambda}_0, \boldsymbol{\lambda}_{1:K}, \boldsymbol{\mu}_{1:K}$ and $\boldsymbol{\upsilon}_{1:K}$ respectively. In other words, a vector of Lagrange multiplier is required for each of the adjoining constraint with which the original objective function is augmented; thus, the total number of Lagrange multipliers is equal to the product of the dynamic states and control steps, Sarma (2005). To this end, a modified objective function $\bar{J}$ is defined as follows:

$$\bar{J}\left(\mathbf{u}_{1:K}, \mathbf{x}_{0:K}, \mathbf{y}_{1:K}, \boldsymbol{\lambda}_{0:K}, \boldsymbol{\mu}_{1:K}, \boldsymbol{\upsilon}_{1:K}\right) \triangleq \sum_{k=1}^{K} \begin{bmatrix} J_k(\mathbf{u}_k, \mathbf{y}_k) \\ +\boldsymbol{\lambda}_0^T(\mathbf{x}(0) - \bar{\mathbf{x}}_0) \\ +\boldsymbol{\lambda}_k^T \mathbf{g}_k(\mathbf{u}_k, \mathbf{x}_{k-1}, \mathbf{x}_k) \\ +\boldsymbol{\mu}_k^T \mathbf{h}_k(\mathbf{u}_k, \mathbf{x}_k, \mathbf{y}_k) \\ +\boldsymbol{\upsilon}_k^T \mathbf{c}_k(\mathbf{u}_k, \mathbf{y}_k) \end{bmatrix} \tag{5.9}$$

Since the additional terms that are ‘adjoined’ to the original objective function J has a net zero-sum for any trajectory, it therefore follows that the value of the modified objective function $\bar{J}$ is the same as the value of J for any allowable $\mathbf{u}_k$ that satisfies the original optimization problem. Thus, a necessary condition for an optimum of the modified objective function $\bar{J}$ is the requirement that any variations of the input control must be non-improving, i.e. all derivatives of $\bar{J}$ with respect to the dependent variables are equal to zero. In Jansen (2012), it was demonstrated that (5.9) can be rearranged into a set of derivatives of $\bar{J}$ which represent the Karush-Kuhn-Tucker (KKT) or first-order necessary conditions for an optimum. These equations include:

$$\frac{\partial \bar{J}}{\partial \mathbf{u}_k} \equiv \frac{\partial J_k}{\partial \mathbf{u}_k} + \boldsymbol{\lambda}_k^T \frac{\partial \mathbf{g}_k}{\partial \mathbf{u}_k} + \boldsymbol{\mu}_k^T \frac{\partial \mathbf{h}_k}{\partial \mathbf{u}_k} + \boldsymbol{\upsilon}_k^T \frac{\partial \mathbf{c}_k}{\partial \mathbf{u}_k} = \mathbf{0}^T, \quad \forall k \in (1, 2, \ldots, K) \tag{5.10}$$

$$\frac{\partial \bar{J}}{\partial \mathbf{x}_0} \equiv \boldsymbol{\lambda}_1^T \frac{\partial \mathbf{g}_1}{\partial \mathbf{x}_0} + \boldsymbol{\lambda}_0^T = \mathbf{0}^T \tag{5.11}$$

$$\frac{\partial \bar{J}}{\partial \mathbf{x}_k} \equiv \boldsymbol{\lambda}_{k+1}^T \frac{\partial \mathbf{g}_{k+1}}{\partial \mathbf{x}_k} + \boldsymbol{\lambda}_k^T \frac{\partial \mathbf{g}_k}{\partial \mathbf{x}_k} + \boldsymbol{\mu}_k^T \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k} = \mathbf{0}^T, \quad \forall k \in (1, 2, \ldots, K-1) \tag{5.12}$$

$$\frac{\partial \bar{J}}{\partial \mathbf{x}_K} \equiv \boldsymbol{\lambda}_K^T \frac{\partial \mathbf{g}_K}{\partial \mathbf{x}_K} + \boldsymbol{\mu}_K^T \frac{\partial \mathbf{h}_K}{\partial \mathbf{x}_K} = \mathbf{0}^T \tag{5.13}$$

$$\frac{\partial \bar{J}}{\partial \mathbf{y}_k} \equiv \frac{\partial J_k}{\partial \mathbf{y}_k} + \boldsymbol{\mu}_k^T \frac{\partial \mathbf{h}_k}{\partial \mathbf{y}_k} + \boldsymbol{\upsilon}_k^T \frac{\partial \mathbf{c}_k}{\partial \mathbf{y}_k} = \mathbf{0}^T, \quad \forall k \in (1, 2, \ldots, K) \tag{5.14}$$

$$\frac{\partial \bar{J}}{\partial \boldsymbol{\lambda}_0} \equiv \left(\mathbf{x}(0) - \bar{\mathbf{x}}_0\right)^T = \mathbf{0}^T, \quad \forall k \in (1, 2, \ldots, K) \tag{5.15}$$

$$\frac{\partial \bar{J}}{\partial \boldsymbol{\lambda}_k} \equiv \mathbf{g}_k^T\left(\mathbf{u}_k, \mathbf{x}_{k-1}, \mathbf{x}_k\right) = \mathbf{0}^T, \quad \forall k \in (1, 2, \ldots, K) \tag{5.16}$$

$$\frac{\partial \bar{J}}{\partial \boldsymbol{\mu}_k} \equiv \mathbf{h}_k^T\left(\mathbf{u}_k, \mathbf{x}_k, \mathbf{y}_k\right) = \mathbf{0}^T, \quad \forall k \in (1, 2, \ldots, K) \tag{5.17}$$

$$\frac{\partial \bar{J}}{\partial \boldsymbol{\upsilon}_k} \equiv \mathbf{c}_k^T\left(\mathbf{u}_k, \mathbf{y}_k\right) = \mathbf{0}^T, \quad \forall k \in (1, 2, \ldots, K) \tag{5.18}$$

Upon close observation, it is easy to see that (5.15), (5.16), (5.17) and (5.18) are respectively identical to the initial condition, systems equation, output equation and equality constraint, and are therefore automatically satisfied.

5.1.2 Computation of the Lagrange Multipliers

In most in-house, commercial and open source reservoir flow simulator; there is often an in-built strategy that enforces $\mathbf{c}_k\left(\mathbf{u}_k, \mathbf{y}_k\right) = \mathbf{0}$ (i.e. the equality constraint) via the implementation of the so-called voidage replacement. Thus, if we temporarily set aside the equality constraint, then Eq. 5.14 allows us to compute the Lagrange multiplier $\boldsymbol{\mu}_{1:K}$. Next, we use Eq. 5.13 to compute the Lagrange multiplier $\boldsymbol{\lambda}_K$ for the final discrete time-step *K*; and thereafter, Eq. 5.12 can be used to compute the Lagrange multipliers $\boldsymbol{\lambda}_k$ for $k = K-1, K-2, \ldots, 1,$ (i.e. recursively or backward in time). Eq. 5.11 allows us to compute the Lagrange multiplier $\boldsymbol{\lambda}_0$; and finally Eq. 5.10 represents the effect of any change in the control input $\mathbf{u}_k$ on the value of the objective function, while keeping all other variables fixed. In other words, Eq. 5.10 is the expression needed to trace the effect of any arbitrary change in $\mathbf{u}_k$ on the objective function; $\mathbf{u}_k$ is adjudged optimal if the effect of its arbitrary change on $\bar{J}$ is non-improving.

Thus, in this technique, the states and outputs are computed using a forward simulation, while the Lagrange multipliers are recursively computed (first for the last time-step *K* and then *K*−1, *K*−2… *k*) using a backward simulation. Note that the computation of the Lagrange multipliers as represented can be interpreted as the solution of an ‘adjoint’ system of discrete-time differential equations, and the magnitude of the Lagrange multipliers is a first-order measure of the effect of violating the corresponding constraints on the value of the objective function, Jansen (2012). Optimal control theory therefore, provides an effective mechanism for the efficient computation of the gradients of the objective function with respect to the manipulated variables. The efficiency of this technique hinges on the fact that irrespective of the number of decision variables, the gradients are computed in as little as two simulation runs – a forward simulation for the computation of the dynamic states and outputs, and a backward simulation for the computation of the vectors of Lagrange multipliers which are subsequently used to determine the gradients. Algorithm 5.1 is a step-by-step procedure of the production optimization problem as adapted from Jansen (2012), and Figure 5.1 depicts a simple schematic flowchart of the process. Owing to its computational efficiency, modern numerical reservoir simulators have in-built adjoint model capabilities for the implementation of this optimization technique.

Algorithm 5.1 Adjoint and Gradient Computation Algorithm

```
1. Choose an initial control vector $\mathbf{u}_{1:K}$, and solve the forward model
   equation using the given initial condition $\mathbf{x}(0) = \bar{\mathbf{x}}_0$.
2. Compute the states $\mathbf{x}_{1:K}$ and the outputs $\mathbf{y}_{1:K}$ using the system
   equation $\mathbf{g}_k(\mathbf{u}_k, \mathbf{x}_{k-1}, \mathbf{x}_k) = \mathbf{0}$ and the output equation
   $\mathbf{h}_k(\mathbf{u}_k, \mathbf{x}_k, \mathbf{y}_k) = \mathbf{0}$ respectively
3. Store the computed dynamic states
4. Compute the objective function of the forward simulation $J$ using Eq.
   5.6; if converged stop, else continue
5. Solve the adjoint model equation by computing the Lagrange
   multipliers $\boldsymbol{\mu}_{1:K}, \boldsymbol{\lambda}_{1:K}$, and $\boldsymbol{\upsilon}_{1:K}$ using equations (5.14), (5.13), and
   (5.12)
6. Using the computed Lagrange multipliers, calculate the gradients of
   the objective function with respect to the controls $dJ/d\mathbf{u}_{1:K}$ in
   accordance to Eq. 5.11
7. With the aid of the computed gradients, determine improved search
   direction and improved control vector $\mathbf{u}_{1:K}$, using a quasi-Newton
   algorithm (or any gradient-based algorithm of choice)
8. Revert to step 2, and repeat all steps until all gradients are zero (or as
   close to zero as possible)
```

Despite the popularity and computational efficiency of this technique, it is important to note that the optimal production trajectory resulting from the production optimization problem above is hardly attainable in practice, Saputelli et al. (2006), Rezapour (2009). Issues ranging from under-modeling to over-modeling of physics-based reservoir model; and the inherent range of geological uncertainty within the model translates to the inevitable reality that model-based optimal production trajectory are not realizable. To this end, a predictive control strategy that is based on data-driven model is employed in tracking the optimal trajectory that results from the well production optimization problem described above.

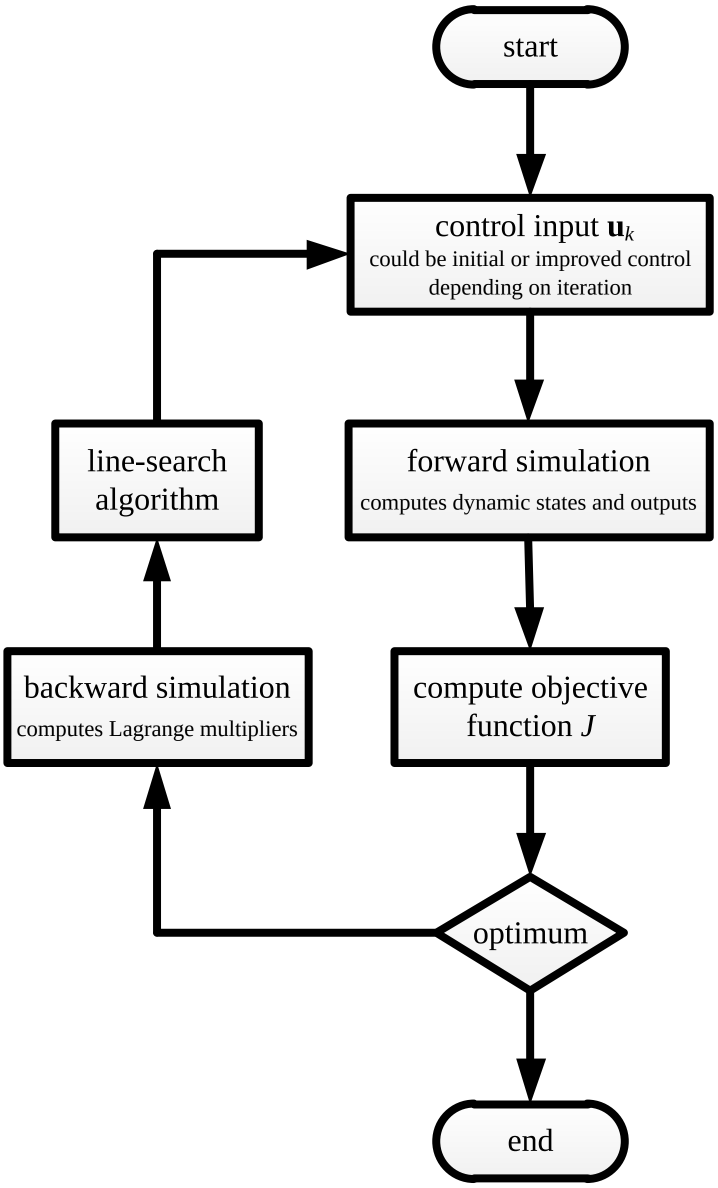

start
control input $\mathbf{u}_k$
could be initial or improved control depending on iteration
forward simulation
computes dynamic states and outputs
compute objective function J
optimum
end
backward simulation
computes Lagrange multipliers
line-search algorithm

Figure 5.1: Flowchart showing the gradient-based optimization using the adjoint formulation.

5.2 Embedded Trajectory Tracking and Control

The embedded trajectory tracking is based on the control strategy known as model predictive control (MPC). The implementation of this control strategy rests on the fundamental control principle in which future control inputs and future process responses are predicted with the aid of data-driven models, and the results are optimized at regular sampling intervals with respect to an objective function. In other words, the controller uses output measurements at sampling time k to predict the dynamic behavior of the system over a finite control or prediction horizon and determines the input such that deviations from the performance objective is minimized. For more information on MPC and its nonlinear extension – nonlinear model predictive control (NMPC), interested readers are referred to Allgower et al. (2000, 2004), Maciejowski (2002), Qin and Badgwell (2003), Camacho and Bordons (2007a, 2007b), Findeisen et al. (2007), Huang and Kadali (2008) and Rawlings and Mayne (2009).

Figure 5.2 shows a pictorial illustration of the control principle from the input and output viewpoints. The corrected control inputs $\tilde{\mathbf{u}}_{1:K} = \mathbf{u}_{k|k}, \mathbf{u}_{k+1|k}, \ldots, \mathbf{u}_{k+N_2-1|k}$ for a given prediction horizon are computed based on the well production settings optimization highlighted in the previous section, and the dynamic behavior of the process $\mathbf{x}_{k+i|k}$ is predicted for a control horizon as measurements $\mathbf{y}_{k+i|k}$ become available.

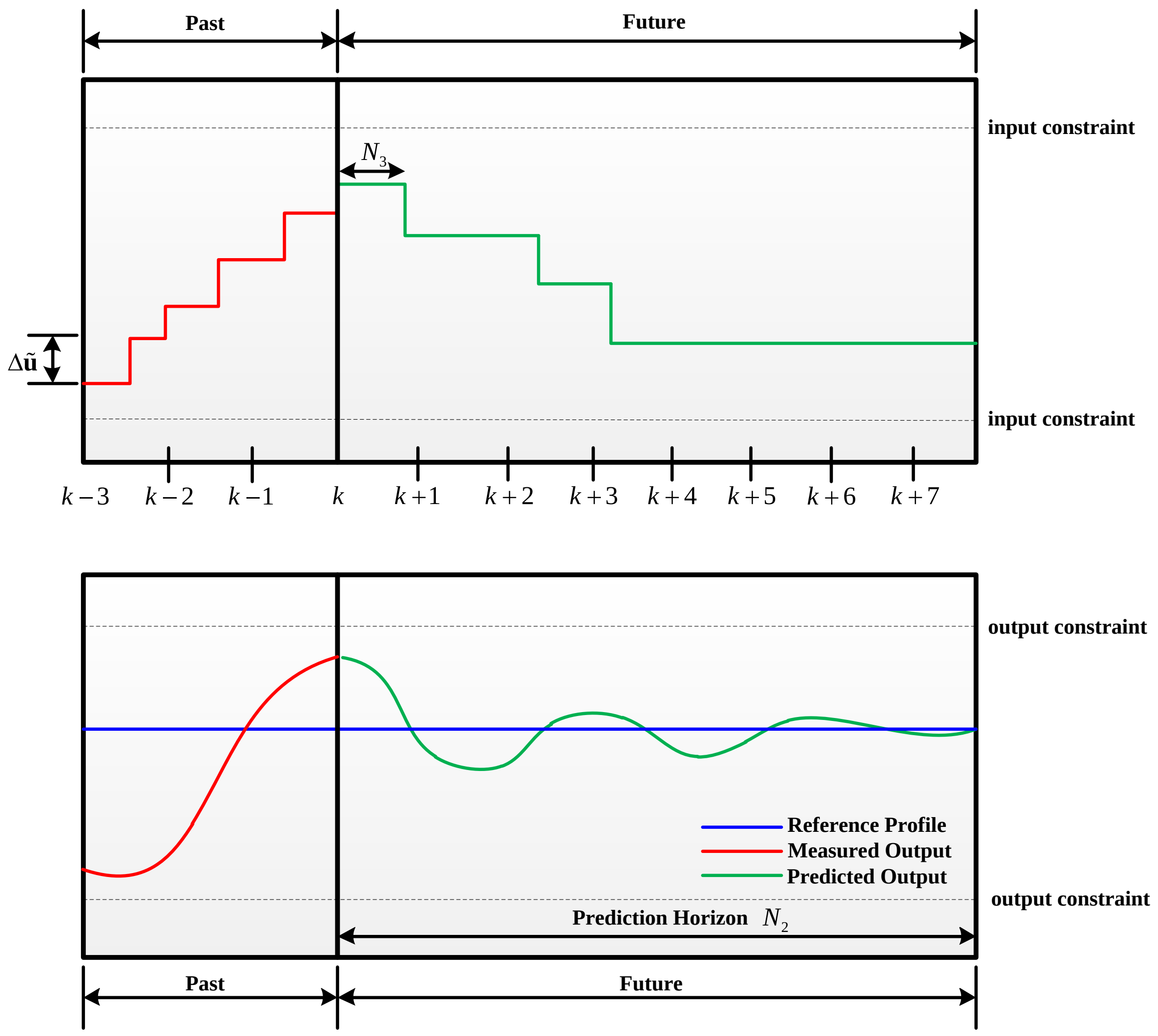


Figure 5.2: Illustrative representation of Model Predictive Control (MPC) from the input (upper) and output (lower) viewpoints, adopted from Currie, (2011).

Note that $\mathbf{u}_{k+i|k}$, $\mathbf{x}_{k+i|k}$ and $\mathbf{y}_{k+i|k}$ respectively denote the input, state and output vectors at time $k+i$, predicted or measured (as the case may be) at time *k*. Feedback is technically incorporated by applying the obtained open-loop input until the next sampling time $k+1$, at which the entire process of prediction and optimization is repeated. The MPC procedure is summarized in Algorithm 5.2.

Algorithm 5.2 Model Predictive Control Algorithm

```
1. Compute the estimates of the current state of the system at sampling
   time $k$
2. Compute an admissible optimal input by minimizing the desired cost
   function over a prediction horizon using the system model and the
   current state estimate for prediction
3. Implement the computed optimal input until the next sampling time
   $k+1$
4. Revert back to step 1
```

Since MPC makes explicit use of a model to predict the future behavior of the process, it cannot be implemented without the identification of appropriate mathematical model of the process. To this end, nonlinear models are identified using **MATLAB**® system identification toolbox. These data-driven nonlinear models have the ability to capture process nonlinearities when deployed in MPC strategies; however, they lead to computationally demanding NMPC. The high computational cost associated with NMPC stems from the fact that the often easy-to-solve quadratic programming (QP) problem that results from linear MPC changes to a more challenging nonlinear programming (NLP) problem when the underlying model is nonlinear. Hence, it is imperative to strike a balance between the need to deploy data-driven nonlinear models that are capable of capturing the inherent nonlinear dynamics of the waterflooding process, and the need to avoid the computational cost of the challenging NLP problem that results from the use of nonlinear models and the ensuing NMPC. Consequently, the identified nonlinear model (which captures the salient nonlinearities of the process) is linearized into a state-space model about an operating regime defined by the state. This linearization is achieved by computing a first-order Taylor series approximation of the nonlinear model; and subsequently, the input, state and output operating points are included as biases inside the MPC controller. The resulting linearized models are often accurate in the neighborhood of the operating points, and importantly, they achieve better simulation fit when compared with simple linear models. Ultimately, this enables the controller to formulate the control problem in computationally friendly quadratic programming format.

