การออกแบบระบบการผลิตปิโตรเลียมที่เหมาะสมโดยใช้อัลกอริทึมทางพันธุกรรม

นายนิพนธ์ ตัญฑโญภิญ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมปิโตรเลียม ภาควิชาวิศวกรรมเหมืองแร่และปิโตรเลียม
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549

DESIGNING AN OPTIMIZED PETROLEUM PRODUCTION SYSTEM USING GENETIC ALGORITHM

Mr.Nipon Tantayopin

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Petroleum Engineering
Department of Mining and Petroleum Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2006

Thesis Title	**DESIGNING AN OPTIMIZED PETROLEUM PRODUCTION SYSTEM USING GENETIC ALGORITHM**
By	Nipon Tantayopin
Field of Study	Petroleum Engineering
Thesis Advisor	Suwat Athichanagorn, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

....Dean of the Faculty of Engineering
(Professor Direk Lavansiri, Ph.D.)

THESIS COMMITTEE

....Chairman
(Associate Professor Sarithdej Pathanasethpong)

....Thesis Advisor
(Assistant Professor Suwat Athichanagorn, Ph.D.)

....Member
(Jirawat Chewaroungroaj, Ph.D.)

นิพนธ์ ตัญฑไญภิญ: การออกแบบระบบการผลิตปิโตรเลียมที่เหมาะสมโดยใช้อัลกอริทึมทางพันธุกรรม (DESIGNING AN OPTIMIZED PETROLEUM PRODUCTION SYSTEM USING GENETIC ALGORITHM) อาจารย์ที่ปรึกษา ผศ.ดร.สุวัฒน์ อธิชนากร จำนวนหน้า 130 หน้า.

วิทยานิพนธ์ฉบับนี้อธิบายถึงวิธีการหามูลค่าปัจจุบันสุทธิที่เหมาะสมโดยการออกแบหลุมผลิตน้ำมันให้สมบูรณ์ การจัดตารางการผลิต และตารางอัดก๊าซเพื่อช่วยในการไหลของผลผลิต ในการออกแบบระบบการผลิตน้ำมันปัจจัยที่มีความสำคัญมากต่ออัตราการผลิตที่ดี ได้แก่ ขนาดของท่อผลิต ขนาดของโช้ค (choke) ขนาดของท่อส่งผลผลิต ความดันของเครื่องแยกน้ำมันและก๊าซ ปริมาณของก๊าซที่อัดเพื่อช่วยในการไหลของผลผลิต และจำนวนของหลุมผลิต

อัตราการผลิตของแหล่งผลิตสามารถคำนวณได้จากการสร้างแบบจำลองการผลิต ซึ่งประกอบด้วยส่วนย่อยต่างๆ ได้แก่ แบบจำลองการไหลของผลผลิต แบบจำลองการไหลในท่อผลิต แบบจำลองการไหลของผลผลิตผ่านโช้ค แบบจำลองการไหลในท่อขนส่ง แบบจำลองเครื่องแยกน้ำมันและก๊าซ

ปัจจัยที่มีความสำคัญต่ออัตราการผลิตที่มีการศึกษาในวิทยานิพนธ์ฉบับนี้ ได้แก่ ขนาดของท่อผลิต ขนาดของโช้ค (choke) ขนาดของท่อส่งผลผลิต ความดันของเครื่องแยกน้ำมันและก๊าซ ปริมาณของก๊าซที่อัดเพื่อช่วยในการไหลของผลผลิต และจำนวนของหลุมผลิต โดยปัจจัยบางชนิดเปลี่ยนแปลงตามเวลา อัตราการผลิตในแต่ละช่วงเวลาจะนำไปคำนวณหาค่าปัจจุบันสุทธิในแบบจำลองทางเศรษฐศาสตร์

การคำนวณหาค่าปัจจุบันสุทธิที่ดีที่สุด โดยการนำเทคนิคในการหาค่าที่เหมาะสมที่สุดมาคำนวณแทนที่การคำนวณหาค่าปัจจุบันสุทธิของชุดปัจจัยทุกชุด ซึ่งเทคนิคดังกล่าวมีชื่อเรียกว่าอัลกอริทึมทางพันธุกรรม

ในการศึกษาครั้งนี้ได้มีการทดลองการหาค่าปัจจัยการผลิตที่เหมาะสม จำนวน 3 กรณี เพื่อศึกษาประสิทธิภาพของอัลกอริทึมทางพันธุกรรมและอิทธิพลของแต่ละปัจจัยต่อมูลค่าปัจจุบันสุทธิ

ภาควิชาวิศวกรรมเหมืองแร่และปิโตรเลียม ลายมือชื่อนิสิต..........................

สาขาวิชาวิศวกรรมปิโตรเลียม ลายมือชื่ออาจารย์ที่ปรึกษา..........................

ปีการศึกษา 2549

##467 16067 21: MAJOR PETROLEUM ENGINEERING
KEY WORD: PETROLEUM PRODUCTION OPTIMIZATION/ GENETIC ALGORITHM

NIPON TANATAYOPIN. THESIS TITLE: DESIGNING AN OPTIMIZED PETROLEUM PRODUCTION SYSTEM USING GENETIC ALGORITHM. THESIS ADVISOR: SUWAT ATHICHANAGORN, Ph.D. 130 pp.

This report describes techniques developed to optimize net present value by designing completion, schedule of production and amount of gas injection for gas lift purposes. In the design of a production system, the determination of production parameters such as tubing diameter, choke diameter, pipeline diameter, separator pressures, volume of gas injected and number of wells are crucial in obtaining the optimal economic value of a project.

The production profile of a reservoir can be predicted by integrating reservoir model, wellbore flow model, choke model, flowline model and separator model.

Production profile changes with different sets of completion and production parameters including gas-lift configuration. The parameters that affect production rate are tubing diameter, choke diameter, pipeline diameter, separator pressures, volume of gas injected and number of wells. Some of these parameters may vary with time. After the production profile is obtained, NPV is calculated in the economic model.

To find the maximum net present value, instead of calculating all sets of production parameters, nonlinear optimization technique is used in order to reduce computation time. Genetic algorithm has been chosen for this project.

Three case studies with different reservoir and economic conditions were performed to see the effectiveness of genetic algorithm in finding the solution and the effect of each parameter on NPV.

Department of Mining and Petroleum Engineering Student's signature ....
Field of study: Petroleum Engineering Advisor's signature ....
Academic year: 2006

Acknowledgement

I would like to thank Asst. Prof. Dr. Suwat Atichanagorn for his guidance and advice with full of patience.

Many of my friends have supported me with data and theory description, thanks to Akarak Suwannarak and Suksombut for their help in Genetic Algorithm, Chatwit Pochan for drilling and completion data, Suppanai and Rossawon for their sincere help. Moreover, thanks to Schlumberger for providing ECLIPSE reservoir simulation program to the department of Mining and Petroleum Engineering as the software was used to run result, for reference.

I appreciate the understanding of my boss, David Kendall (Baker Atlas Songkhla base operation manager) , and my colleagues in Baker Atlas Thailand.

Finally, I would like to express my thanks to my parents, Noppon, Pornteera and Thitima Sudsatha for their encouragement and support.

Contents

List of Figures

Figure **Page**

Figure **Page**

List of Tables

Nomenclature

a_i	Redlich-Kwong parameter, of component *i*.
a_{ij}	Redlich-Kwong cross parameter, for components *i* and *j*.
a_m	Redlich-Kwong parameter, mixture *z*.
A	Constant used in cubic root solution; Areal extent of reservoir, sq. ft.
A_c	Cross-sectional area of choke
b_i	Redlich-Kwong parameter, of component *i*.
b_m	Redlich-Kwong parameter, mixture *Z*.
B	Constant used in cubic root solution
C	Choke discharge coefficient
C_D	Choke discharge coefficient
C_L	No-slip in-situ volume fraction of liquid and specific heat of liquid
C_p	Specific heat of gas at constant pressure
C_v	Specific heat of gas at constant volume
D	Tubing diameter, ft
D_2	Choke diameter, inches
ΔL	Depth change, ft
Δt	Time step length, days
Δp_H	Hydrostatic pressure change over length, psia
Δp_F	Frictional pressure change over length, psia
E_L	In-situ volume fraction of liquid, considering holdup
f_f	Fanning friction factor
f_{ip}	Partial fugacity of component *i* in phase *p*
F	Critical/subcritical boundary function; Objective function
g	Gravitational acceleration, 32.2 ft / second^2
g_c	Gravitational constant, 32.2 lbm-ft/(lbf-second^2)
h	Reservoir thickness, ft
k	Reservoir permeability, md, and specific heat
k_{rp}	Relative permeability of phase *p*
K	Equilibrium ratio
K_i	Equilibrium ratio of component *i* in mixture *Zz* (y_i/x_i)
L	Liquid phase mole fraction, and depth, ft

M	Choke mass rate, lbm/day
M_p	Phase *p* molecular weight, lbm/lb. mole
n	Number of components in mixture, and Polytropic exponent for gas
N	Total reservoir mass flow rate, lb.mole/day
N_k	Total reservoir mass, lb. mol
$N_{k,i}$	Reservoir mass of component *i* at time step *k*, lb. mol
n_p	Phase *p* relative permeability exponent
N_p	Reservoir mass rate of phase *p*, lb. mol/day
p	Constant for cubic root solving, and pressure of interest, psia
p_{ci}	Critical pressure of component *i*, psia
p_{ri}	Reduced pressure of component *i*
p_{wf}	Well flowing pressure, psia
q	Constant for cubic root solving; Volumetric flow rate, bbl/day
q_p	Volumetric flow rate, bbl/day
r	Constant for cubic root solving
R	Universal gas constant
r_e	Radius of reservoir, ft
Re	Reynold's number
r_w	Radius of wellbore, ft
S_p	Phase *p* saturation
S_{pr}	Residual phase *p* saturation
T	Temperature of interest, °R
T_{ci}	Critical temperature of component *i*, °R
T_{ri}	Reduced temperature of component *i*
V	Gas phase mole fraction; Specific volume, cu ft / lb
V_b	Bubble rise velocity, ft / second
V_M	Mixture velocity, ft/second
V_{Msp}	Modified superficial velocity of phase *p*, ft/second
V_{sp}	Superficial velocity of phase *p*, ft/second
V_t	Total gas rise velocity, ft/second
x	Composition of liquid phase of mixture *z*
xi	Mole fraction of component *i* in liquid phase *x*
x_1, x_2, x_3	Solutions of cubic root

y	Composition of gas phase of mixture z
y_i	Mole fraction of component i in gas phase y
y	Composition of gas phase of mixture z and ratio of upstream to downstream pressure
y_c	Critical ratio of upstream to downstream pressure
y_u	Applied ratio of upstream to downstream pressure
z	Composition of mixture, array, see zi
z_i	Mole fraction of component i in mixture z
z_p	Composition of produced fluid
z	Gas compressibility factor

Subscripts & Superscripts

1	Upstream; Separator 1
2	Downstream; Separator 2
3	Separator 3
atm	*Atmosphere*
c	Critical
F	Frictional
g	Gas phase
G	Gas phase
H	Hydrostatic
i	Component number, out of *n* components
inj	Injection
k	Time step index; Optimization step index
L	Liquid phase
M	Mixture; modified
o	Oil phase
p	Phase
r	Reservoir
s	Superficial
sepn	Separator number *n*
st	Stock tank

t	Tubing
u	Applied value
wh	Wellhead
wf	Sandface flowing

Greek Letters

α	Constant used in cubic root solving
β	Constant used in cubic root solving
ε	Tolerance; Absolute pipe roughness, inches
ϕ	Porosity
$\hat{\phi}_i^p$	Partial fugacity constant for component *i*, in phase *p*
μ_p	Viscosity of phase *p*, cp
ω	Acentric factor
ρ_p	Phase *p* density, lbm/cu ft
σ	Interfacial surface tension between oil and gas, dyne/cm
$\sigma_{p1,p2}$	Interfacial surface tension between phase 1 and phase 2, dyne/cm

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

Economic is the most important decision factor in petroleum industry. Therefore the way to make the project to be the most economic should be studied. This study net present value (NPV) is defined to be objective function. The method to find set of completion and production parameters that give optimum net present value is demonstrated in this study.

After the exploration phase, drilling, completion and production phase are performed to get the hydrocarbon production. The main idea of this study is to answer the question of "how to get maximum profit?"

Production profile is one of the most important parameters for predicting the project income. Besides reservoir properties, factors that affect production profile are tubing size, choke configuration, pipeline size and first separator pressure. Moreover, after a period of production, the reservoir pressure is low; and ability to drive liquid hydrocarbon is also low. Gas lift is one of artificial methods to help production flow to surface.

In some cases, if the reservoir is large or the oil price is high enough, it is worth to produce with multiple wells.

Once reservoir properties, drilling, completion factors, number of wells and production configurations are known, the production profile can be determined. The problem is what the best sizes and configurations of these factors are.

To answer that question, economic calculation is performed with production profiles from various sets of decision variables to get net present values, NPV.

Income is from oil and gas sale while cost is the combination of drilling, completion, production facility and operation costs.

To find the maximum net present value, instead of calculating all sets of production parameters, nonlinear optimization technique is used. Genetic algorithm which is proven to be the most suitable for petroleum industry [1] has been chosen for this project. The answer from optimization algorithm is the set of tubing diameter, choke configuration, pipeline diameter, first separator pressure, gas injection rate and number of wells that makes maximum profit.

1.2 Literature review

Production optimization has been studied for a long time by many engineers. Following studies give the guide line to this thesis. In 1990, Carroll [1] studied on production optimization by using Newton's Method, modified Newton's Method with Cholesky factorization, and the polytope heuristic. The production system consists of a reservoir with single production well. The flow performance is determined material balance equation. Only in separator model is calculated with compositional model. In his study, total production rate is the objective function. His study gives an idea of production optimization by designing completion parameters. In 1993, Fujii [2] constructed multiple-well-production model to maximize one of these followings in each case: total production rate, net income and net present value. His production model was calculated by using inflow and outflow performance relationships. In his study, three optimization methods were used; Newton-type methods, the polytope method and genetic algorithms. An idea of production optimization of multiple production wells are from his study. In 1996, nonlinear optimization of well considering gas lift and phase behaviour was studied by Palke [3]. His study was to optimized net present value of a single oil production well with gas lift. Composition phase calculation was used for phase behaviour. Newton-type methods, the polytope method and genetic algorithm were tested to find which one is the best optimization method for petroleum industry. In the study of Palke gives guild line of NPV optimization using genetic algorithm.

CHAPTER II

THEORIES AND CONCEPTS

2.1 Model Description

In order to calculate the net present value, NPV, production profiles need to be determined. Simulation program for obtaining production profile is constructed for this study. Details of the simulation program are described in the following sections.

The field model constructed is an integration of smaller components. The complete model represents a reservoir with optimal multiple gas-lifted wells. The smaller model components include:

- Fluid properties
- Reservoir model
- Well model with gas lift
- Choke model
- Pipeline model
- Separator model

To integrate the small model components, the production path needs to be described first. Starting at the reservoir, produced fluid flows into the tubing. At the point where the lift gas enters the tubing, the two streams combine, and the flow continues up the wellbore. At the surface, the combined fluid passes through the choke into the flow line and the first separator. This process is shown in Figure 2.1. In case that there is more than one well, drainage area is calculated by equally averaging the reservoir area. Each well produces only from its own area. The reservoir pressure for every well is assumed to be the same at the average reservoir pressure. All the production fluids pass through flow lines to the first of the three separators. Gas from the first separator goes into the gas line, and the liquid phase moves into the second separator. Gas from the second separator passes into the gas line, and the liquid goes into the third separator. The gas in the third separator goes into the gas line, and the liquid goes into stock tanks to be sold. Some of the separated gas is compressed, and injected into the tubing-casing annulus for gas lift. The remainder of the gas is sold. Figure 2.2 shows the surface production path.

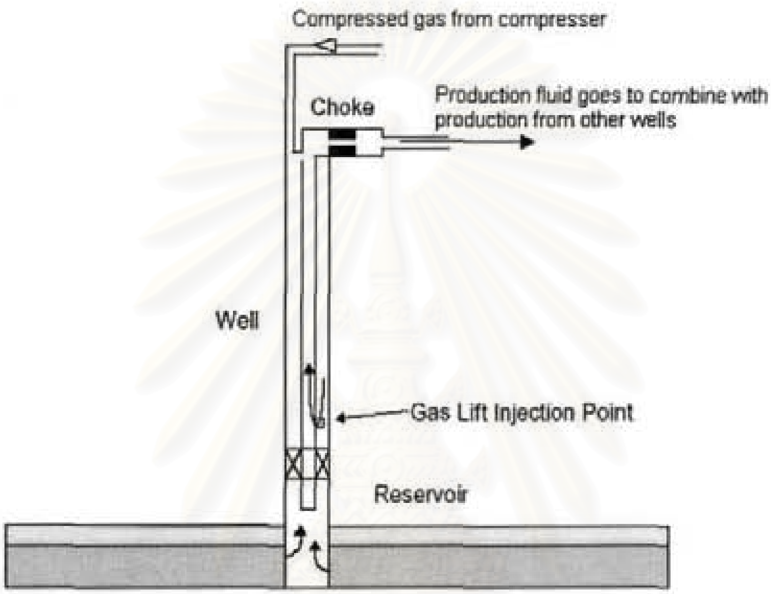

Figure 2.1: A diagram illustrating fluid flow from the reservoir to the wellhead and compressed gas injection for gas lift.