In addition, it is noted that the evolving dynamic states of the identified model are of crucial importance in the implementation of MPC strategy. This is because a system usually evolves according to its state vector equations; therefore, any control strategy that is aimed at influencing future behavior must be based on the current dynamic state of the system. To this end, a Kalman Filter is introduced as an observer for state estimation of the linearized nonlinear model.

5.2.1 Model Identification

Developing a model that can adequately describe the relationship between the input and output data of the waterflooding process is the first step in this control problem. In system identification, the accuracy of the identified model to a large extent depends on the “quality” of the original input-output experimental data. In other words, there is need for proper design of the input signal, and this should be tailored along the characteristics of the underlying physics of the first-principle white-box reservoir model. To achieve this, the experimental design espoused the procedure outlined in Andersson et al. (1998) and van Essen et al. (2010). In this regard, preliminary step-response and impulse-response experiments are carried out to assess the dynamics of the system. These preliminary investigations afford us the necessary information required to determine the system’s time constant, time delay and nonlinearity. Thus, the input signals are excited with various step functions, and the corresponding outputs are measured accordingly. The experiment duration and the bandwidth of the system are determined by rule of thumb; such that the minimal time adopted for experiment duration is five-times the largest time constant, and the system bandwidth is equal to $1/(T_s M)$ – where T_s is the sampling interval, and M is the number of sampling intervals. Furthermore, experimental data is pre-processed, and divided into distinct identification and validation data sets. In this regard, two-third of the experimental data was assigned for identification of the data-driven nonlinear model; while the remaining third is designated for validation purposes.

5.2.2 State Estimation

For effective MPC strategy, the evolving dynamic state of the linearized model is estimated using a state observer which estimates the current state of the model. Usually, Bayesian methods provide a general framework for state estimation, and the approach involves the use of measurement datasets to estimate the probability density function (PDF) of the state. Thus, many state estimation algorithms involve the estimation of the PDF of system states that are not directly observable. Based on the current state, the next state is predicted and subsequently updated from available measurements. Because of their high efficiency, the Kalman Filter (KF) algorithm is employed for this purpose. State estimation of linear models using KF is a popular and well-known technique, Kailath et al. (2000). It is a recursive estimation procedure that uses sequential datasets in such a way that prior knowledge of the state (expressed by the covariance matrix) is improved at each step by taking the prior state estimates and new measurement data for subsequent state estimation, Tayamon (2012).

Consider the discrete-time state space equation of the reservoir model in (3.48) and (3.49). If we recast the equations in the form:

$$\mathbf{x}_{k+1} = \mathbf{A}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{B}(\mathbf{x}_k)\mathbf{u}_k + w_k \tag{5.19}$$

$$\mathbf{y}_k = \mathbf{C}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{D}(\mathbf{x}_k)\mathbf{u}_k + v_k \tag{5.20}$$

with

$$w_k \sim N(0, Q_k), \quad v_k \sim N(0, R_k)$$

$$\mathbf{E}(w_i w_j^T) = Q_i \delta(i-j)$$

$$\mathbf{E}(v_i v_j^T) = R_i \delta(i-j)$$

$$\mathbf{E}(w_k v_k^T) = \mathbf{0}$$

where the vectors $w_k \in \mathbb{R}^n$ and $v_k \in \mathbb{R}^p$ are process noise and measurement noise – both vectors are uncorrelated white noises with zero mean value, **E** denotes the expected value operator, δ is the Kronecker delta, and the matrices $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{p\times p}$ are the covariance matrices of the noise sequences of w_k and v_k.

According to Therrien (1992), the state and output are Gaussian if the model is linear and the input is Gaussian. Thus, the state and output PDFs will always be normally distributed, and what this means is that the mean and covariance are sufficient statistics. In other words, it is not necessary to calculate a full state PDF in order to estimate the state; a mean vector $\hat{\mathbf{x}}$ and covariance matrix P for the state will be sufficient. This is the underlying principle of the Kalman Filter. It is basically a Bayesian estimator decomposed into two steps, namely: the prediction step and the updating step, Orderud (2005).

The Kalman Filtering process is as follows:

1. Prior to measurements or observation, the next state (*priori estimate*) is predicted according to the equation:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}(\mathbf{x}_k)\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}(\mathbf{x}_k)\mathbf{u}_k \tag{5.21}$$

$$P_{k|k-1} = \mathbf{A}(\mathbf{x}_k)P_{k-1|k-1}\mathbf{A}(\mathbf{x}_k)^T + Q_k \tag{5.22}$$

2. After measurements become available, the state is updated (*posteriori estimate*) in accordance to the equation:

$$K_k = P_{k|k-1}\mathbf{C}(\mathbf{x}_k)^T\left(\mathbf{C}(\mathbf{x}_k)P_{k|k-1}\mathbf{C}(\mathbf{x}_k)^T + R_k\right)^{-1} \tag{5.23}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k\left(\mathbf{y}_k - \mathbf{C}(\mathbf{x}_k)\hat{\mathbf{x}}_{k|k-1}\right) \tag{5.24}$$

$$P_{k|k} = \left(\mathbf{I} - K_k\mathbf{C}(\mathbf{x}_k)\right)P_{k|k-1} \tag{5.25}$$

where K is the Kalman gain matrix, used in the update observer, and P is the covariance matrix for the state estimate, containing information about the accuracy of the estimate.

5.2.3 MPC Problem Formulation

In formulating the MPC problem, we consider the discrete-time state space equation of the reservoir model in (3.48). On the basis of the state-space equation, the dynamic state of the reservoir model $\mathbf{x}_{k+i|k}$ evolves according to the prediction model:

$$\mathbf{x}_{k+i+1|k} = \mathbf{A}(\mathbf{x}_{k+i|k})\mathbf{x}_{k+i|k} + \mathbf{B}(\mathbf{x}_{k+i|k})\mathbf{u}_{k+i|k}, \quad i = 0,1,... \tag{5.26}$$

$$\mathbf{x}_{k|k} = \hat{\mathbf{x}}_k \tag{5.27}$$

where (5.27) defines the initial condition at the beginning of the prediction horizon.

The control law is computed by minimizing a performance function which is given by:

$$V(\mathbf{u}_k) = \sum_{i=N_1}^{N_2}\left\|\mathbf{y}_{k+i|k} - \mathbf{r}_{k+i|k}\right\|_{\boldsymbol{Q}}^2 + \sum_{i=1}^{N_3-1}\left\|\tilde{\mathbf{u}}_{k+i|k} - \hat{\mathbf{u}}_{k+i|k}\right\|_{\boldsymbol{R}}^2 \tag{5.28}$$

where $\mathbf{y}_k$ is a vector of measured outputs, N_1 accounts for any possible delays in the input, and $\tilde{\mathbf{u}}_k$ is a vector of corrected inputs which effect is to minimize the discrepancy between the actual measured output $\mathbf{y}_k$, and the reference trajectory $\mathbf{r}_k$. For all intents and purposes, the reference trajectory corresponds to the optimal output resulting from the well production settings optimization; the entire set up is illustrated in Figure 5.3.

Note that $\boldsymbol{Q}$ and $\boldsymbol{R}$ are appropriate matrices that account for penalty weights in states and control prediction respectively. In this case, the cost function penalizes any deviations of the measured outputs from the expected output (reference trajectory); N_2 and N_3 are the prediction and control horizon respectively. Note that we assume that the horizon $N_3 \leq N_2$, so that in all cases $\tilde{\mathbf{u}}_{k+i|k} - \hat{\mathbf{u}}_{k+i|k} = \mathbf{0}$ for $i \geq N_3$.

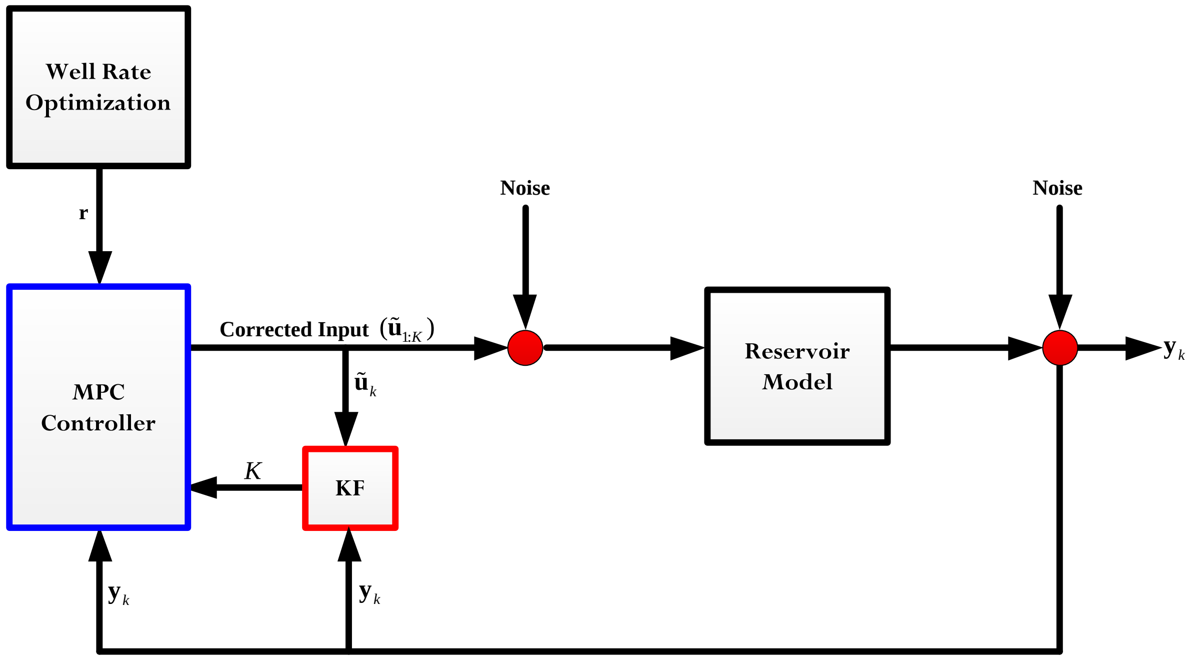


Figure 5.3: Schematic illustration of the working of MPC controller as modified from Currie (2011)

If the predictive controller is expressed in control terms such that $\Delta\hat{\mathbf{u}}_{k|k} = \tilde{\mathbf{u}}_{k|k} - \hat{\mathbf{u}}_{k|k}$; then Eq. 5.28 can be reduced to:

$$V(\mathbf{u}_k) = \|\mathcal{Y}_k - \mathfrak{R}_k\|_{\mathcal{Q}}^2 + \|\Delta\mathcal{U}_k\|_{\mathcal{R}}^2 \tag{5.29}$$

where $\mathcal{Y}_k = \begin{bmatrix} \mathbf{y}_{k+N_1|k} \\ \vdots \\ \mathbf{y}_{k+N_2|k} \end{bmatrix}$, $\mathfrak{R}_k = \begin{bmatrix} \mathbf{r}_{k+N_1|k} \\ \vdots \\ \mathbf{r}_{k+N_2|k} \end{bmatrix}$, $\Delta\mathcal{U}_k = \begin{bmatrix} \Delta\hat{\mathbf{u}}_{k|k} \\ \vdots \\ \Delta\hat{\mathbf{u}}_{k+N_3-1|k} \end{bmatrix}$, and the matrices $\mathcal{Q}$ and $\mathcal{R}$ are the extended weighing matrices which are respectively defined as follows:

$$\mathcal{Q} = \begin{bmatrix} \boldsymbol{Q}_{N_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}_{N_1+1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{Q}_{N_2} \end{bmatrix}, \ \mathcal{R} = \begin{bmatrix} \boldsymbol{R}_0 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_1 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{R}_{N_3} \end{bmatrix}.$$

By defining the variables as above, one can show that the controlled variables vector can take the form:

$$\mathcal{Y}_k = \mathbf{\Psi}\hat{\mathbf{x}}_{k|k} + \mathbf{\Upsilon}\mathbf{u}_{k-1} + \mathbf{\Theta}\Delta\mathcal{U}_k \tag{5.30}$$

where the matrices $\mathbf{\Psi}, \mathbf{\Upsilon}$ and $\mathbf{\Theta}$ are appropriate functions of the state-space model.

If we define a vector of future predicted errors by $\mathcal{E}_k = \mathfrak{R}_k - \mathbf{\Psi}\hat{\mathbf{x}}_{k|k}\mathbf{\Upsilon}\mathbf{u}_{k-1}$, then Eq. 5.29 can be expressed as:

$$V(\mathbf{u}_k) = const - \Delta\mathcal{U}_k\mathcal{H}_1 + \Delta\mathcal{U}_k{}^T\mathcal{H}_2\Delta\mathcal{U}_k \tag{5.31}$$

where $\mathcal{H}_1 = 2\Theta^T\mathcal{Q}\mathcal{E}_k$, $\mathcal{H}_2 = \Theta\mathcal{Q}\Theta + \mathcal{R}$ and neither $\mathcal{H}_1$ nor $\mathcal{H}_2$ depends on $\Delta\mathcal{U}_k$.

Given constraints in the form of: $E = \begin{bmatrix} \Delta\mathcal{U}_k \\ \mathbf{1} \end{bmatrix} \le \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$; $F = \begin{bmatrix} \mathcal{U}_k \\ \mathbf{1} \end{bmatrix} \le \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$; $G = \begin{bmatrix} \mathcal{Y}_k \\ \mathbf{1} \end{bmatrix} \le \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}$ (for appropriate matrices *E*, *F* and *G*); the constrained MPC problem becomes an optimization (minimization) problem as follows:

$$\begin{aligned} &\min && -\Delta\mathcal{U}_k\mathcal{H}_1 + \Delta\mathcal{U}_k{}^T\mathcal{H}_2\Delta\mathcal{U}_k \\ &\text{subject to:} && \Omega\Delta\mathcal{U}_k \le \omega \end{aligned} \tag{5.32}$$

where Ω and ω are derived from the constraints. Note that Eq. 5.32 is solved as a QP, as it has the general form:

$$\begin{aligned} &\min && \frac{1}{2}\theta^T\Phi\theta + \vartheta^T\theta \\ &\text{subject to:} && \Omega\theta \le \omega \end{aligned} \tag{5.33}$$

The identification of nonlinear models from the process data is relatively easy, and so is the linearization (which is required to circumvent the complications associated with nonlinear models, and the challenges of NLP that result from NMPC) of the model. These properties were exploited because of the relative ease of solving the resulting QP.

5.3 Applications

In this section, we consider two examples of the application of the well-rate optimization and control strategy in the maximization of the NPV of waterflooded reservoirs. In both cases, we begin with the well production optimization problem using a gradient optimization technique where the gradients are computed via the adjoint formulation as presented in section 5.1. Here, the input vector $\mathbf{u}_{1:K} = \mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_K$ is iteratively adjusted until such a vector $\hat{\mathbf{u}}_{1:K} = \mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_K$ (optimal input vector) which maximizes the NPV (objective function) is attained. The associated output or total flow rates at the producers that yields a maximum value of the objective function corresponds to the optimal output $\hat{\mathbf{y}}_{1:K} = \mathbf{y}_1, \mathbf{y}_2, ...\mathbf{y}_K$; and for intents and purposes, this vector is set as the reference trajectory $\mathbf{r}$ of the predictive control strategy. With the aid of Eq. 5.28, the controller computes corrected input vector $\tilde{\mathbf{u}}_{1:K} = \mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_K$ such that any deviation between the measured output (resulting from the corrected inputs $\tilde{\mathbf{u}}$) and the reference trajectory $\mathbf{r}$ is as minimal as possible.

5.3.1 Example 1: Reservoir model with an injector and a producer

In the first example, we consider the optimization and control of the well settings in a 2-D reservoir model with two wells – an injector and a producer – which placements correspond with results from the second optimization run of HPSDE algorithm in sub-section 4.5.2. The 2-D reservoir model has $50 \times 50 \times 1$ grid-blocks of dimension $10\,\text{m} \times 10\,\text{m} \times 10\,\text{m}$; and this is depicted in Figure 5.4. The system contains oil and connate water with initial pressure and connate water saturation of $150 \times 10^5\,\text{Pa}$ and 0.2 respectively; both properties are assumed to be uniform throughout the reservoir model, and negligible capillary pressure effect is also assumed. The water injection well is constrained to operate between the bottom-hole pressure of $75 \times 10^5\,\text{Pa}$ and $100 \times 10^5\,\text{Pa}$; while the bottom hole pressure of the production wells are constrained to operate between $30 \times 10^5\,\text{Pa}$ and $45 \times 10^5\,\text{Pa}$. Since the reservoir system is a two-phase oil and water flow model, we employ the Corey model for relative permeability. The Corey exponents are $n_o = n_w = 2.45$; with relative permeability endpoints for oil and water 0.9 and 0.65 respectively. The relative permeability curve used is as depicted in Chapter 3 – see Figure 3.3; and the remaining system properties are given in Table 5.1.

With the aid of these information and the economic parameters given in Table 5.2; the reservoir is simulated for 500days at five time-steps of 100days, and the NPV for the well-rate optimization is computed using gradient formulation as described in section 5.1. A total NPV of \$2.371×10^6 was attained in this example; this resulting NPV as well as the optimal control inputs for the simulation duration are shown in Figures 5.5 and 5.6 respectively.

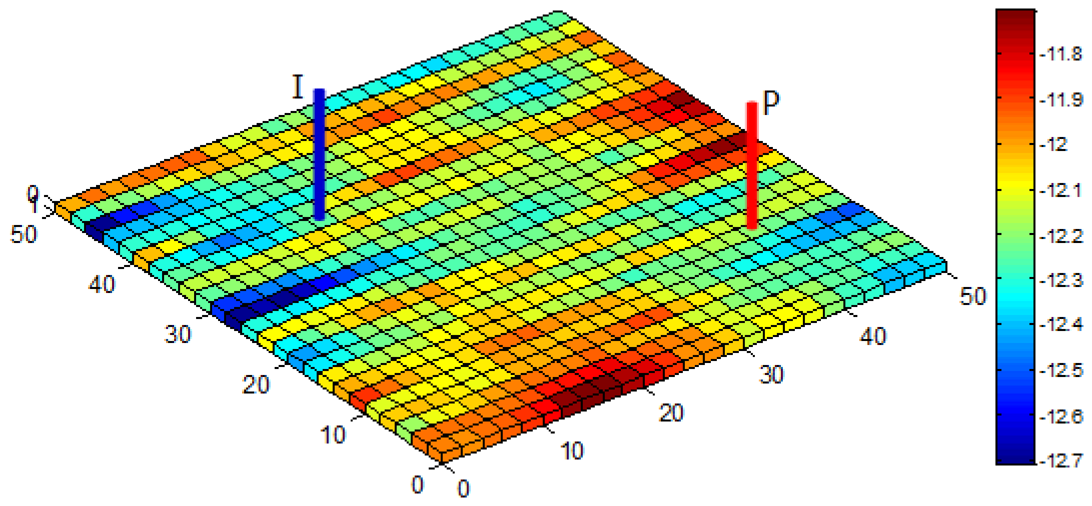


Figure 5.4: 2–Dimensional reservoir model 50×50×1 grid-blocks with two wells – an injector (blue) and a producer (red)

Table 5.1 Reservoir systems properties		
Properties	Symbol	Value
Porosity	ϕ	0.3
Oil viscosity	μ_o	10^{-3} Pas
Water viscosity	μ_w	10^{-3} Pas
Oil density	ρ_o	859 kgm^{-3}
Water density	ρ_w	1014 kgm^{-3}
Oil compressibility	c_o	10^{-10} Pa^{-1}
Water compressibility	c_w	10^{-10} Pa^{-1}
Rock compressibility	c_r	1.8×10^{-10} Pa^{-1}

Table 5.1: Reservoir system properties and their values

Table 5.2 NPV computation parameters		
Parameters	Symbol	Value
Price of oil	p^{oil}	\$300/m^3
Water production cost	$p^{w,p}$	\$50/m^3
Water injection cost	$p^{i,p}$	\$25/m^3
Discount factor	D	0.1

Table 5.2: NPV computation parameters for life-cycle optimization

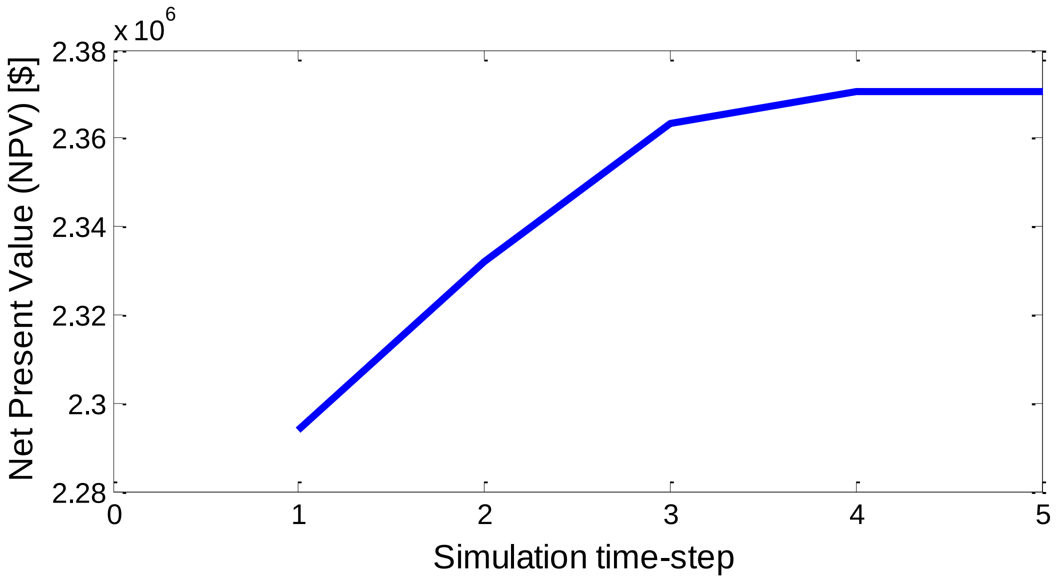


Figure 5.5: Optimal NPV for BHP-constrained rate optimization of example 1 reservoir model

Subsequently, the predictive control strategy is deployed to track actual production along the reference optimal NPV attained. This is necessary because as earlier explained; attaining the optimal NPV resulting from production optimization is not readily achievable by mere implementation of simulated optimal control inputs. To this end, a nonlinear model (**a1**) is identified using the system identification toolbox. The experimental design for identification data was carried out using **ECLIPSE**® simulator, as this provided the enablement for easy grid-block refinement which is required for the generation of persistently exciting signals. This signal type is of crucial importance in the identification of the data-driven model for the MPC strategy; it is more so if the salient nonlinearities of the process is to be captured.

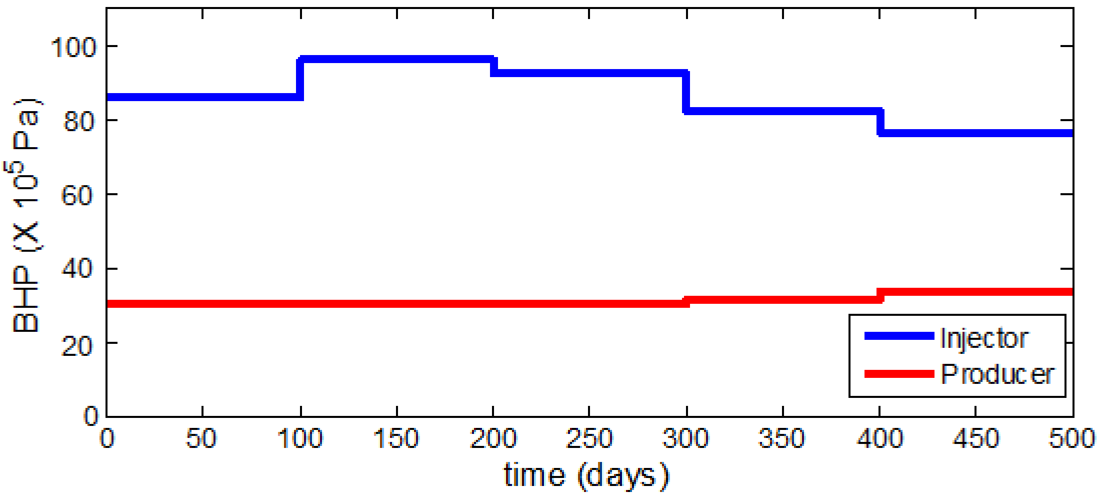


Figure 5.6: BHP control for duration of flooding of example 1 reservoir model

Following the system identification process, the identified nonlinear model is linearized into a state-space model (**a2**) by computing a first-order Taylor series approximation of the model. Using the same experimental data, a linear model (**a3**) is identified via subspace identification technique, the simulation fit of all three models (with respect to the measured output data) are plotted and shown in Figure 5.7 below. Interestingly, the simulation fit of the linearized nonlinear model with respect to measured data was better (albeit marginally) than that of the linear subspace model.

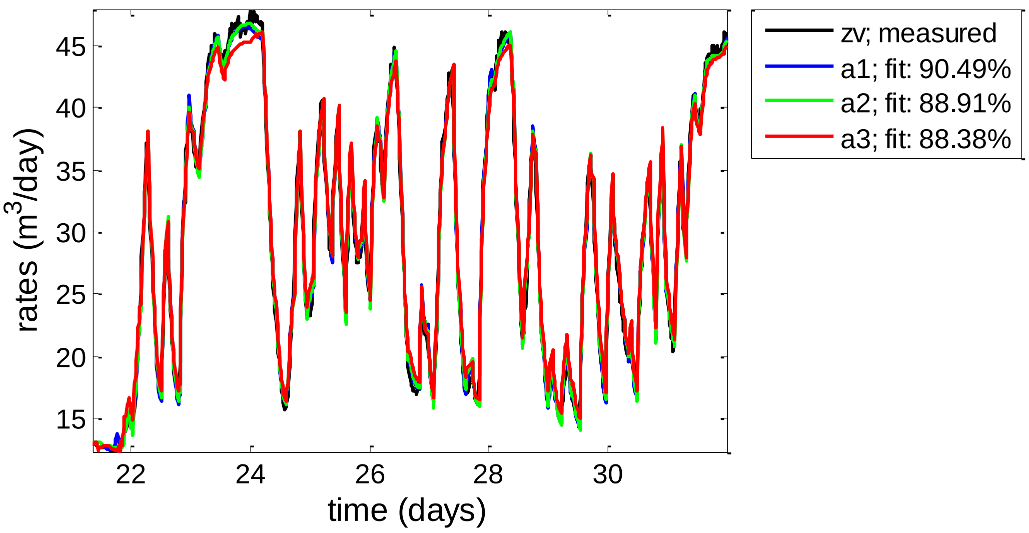


Figure 5.7: Simulation fit of identified and linearized models with respect to measured output

Now, designating the fluid rates at the producer as reference trajectory **r**, the predictive controller algorithm described in sub-section 5.2.3 is employed for the computation of corrected inputs such that the discrepancy between the actual measured outputs $\mathbf{y}_k$ and the reference trajectory **r** is minimized. In so doing, the designated reference trajectory is fed (as input) into a model predictive control algorithm as pictorially described in Figure 5.3. It is instructive to note that the computed corrected inputs are based on the linearized nonlinear model – this linearized model strikes a good balance between process nonlinearity accuracy and computational affordability. Finally, the measured fluid rates resulting from the application of the corrected inputs on the physical reservoir and the fluid profile resulting from the direct application of the control inputs that resulted from the production optimization are used to compute NPV for the different scenarios. The computed NPVs from both scenarios are compared to the original optimal NPV that resulted from the production optimization; and the result of the comparison is shown in Figure 5.8.

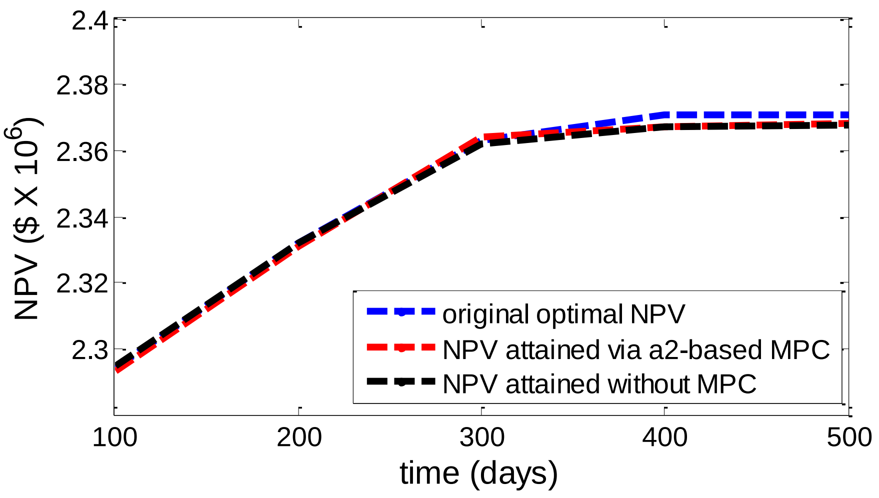


Figure 5.8: NPV accrued from the application of MPC via linearized nonlinear model and NPV accrued from direct application of optimal control resulting from production optimization

5.3.2 Example 2: Reservoir model with nine wells – three injectors and six producers

In this application, the physical reservoir model is a 3-D model with nine (3 injection and 6 production) wells – with placements corresponding to results from the fifth optimization run of HPSDE algorithm in sub-section 4.5.3. The model has $50 \times 50 \times 8$ number of grid-blocks of dimensions of $10\,\text{m} \times 10\,\text{m} \times 10\,\text{m};$ with nine vertically placed wells as depicted in Figure 5.9. The reservoir system contains oil and connate water with initial pressure and saturation of $350 \times 10^5\,\text{Pa}$ and 0.2 respectively; both properties are assumed to be uniform throughout

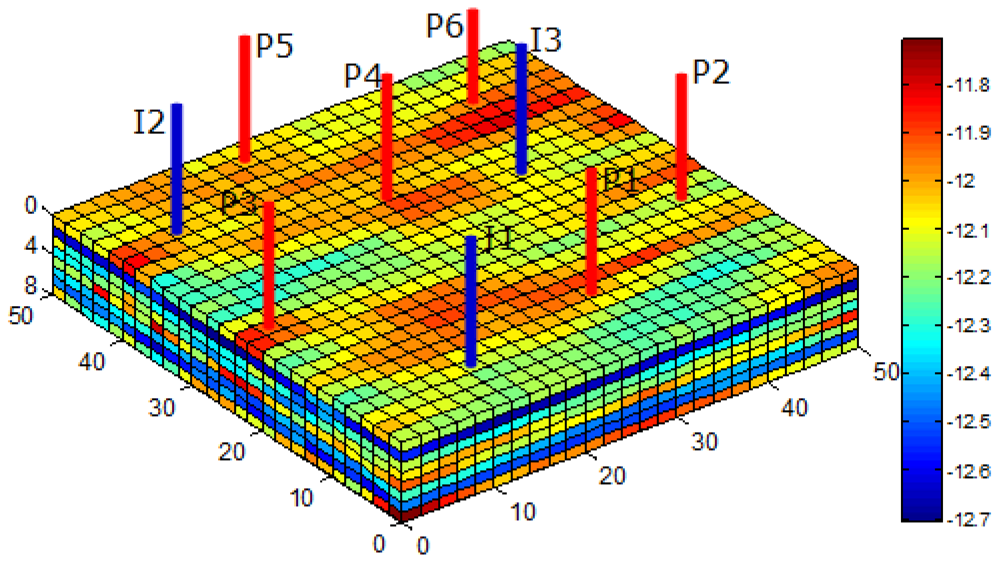


Figure 5.9: 3–Dimensional reservoir model 50×50×8 grid-blocks with nine wells – 3 injectors (blue) and 6 producers (red)

the reservoir model, and negligible capillary pressure effect is also assumed. The water injection wells are constrained to operate between the bottom-hole pressure of 100×10^5 Pa and 140×10^5 Pa; while the bottom hole pressure of the production wells are constrained to operate between 50×10^5 Pa and 85×10^5 Pa. Like in Example 1, relative permeability is based on Corey model, the exponents are given by $n_o = n_w = 2.45$; and relative permeability endpoints for oil and water are 0.9 and 0.65 respectively. The remaining fluid and geological properties are same as given in Table 5.1.

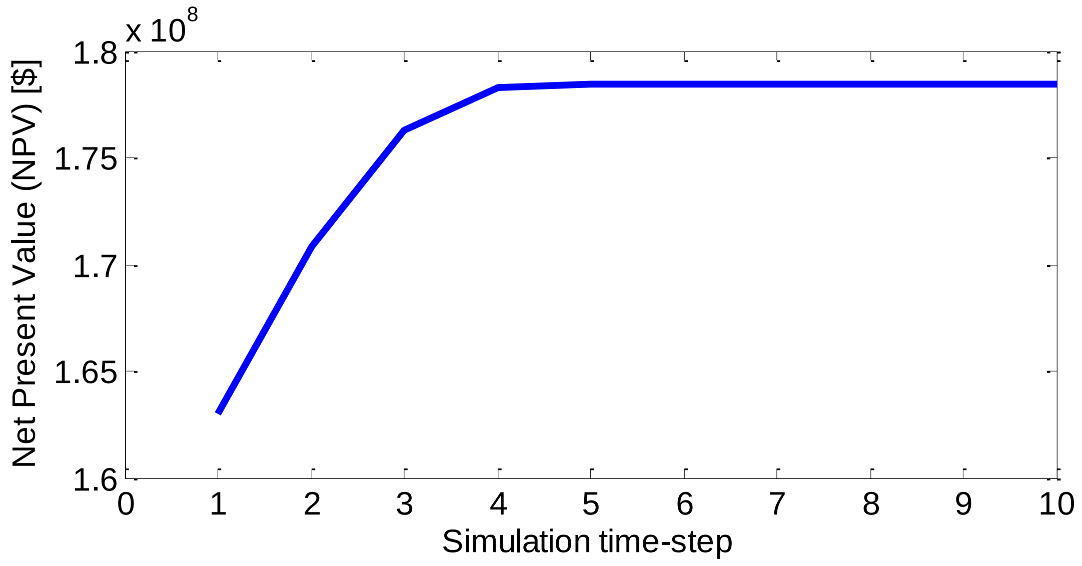


Figure 5.10: Optimal NPV for BHP-constrained rate optimization of the reservoir model

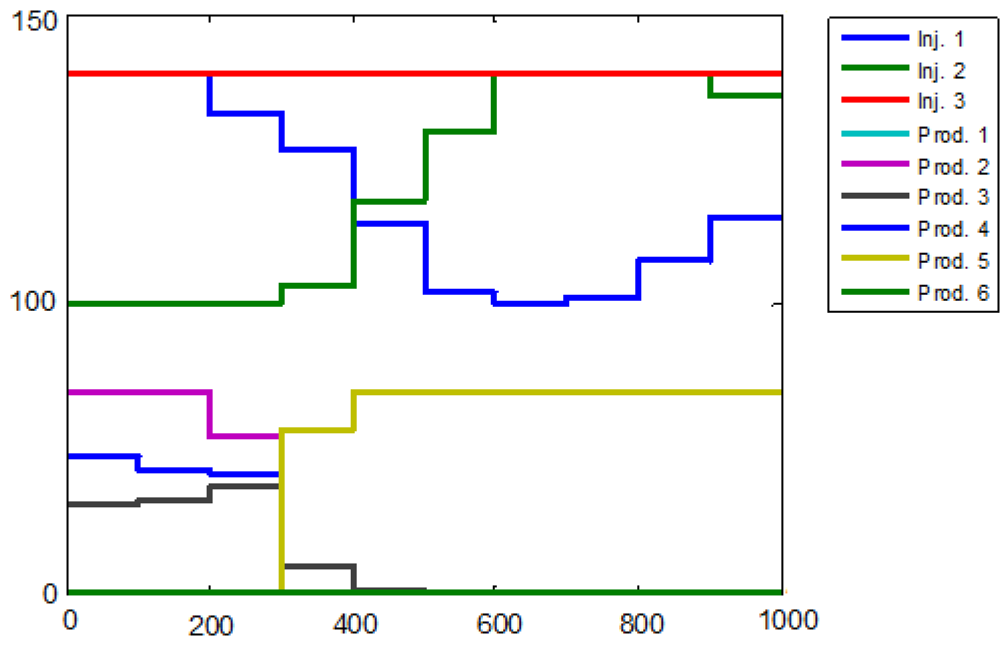


Figure 5.11: BHP control for duration of flooding of the reservoir

Now, using the above reservoir system properties and the economic parameters given in Table 5.2; the reservoir is simulated for 1000days at ten time-steps of 100days, and the NPV for the well-rate optimization is computed using the gradient method. The resulting NPV for this production optimization as well as the optimal control inputs are shown in Figures 5.10 and 5.11 respectively. From Figure 5.10, the NPV attained after 1000days of production is $\$1.7850\times10^8$; and the production wells fluid production profile that led to this NPV is shown in Figure 5.12.

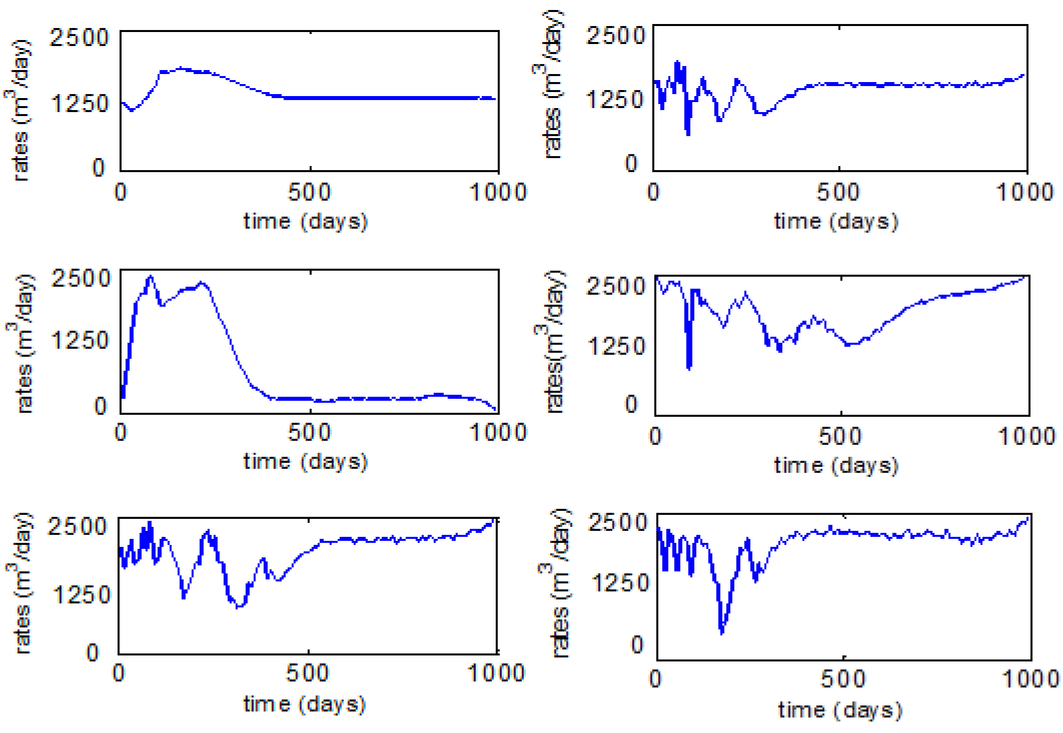


Figure 5.12: Optimum fluid production profile at the production wells

In order to track the fluid production profile of the physical reservoir along this optimal NPV, a nonlinear model (**b1**) was identified. Like in the preceding example, the production optimization simulation was performed using **MRST**® simulator; however, because of the non-availability of grid-refinement functionality on **MRST**® platform, the data-driven model identification experimental design was conducted using the **ECLIPSE**® simulator – as this provides the enablement for easy grid refinement, a requirement for generation of persistently exciting signals. The identification of the model is followed by a linearization process in which the nonlinear model is linearized into a state-space model (**b2**); and a linear model (**b3**) is identified from the same experimental data using subspace identification technique. Figure 5.13 shows the simulation fit of **b1**, **b2** and **b3** with respect to the measured output data.