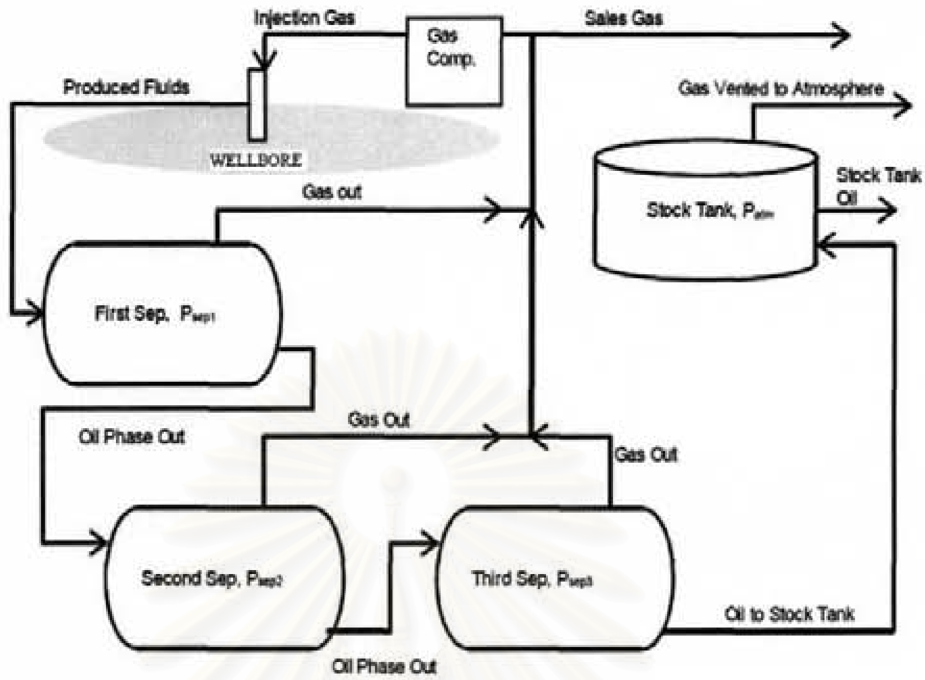

Figure 2.2: A flow diagram of fluid from production wells flowing into separators.

2.2 Fluid properties

In this study, fluid properties are calculated based on fluid composition. These fluid properties are used in the calculation of fluid flow in the reservoir model, tubing model, choke model, flow line model and separator model. Not only pressure and temperature affect fluid properties, but fluid composition also does. Therefore, compositional calculation is suitable for this study. By using the Redlich-Kwong Equation of State, empirical black-oil phase behaviour correlations are not needed.

Vapour phase density

Vapour density can be determined based on the Equation of State of Redlich-Kwong. In order to do Equation of State calculation, flash calculation is also required to determine for its parameters.

Flash Equilibria

Fluid properties vary with pressure and temperature. A flash calculation takes the composition of a mixture and calculates the resulting phase equilibria at a new temperature and pressure, such as the number of phases present and amount of each phase. The flash calculation is iterative and converges when the fugacity of each component is the same in both phases. The basic procedure of a flash calculation is shown in Figure 2.3 and summarized as follows:

1) Given the composition of the mixture, z_i, at a temperature, T, and pressure, p, calculate initial K-factor, by using Wilson Equation [4].

2) Perform Flash calculation to get liquid fraction, vapour fraction, liquid phase composition, x_i, and vapour phase composition, y_i.

3) Calculate the equation of state parameters.

4) Solve the equation of state for vapour phase density.

5) Determine the partial fugacity of the components in each phase to verify whether it is in equilibrium.

6) If the fugacity ratio has not converged to one for each component, then update the Equilibrium Ratio with partial fugacity and proceed with step 2.

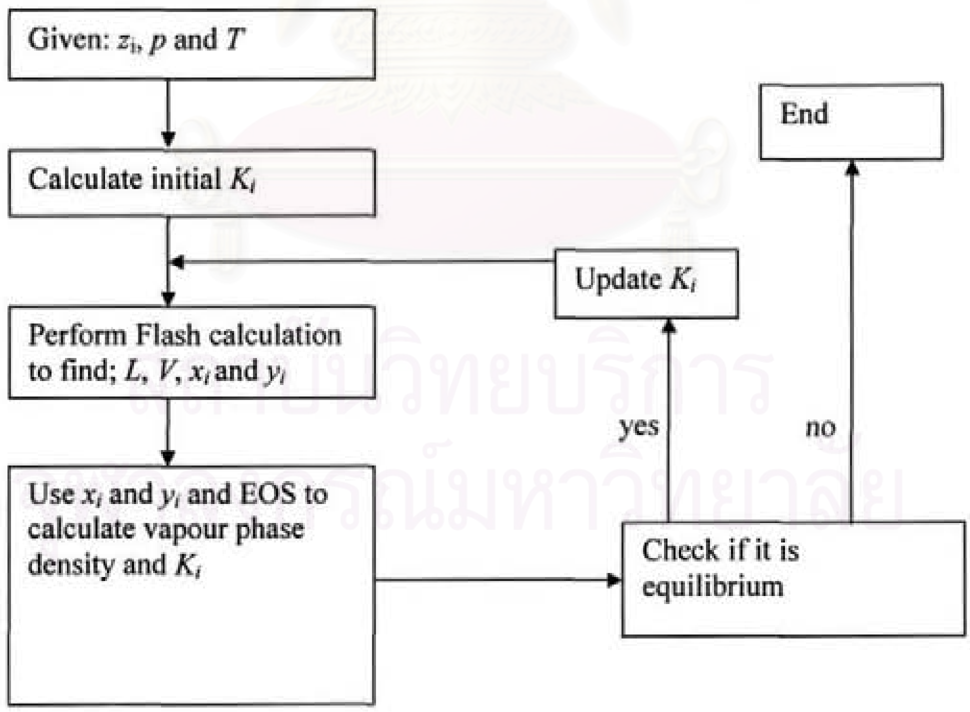

Figure 2.3: Diagram of vapour phase calculation.

The ratio of the vapour mole fraction to the liquid mole fraction for a given component is known as the equilibrium ratio, or alternatively as the K-value, and is defined as:

$$K = \frac{y}{x} \tag{2.1}$$

Empirical correlations can be used to provide an initial estimate of the equilibrium ratios. The Wilson Equation was used in this model.

$$K_i = \frac{e^{[5.37(\omega_i)(1-\frac{1}{T_{ri}})]}}{P_{ri}} \tag{2.2}$$

where ω_i *is* accentric factor,

$$p_{ri} = \frac{p}{p_{ci}} \tag{2.3}$$

$$T_{ri} = \frac{T}{T_{ci}} \tag{2.4}$$

The values of T_{ci} and p_{ci} for C1, C2, C3…C6 are constant but those for C7+ are

$$p_{ci} = \exp\left\{\begin{array}{l} 8.3634-(0.0566/\gamma_{C7+})-\left[(0.24244+(2.2898/\gamma_{C7+})+(0.11875/\gamma_{C7+}^{\ 2}))10^{3}\mathrm{T_B}\right] \\ +\left[1.4685+(3.648/\gamma_{C7+})+(0.47227/\gamma_{C7+}^{\ 2})\right]10^{-7}\mathrm{T_B}^2 \\ -\left[0.42019+(1.6977/\gamma_{C7+}^{\ 2}))\right]10^{-10}\mathrm{T_B}^3 \end{array}\right\} \tag{2.5}$$

$$T_{ci} = 341.7 + 811\gamma_{C7+} + (0.4244 + 0.1174\gamma_{C7+})\mathrm{T_B} + \frac{(0.4669 - 3.2623\gamma_{C7+})10^5}{\mathrm{T_B}} \tag{2.6}$$

$$T_B = \left[4.5579\mathrm{M}_{c7+}^{\ 0.15178}\ \gamma_{C7+}^{0.15427}\ ))\right]^3 \tag{2.7}$$

This initial K-factor is used only for the first iteration of flash calculations. After the first iteration, the K-factor will be updated with the EOS.

Performing a material balance on each component, we know that

$$z_i = Lx_i + Vy_i \tag{2.8}$$

where V and L are the vapour and liquid mole fractions, respectively, and $L = 1 - V$. Using the relation $y_i = K_i / x_i$ and solving for x_i yields

$$x_i = \frac{z_i}{L + (1-L)K_i} \tag{2.9}$$

And letting $x_i = K_i / y_i$ and solving for y_i yields

$$x_i = \frac{z_i}{L + (1-L)K_i} \tag{2.10}$$

From $\sum x_i = \sum y_i = \sum z_i = 1$ (2.11)

So, $\sum X_i - \sum Y_i = 0 = \sum_{i=1}^{n} \frac{Z_i(1-K_i)}{L+(1-L)K_i}$ (2.12)

Liquid fraction could be determined from solving the root of the function

$$F(L_k) = \sum_{i=1}^{n} \frac{Z_i(1-K_i)}{L+(1-L)K_i} = 0 \tag{2.13}$$

This equation can be efficiently solved with Newton-Raphson iteration where

$$L_{k+1} = L_k - \frac{F(L_k)}{\left.\frac{\partial F}{\partial L}\right|_{L_k}} \tag{2.14}$$

and

$$L_{k+1} = L_k - \frac{F(L_k)}{\left.\frac{\partial F}{\partial L}\right|_{L_k}} \tag{2.14}$$

and

$$\frac{\partial F}{\partial L} = -\sum_{i=1}^{n} \frac{z_i (1 - K_i)^2}{(K_i + (1 - K_i)L)^2} \tag{2.15}$$

Once L is determined, the compositions of the liquid and vapour phases are obtained.

Equation of State

The EOS used in this study is Redlich and Kwong [5] equation. The standard form is

$$P = \frac{RT}{V - b_m} - \frac{a_m}{\sqrt{T}V(V + b_m)} \tag{2.16}$$

where

$$a_m = \sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j a_{ij} \tag{2.17}$$

$$b_m = \sum_{i=1}^{n} Y_i b_i \tag{2.18}$$

$$a_{ij} = \sqrt{a_i a_j} \tag{2.19}$$

$$a_i = \frac{0.42748 R^2 T_{ci}^{2.5}}{P_{ci}} \tag{2.20}$$

$$b_i = 0.08664 \frac{RT_{ci}}{P_{ci}} \tag{2.21}$$

$i = C_1, C_2, C_3, ... C_6, and C_{7+}$

$j = C_1, C_2, C_3, ... C_6, and C_{7+}$

With known pressure and temperature, it is more convenient to write the EOS in cubic form:

$$V^3 - \frac{RTV^2}{P} + \frac{1}{P}(\frac{a_m}{\sqrt{T}} - b_m RT - Pb_m^2)V - \frac{a_m b_m}{p\sqrt{T}} = 0 \quad (2.22)$$

In this form EOS can be solved for z by substituting

$$V = \frac{zRT}{P} \quad (2.23)$$

in Equation 2.22

$$(\frac{zRT}{P})^3 - z^2(\frac{RT}{P})^3 + \frac{zRT}{P^2}((\frac{a_m}{\sqrt{T}} - b_m RT - Pb_m^2) - \frac{a_m b_m}{p\sqrt{T}}) = 0 \quad (2.24)$$

By solving for the z factor, if there are 3 real roots, the maximum answer is the vapour phase z-factor.

Then, $V = \frac{n}{M}\frac{zRT}{P}$ (2.25)

where n = number of mole

M = molecular weight

At this point vapour phase density is obtained as

$$\rho_g = \frac{pM}{zRT} \quad (2.26)$$

Partial Fugacity

For the Redlich-Kwong equation of state, the partial fugacity of each component is given by

$$\hat{f}_i = P\exp\left\{\frac{b_i}{b_m}(z-1) - \ln\left[z(1-\frac{b_m}{V})\right] + \frac{1}{b_m RT^{1.5}}\left[\frac{a_m b_i}{b_m} - 2\sqrt{a_m a_i}\right]\ln(1+\frac{b_m}{V})\right\} \tag{2.27}$$

The partial fugacity represents the chemical potential of each component at a given thermodynamic state. When the partial fugacity is equal in each phase, for each component, thermodynamic equilibrium has been reached.

$$\frac{\hat{f}_i^v}{\hat{f}_i^L} = 1 \tag{2.28}$$

where $\hat{f}_i^v$ = Partial fugacity of component i in vapour phase

$\hat{f}_i^L$ = Partial fugacity of component i in liquid phase

Since the algorithm is iterative, an exact solution is difficult. The convergence criterion used in this study was whether the fugacity ratio for each component was within a tolerance of one:

$$abs(\frac{\hat{f}_i^v}{\hat{f}_i^L}) \leq 1 + \varepsilon \tag{2.29}$$

where ε is tolerance, 10^{-3}

If the process has not converged, the K_i values are updated with the following relationship:

$$K_i^{k+1} = \frac{\hat{f}_i^L}{\hat{f}_i^v} K_i^k \tag{2.30}$$

In order to update the K-factor, dimensionless partial fugacity coefficient is required for successive iterations.

$$\hat{\phi}_i^v = \frac{\hat{f}_i^v}{y_i p} \tag{2.31}$$

$$\hat{\phi}_i^L = \frac{\hat{f}_i^L}{x_i p} \tag{2.32}$$

$$\frac{\hat{f}_i^v}{\hat{f}_i^L} = \frac{\hat{\phi}_i^v}{\hat{\phi}_i^L} \tag{2.33}$$

$$K_i^{k+1} = \frac{\hat{f}_i^l}{\hat{f}_i^v} K_i^k \tag{2.34}$$

$$K_i^{k+1} = \frac{\hat{\phi}_i^l}{\hat{\phi}_i^v} \tag{2.35}$$

$$\ln \hat{\phi}_i = \frac{b_i}{b_m}(z-1) - \ln\left[z(1-\frac{b_m}{V})\right] + \frac{1}{b_m RT^{1.5}}\left[\frac{a_m b_i}{b_m} - 2\sqrt{a_m a_i}\right]\ln(1+\frac{b_m}{V}) \tag{2.36}$$

where $\hat{\phi}_i^L$ is Partial fugacity constant for component i of vapour phase

$\hat{\phi}_i^V$ is Partial fugacity constant for component i of liquid phase

The z-factor can be obtained by solving equation 2.24. The minimum root is liquid phase z-factor; while the maximum root is vapour phase z-factor.

At this point, the flash calculations are repeated using the updated K_i values.

Liquid phase density

With know liquid composition, liquid density at each specific pressure and temperature can be determined by using empirical method of McCain [(6)]. The concept of this empirical method is to find liquid density at standard condition and correct it by pressure and temperature correction. The calculation procedure is:

1) Assume initial density, $\rho_{initail}$

2) Find the density at standard conditions

$$\rho_{sta} = \frac{\sum(x_i M_i)}{\sum V_{L,sc}} \tag{2.37}$$

where $V_{L,sc}$ is liquid volume at standard condition.

$$\sum V_{L,sc} = \sum \left(\frac{x_i M_i}{\rho_{sc,i}}\right) \tag{2.38}$$

where $\rho_{sc,i}$ is liquid density at standard condition,

$$\rho_{sc,1} = 0.312 + 0.45\,\rho_{initial} \tag{2.39}$$

$$\rho_{sc,2} = 15.3 + 0.3167\,\rho_{initial} \tag{2.40}$$

3) Compare new the density to the initial density

- If the difference is significant, go back to step 2 and use the new density as initial density.

4) Calculate pressure and temperature correction as follows:

$$\Delta\rho_p = (0.167 + 16.181(10^{-0.0425\rho}))\left(\frac{p}{1000}\right) - 0.01(0.299 + 263(10^{-0.0603\rho}))\left(\frac{p}{1000}\right)^2 \tag{2.41}$$

$$\Delta\rho_T = (0.0032 + 1.505\rho^{-0.951})(T-60)^{0.938} - (0.0216 - 0.0233(10^{-0.0161\rho}))(T-60)^{0.475} \tag{2.42}$$

5) Compute the correct density

$$\text{Liquid density} = \rho_{sta} + \Delta\rho_p - \Delta\rho_T \tag{2.43}$$

Viscosity

The following equation may be used to calculated oil viscosity when its °API gravity is between 5 and 58 °API.

$$\mu_o = 10\text{^}(10\text{^}(1.8653 - 0.025086°API - 0.5644\log(T + 460))) - 1 \tag{2.44}$$

$$°API = \frac{141.5}{\gamma_o} - 131.5 \tag{2.45}$$

For gas, viscosity is calculated by following equations:

$$\mu_g = A(10^{-4})\exp(0.01602B\rho_g{}^{C}) \tag{2.46}$$

$$A = \frac{(9.379 + 0.01607M_g)T^{1.5}}{209.2 + 19.26M_g + T} \tag{2.47}$$

$$B = 3.448 + \frac{986.4}{T} + 0.01009M_g \tag{2.48}$$

$$C = 2.447 - 0.2224B \tag{2.49}$$

where M_g is gas phase molecular weight

2.3 Reservoir model

Some of important assumptions concerning the reservoir model used in this study are:

- The reservoir is homogeneous, isotropic, horizontal, cylindrical, and of uniform thickness.
- Reservoir fluid is still homogenous even it is produced by multiple wells.
- The reservoir is a zero-dimensional single cell that is bounded by no-flow boundaries.
- Production occurs under pseudo-steady state conditions and at a constant rate.
- Capillary pressure, gravity effects, and coning are negligible.
- There is no aqueous phase, and the rock phase is incompressible.
- Damages of reservoir due to drilling and completion are neglected.

Flow rate

The purpose of this section is to describe the procedure to find production flow rate and update reservoir condition at times of producing. Flow rate mostly depends on fluid behaviour and also well flowing pressure which is related to outflow performance, from tubing to separator, which is described in subsequence sections. The basic procedure of flow rate calculation is described as follows:

1) Begin with an average reservoir pressure at time step k, $p_{res,\,k}$, a total reservoir composition z_{res}, and an initial guess of p_{wf}.

2) Estimate reservoir pressure at time step k+1, $p_{res,\ k+1}$ and calculate average reservoir pressure $p_{res,\ k+1/2} = ½(p_{res,k} + p_{res,k+1})$

3) Perform flash calculations to determine fluid properties, mass of reservoir fluid, N_k, and phase compositions, z, for the mixture at the average reservoir pressure.

4) Based on these sandface phase properties, determine S_o and S_g. From these values, determine k_{ro} and k_{rg}. Fluid properties in the flow rate equation are calculated at the average reservoir pressure.

5) Determine q_g and q_o. Use the properties at average reservoir pressure to determine mass flow rates for each phase and composition of the fluids.

6) Calculate new mass of reservoir fluid, N_{k+1}, new reservoir fluid composition, z_{res}, and new reservoir pressure, p_{res}.

7) Check if $p_{res,\ k+1}$ change much from the guessed value

- If yes, adjust $p_{res,\ k+1}$ and repeat step 3-6.
- If not, then proceed on.

8) Go to pipeline model, choke model and tubing model to calculate p_{wf} based on the same production rate and composition.

9) Check if p_{wf} from inflow and outflow are similar.

- if not, adjust p_{wf} and go back to step 2.
- if they merge, the computed production rate and composition are used for economic model, and update reservoir fluid mass, reservoir fluid composition and reservoir pressure.

The main procedure of the reservoir model is shown in Figure 2.4 as a flow chart.

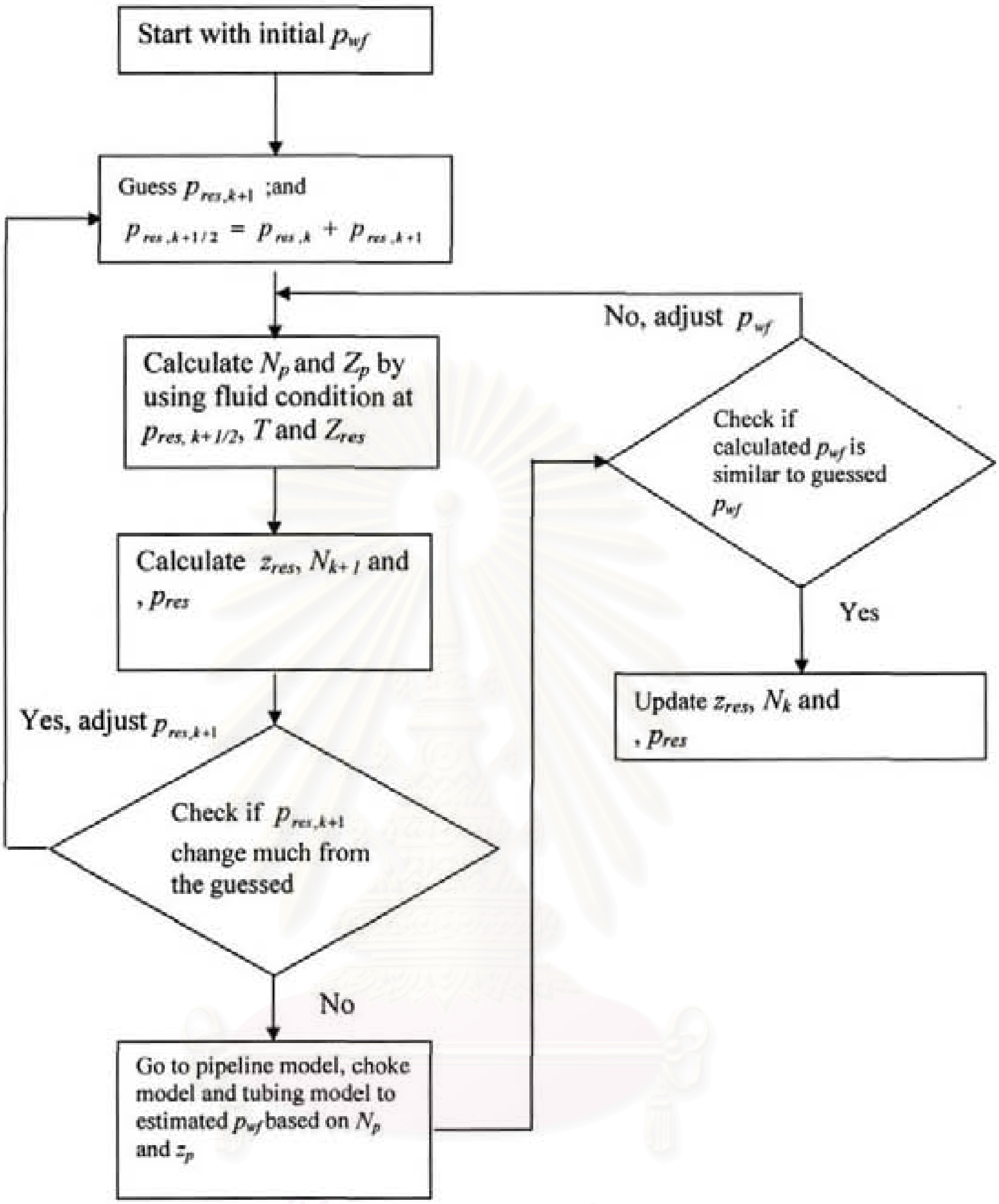

Figure 2.4: Flow Chart of reservoir model calculation.

Production rate in this study is based on this volumetric pseudo-steady state equation:

$$q = \frac{0.00708kk_rh}{\mu}\left(\frac{\overline{p} - p_{wf}}{\ln(r_e/r_w) - 0.75}\right) \qquad (2.50)$$

where q = flow rate, BBL/day

k = reservoir permeability, md

k_{rp} = relative permeability of phase p

h = reservoir thickness, ft

r_e = radius of reservoir, ft

r_w = rRadius of wellbore, ft

Flow rate in liquid phase and vapour phase

The flow rate is calculated separately for the liquid phase and vapour phase. These are flow rate equations for liquid phase and vapour phase, respectively:

$$q_o = \frac{0.00708 k k_{ro} h}{\mu_o}\left(\frac{\bar{p} - p_{wf}}{\ln(r_e / r_w) - 0.75}\right) \tag{2.51}$$

$$q_g = \frac{0.00708 k k_{rg} h}{\mu_g}\left(\frac{\bar{p} - p_{wf}}{\ln(r_e / r_w) - 0.75}\right) \tag{2.52}$$

Determining flow rate equation parameters

The relative permeabilities can be found from the equations:

$$k_{ro} = \left(\frac{(S_o - S_{or})}{(1 - S_{or} - S_{gr})}\right)^{n_{oil}} \tag{2.53}$$

$$k_{rg} = \left(\frac{(S_g - S_{gr})}{(1 - S_{or} - S_{gr})}\right)^{n_{gas}} \tag{2.54}$$

where S_o is liquid phase saturation

S_g is vapour phase saturation

S_{or} is residual liquid phase saturation

S_{gr} is residual vapour phase saturation

The saturation values depend on the specific volume of each phase and liquid mole fraction, L.

$$S_o = \frac{\dfrac{M_o L}{\rho_o}}{\dfrac{M_o L}{\rho_o} + \dfrac{M_g (1-L)}{\rho_g}} \tag{2.55}$$

where

$$M_o = \sum_{i=1}^{n} M_i z_i x_i \tag{2.56}$$

$$M_g = \sum_{i=1}^{n} M_i z_i y_i \tag{2.57}$$

$$S_g = 1 - S_o \tag{2.58}$$

Mass flow rate

However, from Equation 2.50 shows the flow rate in unit of barrel per day which is inconvenient to update the amount of fluid and fluid composition in the reservoir. The flow rate is converted to unit of mole per day, called mass flow rate. Mass flow rate can be determined by these following equations:

$$N_{po} = 5.615 q_o \rho_o / M_o \tag{2.59}$$

$$N_{pg} = 5.615 q_g \rho_g / M_g \tag{2.60}$$

where N_{po} is oil mass flow rate, mole/day

N_{pg} is gas mass flow rate, mole/day

The total mass flow rate is the sum of mass flow rates of liquid and vapour.

$$N_p = N_{po} + N_{pg} \tag{2.61}$$

where N_p is sum mass flow rate, mole/day

Production composition

By calculating the mass flow rate and performing flash calculation at the well flowing pressure, the production composition can be determined by the equation:

$$z_{p,i} = \frac{N_{po} x_i + N_{pg} y_i}{N_p} \tag{2.62}$$

Reservoir mass and composition

After the reservoir has been on production, time steps are made. At each time step, the total reservoir fluid mass is old reservoir fluid mass deducted by produced fluid mass:

$$N_{k+1} = N_k - N_p \Delta t \tag{2.63}$$

where N_{k+1} is fluid mole in reservoir at time step k+1

N_k is fluid mole in reservoir at time step k

The mass of each component in the reservoir is

$$N_{k+1,i} = N_{k,i} - N_p z_{p,i} \Delta t \tag{2.64}$$

New reservoir fluid composition is

$$Z_{res,i} = \frac{N_{k+1,i}}{N_{k+1}} \tag{2.65}$$

Updated reservoir pressure

To update the reservoir pressure, iterations are needs as explained in the following procedure:

1. Calculate reservoir density by

$$\rho_{res} = \frac{N_k M_{res}}{Ah\phi} \tag{2.66}$$

2. Iterative on the pressure in order to calculate vapour phase density, ρ_V by performing flash calculation and using Equation of State. Then, calculate liquid phase density, ρ_L at the pressure that yields convergence. The mixed density is

$$\rho_{mix} = \rho_L L + \rho_V (1 - L) \tag{2.67}$$

3. Check if the density in step 1 and 2 are similar.
   - If yes, the pressure in step 2 is the new reservoir pressure.
   - If no, change the pressure and go back to step 2.

2.4 Wellbore model

Hydrocarbon in the reservoir flows into production line starting at tubing. Production tubing in this study is vertical and one size. At some depth between the wellhead and reservoir depth, there is an injection port for gas lift injection. In order to describe oil and gas flow along the tubing, multiphase flow correlation is needed. In this study Aziz, Govier and Fogarasi multiphase flow correlation [7] is used.

Pressure along the wellbore is calculated by the correlation while temperature is assumed to vary linearly along the wellbore. The calculation procedure along the wellbore is traverse to direction of fluid flow. Starting at the wellhead, at the point before the fluid reaches the choke, the fluid flow is calculated back along the wellbore to the reservoir depth. The basic procedure is described as follows:

1) Based on the pressure at depth L, p_L, assume a downstream pressure at a given change of depth, ΔL. The initial guess for $p_{L+\Delta L}$ can be p_L.

2) Find the average pressure along the calculation depth

$$p_{L+\frac{1}{2}\Delta L} = \frac{1}{2}(p_L + p_{L+\Delta L}) \tag{2.68}$$

3) Flash the flowing mixture at $p_{L+\frac{1}{2}\Delta L}$ and $T_{L+\frac{1}{2}\Delta L}$ to calculate the no-slip properties and compositions of the phases in this step.

4) Use the multiphase flow correlation (AGF in this case) to determine the flow regime, liquid holdup, frictional pressure loss, and hydrostatic head over this pressure step.

5) Use the output of the multiphase flow correlation and p_L to determine $p_{L+\Delta L}$.

- If $p_{L+\Delta L}$ has changed from the initial guess, this step has not converged. Return to Step 2.
- If it has not changed significantly, this step has converged.

6) Set p_L equal to $p_{L+\Delta L}$. Assume now that $p_{L+\Delta L} = p_L + (p_L - p_{L-\Delta L})$ and return to Step 2.

This process is repeated until the ultimate depth of interest is reached. For this project, this calculation is performed twice for each determination of bottom hole pressure. First, it is used to determine the pressure at the point of gas injection, based on the surface pressure and the mixture of produced fluid and lift gas. Second, this technique is used to determine the sandface pressure based on the injection-point pressure and produced fluid rate and composition.

The flow chart of the tubing model calculation is shown in Figure 2.5.

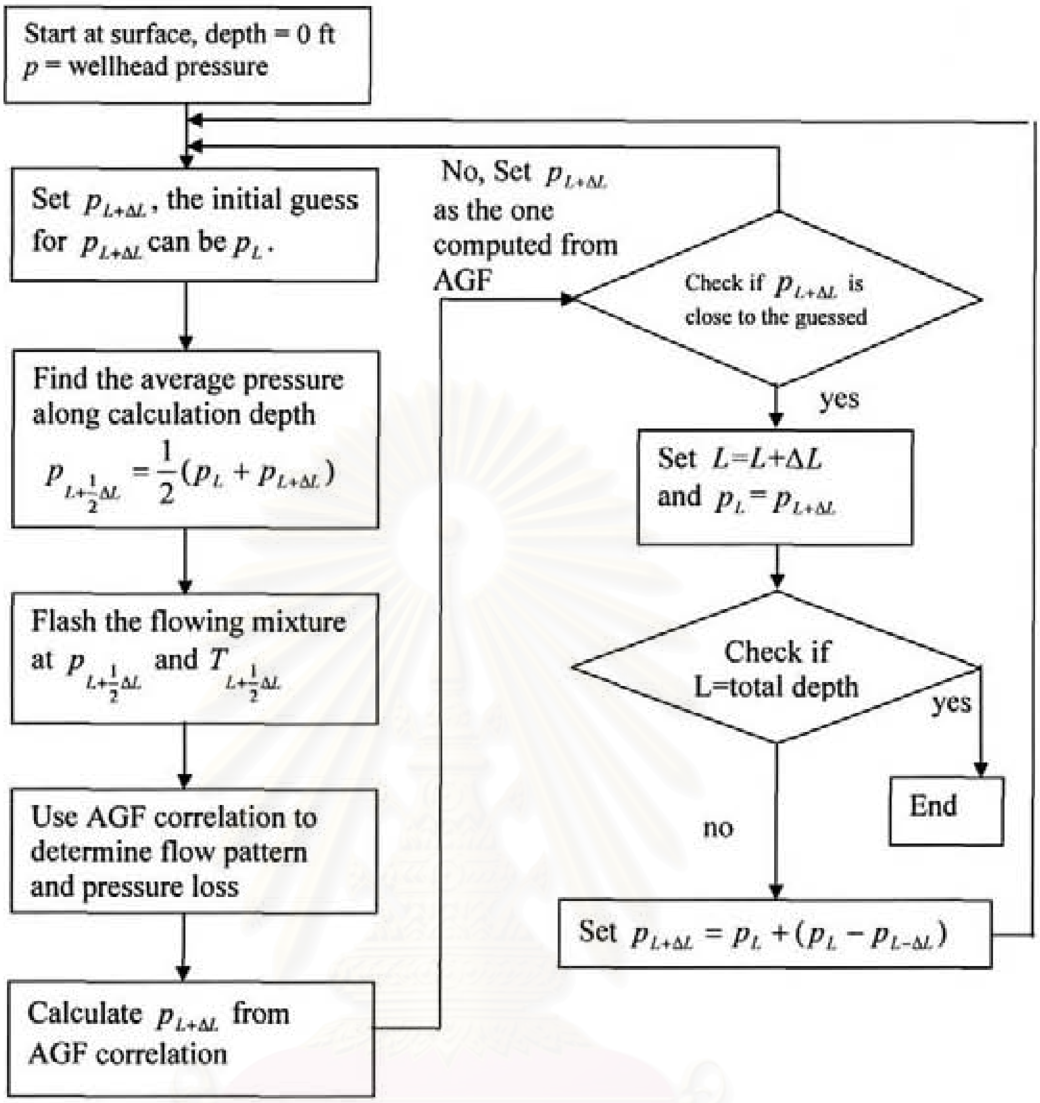

Figure 2.5: Flow chart of tubing model calculation.

Multiphase flow

To determine pressure loss in pipe for multiphase mixtures is much more difficult than calculating the pressure loss for single phase flow. Whereas single-phase flow may be characterized by laminar or turbulent flow, multiphase flow analysis must consider quantities of the phases, flow pattern of the mixture, interfacial tension between the phases, and different velocities of the phases.

No-slip holdup is defined as the ratio of the volume of liquid in a pipe segment divided by the total volume if the gas and liquid flowed at the same velocity. In this case, the liquid holdup can be directly calculated from the liquid and gas flow rates, that is in-situ volume fraction of liquid, C_L

$$C_L = \frac{q_L}{q_L + q_g} \tag{2.69}$$

Typically the phases will move at different velocities due to variation in phase densities and viscosities. The lighter phase moves faster than the denser. While the lighter phase keeps passing through the denser phase, this causes the denser phase to have more cross sectional area.

Since the phases are not moving in tandem, the phase volumes inside the system cannot be directly inferred from the phase flow rates.

The actual in-situ volume fraction of liquid, E_l, is the ratio of volume occupied by liquid to the total pipe volume.

$$E_l = \frac{Volume_L}{Volume_L + Volume_g} \tag{2.70}$$

Aziz, Govier, and Fogarasi (AGF) multiphase flow correlation

Aziz, Govier, and Fogarasi proposed a multiphase flow correlation that was dependent on the flow regime. The Aziz *et al.* correlation has some theoretical justification and is considered to be one of the least empirical correlations available. The steps to follow in the pressure drop calculations are:

1) Determine flow patterns.
2) Determine the liquid holdup appropriate for the existing flow pattern, and the hydrostatic head component of the total pressure drop.

3) Calculate the frictional pressure drop using a friction factor evaluated at Reynolds number appropriate for the flow pattern.
4) Calculate the total pressure loss as the sum of the hydrostatic head component, the frictional pressure loss, and if necessary, the kinetic energy term.