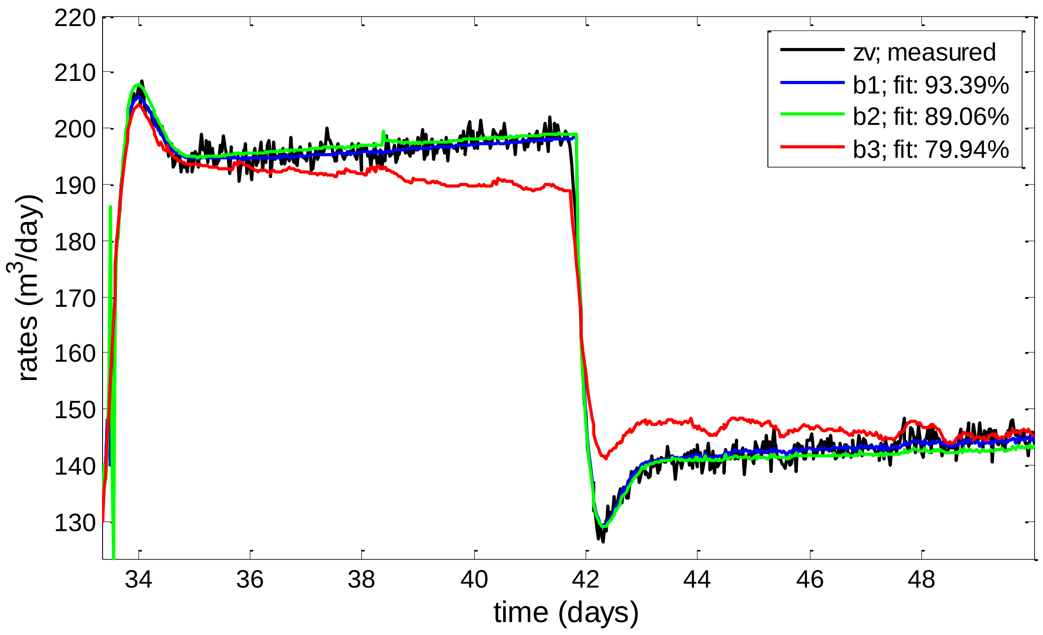


Figure 5.13: Simulation fit of identified and linearized models with respect to measured output

Now, the production optimization output that yielded the maximum NPV is subsequently designated the reference trajectory and fed into the predictive controller as described in Figure 5.3; for the purpose of computing corrected inputs which minimizes the difference between the actual measured outputs and the reference trajectory. Note that the data-driven model employed for the computation of corrected input in the MPC strategy is the linearized nonlinear model. The simulation fit shown in Figure 5.13 indicates that this model strikes a good balance between model accuracy and computational complexity.

The measured fluid rates resulting from the MPC-based control are recorded and plotted. For comparison purposes, the producers' fluid profile resulting from the direct application of the uncorrected control inputs that resulted from the production optimization are recorded and plotted as well. The production wells fluid profile for both scenarios are shown in Figures 5.14 and 5.15 respectively. These fluid profiles are used for the computation of the NPVs that correspond to the different scenario. While a total NPV of $\$177.91\times10^6$ was attained from the scenario that involved embedding an MPC controller, the NPV accrued from the scenario that was devoid of MPC strategy – i.e. that which involves the direct application of the uncorrected optimal control inputs resulting from the production optimization loop – was $\$176.67\times10^6$. The improvement in the value of the objective function and the nearness of this value to the original NPV resulting from the production optimization exercise reinforces the needfulness of this control strategy within production optimization framework.

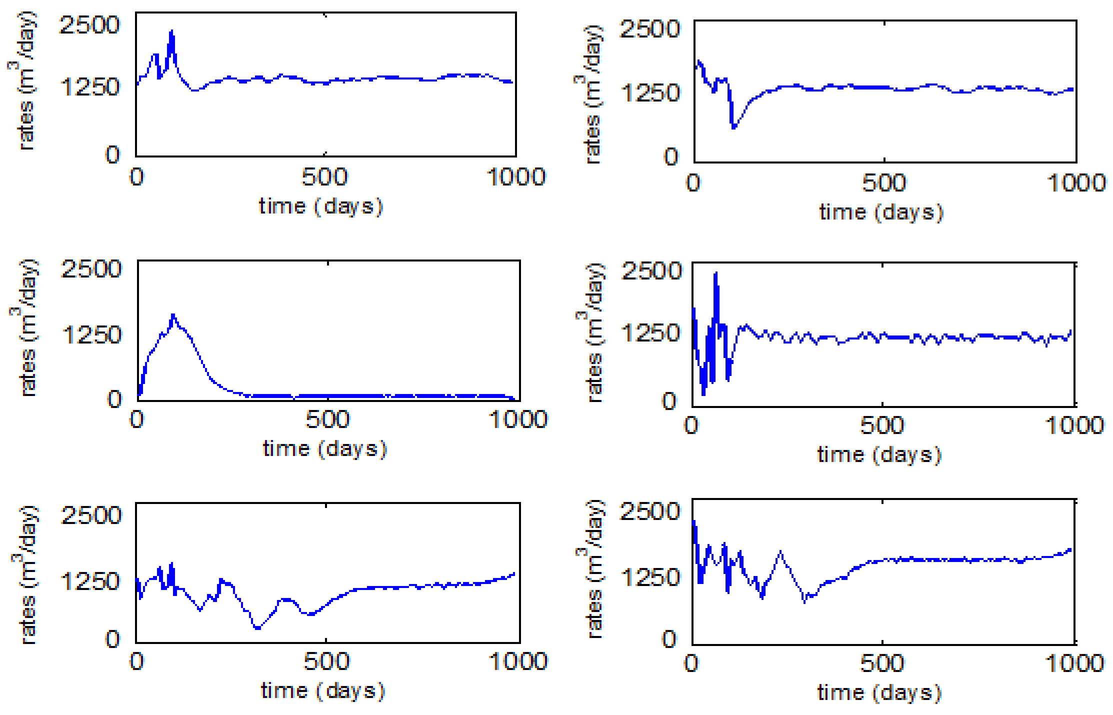


Figure 5.14: Measured fluid production profile at the production wells via linearized nonlinear model

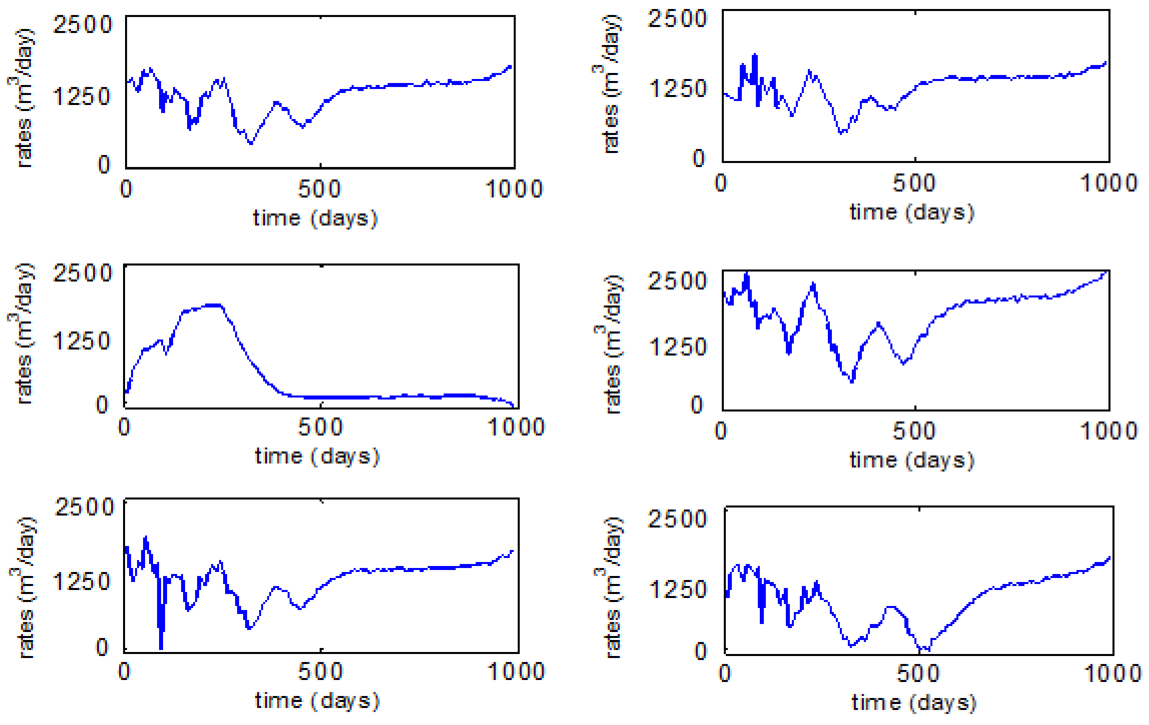


Figure 5.15: Measured fluid production profile at the production wells via the direct application of the optimal control input resulting from production optimization and devoid of MPC

Finally, the resulting NPV from the produced volumes arising from the different scenarios, are plotted and compared with the original optimal NPV arising from the production optimization; this is shown in Figure 5.16.

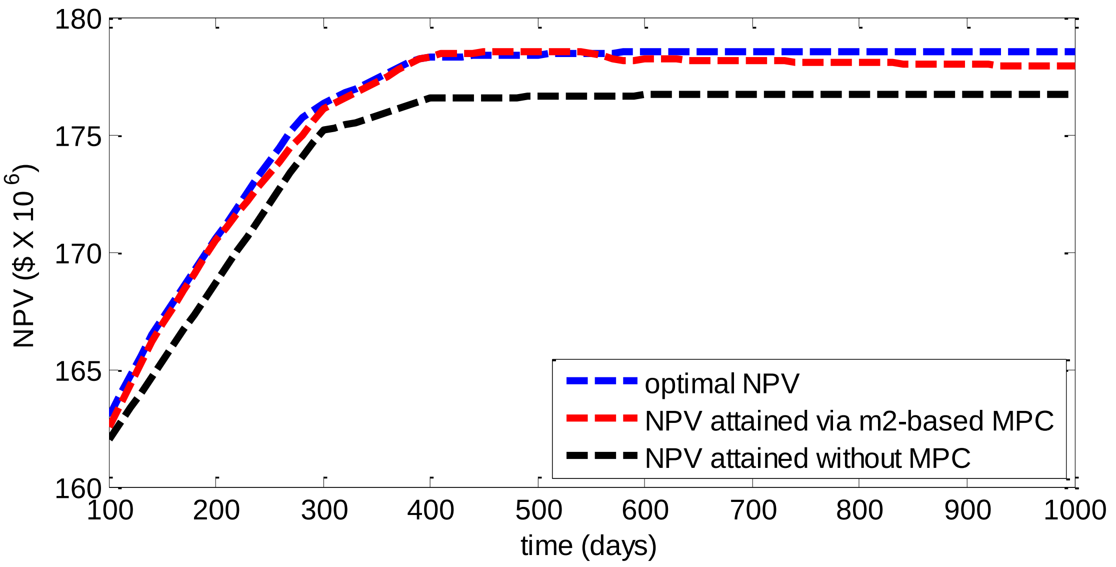


Figure 5.16: NPV accrued from the application of MPC based on linearized nonlinear model and NPV accrued from direct application of optimal control resulting from production optimization

5.4 Discussion

In the two examples considered in this chapter, the NPV attained by the implementation of the corrected controls resulting from MPC strategies based on linearized nonlinear models are $\$2.368\times10^6$ and $\$177.91\times10^6$ for Examples 1 and 2 respectively. Although these values are smaller than the optimal simulated NPV; they represent an improvement when compared to the NPV of $\$2.367\times10^6$ (for Example 1), and $\$176.67\times10^6$ (for Example 2) that was achieved by direct application of the control inputs emanating from the two production optimization examples. On the evidence of Figures 5.8 and 5.16, it can be inferred that an embedded linearized nonlinear model based MPC loop in production optimization problems can significantly enhance the potential to attain as near as possible, the optimal production trajectory resulting from the simulated production optimization loop.

Although the improvement recorded in the reservoir model with a single injection well and a single production well was not as remarkable as the improvement recorded in Example 2; the fact remains that there is a scope for the improvement of cumulative oil production and therefore, the NPV of oil field assets, by the application of corrected control inputs resulting from the implementation of MPC. With respect to the NPV accrued by direct application of the control input from the production optimization loops, there was an improvement of 0.04% in the NPV in first application, and the improvement in the NPV recorded in the second application is in the tune of 0.7%. To put these figures or percentages into perspective, it is worthy to note the fact that the E&P industry is an industry where an improvement in recovery or economic outcome by as little as a fraction of 1% translates into profits in the order of tens or hundreds of millions of dollars. Against this backdrop, an improvement in NPV by as little as 0.04% as recorded in Example 1 is by no means trivial.

5.5 Summary

The focus on this chapter was production optimization and control for waterflooded reservoir models with a defined well configuration. It was pointed out that a number of techniques have been employed in attempts at solving production optimization problem in the petroleum industry, and of these techniques, the most reliable is the gradient-based approach – where the gradients of the objective function with respect to control variables are computed using adjoint formulations. The reliability of this approach stems from the fact that the computation of the derivatives requires only two simulation runs, regardless of the number of decision variables. In implementing this computational-friendly approach, we espoused the approach presented in Sarma (2005) and Jansen (2012).

However, because the optimal trajectory resulting from production optimization exercises is hardly attainable, there is need to employ a path tracking control strategy that computes corrected control inputs that minimize the difference between actual production profiles and the optimal simulated results. To this end, an MPC algorithm was used to compute corrected inputs that minimize the discrepancy between the optimal trajectory (resulting from production optimization) and actual production profile. Usually, such predictive control strategy is implemented with the aid of simple data-driven linear models which leads to easy-to-implement QP problems. However, since the dynamics of the waterflooding process is inherently nonlinear, it is essential that such nonlinearities are reflected in the underlying models for the predictive controller strategy. Despite their higher accuracy, the use of such data-driven nonlinear models lead to more complex NMPC and challenging NLP problems. In other words, inasmuch as the importance of employing nonlinear models (from the view point of accuracy) is underscored; one must not lose sight of the fact that it is much attractive (from the computational cost view-point) to employ simple linear models that possess easy-to-implement controller-design properties. Striking the appropriate balance between accuracy and complexity necessitated the use of linearized nonlinear models. The choice of this model type stems from their ability to capture process nonlinearity and importantly, the fact that they can be deployed in linear MPC if they are adequately linearized in the local neighborhood of an operating point defined by the state of the physical system.

CHAPTER 6

No matter how strong or great man is, he should never challenge his chi – Chinua Achebe

CONCLUSION AND FUTURE WORK DIRECTION

As outlined at the outset of this thesis, the main objective of this research is to develop and deploy efficient optimization, control and estimation techniques that would lead to the maximization of hydrocarbon reservoir recovery factor within the ambit of model-based closed-loop reservoir management framework. In this chapter, a summary of the findings of this work is presented as conclusion, the limitations and scope for improvements are spelt-out and finally, we end by outlining future work directions.

6.1 Conclusions

In this thesis, it was established that one of the fall-outs of the unprecedented rise in world population and robust economic growth in China, India and other emerging economies is the monumental and continuing rise in global energy demand and per capita energy utilization. Considering that oil is the most sought-after energy resource and the fact that it is a non-renewable resource; there is need to develop efficient techniques that would maximize the recovery factor of existing and new oil reservoirs. In addition, the need to develop efficient techniques and strategies for improved recovery of oil from reservoirs is further reinforced if we consider the fact that all credible forecasts on projected global energy demand point to the inevitability of further increase in the demand of this energy resource in the coming decades.

Although it was noted that there are quite a number of ways that would ultimately lead to increased cumulative production of oil, the focus in this research is on defined optimization, control and estimation techniques within a closed-loop reservoir management framework. The two foci of this work are field development optimization (well placement problem) and production settings optimization and control.

6.2 Well Placement Optimization

The importance of well placement optimization was highlighted; it was argued that it is by far the most important field development decision input – as it can ultimately determine the reservoir's production profile, and therefore, the recoverability of the reservoir. Overall, three different problems involving the placements of vertical wells in reservoir models of varying complexities were considered. The differential evolution (DE) and particle swarm optimization (PSO) algorithms were applied in all of these problems, and the results emanating from both algorithms were compared with the results obtained via the application of a third metaheuristic algorithm – hybrid particle swarm differential evolution (HPSDE) – which is a product of the hybridization of DE and PSO algorithms. Using a discounted net present value (NPV) as objective function, the hybrid algorithm consistently outperformed both the DE and PSO algorithms in all three problems considered in this problem domain. In addition, the issue of geological uncertainty arising from the discrepancy between the real physical reservoir and the reservoir model was considered in two of the three problems. In this regards, a *max-mean objective* robust optimization of the objective function was performed; and in both cases, the HPSDE algorithm yielded better results than the duo of DE and PSO. In one of the problems considered, the performance measures of the metaheuristic algorithms were compared with the NPV attained via a number of specific well pattern arrangements. In this regard, the inverted five-spot, inverted seven-spot and the inverted nine-spot patterns were considered; and the interestingly, all three stochastic algorithms yielded higher NPV than the specific well pattern arrangements. The HPSDE algorithm was further compared with more established optimization techniques such as linear programming (LP) and genetic algorithm (GA); but none of both algorithms achieved results that are comparable to HPSDE. However, the GA attained results that were comparable with PSO and to an extent DE; the performance of LP fell way behind the performance of all the stochastic algorithms considered in this work. Also, because the 'parent' algorithms that gave birth to HPSDE are both global algorithms; we further compared the performance of HPSDE against the results from another hybrid algorithm created by the hybridization of a global algorithm (PSO) and a local search (TS) algorithm. Although the new hybrid algorithm outperformed DE, PSO and GA algorithms (in that order); its performance fell behind HPSDE by 2.6%. It was also demonstrated that the relative performance of any one of DE and PSO with respect to the other is fundamentally dependent on the total number of simulations as there was a marked variation in their performance in early, mid and later stages of simulation. In this regard, DE often attained higher performance value than PSO at very low and very high number of total simulation. To further investigate the performance of these stochastic algorithms, their computational complexities were analyzed in terms of runtime and space complexities. The algorithms were also tested on six benchmark problems which reflected opposite sides of complexity factors such as modality, separability and scalability. Using statistical indices as quality indicators, HPSDE algorithm outperformed both DE and PSO algorithms in all but one of the benchmark test functions. Also, the experimental results emanating from HPSDE algorithm yielded the lowest standard deviation in all six benchmark tests; this implies that the probability of attaining better results is higher with HPSDE algorithm than the other two algorithms.

6.2.1 Limitations

Although it is noted that the findings regarding the deployment of HPSDE algorithm in field development optimization are interesting and potentially useful; it is acknowledged that there are limitations that still have to be addressed. Chief among these limitations is the issue of control parameters tuning. In the three problems considered, all three algorithms were deployed without any form of parameter tuning. It is noted that the DE parameters (F = 0.5, CR = 0.1) used in this work are adopted from Storn and Price (1997) and the PSO parameters $(c_1 = c_2 = 1.193, \omega = 0.721)$ are adopted from Onwunalu and Durlofsky (2010). For instance, in the second and third examples, there are instances where DE outperformed PSO, and vice versa. It is important to understand how these behaviors are influenced by relevant control parameters of the underlying metaheuristic algorithm.