Flow pattern classification

Four flow regimes are considered: Bubble, slug, froth, and annular-mist. Aziz *et al.* presented original correlations for the bubble and slug flow regimes and used the method of Duns and Ros [8] for the froth and annular-mist flow regimes. These flow patterns are shown in Figure 2.6.

Bubble Flow

The pipe is almost filled with the liquid phase, and the pipe wall is always contacted with the liquid. The free gas is present in small bubbles. The bubbles have little effect on the pressure gradient.

Slug Flow

The liquid phase is still continuous, but the gas bubbles coalesce and form slugs which almost plug the pipe cross section. The bubble velocity is greater than that of the liquid. Both the liquid and gas phase have significant effects on the pressure gradient.

Transition Flow

There are changes from the liquid phase to the gas phase. Though the liquid effects are significant, the gas phase effects are more dominant.

Mist Flow

Though the pipe wall is still coated with the liquid, the continuous phase is the gas phase. The pressure gradient is now controlled by the gas phase.

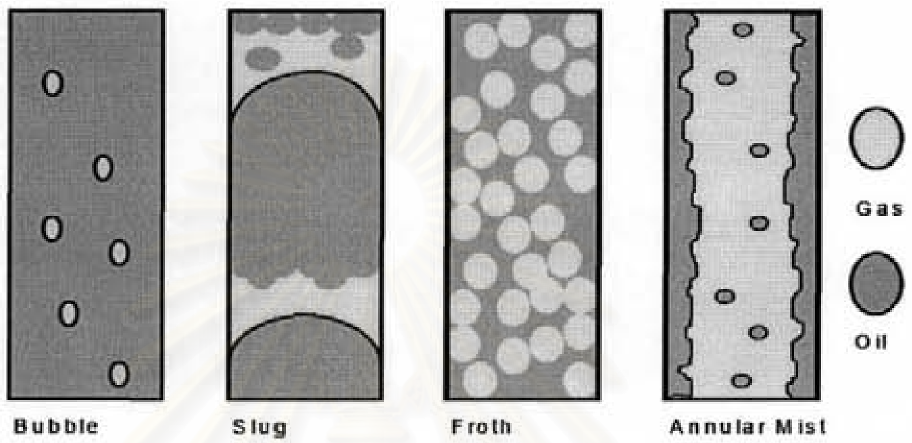

Figure 2.6: Picture of fluid characteristics inside tubing.

In order to classify the flow regime these parameters need to be calculated.

Superficial velocities

$$V_{sL} = \frac{q_L}{A} \tag{2.71}$$

$$V_{sG} = \frac{q_g}{A} \tag{2.72}$$

where V_{sL} is superficial velocity of liquid phase, ft/second

V_{sg} is superficial velocity of vapour phase, ft/second

A is area of wellbore

Modified superficial velocities

$$V_{MsL} = V_{sL}\left(\frac{\rho_L \sigma_{WA}}{\rho_{water}\sigma}\right)^{\frac{1}{4}} \tag{2.73}$$

$$V_{MsG} = V_{sG}\left(\frac{\rho_g}{\rho_{air}}\right)^{\frac{1}{3}}\left(\frac{\rho_l \sigma_{WA}}{\rho_{water}\sigma}\right)^{\frac{1}{4}} \tag{2.74}$$

where

ρ_L = liquid density, lb/ft^3

σ is interfacial surface tension between oil and gas, dyne/cm

σ_{WA} *is* interfacial surface tension between water and air, dyne/cm

$\sigma = 50$

$\sigma_{wA} = 72$

$\rho_{air} = 0.078 lb/ft^3$

$\rho_{water} = 62.37 lb/ft^3$

Mixture velocity is defined as

$$V_M = V_{MsL} + V_{MsG} \tag{2.75}$$

Flow Regimes are classified as the following:

1) If $V_{MsL} > 4$

and

$V_{MsG} < \frac{(100V_{MsL})^{0.17211}}{1.96}$; Bubble Flow

$\frac{(100V_{MsL})^{0.17211}}{1.96} \leq V_{MsG} < 26.5$; Slug Flow

$26.5 \leq V_{MsG}$; Annular-Mist Flow

2) If $V_{MsL} \leq 4$

and

$$V_{MsG} < \frac{(100V_{MsL})^{0.17211}}{1.96} \qquad \text{; Bubble Flow}$$

$$\frac{(100V_{MsL})^{0.17211}}{1.96} \le V_{MsG} < \frac{V_{MsL}}{0.263} + 8.6 \qquad \text{; Slug Flow}$$

$$\frac{V_{MsL}}{0.263} + 8.6 \le V_{MsG} < 70(100V_{MsL})^{-0.152} \qquad \text{; Froth Flow}$$

$$70(100V_{MsL})^{-0.152} \le V_{MsG} \qquad \text{; Annular- Mist Flow}$$

Pressure gradient calculation

Bubble flow regime

To obtain the pressure gradient due to fluid density in the bubble flow regime, Aziz *et al.* proposed to define the liquid holdup as in-situ liquid fraction

$$E_L = 1 - \frac{V_{sG}}{V_t} \tag{2.76}$$

where the absolute bubble rise velocity is

$$V_t = 1.2V_M + V_b \tag{2.77}$$

And the bubble rise velocity is

$$V_b = 1.41\left(\frac{g\sigma(\rho_l - \rho_g)}{\rho_l^2}\right)^{\frac{1}{4}} \text{ ; } \sigma = 95 \tag{2.78}$$

where g = gravitational acceleration, 32.2 ft / second^2

The hydrostatic head component of the total pressure gradient is then

$$\Delta P_H = \Delta L\left(\left.\frac{dP}{dL}\right|_H\right) = \frac{\Delta L}{144}\left(\frac{g}{g_c}(\rho_L E_L + (1 - E_L)\rho_g)\right) \tag{2.79}$$

The frictional pressure loss is

$$\Delta P_f = \frac{2 f_f V_M^2 \rho_L \Delta L}{144 g_c D} \tag{2.80}$$

where f_f is Fanning friction factor

g_c is gravitational constant, 32.2 lbm-ft/(lbf-second^2)

D is tubing diameter, ft

The friction factor can be found by solving the equation

$$\frac{1}{\sqrt{4f_f}} = 1.74 - 2\log\left(\frac{2\varepsilon}{D} + \frac{18.7}{R_e\sqrt{4f_f}}\right) \tag{2.81}$$

$$R_e = 1448\frac{DV_M\rho_L}{\mu_L} \tag{2.82}$$

where ε is absolute pipe roughness, inches

The acceleration component was considered to be negligible in the bubble flow regime.

Slug flow regime

The calculation method for slug flow regime is very similar to that of the bubble flow. The density component in the slug flow regime uses the same definition for liquid holdup in the bubble flow regime.

In-situ liquid fraction is similar to that of the bubble flow (Equation 2.76) However, V_b is defined as:

$$V_b = 0.345\left(\frac{Dg(\rho_L - \rho_g)}{\rho_L}\right)^{\frac{1}{2}} \tag{2.83}$$

V_t is defined in Equation 2.77.

Having obtained the liquid holdup, the hydrostatic head pressure loss and pressure loss due to friction are determined by Equation 2.79 and 2.80 respectively.

As in the bubble flow regime, the acceleration component was considered to be negligible in the slug flow regime.

Annular-Mist flow regime

For the annular-mist flow regime, Aziz *et al.* used the procedure of Duns and Ros. Duns and Ros assumed that the high gas velocity of the annular-mist region would allow no slippage to occur between the phases.

$$E_L = C_L = \frac{V_{sL}}{V_M} \tag{2.84}$$

The hydrostatic pressure drop is determined by Equation 2.79.

And the frictional pressure drop is

$$\Delta P_f = \frac{2 f_f V_{sG}^2 \rho_g \Delta L}{144 g_c D} \tag{2.85}$$

The friction factor is determined by Equation 2.81.

The Reynold's number is calculated only from the gas phase.

$$R_e = 1448 \frac{D V_{sG} \rho_g}{\mu_g} \tag{2.86}$$

The acceleration pressure loss can be accounted by using E_k

$$E_k = \frac{V_M V_{MsG} \rho_{NS}}{g_c p} \tag{2.87}$$

where ρ_{NS} is no slip density

The total pressure loss for annular-mist flow is

$$\Delta p_{total} = \frac{\Delta p_f + \Delta p_h}{1 - E_k} \tag{2.88}$$

Froth Flow Regime

The froth flow region is a region of transition between the slug and the annular-mist flow regions. When the flow occurs within the transition region, the pressure gradient is obtained by performing a linear interpolation between the slug and annular-mist regions, as suggested by Duns and Ros. The interpolation is performed as follows:

$$\Delta P = (\Delta P_1 - \Delta P_2)\left(\frac{V_{sG} - V_{sG2}}{V_{sG3} - V_{sG2}}\right) + \Delta P_1 \tag{2.89}$$

where

ΔP_1 is total pressure loss from slug flow

ΔP_2 is total pressure loss from (annular-mist flow)

$$V_{sG2} = \frac{((V_{MsL}/0.263) + 8.6)}{V_{MsG}} \tag{2.90}$$

$$V_{sG3} = \frac{((100V_{MsL})^{-.152})70)}{V_{MsG}} \tag{2.91}$$

2.5 Choke model

The reasons for having a choking device in the production system are to

- Protect reservoir and surface equipment from pressure fluctuations.
- Maintain stable pressure downstream of the choke for processing equipment.
- Provide the necessary backpressure on a reservoir to avoid formation damage and to prevent sand from entering the wellbore.
- Control flow rates and maintain well allowable.

- Produce the reservoir at the most efficient rate.
- Protect the reservoir and surface equipment from pressure changes.
- Prevent sand production due to excessive draw-down.
- Prevent water and/or gas coning.
- Get the most efficient production from the reservoir.

Generally, the flows of fluid through chokes are classified into two patterns based on the fluid velocity, critical flow and subcritical flow. In the critical flow region, fluids travel faster than sonic velocity. When the velocity of the fluid is greater than the sonic velocity of the fluid, any downstream perturbation is unable to propagate upstream, and the mass flow rate through the choke is solely a function of the upstream parameters. This causes the result as the independence of choke flow from the downstream pressure. In subcritical flow, the fluctuations in flow conditions are transmitted upstream of the choke. Because the effects of wellhead chokes on the production system are quite significant, an accurate choke performance calculation is one of the most important parts in the process of production optimization.

In theory, the choke should be small enough to cause critical flow. This has many advantages. The separator pressure can be changed, within reason, without altering the wellhead or sandface pressures.

In this study, Sachdeva *et al.* [9] correlation is used for choke calculation.

Sachdeva *et al.* correlation

There are some assumptions associated with Sachdeva *et al.* correlation:

- The gas phase contracts isentropically but expands polytropically.
- Flow is one-dimensional.
- Phase velocities are equal at the throat (no slippage occurs between the phases).
- The predominant influence on pressure is accelerational.

- The quality of the mixture is constant across the choke (no mass transfer between the phases).
- The liquid phase is incompressible.

Moreover, the Sachdeva *et al.* model makes no attempt to distinguish between free gas and solution gas, nor does it take into account the effect of different mixtures of liquids. Calculating procedure is by the following

1) Determine critical ratio of upstream to downstream pressure, y_c, by iterating on the upstream pressure, p_1, until y and y_c are merged.
2) Determine upstream pressure, p_1, that yield the same production rate that is obtained from the reservoir model. This can be done by iteratively calculate p_1 until the same production rate is obtained.
3) At this point, the upstream pressure is obtained.

The first step in the Sachdeva *et al.* method is to find the critical-subcritical boundary.

This is done by iterating and converging on y_c in the expression:

$$y = \frac{p_2}{p_1} \tag{2.92}$$

where p_2 is downstream pressure, psia

p_1 is upstream pressure, psia

$$y_c = \left\{ \frac{\dfrac{k}{k-1} + \dfrac{(1-x_1)V_L(1-y)}{x_1 V_{G1}}}{\dfrac{k}{k-1} + \dfrac{n}{2} + \dfrac{n(1-x_1)V_L}{x_1 V_{G2}} + \dfrac{n}{2}\left(\dfrac{(1-x_1)V_L}{x_1 V_{G2}}\right)^2} \right\}^{\frac{k}{k-1}} \tag{2.93}$$

$k = C_p/C_v$

$$n = 1 + \frac{x_1(C_p - C_v)}{x_1 C_v + (1 - x_1)C_L} \quad \text{= Polytropic exponent for gas} \tag{2.94}$$

x_1 = vapour fraction (inlet)

V_L = upstream liquid specific volume, cuft/lb

V_{G1} = upstream vapour specific volume, cuft/lb

V_{G2} = downstream vapour specific volume, cuft/lb

C_p = Specific heat of gas at constant pressure

C_v = Specific heat of gas at constant volume

C_L = Specific heat of liquid

$$V_{G2} = V_{G1} y^{\left(\frac{-1}{k}\right)} \tag{2.95}$$

$$\rho_{m2} = (x_1 V_{G1} y^{(-\frac{1}{k})} + (1 - x_1)V_L)^{-1} \tag{2.96}$$

After the critical ratio is found the condition of fluid flow can be determined whether it is critical flow or not. If y is equal or less than y_c, it is critical flow.

Mass flow rate

$$M = 86400 A_c C_D \sqrt{(2g_c 144 P_1 \rho_{m2}^2 \left[\frac{(1 - x_1)(1 - y\ )}{\rho_L} + \frac{x_1 k}{k - 1}(V_{G1} - yV_{g2}) \right]} \tag{2.97}$$

where A_c is cross-sectional area of choke

C_D is choke discharge coefficient

If the flow is critical, y is then equal to y_c. If the flow is subcritical, y is $\frac{p_2}{p_1}$.

Liquid flows through restriction

One of the limitations with the Sachdeva *et al.* choke model is that it cannot handle single-phase liquid flow. Fortunately, there are good single-phase liquid flow models.

For single-phase liquid flow, the pressure drop through the choke is assumed to be equal to the kinetic energy pressure drop divided by the square of a drag coefficient [10].

$$q = 5.615 * 22800C(D_2)^2 \sqrt{\frac{\Delta p}{\rho}} \tag{2.98}$$

C is a flow coefficient of the choke, based on the choke diameter and Reynold's number. This coefficient ranges from 0.92 to 1.2.

D_2 = choke diameter, inch

Mass flow rate is calculated by multiplying, q, by liquid density, ρ, and dividing by liquid molecular weight, M.

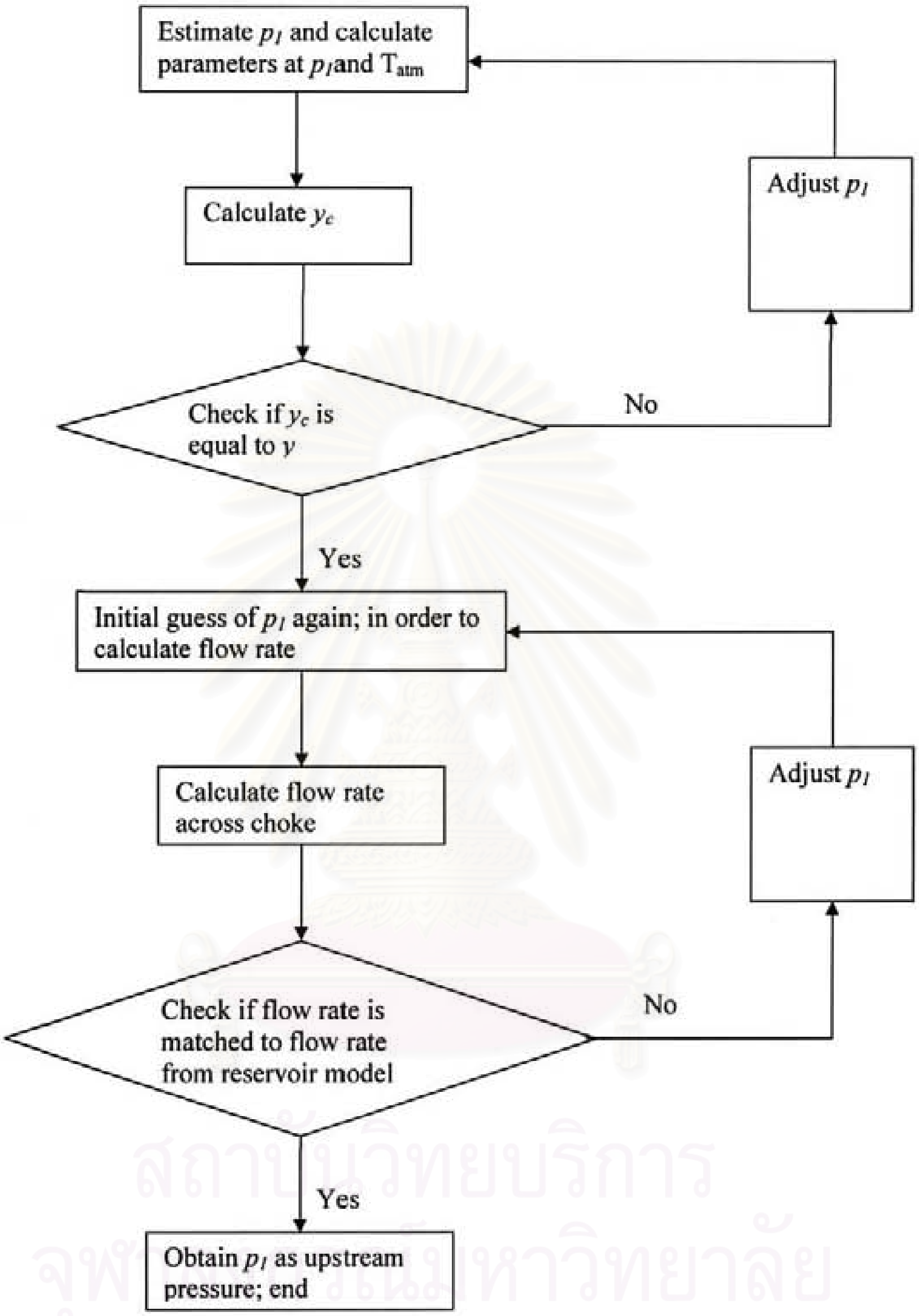

Figure 2.7: Flow chart of choke model calculation.

2.6 Pipeline model

After the production fluid passes through choke, there is horizontal flow through pipeline to the separator. This model is used to calculate the pressure drop in the flow line to separator. Aziz, Govier and Fogarasi (AGF) multiphase flow correlation can still be used for horizontal flow.

The horizontal flow pressure loss calculation is considered to be similar to that of vertical flow without hydrostatic pressure loss.

2.7 Separator model

Before the production goes to sale line, mixed fluids is separated to liquid phase, gas phase and water phase. However, there is no water in this study. Three-stage separation and one stock tank are used in this study because it is necessary to reduce fluid pressure to ambient before it goes into the stock tank.

Separators are controlled by adjusting the internal pressure. The amount of each output stream depends upon the separator pressure. In each separator, the fluids are flashed into liquid and gas. The liquid, by gravity, is at the bottom part, and gas phase is on the top. Figure 2.8 shows cross sectional picture of the separator.

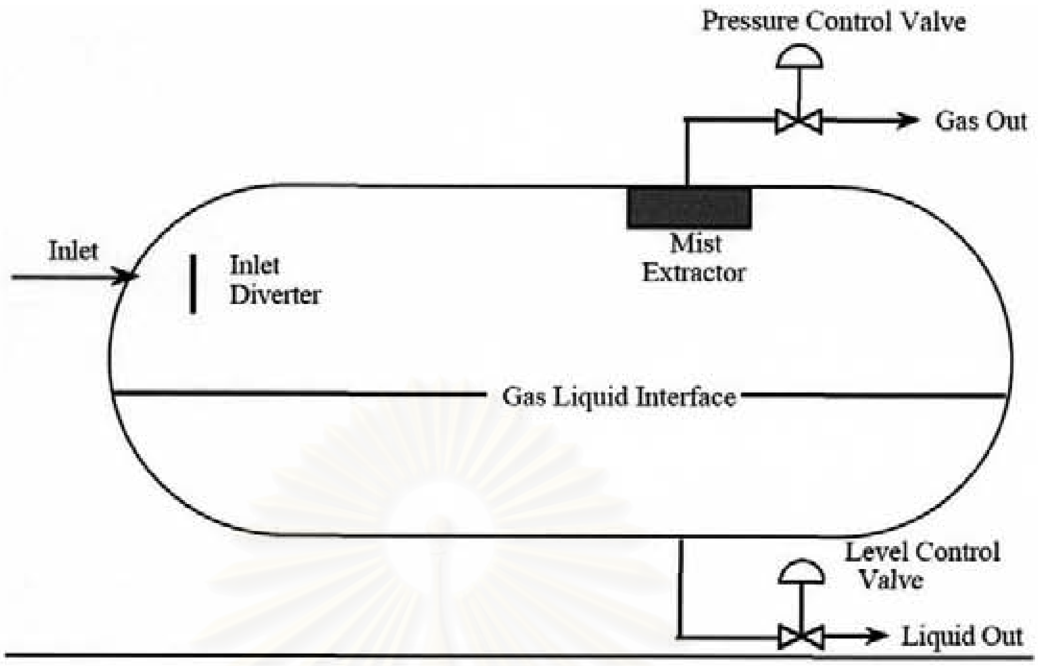

Figure 2.8: Cross section of a separator.

Calculation steps of three-stage separations and one stock tank are as follows:

1) Begin with a mixture of the produced fluid and the lift gas. This mixture has a mass flow rate of N_p, and a composition of z, (same composition as in the reservoir)
2) Perform a flash calculation of the mixture at first stage separator, p_{sep1}. The liquid mole fraction at this stage is L_1. The gas phase composition is y_1, with mass rate of N_{wh} $(1-L_1)$. The oil phase mass flow rate is $N_{wh}L_1$ with composition x_1.
3) Flash first separator liquid composition x_1 at second separator pressure, p_{sep2}. The liquid mole fraction at this stage is L_2. The gas phase mass rate is $N_{wh}L_1(1-L_2)$ with composition y_2. The oil phase mass rate is $N_{wh}L_1L_2$ with composition x_2.
4) Flash second separator liquid composition x_2 at third separator pressure, p_{sep3}. The liquid mole fraction at this stage is L_3. The gas phase mass rate is $N_{wh}L_1L_2(1-L_3)$ with composition y_3. The oil phase mass rate is $N_{wh}L_1L_2L_3$ with composition x_3.
5) Flash third separator liquid composition x_3 at ambient pressure, p_{atm}. The liquid mole fraction in the stock tank is L_{st}. The sales oil mass rate is $N_{wh}L_1L_2L_3L_{st}$ with composition X_{st}. The sales mass rate (lb/day) can be figured by multiplying the mass

rate (mole/day) by the molecular weight of X_{st}. The volume rate can be figured by dividing this mass rate by the fluid's density.

6) The gas stream can be determined by combining the three separator gas streams. The mass rate (lbmol/day) is $N_{wh}(1-L_1)+ N_{wh}L_1(1-L_2)+ N_{wh}L_1L_2(1-L_3)$. Notice that gas from the stock tank is lost to the atmosphere

7). The oil produced from the stock tank is sold. Partion of the gas from the first three separators may be compressed and injected into the tubing-casing annulus in the gas-lift process.

Picture of four-stage separator with production path inside the separators are shown in Figure 2.9.

For multi-stage separator, there are correlations to find separator pressures, related to first stage separator pressure and stock tank pressure. The simplest of these methods assumes an equal pressure ratio between the stages for optimum performance (Campbell) [11]

$$r = \left(\frac{p_1}{p_{st}}\right)^{\frac{1}{n}} \tag{2.99}$$

$$p_{sep,i} = p_{st} r^{n-i+1} \tag{2.100}$$

2.8 Economics

In this study net present value (NPV) is defined to be an objective function. Net present value provides the discounted value of a future cash flow. Economics model is used to calculate NPV from all income and costs for each project. Costs are from the cost of drilling, completion, facility, operation cost and abandonment cost, while income is from oil and gas sales [12].

Costs are grouped as initial cost, operation cost and abandonment cost. The initial cost includes drilling, completion, and facility costs which all are affected by the decision variable and are the cost at the first time step. The operation cost is a cost incurred at each time step while the abandonment cost occurs at the final time step. Incomes from oil and gas sales are also evaluated at each time step.

It needs to be remarked that tax and depreciation are not calculated in this economic model.

Table 2.2 is a sample calculation of net present value for a project with an initial cost of \$60,000. This example is provided by Thompson and Wright [13].

In this study, a fixed discount rate is used for Net Present Value calculation.

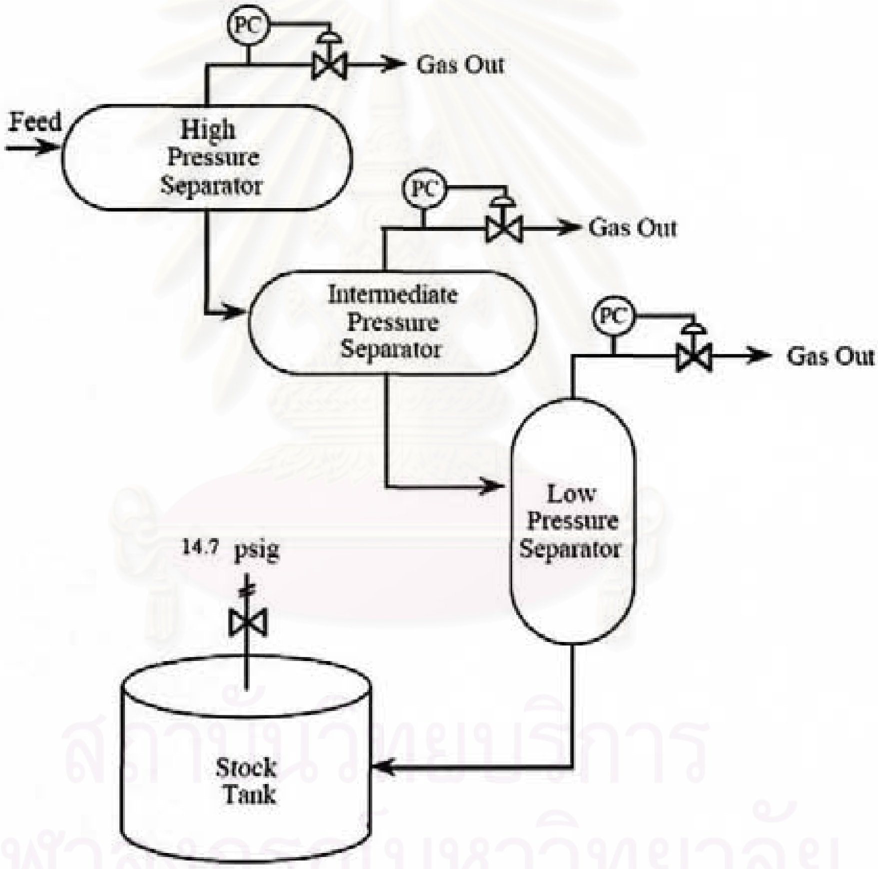

Figure 2.9: Four-stage separator (from Carroll, 1990) [1].

Table 2.1: Net present value illustration from Thompson and Wright [13].

Year	Net Cash Flow	Present Value Discounted at 10%	
0	-60,000		-60,000
1	37,100		33,727
2	16,800		13,884
3	12,200		9,166
4	8,640		5,901
5	5,440		3,378
6	250		141
		Net Present Value for Project	$ 6,198

The net present value depends on the discount factor and distribution of cash flows.

2.9 Integrated model

This model combines all compartmentalized models together in order to simulate the production of fluid from the reservoir to the separator system. The calculation starts at initial reservoir conditions. Along the way from the reservoir to the separator, fluid properties are calculated using compositional models. At the separator, the fluids are flashed into oil and gas phases. Some of gas production is injected back to the well for artificial lift. The rest of gas production and all of oil production go for sales. At the end of each time step, reservoir pressure and fluid composition in the reservoir are updated. The production is stopped when reservoir pressure is equal to or less than abandonment pressure.

The computation at each time step can be summarized as follows:

1) Beginning at time t and reservoir pressure $p_{res,t}$, assume a value for the well flowing pressure p_{wf} for the time step. Also, assume a value for pressure at the new time step $p_{res,t+1}$

2) Find $p_{res,t+1/2}$ from $p_{res,t}$ and $p_{res,t+1}$.

3) Calculate the mass flow rate. Also, find the produced fluid composition.

4) Find the wellhead pressure which allows the produced fluid and lift gas to flow across the choke against the first separator pressure.

5) Calculate pressure loss along pipeline, choke and wellbore, from wellhead to injection depth, with the combined fluid composition, z_{p+inj} and mass flow rate, N_{p+inj} (produced flow rate plus gas-lift flow rate) to find the pressure at the point of injection.

6) Calculate pressure loss along wellbore, from injection depth to sandface depth, with the produced fluid composition, z_p, and mass flow rate, N_p, to find the tubing pressure at the sandface, well flowing pressure p_{wf}.

7) If the difference between assumed p_{wf} and computed p_{wf} is too large, change p_{wf} and go back to Step 3.

8) Combine the produced fluid and lift gas mass rate, N_{p+inj}, and composition, z_{p+inj}, and use the separator model to determine the produced mass rate, N_{atm}, new lift gas composition, z_{inj}.

9) Find the new reservoir pressure $p_{res,t+1}$ from producing the fluids at the mass rate computed in step 3 and new reservoir composition. If $p_{res,t+1}$ has changed significantly, return to Step 2. Otherwise, all quantities can be updated at this point.

10) Check if the flow rate is more than the minimum rate. If yes, begin calculation for the new time step.

The calculation procedure is shown as a flow chart in Figure 2.10.

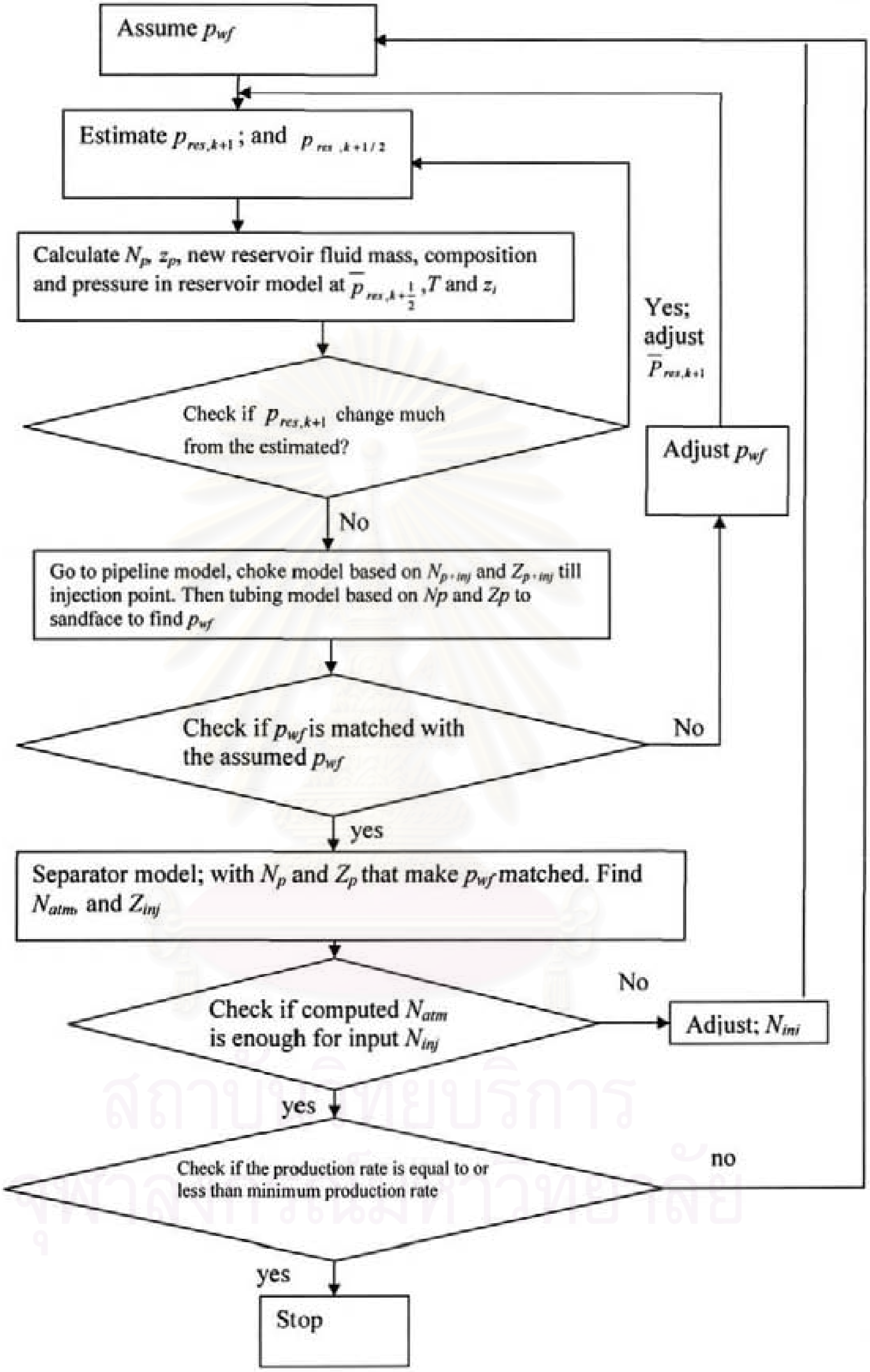

Figure 2.10: Flow chart of integrated model calculation.

2.10 Optimization method: Genetic algorithm

Genetic algorithm is an optimization method that draws an analogy to the process of natural selection (Goldberg) [14]. It is a search technique used in computing to find true or approximate solutions to optimization and search problems, and is often abbreviated as GA. Based on the random generation of decision variables and the development of the sets of variables using direct function value comparisons, the GA does not require any mathematical computations. Genetic algorithms uses techniques similar to evolutionary of biology such as mutation, selection, and crossover.

Without the idea of a convergence, a GA criterion depends upon user satisfaction. Consequently, the use of GA should be carefully examined based on the problem types, dimensions and computer capacities.

In order to optimize the problem, genetic algorithms generate population of decision variables (called chromosomes) of candidate solutions (objective funciton) for better solutions. Normally, populations are represented in binary as strings of 0s and 1s, but other encodings are also possible. The genetic algorithm starts with population of randomly generated. In each generation, the fitness of every individual in the population is evaluated. Then, in the next generations populations are selected from the current population (based on their fitness), and modified to form a new population. These processes are repeated until it reaches its critiria.

GA procedure

Typical genetic algorithm concepts are described as

1) Genetic representation of the solution domain
2) Fitness function to evaluate the solution domain

A standard representation (population) of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. This makes GA itself easily reproduce the population by crossover or mutation.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. In this study, the Net Present Value (NPV) is to be maiximized.