Another limitation is the issue of usability in practical field development optimization scenarios. This is so because the viability of hybridized metaheuristic optimization algorithm such as HPSDE as a serious alternative in practical field development scenarios and indeed in other reservoir engineering problem domains; depends to some extent on their relative ease-of-implementation in practice. Thus, the issues of usability have to be addressed before these algorithms can be deployed for practical use in the industry. In a sense, the usability limitation is intertwined with parameter tuning. It is noted that the usability of HPSDE algorithms would be greatly enhanced if the issue of parameter tuning is sorted out at the design-end (against the user-end) of the algorithm. This is so because it is generally unrealistic for industrial end-users to waste expensive function evaluations in correcting the weakness of the design phase of an algorithm.

Also, while it is acknowledged that there are many sources of uncertainty in petroleum reservoir models; this work assumed that the only source of uncertainty is the permeability distribution of the reservoir model. It is important to investigate the effects of other sources of uncertainty. Finally, it is important to point out that the reservoir models in all of the three applications considered in this problem domain were synthetic or laboratory model. The use of real reservoir models would surely bring the deployment of HPSDE algorithm in real field development optimization scenarios closer to fruition.

6.2.2 Recommendations and Scope for Improvement

Based on the limitations highlighted above, the following recommendations are suggested as they would invariably improve the scope of the deployment of the algorithm in practical reservoir engineering field development optimization problems

- effective control parameter tuning to further improve algorithmic performance
- incorporation of prior-knowledge such as problem structure and other relevant information about the underlying optimization problem into the algorithm
- since this algorithm is population based, it is desirable to incorporate techniques or mathematical concepts that would increase the diversity of the population
- deployment of the algorithm in real reservoir models at the development stage of its life-cycle
- the use of the algorithm in conjunction with 4-D seismic data for optimal placement of infill wells
- besides the placement optimization for vertical wells, well trajectory optimization for deviated wells may also be tackled using HPSDE algorithm
- it is important to investigate the effects of other sources of uncertainty in the reservoir models, a situation where the permeability distribution of the reservoir is assumed to be the only source of uncertainty is certainly fraught with inaccuracies
- it is equally important to re-define the system and the system boundary so that the system is not restricted to our assumption of two-phase flow; and the system boundary may be extended to include the surface facilities as well as the interaction

between the fluid flowing from the reservoir into the wellbore and indeed into the surface facilities

6.3 Production Optimization and Control

In production optimization and control, the focus was on hydrocarbon production optimization and control for waterflooded reservoir models with defined well configuration. We pointed out that a number of techniques have been employed in attempts at solving this optimization problem, and that of these techniques; the most reliable is the gradient-based approach where the gradients of the objective function with respect to control are computed using adjoint formulations. The reliability of this approach stems from the fact that the computation of the derivatives requires only two simulation runs, irrespective of the number of decision variables. In implementing this computationally-friendly approach, we espoused the approach advocated in Sarma (2005) and Jansen (2012). Despite the aforementioned computational efficiency of this approach, the optimal trajectory arising from the production optimization is hardly attainable. To this end, a predictive controller algorithm was used to compute corrected inputs such that the discrepancy between the optimal trajectory (resulting from production optimization) and actual production profile is minimized. Usually, such predictive control strategy is implemented with the aid of simple data-driven linear models. However, the dynamics of the waterflooding process is nonlinear, it is essential that such nonlinearities are reflected in the underlying models for any predictive controller strategy. The use of linear models in MPC strategies leads easy-to-implement QP problems, while nonlinear data-driven models lead to NMPC and its associated complex NLP problems. In as much as nonlinear models are good for accuracy reasons, there is need to ‘circumvent’ the hassles associated with NLPs. Striking the appropriate balance between accuracy and complexity necessitated the use of linearized nonlinear models. In the examples considered, improved NPVs were achieved by the implementation of the corrected control inputs resulting from this control strategy as against the results obtained by direct application of the controls resulting from production optimization exercise. Therefore, an MPC strategy which is based on linearized nonlinear models enhances the opportunity to attain as near as possible, the optimal production trajectory resulting from simulated production optimization. Although the results were promising and potentially useful, there are limitations and issues that still need to be addressed; these are issues and limitations are discussed in the next sub-section.

6.3.1 Limitations

The use of model predictive control strategy to attain optimal or near optimal production trajectory can potentially play a significant role in waterflooding production operations and indeed in other production operation strategies, as have been the case in the downstream refining and marketing sector. However, there are limitations that still need to be addressed. These limitations include simplifications such as the assumption that process noise and measurement noise are absent. It is important to note that this is an ideal condition that is hardly the case in practical field production situations. With the introduction of measurement and process noise, it will be interesting to study the outcomes resulting from the identification of different sets of model structures, as well as other linearization techniques to see if they will be provide better approximations of the underlying process model. Also, in espousing Andersson et al. (1998) in the experimental design, the use of rules of thumb as suggested therein is a limitation of its own. Indeed, there is need to avoid such trial-and-error approach during design of experiment for the identification of 'acceptable' models, as this would inevitable enhance the quality of the models.

Another limitation in this work is the use of total fluid rate at the producers as output. In our view, it will be interesting to investigate the use of fractional flow rates of oil by introducing fractional flow meters or separators in the flow system. Again, it is important to point out that the reservoir models in the applications considered in this chapter were synthetic models. The implementation of this optimization and control approach on real reservoir models would be of interest in terms of making more generalized and far-reaching conclusions.

6.3.2 Recommendations and Scope for Improvement

Based on the limitations highlight above, the following recommendations are suggested:

- introducing process noise and measurement noise into the system as these are inevitable realities in real oil field waterflooding production scenario
- the deployment of predictive control strategy in real reservoir models
- it is equally important to re-define the system and the system boundary so that the system is not restricted to our assumption of two-phase flow, and the system boundary

is extended to include the surface facilities and the prevailing interaction between the fluid flowing from the reservoir into the wellbore and indeed into the surface facilities

- the use of other model structures and other linearization approaches
- other ways of achieving non-prohibitive computational complexity without sacrificing accuracy and easy-to-implement properties

6.4 Future Work Directions

In the future, we intend to improve this work by implementing dual-purpose optimization strategies in the well placement optimization problem. Dual-purpose optimization strategies consist of *superficial* and *ultimate* optimization procedures. The purpose of the '*superficial*' optimization is to determine optimum values for the HPSDE algorithm control parameters, after which the '*ultimate*' optimization is activated for solving the well configuration problem. In other words, the optimum control parameter values resulting from the embedded '*superficial*' optimization are subsequently used in the '*ultimate*' optimization loop to solve the given well configuration problem. This algorithm will also be extended to well trajectory optimization. In this regard, we will explore its applicability and effectiveness in multilateral and deviated wells problems. Furthermore, we intend to explore the application of HPSDE algorithm in a combined well placement and well rate optimization problem. Although the potential computational cost of the combined optimization problem could be prohibitive; it is however, expected that such high computational cost would be met by the deployment of surrogate models and the use of multi-processor parallel computing resources.

We also intend to carry out further study in data-driven identification of other nonlinear model structures that would not sacrifice process nonlinearities. We will also investigate the potential benefit of other linearization techniques, with the view that resulting linearized models are compact, accurate and robust. This will no doubt improve the use of data-driven MPC strategies in reservoir production operations. With the recent release of the *Brugge Field* data (as made available to us by The Netherlands Organization for Applied Scientific Research) we intend to apply these techniques to a real reservoir model; importantly, since the production history data is available, we will investigate the performance of HPSDE at finding reservoir model parameter values that minimizes an objective function that represents the mismatch between simulated and measured production.

Overall, this thesis promotes a non-discriminant use of system-theoretic concepts as well as metaheuristic and deterministic based algorithms within the ambit of a closed-loop reservoir management framework. All these will in no small way create the enablement for the transition of this approach from laboratory to field applications.

REFERENCES

Aanonsen, S. I., Nædval, G., Oliver, D. S., Reynolds, A. C., and Valles, B., 2009, The Ensemble Kalman Filter in Reservoir Engineering - a Review: SPE Journal, 14, no. 3, 393–412 (SPE 117274–PA)

Aarnes, J.E., Gimse, T. and Lie, K.-A. (2007). An introduction to the numerics of flow in porous media using Matlab. *Geometric Modelling, Numerical Simulation, and Optimization Applied Mathematics at SINTEF*, Eds. G. Hasle, K.-A. Lie, and E. Quak, Springer Berlin Heidelberg, pp. 265-306

Ahmed, T.H. (2001): Reservoir Engineering Handbook, Second Edition, Gulf Professional Publishing, ISBN 0-88415-770-9

Aitokhuehi, I., L. J. Durlofsky, V. Artus, B. Yeten and K. Aziz (2004). Optimization of advanced well type & performance. In 9th Euro Conf. on the Math. of Oil Recovery, Cannes

Alessandri A. and Coletta P. (2001). Design of Luenberger Observers for a Class of Hybrid Linear Systems HSCC 2001, Lecture Notes on Computer Science, Springer-Verlag Berlin Heidelberg http://www.control.auc.dk/~raf/News/ObserverHybridsystems4.pdf (last accessed on 5 June 2012)

Allgower, F., Findeisen, R. and Nagy, Z. K. Nonlinear model predictive control: From theory to application, J. Chin. Inst. Chem. Engrs., Vol. 35, no. 3, pp. 299315, 2004.

Allgower, F., Findeisen, R., Nagy, Z., Diehl, M., Bock, H., and Schloder, J. (2000). Efficient nonlinear model predictive control for large scale constrained processes. Proceedings of the Sixth International Conference on Methods and Models in Automation and Robotics, Miedzyzdroje, Poland, 2000, pp. 43-54.

Andersson, L., Jönsson, U., Johansson, K. H., & Bengtsson, J. (1998). A manual for system identification. Laboratory Exercises in System Identification. KF Sigma i Lund AB. Department of Automatic Control, Lund Institute of Technology, Box, 118.

Angeline, P.J.: Using selection to improve particle swarm optimization. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC), pp. 84–89. Anchorage, AL, USA (1998)

Asheim, H. (1987). Optimal control of water drive. SPE Journal (SPE 15978)

Asheim, H. (1988). Maximization of water sweep efficiency by controlling production and injection rates. In SPE European Petroleum Conference (SPE 18365), London

Åström, K. and Bohlin, T. (1965). Numerical identification of linear dynamic systems from normal operating records, Proc. IFAC Sym. on Self-Adaptive Systems, Teddington, UK

Aziz, K. and Settari, A. (1979), Petroleum Reservoir Simulation, App. Sci. Pub., London.

Bangerth, B. L., W. H. Klie, M. F. Wheeler, P. L. Stoffa and M. K. Sen (2006). On optimization algorithms for the reservoir oil well placement problem. Computational Geosciences 10, 303–319

Banks, A., Vincent, J., Anyakoha, C.: A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat. Comput.: Int. J. **7**(1), 109–124 (2008)

Beckner, B. L. and X. Song (1995). Field development planning using simulated annealing - optimal economic well scheduling and placement. In SPE Annual Technical Conference and Exhibition (SPE 30650), Dallas

Bennett, A. F., (2002), Inverse Modelling of the Ocean and Atmosphere: Camb. Univ. Press

Bennett, A. F., Chua, B. S., and Leslie, L. M., (1996), Generalized inversion of a global numerical weather prediction model: Meteorology and Atmospheric Physics, 60, 165–178

Bissell, R. C. (1994). Calculating optimal parameters for history matching. In 4th European Conference on the Mathematics of Oil Recovery, Roros

Bittencourt, A. C. and R. N. Horne (1997). Reservoir development and design optimization. In SPE Annual Technical Conference and Exhibtion (SPE 38895), San Antonio

Bloemen, H. (2002): Predictive Control Based on Black Box State-Space Model. PhD Thesis Delft University of Technology (2002)

Bouzarkouna, Z., Ding, D.Y., and Auger, A.: Well placement optimization with the covariance matrix adaptation evolution strategy and meta-models Comput. Geosci. (2011)

Brain, Z., Addicoat, M. (2010): Using Meta-Genetic Algos. to tune parameters of GA to find lowest energy Molecular Conformers. Proc. of the Alife XII Conf., Odense, Denmark, 2010

Brillinger, D. R. (1981). Time Series: Data Analysis and Theory. McGraw-Hill, New York

Brouwer, D. R. (2004). Dynamic Water Flood Optimization With Smart Wells Using Optimal Control Theory. Ph.D. thesis, Delft University of Technology

Brouwer, D. R. and J. D. Jansen (2004). Dynamic optimization of waterflooding with smart wells using optimal control theory. SPE Journal (SPE 78278) 9(4), 391–402.

Bryson, A. E. and Y. C. Ho (1975). Applied optimal control: optimization, estimation and control. Hemisphere-Wiley.

Bryson, A.E., (1999). Dynamic Optimization. Addison-Wesley, Menlo Park, California.

Caers, J. (2003). History matching under training-image-based geological model constraints. SPE Journal 8(3), 218–226.

Camacho, E.F., and Bordons, C. (2007a): Model Predictive Control (Advanced Textbooks in Control and Signal Proc.), Springer; 2nd printing 2007 edition, ISBN-13: 978-1852336943

Camacho, E.F., and Bordons, C. (2007b): Nonlinear Model Predictive Control: An Introductory Review. Assessment and Future Directions of Nonlinear Model Predictive

Control, Lecture Notes in Control and Information Sciences, 2007, Volume 358/2007, 1-16, DOI: 10.1007/978-3-540-72699-9_1

Cartes, C., and de Marsily, G., 1991, Application of the Pilot Point Method to the Identification of Aquifer Transmissivities: Advances in Water Resources, 14, no. 5, 284–300.

Centilmen, A., T. Ertekin and A. S. Grader (1999). Applications of neural networks in multiwell field development. In SPE Annual Technical Conference and Exhibition (SPE 56433), Houston.

Chavent, G. M., DuPuy, M., and Lemonnier, P., (1975), History Matching by Use of Optimal Theory: SPE Journal, 15, no. 1, 74–86 (SPE 4627–PA).

Chen, W. H., G. R. Gavalas, J. H. Seinfeld and M. L. Wasserman (1974). A new algorithm for automatic history matching. SPE Journal 14(4), 593–608.

Chen, Z. (2007). Reservoir Simulation: Mathematical Techniques in Oil Recovery, in the CBMS-NSF Regional Conference Series in Applied Mathematics, Vol. 77, SIAM, Philadelphia, PA, 2007.

Chen, Z., Huan, G., and Ma, Y. (2006) Computational Methods for Multiphase Flows in Porous Media, in the Computational Science and Engineering Series, Vol. 2, SIAM, Philadephia, PA, 2006, ISBN-0-89871-606-3.

Cheng, H., X. Wen, W. J. Milliken and A. Datta-Gupta (2004). Field experiences with assisted and automated history matching. In SPE paper 89857 presented at the SPE Annual Technical Conference and Exhibition, Houston, USA.

Chierici, G. L. (1992). Economically improving oil recovery by advanced reservoir management. Journal of Petroleum Science and Engineering 8, 205–219.

Ciaurri, D.E., Mukerji, T., and Durlofsky, L.J.: Derivative-Free Optimization for Oil Field Operations Studies in Computational Intelligence, Volume 359, Computational Optimization and Applications in Engineering and Industry, pp. 19-55 (2011)

Clerc, M.: Particle Swarm Optimization. iSTE, London (2006)

Currie, J. (2011): jMPC Toolbox v3.11 User's Guide AUT University, New Zealand

Currie P.K. and Jansen, J. D. (2004). Modelling and Optimisation of Oil and Gas Production Lecture notes for course TA4490 ‘Production Optimisation’ Systems TUDelft

D.G. Luenberger (1971). An Introduction to observers, IEEE Transaction on Automatic Control, vol. AC-16, No. 6 December 1971 http://www.stanford.edu/dept/MSandE/cgi-bin/people/faculty/luenberger/pdfs/aito.pdf

Das, S., Abraham, A., Konar, A.: Particle Swarm Optimization and Differential Evolution Algorithms: Tech. Analysis, App. & Hybridizat. Persp. www.softcomputing.net/aciis.pdf. Accessed: 18 Mar 2011

Das, S., Suganthan, P.N. (2011): Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions On Evolutionary Computation, Vol. 15, NO. 1, FEBRUARY 2011

Davendra, D., Zenlinka, I., Onwubolu, G.: Hybrid Differential Evolution—Scatter Search Algorithm for Permutative Optimization Evol. Computation, InTech, Vienna, Austria (2009)

de Marsily, G., G. Lavedan, M. Boucher and G. Fasanino (1984). Interpretation of interference tests in a well field using geostatistical techniques to fit the permeability distribution in a reservoir model. Geostatistics for Natural Resources Characteriz. 831 849.

Deep, K., Bansal, J.C.: Hybridization of particle swarm optimization with quadratic approximation. J. Oper.Res. **46**, 3–24 (2009)

Deep, K., Das, K.N.: Quadratic approximation based hybrid genetic algorithm for function optimization. Appl.Math. Comput. **203**(1), 86–98 (2008)

Diglakis, J.G. and Margaritis, K.G. (2000): On benchmarking functions for genetic algorithms, Intern. J. Computer Math., Vol. 00, pp. 1-27

Domingo, O.B., Cesar, H.M., Nicolas, G.P., (2005): A Crossover Operator for Evol. Algos. Based on Population Features, Dept. of Comp. & Num. Ana., Univ. of Córdoba, Spain.

Dong, X., Wu, Z., Dong, C., Chen X and Wang, H.: Optimization of vertical well placement by using a hybrid particle swarm optimization Wuhan University Journal of Natural Sciences, 16 (3), 237-240 (2011)

Douma, S. G., (2009), Equivalence between Gauss-Newton method and representer method (personal communication).

Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43. Nagoya, Japan (1995)

Emerick, A.A and Reynolds, A.C. (2012). History matching time-lapse seismic data using the ensemble Kalman filter with multiple data assimilations Comput Geosci (2012) 16:639–659 DOI 10.1007/s10596-012-9275-5

Engelbrecht, A.P.: Fundamentals of Computational Swarm Intel. Wiley, West Sussex (2005)

Evensen, G., 1994, Sequential data assimilation with nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics: Journal of Geophysical Research, 99, No. C6, 10143–10162.

Evensen, G., 2007, Data Assimilation. The Ensemble Kalman Filter: Springer.

Eze C. A. O (2002). Energy Conversion and Power Generation. Lecture Notes on MEC 546, Nnamdi Azikiwe University, Awka, Nigeria.

Farshi, M.M.: Improving genetic algorithms for optimum well placement. Master's Thesis, Stanford University (2008)

Feehery, W.F., J.E. Tolsma and P.I. Barton, 1997. Efficient sensitivity analysis of large-scale differential algebraic systems. Applied Numerical Mathematics, 25(1): 41- 54.

Findeisen, R., Raff, T., and Allgower, F.: Sampled-Data Nonlinear Model Predictive Control for Constrained Continuous Time Systems. Advanced Strategies in Control Systems with Input and Output Constraints, Lecture Notes in Control and Information Sciences, 2007, Volume 346/2007, 207-235, DOI: 10.1007/978-3-540-37010-9_7

Gallo, Y. L. and Ravalec-Dupin, M. L. (2000). History Matching Geostatistical Reservoir Models with Gradual Deformation Method, SPE Annual Technical Conference and Exhibition, 1-4 October 2000, Dallas, Texas, ISBN 978-1-55563-910-5

Gavalas, G. R., Shah, P. C., and Seinfeld, J. H., (1976), Reservoir History Matching by Bayesian Estimation: SPE Journal, 16, no. 6, 337–350 (SPE 5740–PA).

Gerksic, S., Juricic, D., Strmcnik, S, Matko, D.: Wiener model based nonlinear predictive control. International Journal of Systems Science, 31(2) 189–202 (2000)

Gheitanchi, S., Ali, F., Stipidis, E.: Trained Particle Swarm Optimization for Ad-Hoc Collaborative Computing Networks. Swarm Intell. Algo. & App. Symp., ASIB 2008, UK.

Gildin, E., and Wheeler, M.F. (2008). Control of Subsurface Flow using Model Predictive Control Techniques. Proceedings of International Conference on Engineering Optimization (EngOpt) Rio de Janerio, Brazil, 1-5 June, 2008

Glover, F.W. (1986). "Future Paths for Integer Programming and Links to Artificial Intelligence". Computers and Operations Research 13 (5): 533–549.