Once we have the genetic representation and the fitness function defined the following steps needed to be proceeded

1) Initialize a population of solutions randomly.
2) Evaluate the fitnesses of individuals in the population.
3) Improve it by the application of mutation, crossover, and selection.
4) Evaluate the individual fitnesses of the population.
5) Replace worst ranked part of the population with offsprings until critiria is reached.

In the first generation, the GA evaluates each population according to the fitness function. The randomly generated candidates which have small fitness will be deleted. However, purely by chance, a few may hold promise. They may show activity, even if only weak and imperfect activity, toward solving the problem. These candidates are kept and allowed to reproduce. New populations are randomly reproduced from them. The candidates that make better fitness will be allowed to go to next generation. Those candidate solutions which were worsened or made no better are again deleted. The good individuals are selected and copied over into the next generation with random changes, and the process repeats.

The expectation is that the average fitness of the population will increase each round, and so by repeating this process for hundreds or thousands of rounds, very good solutions to the problem can be discovered.

Initiaization

Initially, the first sets of decision variable are randomly generated. The population size depends on criterion of the problem, but typically contains several hundreds or thousands of possible solutions. In this study, the population is generated randomly, covering the entire range of possible solutions.

Selection

During each process, a proportion of the existing population is selected to form a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions.

There are many different techniques which a genetic algorithm can use to select the individuals to be copied over into the next generation, but Roulette-wheel selection is some of the most common methods.

Roulette-wheel selection

Base on the fitness of each individual in last generation, genetic algorithm provides the probability of an individual. The fitter of the population, the more chance to be selected to next generation. (Conceptually, this can be represented as a game of roulette - each individual gets a slice of the wheel, but more fit ones get larger slices than less fit ones. The wheel is then spun, and whichever individual owns the section on which it lands each time is chosen.) [14]

Reproduction

The next step is to generate new generation populations from those selected through genetic operators: crossover and, or mutation.

For each new solution to be produced, a pair of parent solutions is selected for breeding from the sets of parameters selected previously. To produce new population, the methods of crossover and mutation are used, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each child, and the process continues until a new population of solutions of appropriate size is generated.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best parameters from the first generation are selected for breeding, along with a small proportion of less fit solutions.

Crossover techniques

Many crossover techniques exist for organisms which use different data structures to store themselves.

One point crossover

A crossover point on the parent organism string is selected. All data beyond that point in the organism string is swapped between the two parent organisms. The resulting organisms are the children, shown in Figure 2.11.

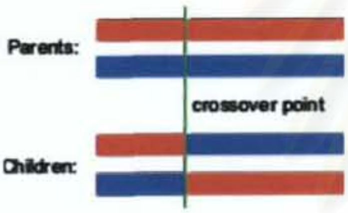

Figure 2.11: Picture of one point crossover.

Two point crossover

Two point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, becoming two child organisms, shown in Figure 2.12.

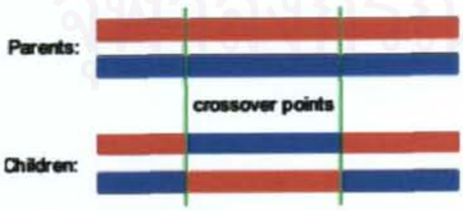

Figure 2.12: Picture of two point crossover.

Crossover for ordered chromosomes

Depending on how the chromosome represents the solution, a direct swap may not be possible.

After a crossover point is selected on the parents. Another population is selected by order. The chromosome that has not existed on the first population before the crossover point will be choosen to be new member. This choosing process is continued until the chromosomes of new populations are completed. For example, if our two parents are ABCDEFGHI and IGAHFDBEC and our crossover point is after the fourth character, then the resulting children would be ABCDIGHFE and IGAHBCDEF.

Mutation technique

In genetic algorithms, mutation is used to maintain genetic diversity from one generation of a population of chromosomes to the next. It is analogous to biological mutation.

Example is by replacinging a chromosome in parent's organism by a candidated chromosome. This might be at any position in the parents. This random variable tells wheter the new chromosome or the replaced chromosome is fit or not.

The purpose of mutation in genetic algorithm is to prevent the population of chromosomes from becoming too similar to each other which causes slowing or even stopping evolution. This reasoning also explains the fact that most genetic algorithm systems avoid only taking the fittest of the population in generating the next but rather a random selection with a weighting toward those that are fitter.

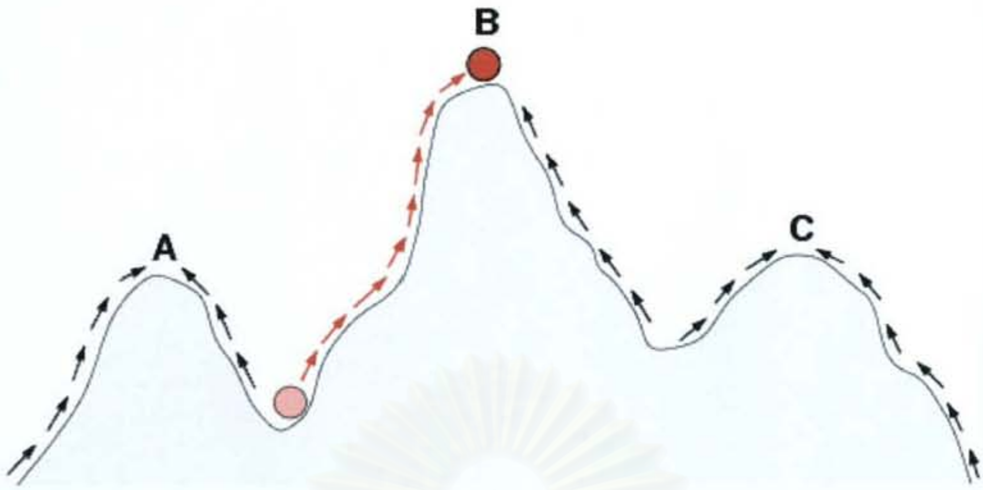

Figure 2.13: Sketch of a fitness landscape. The arrows indicate the preferred flow of a population on the landscape, and the points A, B, and C are local optima. The red ball indicates a population that moves from a very low fitness value to the top of a peak (after Wilke) [15].

Termination

This generational process is repeated until a termination condition has been reached. Common terminating conditions are

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

2.11 Summary of theory and concept

Conclusionly, there are 3 mains program combined into the integrated program. There are production profile program, economic program and genetic algorithm program. At the first generation the sets of decision variable are randonly created by genetic algorithm. Then, the production profile program is run and passes the production profile to the economic program. Finally the NPV is send to genetic algorithm program. For the second generation, sets decision variables are reproduced by genetic algorithm. Then all three programs are run through to get the NPV. This process is repeated until it reached the criteria.

CHAPTER III

MODEL TESTING AND CASE STUDIES

In this section, the results of the model testing and 3 examples of optimization calculations are presented. The integrated model was run and compared the result with that of the ECLIPSE program. Then, three optimization case studies are done to find maximum NPV. Genetic algorithm is used to save time of calculation.

3.1 Model Testing

After all models were integrated, the program was tested, and the results were compared with those from one of most reliable commercial simulation program, ECLIPSE.

One limitation of this task is that there is no function to limit target production rate in the program. The flow rate is determined by matching the well flowing pressure between inflow and outflow model, as described before. Therefore, to compare the results to those obtained from the ECLIPSE, there is a need to adjust the choke size at each time step in order to match the production rate to that of the ECLIPSE.

Reservoir parameters of model testing

The reservoir parameters used in testing the model are

Reservoir radius = 400 ft
Reservoir thickness = 100 ft
Reservoir permeability = 10.85 md
Reservoir porosity = 20%
Initial reservoir pressure = 3000 psi
Reservoir temperature = 300 F

Reservoir depth = 8000 ft

Residual gas saturation = 0.15

Residual oil saturation = 0.01

It needs to be remarked that in ECLIPSE reservoir is square shape, not circular like that of the integrated program. With same reservoir thickness, the reservoir radius of the integrated program is calculated in order to make reservoir volume equal to that of the ECLIPSE.

Reservoir fluid is gas condensate with composition as following:

Component C1 = 59.991

Component C2 = 8.4326

Component C3 = 6.3988

Component i-C4 = 3.4127

Component n-C4 = 3.8989

Component i-C5 = 1.4286

Component n-C5 = 1.3988

Component C6 = 7.2718

Component C7+ = 7.7660

Variable inputs of model testing

1. Tubing ID size = 0.625 ft
2. Pipeline = Not Available
3. Choke size is adjustable to match with ECLIPSE production rate
4. Separator Pressure = 14.7 psia
5. Number of well = 1 well
6. Gas injection rate = zero

Result of model testing

Difficulty of this testing is that the choke needs to be adjusted along the time of calculation in order to match the production rate. Production rate is very sensitive to choke size. By adjusting the choke size only little, the production rate is fluctuating, as can be seen in Figures 3.1 and 3.2.

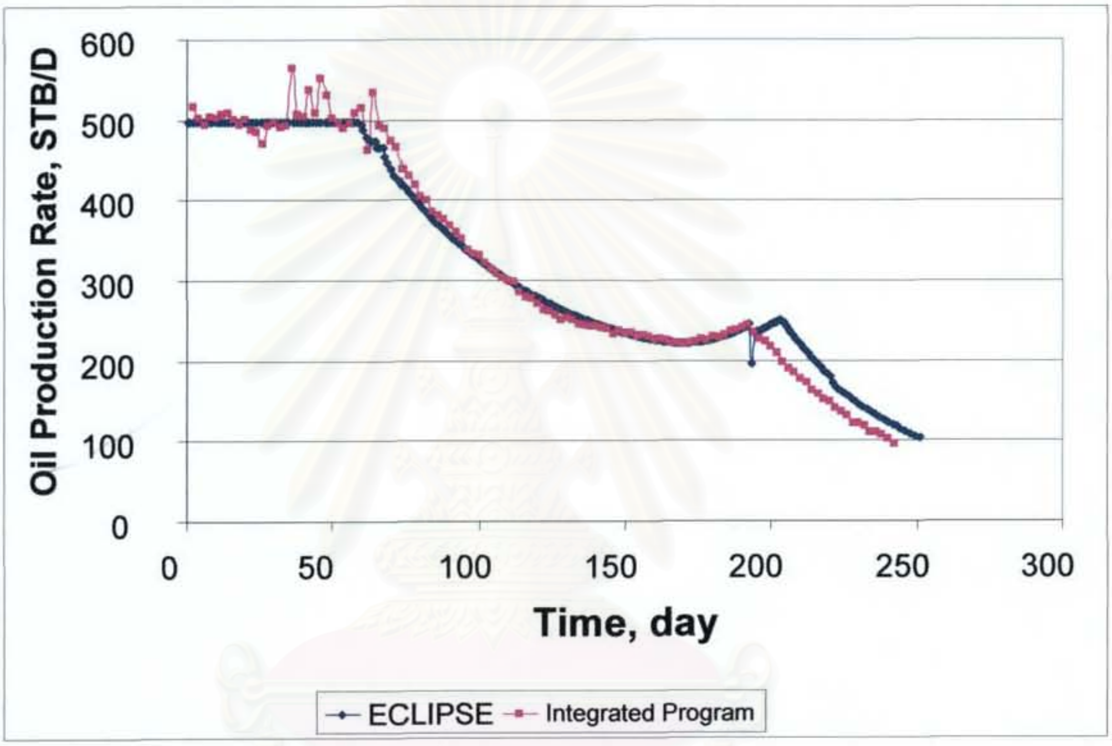

Figure 3.1: Comparison of oil production rate obtained from the integrated model and ECLIPSE.

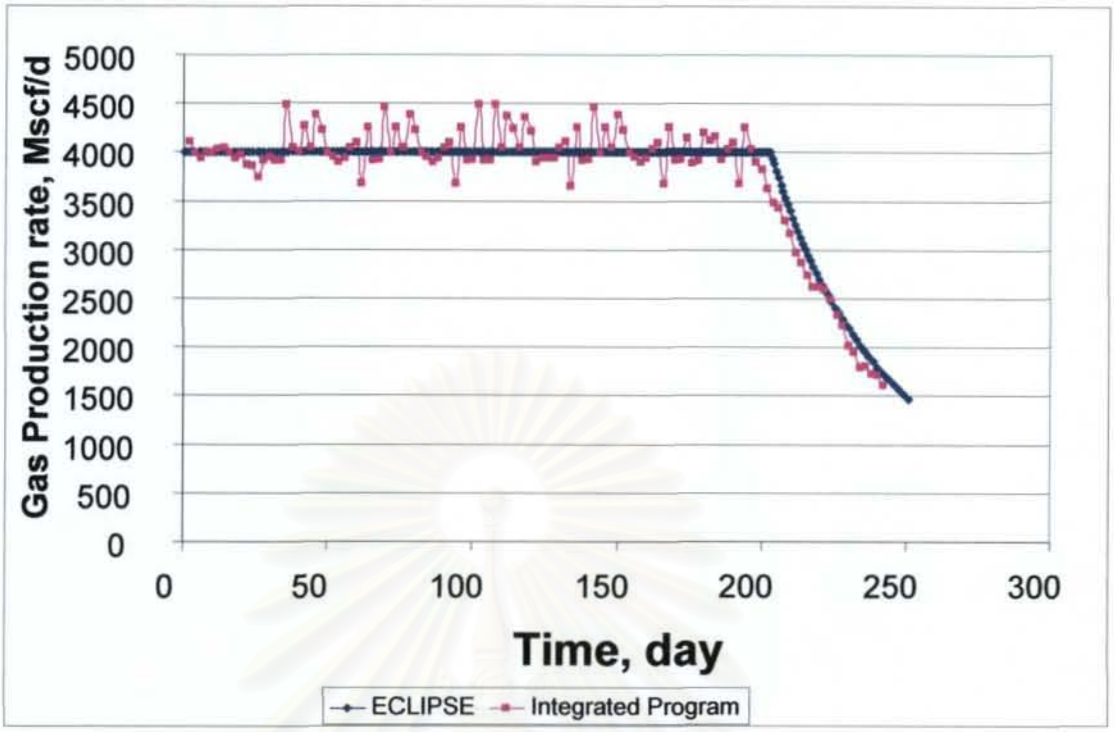

Figure 3.2: Comparison of gas production rate obtained from the integrated model and ECLIPSE.

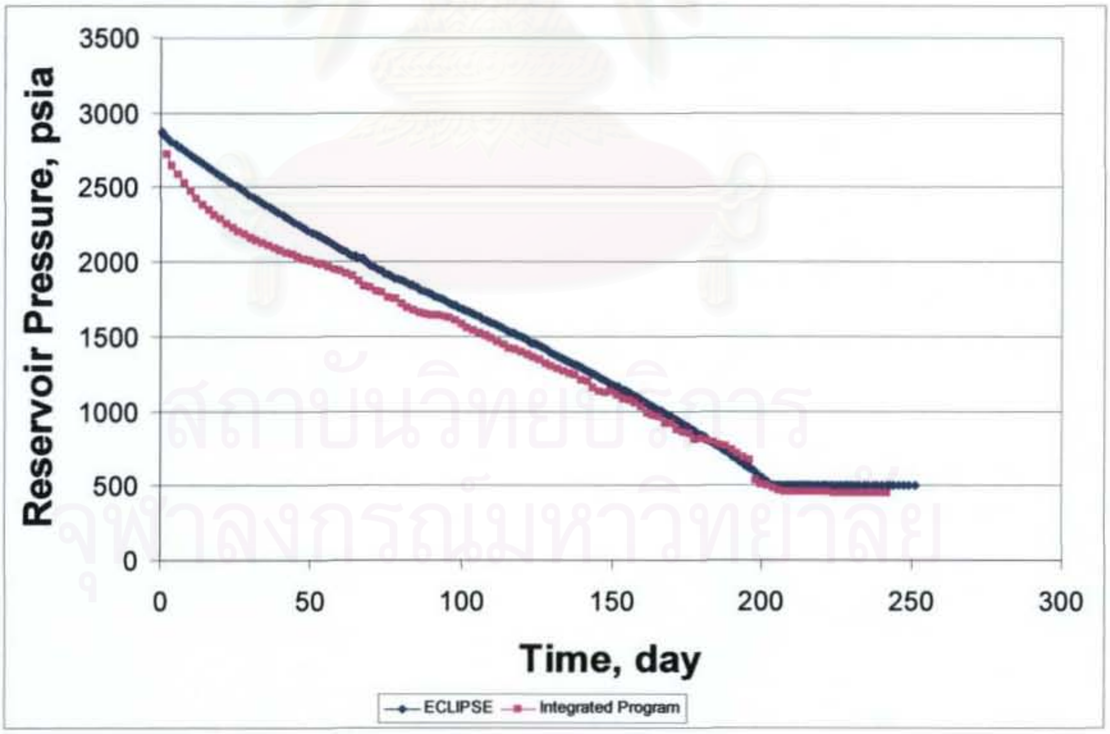

Figure 3.3: Comparison of reservoir pressure obtained from the integrated model and ECLIPSE.

In any case, the errors between these two programs are not significantly different. The maximum error in flow rate is 15% while average the error of flow rate is about 8.5%. This might cause by the difference of reservoir shape. Even the reservoir volumes are equal, that difference of reservoir shape does make reservoir in flow performance difference.

At the early time, fluid composition in the reservoir is still similar to the initial composition. This gas oil ratio is more or less constant in the first 60 days. This is because the fluid pressure is above its dew point. After that period, fluid composition inside the reservoir and also reservoir pressure start to change, this makes production composition changed. This change affects the produced gas oil ratio.

When the reservoir fluid is initially produced at a constant rate, the reservoir pressure reduces quite linearly. After the reservoir could not make the target rate and the production rate starts declining, the reservoir pressure decreases at a slower pace.