Glover, F.W. (1989). "Tabu Search - Part 1". ORSA Journal on Computing 1 (2): 190–206.

Golder Associates, Circle Ridge Fractured Reservoir Project, technical report from www.fracturedreservoirs.com, Redmond, WA, 2000.

Goldberg, D. E., Deb, K., and Korb, B. (1991). Do Not Worry, Be Messy. In Proceedings of the Fourth International Conference on Genetic Algorithms, pp 15–30.

Grimstad, A. A. and Mannseth, T. (2000). Nonlinearity, scale and sensitivity for parameter estimation problems. SIAM Journal for Scientific Computing 21(6), 2096–2113.

Gu, H. and D.S. Oliver (2005). History matching of the PUNQ-S3 reservoir model using the ensemble kalman filter. SPE Journal, Volume 10, Number 2. pp. 217-224.

Guyaguler, B., Gumrah, F. (1999). Comparison of GA with Linear Programming for the Optimization of an Underground Storage Field. IN SITU, 23(2), pp 131–150

Guyaguler, B., Horne, R.N., Rogers, L., and Rosenzweig, J.J: "Optimization of well placement in Gulf of Mexico waterflooding project", SPE 63221, SPE Annual Technical conference and Exhibition, Dallas, Texas, USA., Oct. 1-4, 2000.

Haber, R., Bars, R., Lengyel, O.: Longrange predictive control of the parametric Hammerstein model. In Proceedings of the 4th IFAC Nonlinear Control Systems Design Symposium, Enschede, Netherlands, pp. 434–439, (1998)

Hajizadeh, Y., Demyanov, V., Linah Mohamed, L., and Christie, M. (2011). Comparison of Evolutionary and Swarm Intelligence Methods for History Matching and Uncertainty Quantification in Petroleum Reservoir Models, Intelligent Computational Optimization in Engineering – Studies in Computational Intelligence, Volume 366/2011, pp. 209-240

Haupt, R.L and Haupt, S.E. *Practical Genetic Algorithms, Second Edition*, ISBN 0-471-4556 -5-2 John Wiley & Sons, Inc. (2004).

Haus, U-U., Michaels, D., Savchenko, A., (2008). Extended Formulations for MINLP Problems by Value Decompositions. International Conference on Engineering Optimization, EngOpt, Rio de Janeiro, Brazil

Hendtlass, T., Randall, M.: A survey of ant colony and particle swarm metaheuristics and their application to discrete optimization problems. In: Proc. of the Inaugural Workshop on Artificial Life (AL'01), pp. 15–25 (2001)

Hendtlass, T.: A combined swarm differential evolution algorithm for optimization problems. In: Proceedings of the 14th Int. Conf. on Ind. and Eng. App. of Artificial Intell. and Expert Systems. Lecture Notes in Computer Science, vol. 2070, pp. 11–18 Springer, Berlin (2001)

Higashi, N., Iba, H.: Particle swarm optimization with Gaussian mutation. In: Proceedings of the IEEE Swarm Intell. Symposium, pp. 72–79. Indianapolis, IN (2003)

Ho, B. and Kalman, R. (1965). Effective construction of linear state-variable models from input-output functions, Regelungstechnik 12: 545–548.

Hu, L. Y. and S. Jenni (2005). Historymatching of object-based stochastic reservoir models. SPE Journal 10, 312–323.

Huang, B. and Kadali, R. (2008): Model Predictive Control: Conventional Approach. Dyn. Modeling, Predict. Ctrl. & Perform.Monitor., LNCIS 374, pp. 101–119, springerlink.com Springer-Verlag London Limited 2008

Ingber, L. (1989): Very fast simulated reannealing. Math. Comput. Model. 12, 967–993

Jacquard, P. and C. Jain (1965). Permeability distribution from field pressure data. SPE Journal (SPE 1307) (December), 281–294.

Jafarpour, B., and McLaughlin, D. B., (2007a), Efficient Permeability Parameterization With the Discrete Cosine Transform: SPE Reservoir Simulation Symposium, 26-28 February 2007, Houston, Texas (SPE 106453-MS).

Jafarpour, B., and McLaughlin, D. B., (2007b), History Matching With an Ensemble Kalman Filter and Discrete Cosine Transform: SPE Annual Technical Conference and Exhibition, 11-14 November 2007, Anaheim, California (SPE 108761-MS).

Jahns, H. O. (1966). A rapid method for obtaining a two-dimensional reservoir description from well pressure response data. SPE Journal (SPE 1473-PA) 6(4), 315–327.

Jansen, J. D. (2012). Systems Theory for Reservoir Management. Lecture notes for the course AES1490, TUDelft.

Jansen, J. D., D. R. Brouwer, G. Naevdal and C. P. J. W. van Kruijsdijk (2005). Closed-loop reservoir management. First Break 23, 43–48.

Jansen, J. D., S. D. Douma, D. R. Brouwer, P. M. J. V. den Hof, O. H. Bosgra and A. W. Heemink (2009). Closed-loop reservoir management. In *SPE Reservoir Simulation Symposium*, TheWoodlands, Texas, US.

Jansen, J.D. (2010): Closed-Loop Reservoir Mgt, SPE Golden Gate Section. 8 Dec. 2010

Jazwinski, Andrew H. (1970). Stochastic Processes and Filtering. Mathematics in Science and Engineering. New York: Academic Press. pp. 376. ISBN 0-12-381550-9.

Juang, C.F.:Ahybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **34**(2), 997–1006 (2004)

Julier, S.J.; Uhlmann, J.K. (1997). "A new extension of the Kalman filter to nonlinear systems". Int. Symp. Aerospace/Defense Sensing, Simul. & Controls 3.

Kailath, T., Sayed, A.H., Hassibi, B. (2000): Linear Estimation. Prentice Hall Information and System Sciences Series

Kalman, R. E., (1960), A new approach to linear filtering and prediction problems: Transactions of the ASME – Journal of Basic Engineering, Series D, 82 (1960), pp. 35–45.

Keerthi, S. S. and Gilbert E. G. (1988): Optimal Infinite-Horizon Feedback Laws for a General Class of Constrained Discrete-Time Systems: Stability and Moving-Horizon Approximations, *Journal of Optimization Theory & App.*, Vol. 57(2), pp. 265-293, 1988.

Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE, Neural Networks Council Staff, IEEE Neural Networks Council (eds.) Proc. IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE, Los Alamitos (1995)

Kennedy, J.: Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: Proceedings of Cong. of Evolutionary Computation, vol. 3, pp. 1931–1938. IEEE, New York (1999)

Key World Energy Statistics KWES 2010: The International Energy Agency (IEA) 2010 www.**iea**.org/publications/freepublications/publication/kwes.pdf (accessed in Nov. 2012)

Konar, A. (2005), Computational Intelligence: Principles, Techniques and Applications, Springer Berlin Heidelberg New York ISBN 3-540-20898-4

Kosmidis, V. D., J. D. Perkins and E. N. Pistikopoulos (2005). A mixed integer optimization formulation for the well scheduling problem on petroleum fields. Computers & Chemical Engineering 29(7), 1523–1541.

Kou, J., and Sun, S. (2010): On Iterative IMPES formulation for two-phase flow with capillarity in heterogeneous porous media, Intl. Journal of Numerical Analysis and modeling, Series B, Vol. 1, No. 1, Pp. 20–40

Kouvaritakis, B., Wang, W, Lee, Y.I.: Observers in nonlinear model-based predictive control. International Journal of Robust and Nonlinear Control, 10 749–761 (2000)

Kraaijevanger, J. F. B. M., Egberts, P. J. P., Valstar, J. R., and Buurman, H. W. (2007). Optimal Waterflood Design Using the Adjoint Method. Paper SPE 105764 presented at 2007 SPE RRS, Houston, TX, U.S.A, 26–29 February.

Lampinen, J.: A constraint handling approach for the differential evolution algorithm. In: Proc. the Congress on Evolutionary Computation, vol. 2, pp. 1468–1473 (2002)

Larimore, W.E. (1997), Optimal Reduced Rank Modeling, Prediction, Monitoring, and Control using Canonical Variate Analysis, Proc. IFAC 1997 Int. Symp. on Advanced Control of Chemical Processes, held June 9-11, 1997, Banff, Canada, pp. 61-6.

Lee, T. Y. and J. H. Seinfeld (1987). Estimation of two-phase petroleum reservoir properties by regularization. Journal of Computational Physics 69, 397–419.

Li, R., Reynolds, A. C., and Oliver, D. S., (2003), History matching of three-phase flow production data: SPE Journal, 8, no. 4, 328–340 (SPE 87336–PA).

Li, Z., Zheng, D., Hou, H.: A Hybrid Particle Swarm Optimization Algorithm Based on Nonlinear Simplex Method and Tabu Search, Advances in Neural Networks - ISNN 2010, Lecture Notes in Computer Science Volume 6063, 2010, pp 126-135

Liu, Y., Wang, G., Chen, H., Dong, H., Zhu, X., Wang, S.: An Improved Particle Swarm Optimization for Feature Selection. Science Direct Jour. of Bionic Engr. (2011) Vol.8 (2)

Ljung, L. (1987). System Identification: Theory for the User. Prentice-Hall Information and System Sciences Series. PTR Prentice-Hall, Inc., 1987.

Ljung, L. (1999). System Identification - Theory for the User. Prentice-Hall, Upper Saddle River, N.J., 2 edition, 1999.

Logson, J.S. and L.T. Biegler, (1992). Decomposition strategies for large-scale dynamic optimization problems. Chem. Engng. Sci., 47: 851.

Lorentzen, R. J., G. Nævdal and A. C. V. M. Lage (2003). Tuning of parameters in a two phase flow model using an ensemble kalman filter. International Journal of Multiphase Flow 29(8), 1283–1309.

Løvbjerg, M., Krink, T.: Extending particle swarms with self-organized criticality. In: Proc. of the 4th Congress on Evolutionary Computation, pp. 1588–1593 (2002)

Løvbjerg, M., Rasmussen, T., Krink, T.: Hybrid particle swarm optimizer with breeding and subpopulations. In: Proc. of the 3rd Genetic and Evolutionary Computation Conference (GECCO-2001), vol. 1, pp. 469–476 (2001)

Luenberger, D.G. (1979). Introduction to dynamic systems: Theory, models and applications. John Wiley & Sons.

Maciejowski, J. M. *Predictive Control with Constraints*. Prentice-Hall, 2002.

Mehra, R., and Davis, R., (1972). A Generalized Gradient Method for Optimal Control Problems with Inequality Constraints and Singular Arch. IEEE Transactions on Automatic Control, 17:69–79.

Mersmann, O., Preuss, M., Trautmann, H. (2010): Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. R. Schaefer et al. (Eds.): PPSN XI, Part I, LNCS 6238, pp. 73–82, Springer-Verlag Berlin Heidelberg 2010

Meum, P., Tøndel, P., and Aamo, O. (2008). Optimization of smart well production through nonlinear model predictive control. SPE 112100.

Michalewicz, Z. and Schoenauer, M.: Evolutionary algorithms for constrained parameter optimization problems. Evol. Comput. **4**(1), 1–32 (1996)

Miranda, V., Fonseca, N.: New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control. In: The 14th Power Sys. Comp. Conf. (PSCC'02), Sev. Spain (2002)

Mohamed, L., Christie, M., Demyanov, V., Robert, E., and Kachuma, D. (2010) Application of particle swarms for history matching in the Brugge reservoir. In: Proceedings of the SPE Annual Tech. Conf. and Exhibition (ATCE), SPE 135264, 20–22 Sept., Florence, Italy.

Montes, G., Bartolome, P., Udias, A.L. (2001). The use of genetic algorithms in well placement optimization. In: SPE Latin American and Caribbean Petroleum Engineering Conference, SPE69439.

Montleau, P. de., Cominelli,A. , Neylon,K., Rowan, D., Pallister,I., Tesaker, O., and Nygard,I. (2006). Production Optimization under Constraints Using Adjoint Gradient In Proceedings of ECMOR X-10th European Conference on the Mathematics of Oil Recovery number A041, Amsterdam, The Netherlands. EAGE.

Murthy, B.S.N., K. Gangiah and A. Husain, (1980). Performance of various methods in computing optimal policies. Chem. Eng. Journal., 19: 201- 208.

Nævdal, G., L. M. Johnsen, S. I. Aanonsen and E. H. Vefring (2005). Reservoir monitoring and continuous model updating using ensemble kalman filter. SPE Journal (SPE 84372) 10(1), 66–74

Norquay, S.J., Palazoglu, A., Romagnoli, J.A.: Model predictive control based on Wiener models. Chemical Engineering Science, 53(1) 75–84 (1998)

Oil & Gas Journal, "Worldwide Look at Reserves and Production," Oil & Gas Journal, Vol. 106, No. 47 (December 6, 2010), pp. 46-49, website www.ogj.com

Oliver, D. S., (1996), Multiple Realizations of the Permeability Field From Well Test Data: SPE Journal, 1, no. 2, 145–154 (SPE 27970–PA).

Oliver, D. S., Reynolds, A. C., and Liu, N., (2008), Inverse Theory for Petroleum Reservoir Characterization and History Matching: Cambridge University Press.

Onwunalu, J., Durlofsky, L.J.: Application of a particle swarm optimization algorithm for determining optimum well location and type. Comput. Geosci. 14(1), 183–198 (2010)

Onwunalu, J.: Optimization of nonconventional well placement using genetic algorithms and statistical proxy. Master's Thesis, Stanford University (2006)

Orderud, F. (2005): Comparison of Kalman Filter Estimation Approaches for State Space Models with Nonlinear Measurements (2005).

Peaceman, D.W. (1977), Fundamentals of Numerical Reservoir Simulation, Elsevier, NY.

Peaceman, D.W. (1978), Interpretation of Well-Block Pressures in Numerical Reservoir Simulation, Paper SPE 6893, presented at the 52nd Annual Fall Technical Conference and Exhibition, Denver, CO.

Pelikan and Lobo (1999). Parameter-less Genetic Algorithm: A Worst-case Time and Space Complexity Analysis, (IlliGAL Report No. 99018). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Pintelon, R. and J. Schoukens (2012). System Identification: a Frequency Domain Approach. IEEE Press: Piscataway, New Jersey (second edition).
Podnar, H., Kapov, J. (2003): Genetic Algorithm for Network Cost Minimization Using Threshold Based Discounting. Jou. Of App. Math. and Decision Sciences, 7(4), 207–228

Pohlheim, H (2006). GEATbx Examples, Examples of Objective Functions

Poli, R., Di Chio, C., Langdon, W.B.: Exploring extended particle swarms: a genetic programming approach. In: Beyer, H.-G., et al. (eds.) GECCO 2005: Proceedings of the 2005 Conf. on Genetic and Evolutionary Computation, pp. 169–176. Washington, DC (2005)

Poli, R., Langdon, W.B., Holland, O.: Extending particle swarm optimization via genetic programming. In: Keijzer,M.,et al. (eds.) Lecture Notes in Computer Science. Proceedings of the 8th European Conference on Genetic Programming, vol. 3447, pp. 291–300. Springer, Berlin, Lausanne, Switzerland (2005)

Przybysz-Jarnut J.K. (2010). Hydrocarbon Reservoir Parameter Estimation Using Production Data and Time-Lapse Seismic. Ph. D. thesis, TUDelft

Qin, S. J. and Badgwell, T. A., A survey of industrial model predictive control technology, *Control Engineering Practice*, Vol. 11, no. 7, pp. 733 764, 2003.

RamaRao, B. S., Venue, A. M. L., de Marsily, G., and Marietta, M. G., 1995, Pilot point methodology for automated calibration of an ensemble of conditionally simulated transmissivity fields. Theory & Comp. Expt: Water Resources Research, 31, no. 3, 475–493.

Ramirez, W. F. (1987). Applications of Optimal Control Theory to Enhanced Oil Recovery. Elsevier Science Publishers

Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Self-organizing hierarchical particle swarm optimizer with time-varying accelerating coefficients. IEEE Trans. Evol. Comput. **8**(3), 240–255 (2004)

Rawlings, J.B., Mayne, D.Q (2009): Model Predictive Control Theory and Design. Nob Hill Publishers, ISBN-13: 978-0975937709

Reynolds, A. C., N. He, L. Chu and D. S. Oliver (1996). Reparameterization techniques for generating reservoir descriptions conditioned to variograms and well-test pressure data. SPE Journal (SPE 30588-PA) 1(4), 413–426.

Rezapour, A. (2009): Improved Waterflooding Performance Using Model Predictive Control. MSc Thesis TUDelft

Ringset, R., Imsland, L., and Foss, B. A. (2010): On gradient computation in single-shooting nonlinear model predictive control. In Proc. of IFAC DYCOPS 2010, Leuven, Belgium.

Rogalsky, T., Derksen, R.W., Kocabiyik, S.: Differential evolution in aerodynamic optimization. Can. Aeronaut. Space Inst. J. **46**, 183–190 (2000)

Roggero, F. and L. Y. Hu (1998). Gradual deformation of continuous geostatistical models for history matching.

Roggero, F., Mezghani, M., Hu, L. Y., Ravalec-Dupin, M. L. and Fenwick, D.: (2002). Constraining Stochastic Reservoir Models to Dynamic Data: An Integrated Approach. In the proceedings of 17th World Petroleum Congress, Sept. 1 - 5, 2002, Rio de Janeiro, Brazil

Romero, C., Carter, J., Gringarten, A., Zimmerman, R (2000b) A Modified Genetic Algorithm for Reservoir Characterization, SPE 64765, International Oil and Gas Conference and Exhibition, Beijing, China 7-10 November

Romero, C., Carter, J., Zimmerman, R., Gringarten, A (2000a) Improved Reservoir Characterization Through Evolutionary Computation, SPE 62942, Annual Technical Conference and Exhibition, Dallas, Texas, USA, 1-4 October

Rommelse, J. R., O. Kleptova, J. D. Jansen and A.W. Heemink (2006). Data assimilation in reservoir management using the representer method and the ensemble kalman filter. In 10th European Conference on the Mathematics of Oil Recovery, Amsterdam.

Rossi, D., Carney, M., Kontchou, J.N., Lancaster D., and McIntyre, S. (2002). Reservoir and Production Optimization, white paper from www.slb.com

Saputelli, L., M. Nikolaou and M. J. Economides (2006). Real-time reservoir management: A multiscale adaptive optimization and control approach. Comp. Geos. 10(1), 61–96.

Sarma P. (2006): Efficient closed-loop optimal control of petroleum reservoirs under uncertainty. PhD Thesis Stamford University

Sarma, P., Aziz, K. and Durlofsky, L.J., (2005): Implementation of adjoint solution for optimal control of smart wells. Paper SPE 92864 presented at the *SPE Reservoir Simulation Symposium*, Houston, USA, 31 January – 2 February. DOI: 10.2118/92864-MS.

Sarma, P., Chen, W.H. (2008): Efficient well placement optimization with gradient-based algorithms and adjointmodels. Paper SPE112257 presented at the 2008 SPE Intelligent Energy Conference and Exhibition, Amsterdam, 25–27 February 2008

Sarma, P., Chen,W. H., Durlofsky, L. J., and Aziz, K. (2008a). Production Optimization with Adjoint Models Under Nonlinear Control-State Path Inequality Constraints. SPEREE, 11(2):326–339. Paper SPE 99959.

Sarma, P., Durlofsky, L., and Aziz, K., (2008b), Kernel Principal Component Analysis for Efficient Differentiable Parameterization of Multipoint Goestatistics: Mathematical Goesciences, 40, no. 1, 3–32.

Sarma, P., L. J. Durlofsky, K. Aziz and W. H. Chen (2006). Efficient real-time reservoir management using adjoint-based optimal control and model updating. Computational Geosciences **10**(1), 3–36.

Sarma, P., L. J. Durlofsky, K. Aziz and W. H. Chen (2007). A new approach to automatic history matching using kernel pca. In SPE Res. Sim. Sym. (SPE 106176), Houston.

Schulze-Riegert, R. and Ghedan, S. (2007). Modern Techniques for History Matching, 9th International Forum on Reservoir Simulation, December 9 – 13, 2007 Abu Dhabi, UAE

Schulze-Riegert, R., Krosche, M., Pajonk, O., Mustafa, H (2009) Data Assimilation Coupled to Evolutionary Algorithms – A Case Example in History Matching, SPE 125512, SPE/EAGE Reservoir Characterization & Simulation Conf., Abu Dhabi, UAE, 19-21 Oct.