In conclusion, based on the results of this test run, the integrated program yields a 15% maximum error in flow rate compared with the outputs from ECLIPSE.

3.2 Case studies

To maximize the objective function which is NPV, the integrated model is run for the answer which is a set of parameters for completion design and production.

In these following case studies, economic parameters are rated for incremental production system. Main production facilities are existed. The reservoir in each case study is to expand the field and is produced by adding up to the main facility.

3.2.1 Case 1

Reservoir parameters

Reservoir radius = 1000 ft

Reservoir thickness = 100 ft

Reservoir permeability = 10 md

Reservoir porosity = 20%

Initial reservoir pressure = 3000 psi

Reservoir temperature = 300 F

Reservoir depth = 8000 ft

Residual gas saturation = 0.15

Residual oil saturation = 0.01

Surface temperature = 60 F

Surface pressure = 14.7 psi

Reservoir fluid composition [6] is

Component C1 = 20%

Component C2 = 15%

Component C3 = 20%

Component i-C4 = 7.5%

Component n-C4 = 7.5%

Component i-C5 = 7%

Component n-C5 = 7%

Component C6 = 10%

Component C7+ = 6%

Molecular weight of C_{7+} is 142 lb/mole

Economic parameters

Discount rate is 10 percent.

Oil price is $60 per STB

Gas price is $15.71 per Mscf

Drilling cost is $1,000,000

Well completion and equipment cost are

- 2.875-inch OD well $1,200,000 per well
- 4-inch OD well = $1,450,000 per well
- 5.5-inch OD well = $1,800,000 per well

Production equipment cost of oil [19] is 540,000+52.50*(Maximum oil rate, STB/day) per well

Production equipment cost of gas [19] is 675*(Maximum gas rate, MMscf/day) per well

Production operation cost is $1,400,000 per year per well

Gathering line costs ($) = 500 * length (ft) * number of well

In case there is compressor.

- The well completion and equipment cost is $500,000 per well extra.
- Additional operation cost per well is
    - $150,000 per year for 50Mscf per day injection.
    - $300,000 per year for 100Mscf per day injection.

Decommissioning cost is $500,000 per well

Variable inputs

The model was run with these following decision variables

1. Tubing ID size: 2.441 inch, 3.548 inch and 4.950 inch. All the wells are assumed to have the same tubing size.
2. Choke sizes are varied every 5-year period by following sizes: 16/64, 32/64, 48/64 and 64/64. All the wells are assumed to have the same choke size.
3. Pipeline ID size: 7.921 inch, 10.050 inch and 11.084 inch.
4. Pressure of the first separator: 100 psia, 150 psia and 200 psia.

Pressures of second and third separators are

First separator, psia	Second separator, psia	Third separator, psia
100	62	38
150	84	47
200	104	54

5. Gas injection rate: Zero, 50000 scf/day and 100000 scf/day. All the wells are assumed to have same gas injection rate.
6. Number of wells: 1, 2, 3, 4 and 5.

Result and discussion

Based on six decision variables, the search space included a total of 6480 possible combinations. Each generation had a population of 10 members. Each member has 9 chromosomes. In this case, the program was run for 80 generations. So, the model computed the output 800 times using different sets of control variables to get the fittest answer. After 80 generations, the maximum NPV was found to be $140,556,637.

Table 3.1 shows combinations of production parameters that yield the highest NPV in each generation. The value of NPV in each generation is plotted in Figure 3.4. As seen from the figure, generation 69 provides the highest NPV. The production parameters that result in this highest NPV are summarized in Table 3.2.

Table 3.1: Production parameters that provide the highest NPV in each generation.

Generation	Tubing Size	Pipeline Size	Choke1	Choke2	Choke3	Choke4	Psep	Gas inject rate	No. of well	NPV	life
	Inch	Inch					psia	scf/day		$	year
1	2.875	12	32	48	48	64	100	0	1	80,391,295	18
2	2.875	8	32	32	64	64	100	0	1	73,591,177	20
3	5.5	10	16	16	16	16	100	0	1	21,145,299	17
4	4	12	16	64	64	64	100	0	1	75,384,679	9
5	2.875	8	32	48	48	64	100	0	2	99,849,717	9
6	2.875	10	16	16	16	16	100	50000	1	13,826,497	6
7	5.5	12	16	16	48	16	100	50000	1	45,356,987	15
8	2.875	8	16	16	16	16	100	0	1	13,933,681	24
9	4	10	32	48	16	64	100	0	4	108,742,187	5
10	5.5	10	32	32	32	32	150	0	1	88,405,799	12
11	2.875	8	32	32	32	32	100	0	1	65,966,100	15
12	2.875	10	16	16	16	16	100	100000	1	10,768,458	25
13	2.875	10	16	16	16	16	100	50000	1	13,826,497	24
14	4	12	16	16	16	16	200	100000	1	19,268,720	17
15	2.875	10	16	16	48	48	100	0	1	11,894,578	12
16	2.875	12	16	16	16	16	100	50000	1	14,033,681	6
17	2.875	10	16	16	16	16	100	0	1	11,894,578	5
18	2.875	12	16	32	48	64	100	0	1	52,956,712	23
19	5.5	12	48	48	64	64	100	50000	1	105,569,331	13
20	2.875	12	48	48	48	48	100	0	1	98,614,354	16
21	2.875	12	32	32	32	32	200	50000	1	59,672,513	12
22	2.875	12	32	32	64	64	100	0	1	73,591,177	20
23	2.875	12	16	16	64	64	100	100000	3	75,167,057	5
24	2.875	12	32	32	64	64	100	50000	1	71,269,853	20
25	4	12	32	64	64	16	100	0	1	74,235,698	10

Table 3.1: Production parameters that provide the highest NPV in each generation. (continued).

Generation	Tubing Size	Pipeline Size	Choke1	Choke2	Choke3	Choke4	Psep	Gas inject rate	No. of well	NPV	life
26	5.5	12	32	32	64	16	150	100000	1	86,258,637	15
27	2.875	12	32	32	64	64	150	0	3	102,453,814	6
28	2.875	12	32	32	64	64	200	100000	2	87,357,126	10
29	5.5	10	48	48	64	64	100	0	1	105,674,951	11
30	4	10	16	48	48	64	100	50000	1	79,326,358	13
31	4	10	32	64	64	64	100	50000	2	120,465,725	11
32	5.5	10	16	48	48	64	100	0	3	121,167,198	5
33	4	12	48	48	48	48	100	0	1	98,306,836	16
34	2.875	12	48	48	48	48	150	0	1	95,251,962	15
35	4	12	48	48	48	64	200	0	1	96,363,842	9
36	5.5	12	16	16	48	48	200	0	1	66,637,149	18
37	4	10	16	64	64	64	200	0	3	107,269,852	5
38	2.875	12	16	48	48	48	100	0	1	63,626,852	21
39	5.5	8	48	48	48	48	100	0	1	98,306,836	16
40	4	10	32	32	32	32	100	0	1	104,598,885	12
41	2.875	12	48	48	64	64	100	0	1	100,795,606	15
42	5.5	12	32	48	48	48	150	50000	1	101,129,354	8
43	5.5	10	32	48	48	48	150	50000	1	96,583,415	10
44	5.5	12	64	48	32	64	100	0	1	75,384,679	9
45	4	12	16	16	48	16	100	50000	1	43,677,451	17
46	4	10	16	48	64	64	200	0	3	107,269,852	5
47	5.5	10	16	48	64	64	100	0	3	121,167,198	5
48	4	10	16	32	32	32	100	0	2	106,432,147	8
49	2.875	12	48	48	64	64	100	50000	2	122,357,428	8
50	2.875	12	64	64	64	64	100	100000	2	116,375,429	10
51	2.875	12	32	48	48	32	200	50000	1	99,672,513	12
52	2.875	12	32	32	48	48	100	0	4	78,824,365	6
53	4	12	64	48	48	64	100	0	1	93,832,715	7
54	5.5	12	16	64	64	64	100	0	1	103,468,975	5
55	5.5	12	64	48	48	64	100	0	1	91,583,741	15
56	4	10	16	32	32	48	100	0	1	56,398,284	15
57	2.875	12	48	48	64	64	100	0	2	121,235,207	8
67	5.5	12	48	64	64	64	100	0	5	102,165,479	3
68	5.5	12	32	32	32	32	100	0	5	107,009,400	4
69	5.5	12	64	64	64	64	100	0	2	140,755,293	5
70	4	12	48	48	64	64	100	0	3	124,574,124	11
71	4	12	48	64	64	64	100	0	3	126,675,487	8
72	2.875	12	48	48	48	48	100	0	3	113,195,498	6
73	4	12	64	32	16	48	100	0	2	105,347,269	7
74	5.5	12	64	64	64	64	100	0	3	138,387,994	9
75	5.5	10	16	64	64	64	100	0	2	116,396,481	7
76	5.5	12	32	64	64	64	100	0	2	124,595,308	6
77	5.5	12	48	48	48	48	100	0	2	126,257,824	6
78	5.5	12	32	48	48	64	100	0	2	119,526,371	8
79	5.5	12	48	64	64	64	100	0	2	135,629,584	5
80	5.5	12	48	48	64	64	100	0	2	133,279,583	6
Final	5.5	12	64	64	64	64	100	0	2	140,556,637	5

Results from the genetic algorithm show that in the first 20 generation, the value of the best NPV in each generation is quite fluctuating. Then, the objective function tends to increase with less difference comparing to previous generations. At generation 69th, it reaches the best answer of the case being studied. In many case, best answer from previous is brought to be an offspring of next generation. So the best answer in later generation is always equal or higher than the former, as shown in Figure 3.5. Before the program stopped, the best answer was not improved for 11 generations.

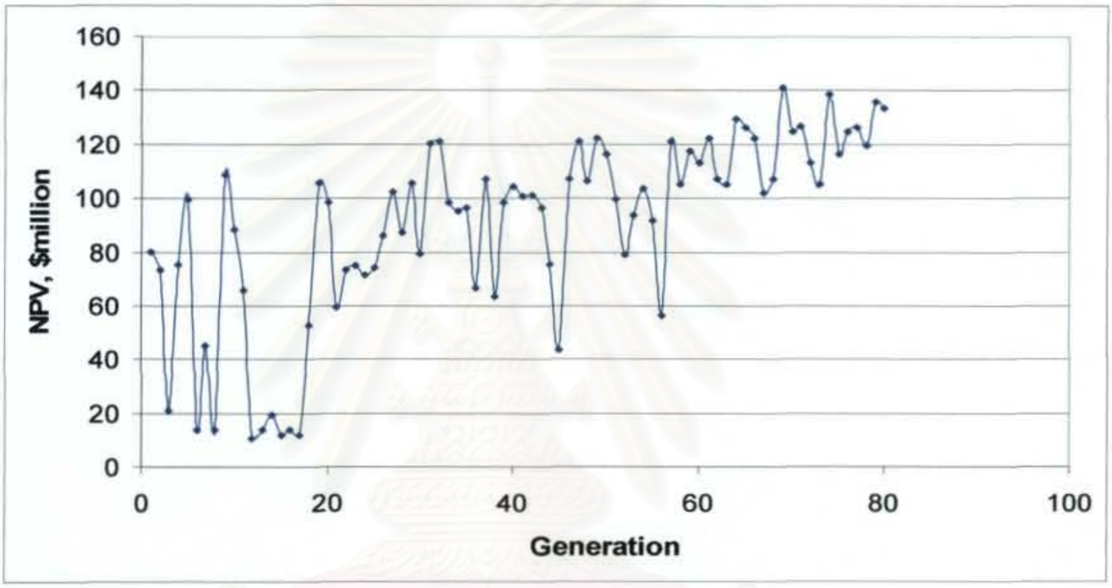

Figure: 3.4 Net present value as a function of generation.

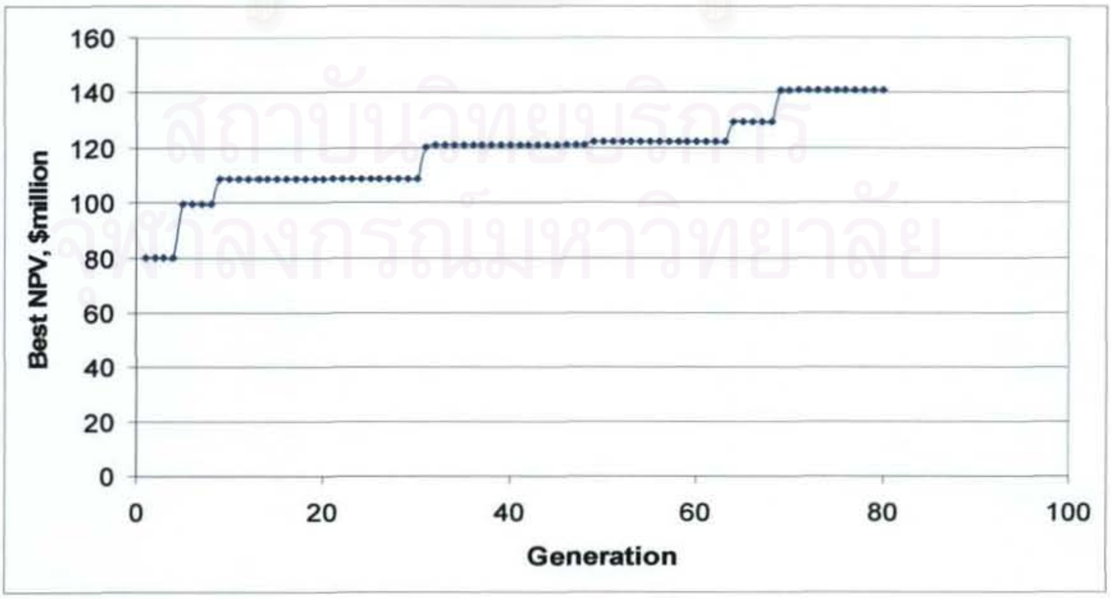

Figure: 3.5 Net present value as a function of generation (keep the best answer).

Table 3.2: Set of variables that gives the best answer of case 1.

Parameters	Value
Tubing Diameter, inch	5.5
Pipeline Diameter, inch	12
Choke1, by 64 inch	64
Choke2, by 64 inch	64
Choke3, by 64 inch	64
Choke4, by 64 inch	64
First separator pressure, psia	100
Injection rate, scf per day	0
No. of well	2
NPV, $	140,556,637

The oil and gas production rates of the best production scenario (generation 69) are shown in Figures 3.6 and 3.7, respectively. As seen in the figures, the oil and gas production rates start with very high rates and rapidly decline. This behavior gives high values of cash flows in the early years of production, resulting in a relatively high NPV.

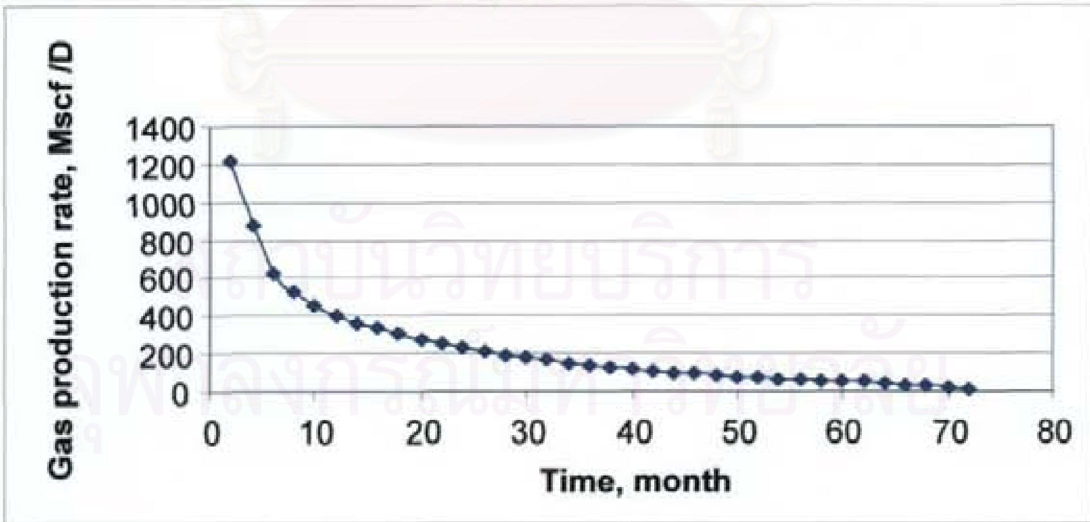

Figure 3.6: Gas production rate profile of case 1 best answer.

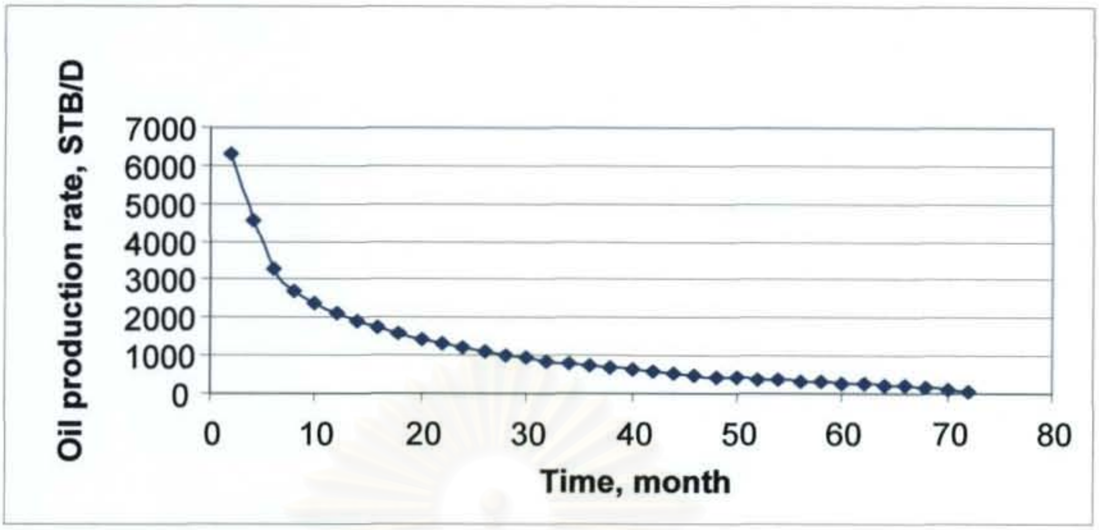

Figure 3.7: Oil production rate profile of case 1 best answer.