Sen, M., Datta-Gupta, A., Stoffa, P., Lake, L., Pope, G (1995). Stochastic Reservoir Modeling Using Simulated Annealing and Genetic Algorithms, SPE 24754, SPE Formation Evaluation, volume 10, number 1, 49-55

Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proc. IEEE International Conf. on Evol. Comput., pp. 69–73. IEEE, Piscataway, NJ (1998)

Skjervheim, J.-A., G. Evensen, Aanonsen S. I. T. A. Johansen (2007). Incorporating 4D seismic data in reservoir sim. model using ensemble kalman filter. SPE Jou. 12(3), 282–292.

Slater, G., Durrer, E (1970). Adjustment of Reservoir Simulation Models to Match Field Performance, SPE 2983, 45th Annual Fall Meeting, Houston, Texas, USA, 4-7 October

Spall, J.C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Transactions on Automatic Control 37(3), 332–341.

Storn, R., and Price, K. Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over continuous spaces. Tech. Rep. TR-95-012, International Computer Science Institute (ICSI), 1995.

Storn, R., Price, K.: Differential evolution—simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)

Sudaryanto, B. and Y. C. Yortsos (2000). Optimization of fluid front dynamics in porous media using rate control 1. equal mobility fluids. Physics of Fluids 12(7), 1656–1670.

Sudaryanto, B. and Y. C. Yortsos (2001). Optimization of displacements in porous media using rate control. In SPE Annual Technical Conference and Exhibition (SPE 71509).

Suwartadi, E., Krogstad, S., & Foss, B. (2010). A Lagrangian-Barrier Function for Adjoint State Constraints Optimization of Oil Reservoir Water Flooding. In Proceeding of IEEE Conference on Decision and Control 2010, Atlanta, Georgia, USA.

Talbi, N., Belarbi, K. (2011): Fast Hybrid PSO and Tabu Search Approach for Optimization of a Fuzzy Controller. Intl. Journal of Comp. Sci. Issues, Vol. 8, (5), No 2, Sept. 2011

Tarantola, A., (1987), Inverse Problem Theory. Methods for Data Fitting and Model Parameter Estimation: Elsevier.

Tayamon, S.: Nonlinear system identification with applications to selective catalytic reduction systems. PhD Thesis, Uppsala University, Sweden. April, 2012.

Teo, K., G. Goh and K.A. Wong, (1991). Unified Computational Approach to Optimal Control Problems. Pitman Monographs and Surveys in Pure and Applied Mathematics. John Wiley & Sons, Inc., New York.

Thangaraj, R., Pant, M., Abraham, A., Bouvry, P.: Particle swarm optimization: hybridization perspectives and experimental illustrations. Appl. Math. Comput. **217**, 5208–5226 (2011)

Therrien, C.W: Discrete Random Signals and Statistical Signal Proc. Prentice Hall, 1992.

TNO–ISAPP: http://www.tno.nl/downloads/TNO-JRV180511-02%20production%20optimi-sation.pdf (last accessed on 12 March 2012)

U.S. Energy Information Administration (2008), World Coal Production, Most Recent Estimates 1980–2007

U.S. Energy Information Administration, International Energy Outlook 2011, DOE/EIA-0484(2011) (Wash. DC, Sept. 2011), http://www.eia.gov/forecasts/ieo/pdf/0484 (2011).pdf

van Essen, G., Rezapour, A., Van den Hof, P.M.J., and Jansen, J.D. Integrated dynamic optimization and control in reservoir engineering using locally identified linear models. 49th IEEE Conference on Decision and Control (CDC), 2010 van Essen, G.M., Zandvliet, M.J., Van den Hof, P.M.J., Bosgra, O.H., Jansen, J.D.: Robust waterflooding optimization of multiple geological scenarios. SPE J. **14**(1), 202–210 (2009)

van Overschee, P. and de Moor, B. (1996): Subspace Identification for Linear System, Theory, Implementation and Applications. Kluwer Academic Publishers

Vasco, D. W. and A. Datta-Gupta (1997). Integrating field production history in stochastic reservoir characterization. SPE Formation Evaluation 12(3), 149–156 (36567–PA).

Vasco, D. W., S. Soon and A. Datta-Gupta (1999). Integrating dynamic data into high resolution reservoir models using streamline-based analytic sensitivity coefficients. SPE Journal 4(4), 389–399.

Vasiljevic, D., Golobic, J.: Comparison of the classical dumped least squares and genetic algorithm in the optimization of doublets. In: Proceedings of the First Workshop on Soft Computing, pp. 200–204. Nagoya, Japan (1996)

Verhaegen, M. and Verdult, V. (2007): Filtering and System Identification: A Least Squares Approach. Cambridge University Press, UK, ISBN-13: 978-0521875127

Voss, S. and T. Patel (2007). Total, Shell Chief Executives Say “Easy Oil” Is Gone. http://www.bloomberg.com/apps/news?pid=newsarchive&sid=aH57.uZe.sAL (last accessed on August 20, 2012).

Wang, C., Li, G., Reynolds, A.C.: Optimal well placement for production optimization. Paper SPE111154 presented at the 2007 SPE Eastern Reg. Meeting, Lexington, 11–14 Oct. 2007

Wang, J and Buckley, J.S.: (2006). Automatic History Matching Using Differential Evolution Algorithm International Symposium of the Society of Core Analysts Trondheim, Norway

Wang, W., Hendriksen, R.: Generalized predictive control of nonlinear systems of the Hammerstein form. Modeling, identification and control, 15(4) 253–262 (1994)

Wang, Y., Zhao, Z., Ren, R. Hybrid Particle Swarm Optimizer with Tabu Strategy for Global Numerical Optimization, IEEE Congress on Evolutionary Computation (CEC 2007)

Wasserman, M. L., A. S. Emanuel, J. H. Seinfeld (1975). Practical app. of optimal-control theory to history-matching multiphase simulator models. SPE Journal 15(4), 347–355.

Watson, A. T., J. H. Seinfeld, G. R. Gavalas and P. T.Woo (1980). History matching in two phase petroleum reservoirs. SPE Journal 20(6), 521–532.

Wen, X.-H., C. V. Deutsch and A. S. Cullick (1998). Integrating pressure and fractional flow data in reservoir modeling with fast streamline-based inverse method. In SPE paper 48971 presented at the SPE Annual Technical Conference and Exhibition, New Orleans, USA.

Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computations, 1, 67–82 (1997)

Wu, Z. and A. Datta-Gupta (2002). Rapid history matching using a generalized travel time inversion method. SPE Journal 7, 113–122.

Yang, P., A. T.Watson and R. V. Armasu (1988). Automatic history matching with variable metric methods. SPE Reservoir Evaluation and Engineering 3(3), 995–1001.

Yeten, B. (2003). Optimum Deployment of NonconventionalWells. Ph. D. thesis, Stanford.

Yeten, B., Durlofsky, L.J., Aziz, K.: Optimization of nonconventional well type, location and trajectory. SPE J. 8, 200–210 (2003)

Zakirov, I. S., S. I. Aanonsen, E. S. Zakirov and B.M. Palatnik (1996). Optimization of reservoir performance by automatic allocation of well rates. In 5th European Conference on the Mathematics of Oil Recovery, Leoben.

Zandvliet, M. J., O. H. Bosgra, J. D. Jansen, P. M. J. Van den Hof and J. F. B. M. Kraaijevanger (2007). Bang-bang control and singular arcs in reservoir flooding. Journal of Petroleum Science and Engineering 58, 186–200.

Zandvliet, M., (2008), Model-based lifecycle optimization of well locations and production settings in petroleum reservoirs: Ph.D. thesis, Delft University of Technology.

Zandvliet, M.J., Handels, M., van Essen, G.M., Brouwer, D.R., Jansen, J.D. (2008): Adjoint-based well-placement optimization under production constraints. SPE J. 13(4), 392–399

Zhang, W.J., Xie, X.F.: DEPSO: hybrid particle swarm with differential evolution operator. In: IEEE International Conference on Systems,Man and Cybernetics (SMCC), pp. 3816–3821. Washington DC, USA (2003)

Ziegler, J.: Tutorial on Library of Efficient Data Types and Algorithms (LEDA), Max-Planck Institut für Informatik (2002)

Zielinski, K., Peters, D., Laur, R. “Run time analysis regarding stopping criteria for differential evolution and particle swarm optimization,” in Proc. 1st Int. Conf. Exp. Process on System Modelling, Simulation and Optimization, 2005

APPENDICES

```
function [xpt, ftp, stat] = wellopt1(objfct,D,K,Xmin,Xmax,param,strategy)
% source code for differential evolution algorithm
% for a D-dimensional maximization problem with the search space % bounded by a box
%
% Input:
% objfct - a handle to the objective function
% D - the number of dimensions
% Xmin - the lower bounds of the box constraints
% Xmax - the upper bounds of the box constraints
% K - the number of function evaluations used
% by the optimization algorithm
% epsilon - not applicable: set it to -1
% strategy – rand/1/bin
% involves a process where mutation target is randomly selected from target and mutation
% performed using a single vector as well as a uniform binomial crossover
% param - a MATLAB structure containing algorithm
% control input parameters
%   Np – population of candidate solution
%   F – scaling factor
%   CR – crossover rate
%
% Output:
% xpt - a vector containing the location of the optimum
% fpt - the objective function value of the optimum
% stat - a MATLAB structure containing algorithm
% specific output statistics
%   histf - the best objective function value history
%
% Parameter Settings
if epsilon ~= -1
   warning ('pso : epsilon_ignored', 'epsilon is ignored')
end
% Np (population number) >= 10
if isfield (param,'Np')
   Np = param.Np;
else
   Np >= 10;
end
% F (scaling factor) = 0.5
if isfield (param, 'F')
```

```
    F = param.F;
else
    F = 0.5;
end
% CR (crossover rate) = 0.1
if isfield (param, 'CR')
    CR = param.CR;
else
    CR = 0.1;
end
% Population Initialization
for i=1:Np
    for j=1:D
        A(1,i,j)=Xmin(j) + (rand*(Xmax(j)-Xmin(j)));
    end
end
for i=1:Np
    sum(1,i) = 0;
    for j=2:D
        sum(1,i)= sum(1,i)+A(1,i,j);
    end
    objfct1(1,i)= A(1,i,1);
    objfct2(1,i)= g(1,i).*(1-(sqrt(A(1,i,1)/g(1,i))));
end
for k = 1:K
    for i=1:Np
        sum(k,i) = 0;
        for j=2:D
            sum(k,i)= sum(k,i)+A(k,i,j);
        end
        g(k,i) = 1 + ((9/(D-1)).*sum(k,i));
        objfct2(k,i)= g(k,i)*(1-((objfun1(k,i)/g(k,i))^2));
        objfct1(k,i)= A(k,i,1);
        objfct2(k,i)= g(k,i).*(1-(sqrt(A(k,i,1)/g(k,i))));
        %Mutation Process
        r=randperm(Np);
        if (r(1)==i)
            r(1)=r(Np-n);
        end
        if (r(2)==i)
            r(2)==r(Np-(n+1));
        end
        if (r(3)==i)
            r(3)=r(Np-(n+2));
        end
        for j=1:D
            if str == 1
                U(k,i,j)=A(k,r(1),j)+F*((A(k,r(2),j))-(A(k,r(3),j)));
            end
            if U(k,i,j)<lb(j)
                U(k,i,j)=Xmin(j);%+ (rand*(Xmax(j)-Xmin(j)));
            end
            if U(k,i,j)>Xmax(j)
                U(k,i,j)=Xmax(j);%Xmin(j)+ (rand*(Xmax(j)-Xmin(j)));
            end
        end
    end
    %Binomial Crossover Process
    %
    for i = 1:Np
        if crossover == 2
            for j=1:D
                p=rand;
                w=randperm(D-1);
                if ((p<=CR)|(j==w(1)))
                    T(k,i,j)=U(k,i,j);
                else
                    T(k,i,j)=A(k,i,j);
                end
            end
        end
    end
    %Evaluating Objective Function for Selection
    for i=1:Np
        for j=1:D
            A(1,i,j)=lb(j) + (rand*(ub(j)-lb(j)));
        end
    end
    for k = 1:K
        for i=1:Np
            if str == 1
                U(k,i,j)=A(k,r(1),j)+F*((A(k,r(2),j))-(A(k,r(3),j)));
            end
            if U(k,i,j)<Xmin(j)
                U(k,i,j)=Xmin(j);%+ (rand*(Xmax(j)-Xmin(j)));
            end
```

```
            if U(k,i,j)>Xmax(j)
                U(k,i,j)=Xmax(j);%Xmin(j)+ (rand*(Xmax(j)-Xmin(j)));
            end
        end
    end
end
end
```

```
function [xpt, fpt, stat] = wellopt2(objfct, D, lb, ub, nfe, epsilon, param)

% source code for particle swarm optimization algorithm
% for a D-dimensional maximization problem with the search space % bounded by a box
%
% Input:
% objfct - a handle to the objective function
% D - the number of dimensions
% lb - the lower bounds of the box constraints
% ub - the upper bounds of the box constraints
% nfe - the number of function evaluations used
% by the optimization algorithm
% epsilon - not applicable: set it to -1
% param - a MATLAB structure containing algorithm
% control input parameters
%   Np - number of particles in the swarm
%   c1 - cognitive parameter
%   c2 - social parameter
%   nbh - neighborhood incidence matrix of the swarm population
%   inw – inertia weight
%
% Output:
% xpt - a vector containing the location of the optimum
% fpt - the objective function value of the optimum
% stat - a MATLAB structure containing algorithm
% specific output statistics
%   histf - the best objective function value history
%
% Parameter Settings
if epsilon ~= -1
   warning ('pso : epsilon_ignored', 'epsilon is ignored')
end
% Np (particle count) >= 10
if isfield (param,'Np')
   Np = param.Np;
else
   Np >= 10;
end
% inw (inertia weight) = 0.721
if isfield (param, 'inw')
   inw = param.inw;
else
   inw= 0.721;
```

```
end
% c1 (cognitive parameter) = 1.193
if isfield (param, 'c1')
   c1 = param.c1;
else
   c1 = 1.193;
end
% c2 (social parameter) = 1.193
if isfield (param, 'c2')
   c2 = param.c2;
else
   c2 = 1.193;
end
% nbh (neighbourhood matrix) initialization at local best topology
if isfield (param,'nbh')
   nbh = param.nbh;
else
   nbh = nbh_local_best (Np);
end
% Swarm Population Initialization
% Initialize v (velocity) uniform and randomly between lb and ub
v = repmat (lb', Np, 1) + repmat (ub'-lb', Np, 1).*rand (Np, D);
%
% Initialize x (position) uniform randomly between lb and ub
x = repmat (lb', Np, 1) + repmat (ub'-lb', Np, 1).*rand (Np, D);
%
% Initially the personal best position is the starting position
p = x;
%
% Initialize the personal best objective function evaluation to infinity to always allow
improvement
p_best = ones (Np, 1)*Inf;
%
% Initialize the number of objective function evaluations to one
evalcount = 1;
%
% Preallocate an array that will hold the objective function evaluations
f = zeros (Np, 1);
%
% Preallocate the stat.histf array
stat = struct ();
stat.histf = zeros (nfe, 1);
%
```

```
% Loop while number of objective function evaluations does not exceeds the stop criterion
while evalcount < nfe
   %
   % Evaluate for all particles the objective function
   for i = 1 : Np
      f(i) = feval (objfct, x(i, :));
      %
      % Update stat.histf array
      stat.histf (evalcount + i) = max (f);
   end
   %
   % Update the personal best positions if the current position is better than the current
personal best position
   p = repmat (f < p_best, 1, N).*x + repmat (~( f < p_best ), 1, D).*p;
   %
   % Update the personal best objective function evaluation
   p_best = max(f, p_best);
   %
   % Calculate the best particle in each neighborhood
   [l_best, g] = max( repmat (p_best, 1, Np).*nbh);
   %
   % Update the velocities using velocity update equation
   for i = 1 : Np
      v = inw*(v +(p - x).*rand(Np,D)*c1+(p(g,:)- x).*rand(Np,D) * c2);
   end
   % Update the positions
   x = x + v;
   % Update the number of objective function evaluations used
   evalcount = evalcount + Np;
end
% Select the optimum from the personal best objective function evaluations
[fopt , g] = min (p_best );
% Select the optimum solution
xopt = p(g, :)
end
```

```
function [nbh] = nbh_local_best (Np)

% source code for generating local best ring topology neighborhood in PSO  algorithm
% for a swarm population Np

% [nbh] = nbh_local_best (Np)
% Local Neighborhood Incidence Matrix Generator
% creates an incidence matrix of a ring topology of size NpxNp
% note: incidence is reflexive
%
% Input:
% Np - the number of particles
%
% Output:
% nbh - the incidence matrix:
% 0 represents no incidence
% 1 represents an incidence
nbh = diag (ones (Np, 1)) + diag (ones (Np - 1, 1), 1) + diag ( ones (Np - 1, 1), -1) + ...
diag ( ones (1, 1), Np - 1) + diag ( ones (1, 1), -(Np - 1));
end
```

```
function [xpt, ftp, stat] = wellopt3(objfct,D,K,Xmin,Xmax,param,strategy)

% source code for hybrid particle swarm differential evolution algorithm
% for a D-dimensional maximization problem with the search space % bounded by a box
%
% Input:
% objfct - a handle to the objective function
% D - the number of dimensions
% Xmin - the lower bounds of the box constraints
% Xmax - the upper bounds of the box constraints
% K - the number of function evaluations used
% by the optimization algorithm
% epsilon - not applicable: set it to -1
% strategy – rand/1/bin
% involves a process where mutation target is randomly selected from target and mutation is
% % performed using a single vector as well as a uniform binomial crossover
% param - a MATLAB structure containing algorithm
% control input parameters
%   Np – population of candidate solution
%   F – scaling factor
%   CR – crossover rate
%   c1 - cognitive parameter
%   c2 - social parameter
%   nbh - neighborhood incidence matrix of the swarm population
%   inw – inertia weight
%
% Output:
% xpt - a vector containing the location of the optimum
% fpt - the objective function value of the optimum
% stat - a MATLAB structure containing algorithm
% specific output statistics
%   histf - the best objective function value history
%
% Parameter Settings
if epsilon ~= -1
   warning ('pso : epsilon_ignored', 'epsilon is ignored')
end
% Np (population number) >= 10
if isfield (param,'Np')
   Np = param.Np;
else
   Np >= 10;
end
```

```
% F (scaling factor) = 0.5
if isfield (param, 'F')
   F = param.F;
else
   F = 0.5;
end
% CR (crossover rate) = 0.1
if isfield (param, 'CR')
   CR = param.CR;
else
CR = 0.1;
end

% inw (inertia weight) = 0.721
if isfield (param, 'inw')
   inw = param.inw;
else
   inw= 0.721;
end

% c1 (cognitive parameter) = 1.193
if isfield (param, 'c1')
   c1 = param.c1;
else
   c1 = 1.193;
end
% c2 (social parameter) = 1.193
if isfield (param, 'c2')
   c2 = param.c2;
else
   c2 = 1.193;
end
% nbh (neighbourhood matrix) initialization at local best topology
if isfield (param,'nbh')
   nbh = param.nbh;
else
   nbh = nbh_local_best (Np);
end

% Population Initialization
for i=1:Np
   for j=1:D
      A(1,i,j)=Xmin(j) + (rand*(Xmax(j)-Xmin(j)));
```

```
    end
end
for i=1:Np
    sum(1,i) = 0;
    for j=2:D
        sum(1,i)= sum(1,i)+A(1,i,j);
    end
    objfct1(1,i)= A(1,i,1);
    objfct2(1,i)= g(1,i).*(1-(sqrt(A(1,i,1)/g(1,i))));
end
for k = 1:K
    for i=1:Np
        sum(k,i) = 0;
        for j=2:D
            sum(k,i)= sum(k,i)+A(k,i,j);
        end
        g(k,i) = 1 + ((9/(D-1)).*sum(k,i));
        objfct2(k,i)= g(k,i)*(1-((objfun1(k,i)/g(k,i))^2));
        objfct1(k,i)= A(k,i,1);
        objfct2(k,i)= g(k,i).*(1-(sqrt(A(k,i,1)/g(k,i))));

        %Mutation Process
        r=randperm(Np);
        if (r(1)==i)
            r(1)=r(Np-n);
        end
        if (r(2)==i)
            r(2)==r(Np-(n+1));
        end
        if (r(3)==i)
            r(3)=r(Np-(n+2));
        end
        for j=1:D
            if str == 1
                U(k,i,j)=A(k,r(1),j)+F*((A(k,r(2),j))-(A(k,r(3),j)));
            end
            if U(k,i,j)<lb(j)
                U(k,i,j)=Xmin(j);%+ (rand*(Xmax(j)-Xmin(j)));
            end
            if U(k,i,j)>Xmax(j)
                U(k,i,j)=Xmax(j);%Xmin(j)+ (rand*(Xmax(j)-Xmin(j)));
            end
        end
    end
    %Binomial Crossover Process
    %
    for i = 1:Np
        if crossover == 2
            for j=1:D
                p=rand;
                w=randperm(D-1);
                if ((p<=CR)|(j==w(1)))
                    T(k,i,j)=U(k,i,j);
                    else p = repmat (f < p_best, 1, N).*x + repmat (~( f <  p_best ), 1,D).*p;
                        %
                        % Update the personal best objective function evaluation
                        p_best = max(f, p_best);
                        %
                        % Calculate the best particle in each neighborhood
                        [l_best, g] = max( repmat (p_best, 1, Np).*nbh);
                        %
                        % Update the velocities using velocity update equation
                        for i = 1 : Np
                            v = inw*(v + (p - x).* rand (Np, D) * c1 + (p(g, :) -   x) .*rand (Np, D) * c2);
                        end
                        % Update the positions
                        x = x + v;
                        % Update the number of objective function evaluations used
                        evalcount = evalcount + Np;
                end
                % Select the optimum from the personal best objective function evaluations
                [fopt , g] = min (p_best );
                % Select the optimum solution
                xopt = p(g, :);
            end
        end
    end
end
end
```

```
function [xpt, ftp, stat] = wellopt4(objfct,D,K,Xmin,Xmax,param)

% source code for genetic algorithm
% for a D-dimensional maximization problem with the search space % bounded by a box
%
% Input:
% objfct - a handle to the objective function
% D - the number of dimensions
% Xmin - the lower bounds of the box constraints
% Xmax - the upper bounds of the box constraints
% K – maximum iteration by the optimization algorithm
% param - a MATLAB structure containing algorithm
% control input parameters
%   Np – Population of candidate solution
%   CR – crossover rate
%   MR – mutation  rate
%
% Output:
% xpt - a vector containing the location of the optimum
% fpt - the objective function value of the optimum
% stat - a MATLAB structure containing algorithm specific output statistics
% histf - the best objective function value history
%
if ~exist('DisplayFlag', 'var')
   DisplayFlag = true;
end

Xover_Type = 1; % crossover type: 1 = single point, 2 = two point, 3 = uniform
param.pcross = 0.6; % crossover probability
param.pmutate = 0.017; % mutation probability

% Begin the evolution loop
for GenIndex = 1 : K
   % Compute the inverse of the cost. Fitness increases with inverse cost.
   Cost = [];
   for i = 1 : Np
      Cost = [Cost, 1 / Np(i).cost];
   end
   for k = Keep+1 : 2 : Np % begin selection/crossover loop
      % Select two parents to mate and create two children - roulette wheel selection
      mate = [];
      for selParents = 1 : 2
         Random_Cost = rand * sum(Cost);
```

```
            Select_Cost = Cost(1);
            Select_index = 1;
            while Select_Cost < Random_Cost
                Select_index = Select_index + 1;
                if Select_index >= param.popsize
                    break;
                end
                Select_Cost = Select_Cost + Cost(Select_index);
            end
            mate = [mate Select_index];
        end
        Parent(1, :) = Np(mate(1)).chrom;
        Parent(2, :) = Np(mate(2)).chrom;
        % Crossover
        switch Xover_Type
            case 1
                % single point crossover
                if param.pcross > rand
                    % crossover the parents
                    Xover_Pt = ceil(rand * param.numVar);
                    % x = genes in parent 1 that are not in parent 2 (after crossover point)
                    x = setdiff(Parent(1, Xover_Pt:param.numVar), Parent(2,
Xover_Pt:param.numVar));
                    % y = genes in parent 2 that are not in parent 1 (after crossover point)
                    y = setdiff(Parent(2, Xover_Pt:param.numVar), Parent(1,
Xover_Pt:param.numVar));
                    child(k-Keep, :) = [Parent(1, 1:param.numVar-length(y)), y];
                    child(k-Keep+1, :) = [Parent(2, 1:param.numVar-length(x)), x];
                else
                    % clone the parents
                    child(k-Keep, :) = Parent(1, :);
                    child(k-Keep+1, :) = Parent(2, :);
                end
            case 2
                % multipoint crossover
                if param.pcross > rand
                    Xover_Pt1 = ceil(rand * param.numVar);
                    Xover_Pt2 = ceil(rand * param.numVar);
                    if Xover_Pt1 > Xover_Pt2
                        temp = Xover_Pt2;
                        Xover_Pt2 = Xover_Pt1;
                        Xover_Pt1 = temp;
                    end
                child(k-Keep, :) = [Parent(1, 1:Xover_Pt1) Parent(2, Xover_Pt1+1:Xover_Pt2)
Parent(1, Xover_Pt2+1:param.numVar)];
                child(k-Keep+1, :) = [Parent(2, 1:Xover_Pt1) Parent(1,
Xover_Pt1+1:Xover_Pt2) Parent(2, Xover_Pt2+1:param.numVar)];
            else
                child(k-Keep, :) = Parent(1, :);
                child(k-Keep+1, :) = Parent(2, :);
            end
        case 3
            % uniform crossover
            for i = 1 : param.numVar
                if param.pcross > rand
                    child(k-Keep, i) = Parent(1, i);
                    child(k-Keep+1, i) = Parent(2, i);
                else
                    child(k-Keep, i) = Parent(2, i);
                    child(k-Keep+1, i) = Parent(1, i);
                end
            end
    end
end % end selection/crossover loop
% Replace the non-elite Np members with the new children
for k = Keep+1 : 2 : param.popsize
    Np(k).chrom = child(k-Keep, :);
    Np(k+1).chrom = child(k-Keep+1, :);
end
% Mutation
for individual = Keep + 1 : param.popsize % Don't allow the elites to be mutated
    for parnum = 1 : param.numVar
        if param.pmutate > rand
            Np(individual).chrom(parnum) = floor(Xmin + (Xmax - Xmin + 1) * rand);
        end
    end
end
% Make sure the Np does not have duplicates.
Np = ClearDups(Np, Xmax, Xmin);
% Make sure each individual is legal.
Np = FeasibleFunction(param, Np);
% Calculate cost
Np = CostFunction(param, Np);
% Sort from best to worst
Np = PopSort(Np);
% Compute the average cost of the valid individuals
```

```
    MaxCost = [MaxCost Np(1).cost];
    AvgCost = [AvgCost AverageCost];
    if DisplayFlag
        % Select the optimum from the personal best objective function evaluations
                [fopt , g] = max (sol, g);
                % Select the optimum solution
                xopt = p(g, :);
    end
end
end
```

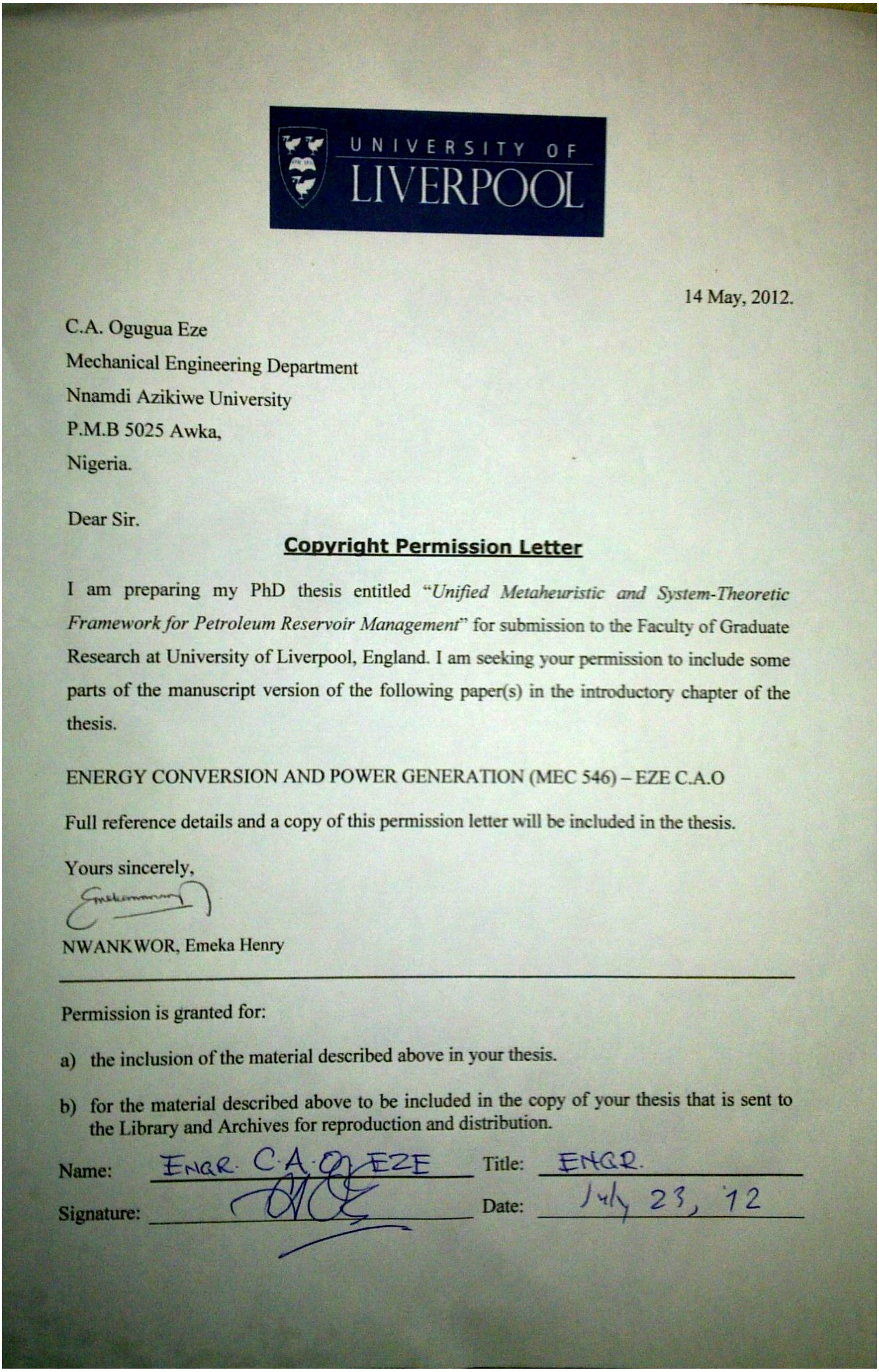
```
function [xpt, fpt, stat] = wellopt5(objfct, D, lb, ub, nfe, epsilon, param)

% source code for particle swarm optimization algorithm
% for a D-dimensional maximization problem with the search space % bounded by a box
%
% Input:
% objfct - a handle to the objective function
% D - the number of dimensions
% lb - the lower bounds of the box constraints
% ub - the upper bounds of the box constraints
% nfe - the number of function evaluations used
% by the optimization algorithm
% epsilon - not applicable: set it to -1
% param - a MATLAB structure containing algorithm
% control input parameters
%   Np - number of particles in the swarm
%   c1 - cognitive parameter
%   c2 - social parameter
%   nbh - neighborhood incidence matrix of the swarm population
%   inw – inertia weight
%
% Output:
% xpt - a vector containing the location of the optimum
% fpt - the objective function value of the optimum
% stat - a MATLAB structure containing algorithm
% specific output statistics
%   histf - the best objective function value history
%
% Parameter Settings
if epsilon ~= -1
   warning ('pso : epsilon_ignored', 'epsilon is ignored')
end
% Np (particle count) >= 10
if isfield (param,'Np')
   Np = param.Np;
else
   Np >= 10;
end
% inw (inertia weight) = 0.721
if isfield (param, 'inw')
   inw = param.inw;
else
   inw= 0.721;
```

```
end
% c1 (cognitive parameter) = 1.193
if isfield (param, 'c1')
   c1 = param.c1;
else
   c1 = 1.193;
end
% c2 (social parameter) = 1.193
if isfield (param, 'c2')
   c2 = param.c2;
else
   c2 = 1.193;
end
% nbh (neighbourhood matrix) initialization at local best topology
if isfield (param,'nbh')
   nbh = param.nbh;
else
   nbh = nbh_local_best (Np);
end
% Swarm Population Initialization
% Initialize v (velocity) uniform and randomly between lb and ub
v = repmat (lb', Np, 1) + repmat (ub'-lb', Np, 1).*rand (Np, D);
%
% Initialize x (position) uniform randomly between lb and ub
x = repmat (lb', Np, 1) + repmat (ub'-lb', Np, 1).*rand (Np, D);
%
% Initially the personal best position is the starting position
p = x;
%
% Initialize the personal best objective function evaluation to infinity to always allow
improvement
p_best = ones (Np, 1)*Inf;
%
% Initialize the number of objective function evaluations to one
evalcount = 1;
%
% Preallocate an array that will hold the objective function evaluations
f = zeros (Np, 1);
%
% Preallocate the stat.histf array
stat = struct ();
stat.histf = zeros (nfe, 1);
%
```

```
% Loop while number of objective function evaluations does not exceeds the stop criterion
while evalcount < nfe
   %
   % Evaluate for all particles the objective function
   for i = 1 : Np
       f(i) = feval (objfct, x(i, :));
       %
       % Update stat.histf array
       stat.histf (evalcount + i) = max (f);
   end
   %
   % Update the personal best positions if the current position is better than the current
personal best position
   p = repmat (f < p_best, 1, N).*x + repmat (~( f < p_best ), 1, D).*p;
   %
  nbr_cost = inf(dim12);

  for i=1:dim1-2

    for j=i+1:dim1-1

      if i==1

        if j-i==1

          nbr_cost(crnt_position(i),crnt_position(j))=crnt_position_cost-
d(1,crnt_position(i))+d(1,crnt_position(j))-
d(crnt_position(j),crnt_position(j+1))+d(crnt_position(i),crnt_position(j+1));
          best_i=i;
          best_j=j;
          best_nbr_cost = nbr_cost(crnt_position(i),crnt_position(j));
          tabu_node1 = crnt_position(i)
          tabu_node2 = crnt_position(j)
        else
          nbr_cost(crnt_position(i),crnt_position(j))=crnt_position_cost-
d(1,crnt_position(i))+d(1,crnt_position(j))-
d(crnt_position(j),crnt_position(j+1))+d(crnt_position(i),crnt_position(j+1))-
d(crnt_position(i),crnt_position(i+1))+d(crnt_position(j),crnt_position(i+1))-
d(crnt_position(j-1),crnt_position(j))+d(crnt_position(j-1),crnt_position(i));
        end
      else
        if j-i==1
```

```
nbr_cost(crnt_position(i),crnt_position(j))=crnt_position_cost-d(crnt_position(i-
1),crnt_position(i))+d(crnt_position(i-1),crnt_position(j))-
d(crnt_position(j),crnt_position(j+1))+d(crnt_position(i),crnt_position(j+1));
        else
           nbr_cost(crnt_position(i),crnt_position(j))=crnt_position_cost-d(crnt_position(i-
1),crnt_position(i))+d(crnt_position(i-1),crnt_position(j))-
d(crnt_position(j),crnt_position(j+1))+d(crnt_position(i),crnt_position(j+1))-
d(crnt_position(i),crnt_position(i+1))+d(crnt_position(j),crnt_position(i+1))-
d(crnt_position(j-1),crnt_position(j))+d(crnt_position(j-1),crnt_position(i));
        end
     end
     if nbr_cost(crnt_position(i),crnt_position(j)) < best_nbr_cost
        best_nbr_cost = nbr_cost(crnt_position(i),crnt_position(j));
        best_i=i;
        best_j=j;
        tabu_node1 = crnt_position(i);
        tabu_node2 = crnt_position(j);
     end
   end
   end
end
while (tabu_tenure(tabu_node1,tabu_node2))>0

   if best_nbr_cost < best_obj      %(TABU solution better than the best found so far)
         fprintf('\nbest nbr cost = %d\t and best obj = %d\n, hence breaking',best_nbr_cost,
best_obj);
      break;
   else
     nbr_cost(tabu_node1,tabu_node2)=nbr_cost(tabu_node1,tabu_node2)*1000;
     best_nbr_cost_col = min(nbr_cost);
     best_nbr_cost = min(best_nbr_cost_col);
     [R,C] = find((nbr_cost==best_nbr_cost),1);
     tabu_node1 = R;
     tabu_node2 = C;
   end
end
if best_nbr_cost > crnt_position_cost
   min_d_col = min(d);
   penal_nbr_cost = nbr_cost + min(min_d_col)*frequency;
   penal_best_nbr_cost_col = min(penal_nbr_cost);
   penal_best_nbr_cost = min(penal_best_nbr_cost_col);
   [Rp,Cp] = find((penal_nbr_cost==penal_best_nbr_cost),1);
   tabu_node1 = Rp;
```

```
  tabu_node2 = Cp;
  best_nbr_cost = nbr_cost(tabu_node1,tabu_node2);
end
for row = 1:dim1-1
  for col = row+1:dim1
    if tabu_tenure(row,col)>0
      tabu_tenure(row,col)=tabu_tenure(row,col)-1;
      tabu_tenure(col,row)=tabu_tenure(row,col);
    end
  end
end
```



UNIVERSITY OF LIVERPOOL

14 May, 2012.

C.A. Ogugua Eze
Mechanical Engineering Department
Nnamdi Azikiwe University
P.M.B 5025 Awka,
Nigeria.

Dear Sir.

Copyright Permission Letter

I am preparing my PhD thesis entitled "*Unified Metaheuristic and System-Theoretic Framework for Petroleum Reservoir Management*" for submission to the Faculty of Graduate Research at University of Liverpool, England. I am seeking your permission to include some parts of the manuscript version of the following paper(s) in the introductory chapter of the thesis.

ENERGY CONVERSION AND POWER GENERATION (MEC 546) – EZE C.A.O

Full reference details and a copy of this permission letter will be included in the thesis.

Yours sincerely,

[illegible]

NWANKWOR, Emeka Henry

Permission is granted for:

a) the inclusion of the material described above in your thesis.

b) for the material described above to be included in the copy of your thesis that is sent to the Library and Archives for reproduction and distribution.

Name:	ENGR. C.A.O. EZE	Title:	ENGR.
Signature:	[illegible]	Date:	July 23, '12

UNIVERSITY OF LIVERPOOL

03 December 2012.

Prof. Jan-Dirk Jansen
HOD Geoscience and Engineering Department
Faculty of Civil Engineering and Geosciences
Delft University of Technology
P.O. Box 5048
2600 GA Delft, The Netherlands.

Dear Sir.

Copyright Permission Letter

I am preparing my PhD thesis entitled "*A Unified Metaheuristic and System-Theoretic Framework for Petroleum Reservoir Management*" for submission to the Faculty of Graduate Research at University of Liverpool, England. I am seeking your permission to include some parts of **pages 166–169** of the manuscript version of the following lecture note(s) in Chapter 5 of my thesis.

Jansen J.D (2012): Systems Theory for Reservoir Management, Lecture Notes for Course AES1490, Version 6f, August 2012.

Full reference details and a copy of this permission letter will be included in the thesis.

Yours sincerely,

[illegible]

NWANKWOR, Emeka Henry

Permission is granted for:

a) the inclusion of the material described above in your thesis.

b) for the material described above to be included in the copy of your thesis that is sent to the Library and Archives for reproduction and distribution.

Name: J.D JANSEN Title: PROFESSOR

Signature: [illegible] Date: 5 DEC 2012