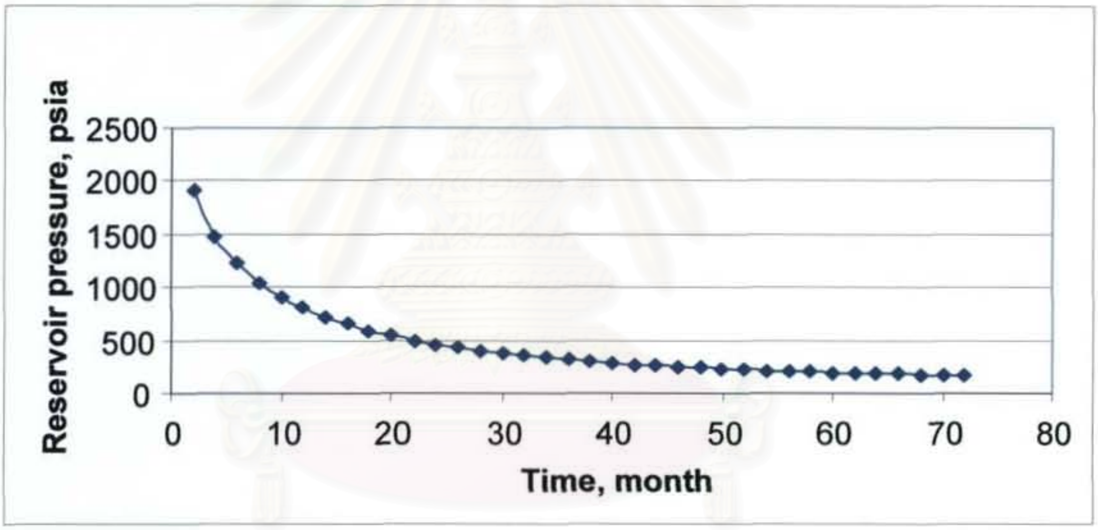

Figure 3.8: Reservoir pressure profile of case 1 best answer.

Effect of choke size

To illustrate the effect of choke size on NPV, a simulation run based on another choke profile was made so that its result can be compared with those obtained from the best scenario. In this simulation run choke diameter is kept at 32/64 choke for the first 5 years and changed to 64/64 afterward. The cumulative oil production, gas production, net income and NPV of the adjusted choke size and the best scenario (fixed choke size) are shown in Figures 3.8, 3.9, and 3.10, respectively.

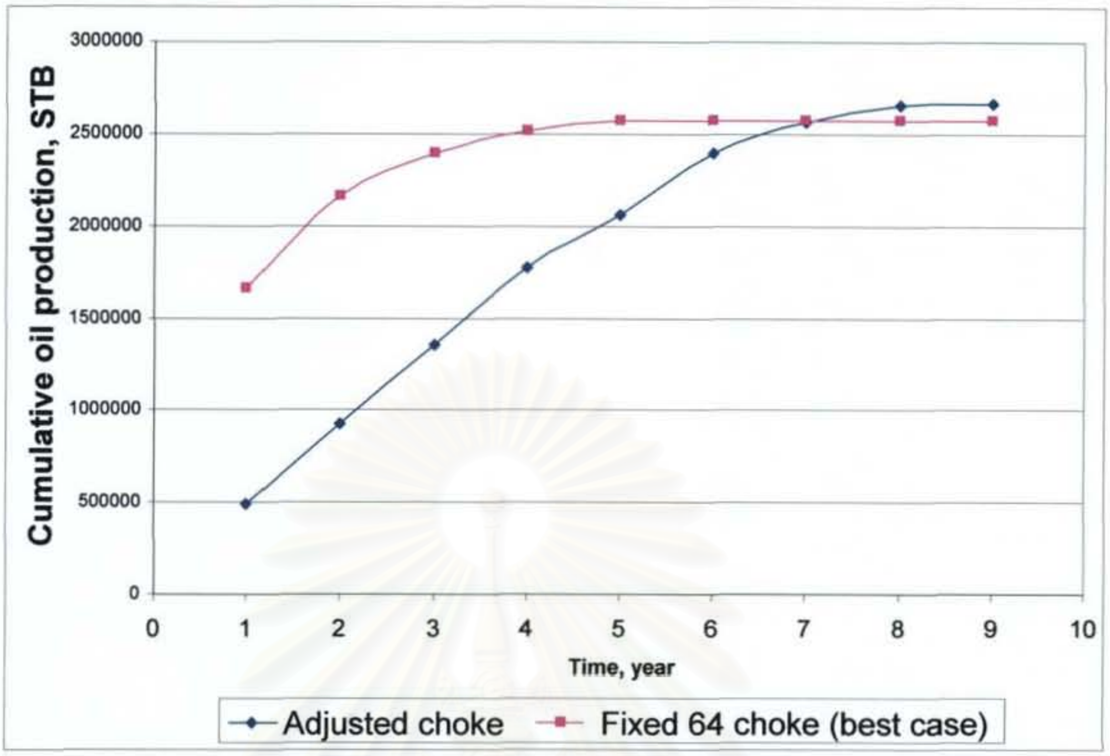

Figure 3.9: Comparison of cumulative oil productions between fixed 64 choke and 32-64 adjusted choke.

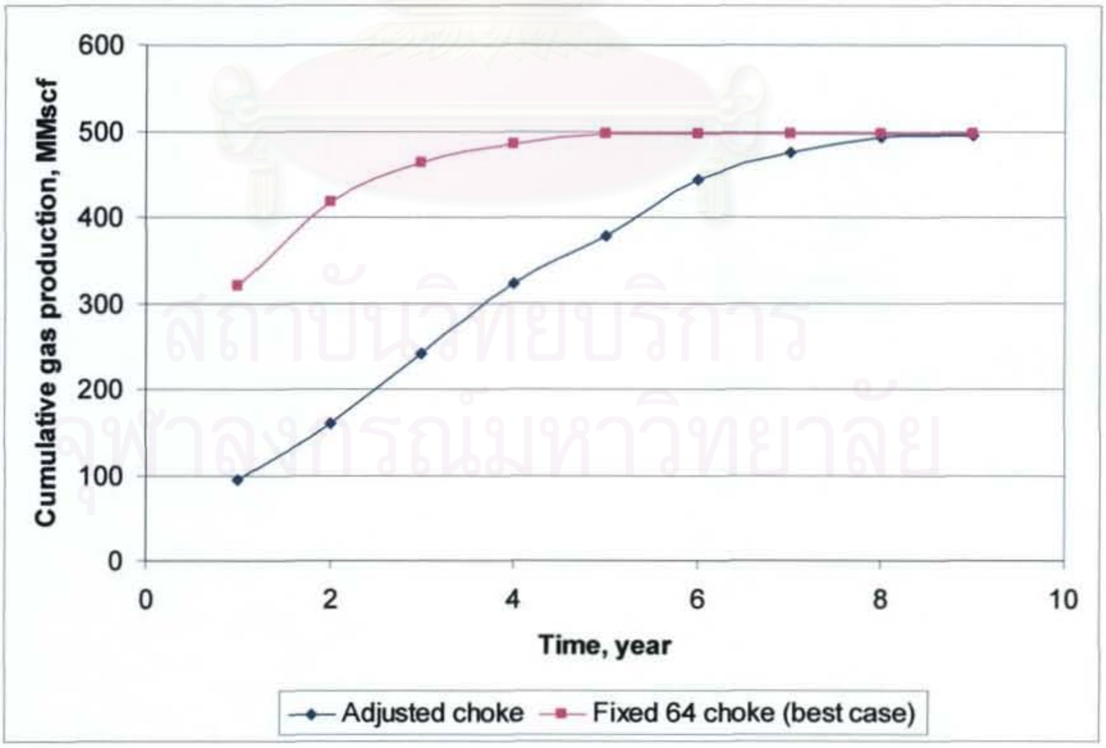

Figure 3.10: Comparison of cumulative gas productions between fixed 64 choke and 32-64 adjusted choke.

Figure 3.9 shows that, in the first seventh years, cumulative oil production of 32-64 adjusted choke is less than that of the best answer. Then on the eighth year, the cumulative oil production of 32-64 adjusted choke becomes higher. The cumulative gas production of 32-64 adjusted choke is less than that of the best answer on the first eighth year, as seen in Figure 3.10. The final cumulative income of 32-64 adjusted choke is more than that of best answer but the effect of discount rate and yearly costs makes NPV of 32-64 adjusted choke lesser, as seen in Figure 3.11. The important factors are oil and gas prices. With high oil and gas production in the early year, the cash flows are large during this early period. Large amount of money is not reduced while NPV of long-life production is less even though cumulative amount of income is more.

Effect of choke size can be seen by controlling the rate with specific economic condition does affect much on NPV in this case study.

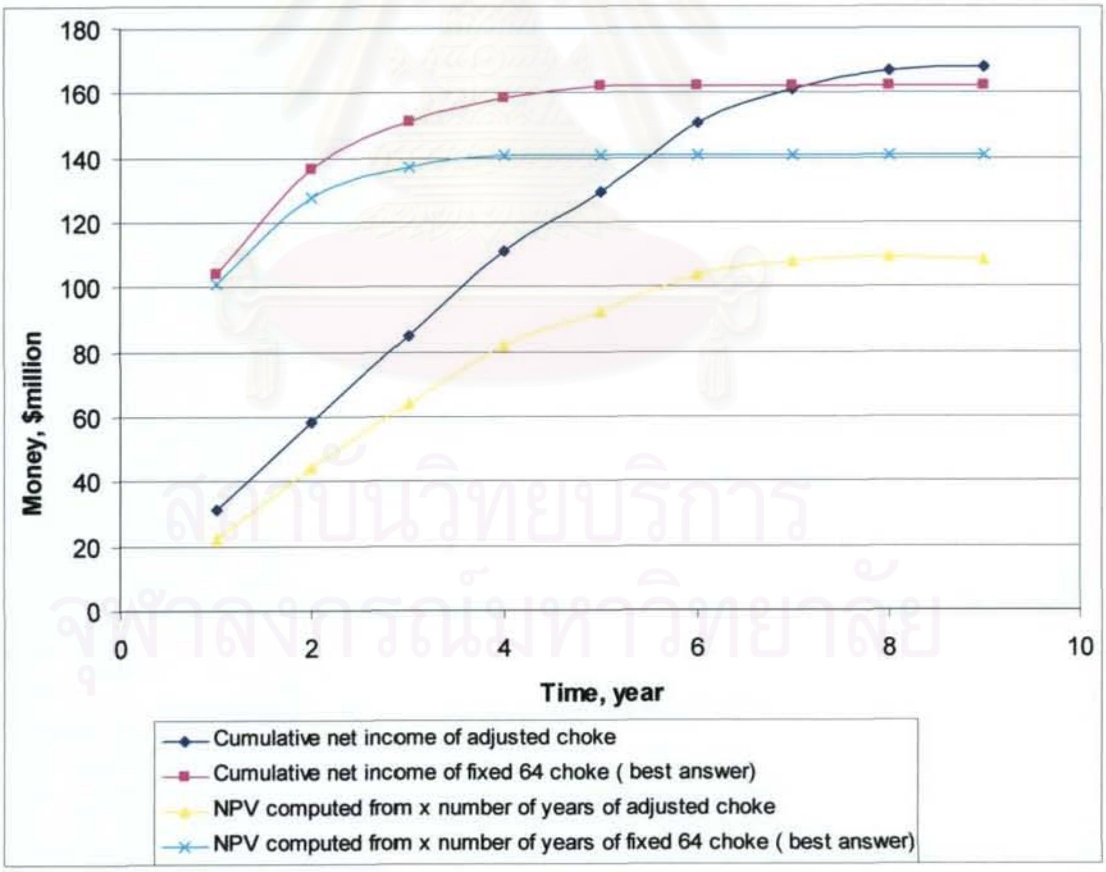

Figure 3.11: Comparison of cumulative net income and NPV between fixed 64 choke and 32-64 adjusted choke.

Effect of multiple wells

Number of production wells affects strongly on the amount of production and costs of drilling and completion, and production facility. In the best scenario (generation 69), two wells provide the best NPV. To show the effect of number of wells, another simulation with 3 wells was performed.

Figure 3.12 shows oil production profile of the case with 2 and 3 production wells. In the first year, the total oil rate from three wells is higher than that from two wells, but in the second year the total oil rate from 3 wells decreases very fast and becomes lower than the oil rate from 2 wells. Moreover, the 3-well production dies 2 year faster than the 2-well case.

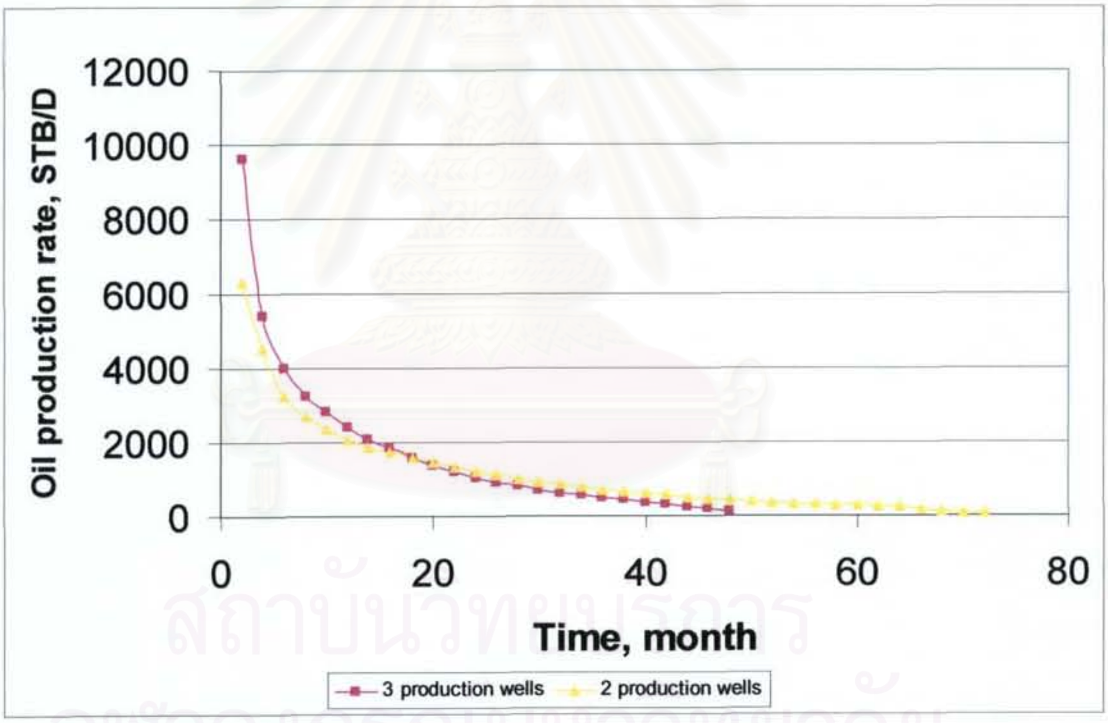

Figure 3.12: Comparison of oil production rate profile between 2 and 3 production wells.

Table 3.3: Cash flow calculation of 3 production wells.

Time year	Total sale $	Drilling and completion cost, $	Operation cost, $	Facility cost, $	Net income $	Discount rate=10%	NPV $
1	110,269,814	8,400,000	4,200,000	796,428	96,873,386	1.0000	96,873,386
2	38,438,919		4,200,000		34,238,919	1.1000	31,126,290
3	15,746,256		4,200,000		11,546,256	1.2100	9,542,360
4	6,825,971	1,500,000	4,200,000		1,125,971	1.3310	845,958
							138,387,994

It can be seen in Figure 3.12 that with three production wells, early year productions are much higher than that of the two production wells but the initial cost and operation cost are 1.5 times higher. From Tables 3.3 and 3.4, high production rate of 3-production well in the first year makes cash flow much higher than that of the best answer while drilling and completion cost and operation cost make the final NPV of the three wells lesser. Moreover, with three production wells the production depletes faster than the best answer.

Table 3.4: Cash flow calculation of case 1 best answer.

Time year	Total sale $	Drilling and completion cost, $	Operation cost, $	Facility cost, $	Net income $	Discount rate=10%	NPV $
1	86,578,000	5,600,000	2,800,000	738,809	77,439,191	1.0000	77,439,191
2	34,416,133		2,800,000		31,616,133	1.1000	28,741,939
3	25,407,555		2,800,000		22,607,555	1.2100	18,683,930
4	19,315,912		2,800,000		16,515,912	1.3310	12,408,649
5	8,606,535	1,000,000	2,800,000		4,806,535	1.4641	3,282,928
							140556637

Effect of tubing size

Figure 3.13 shows comparison of 5-inch production tubing (best case) and 2-inch production tubing. As seen in the figure, the smaller tubing size show lower production rate and longer production life. This effect is similar to choke size but much weaker.

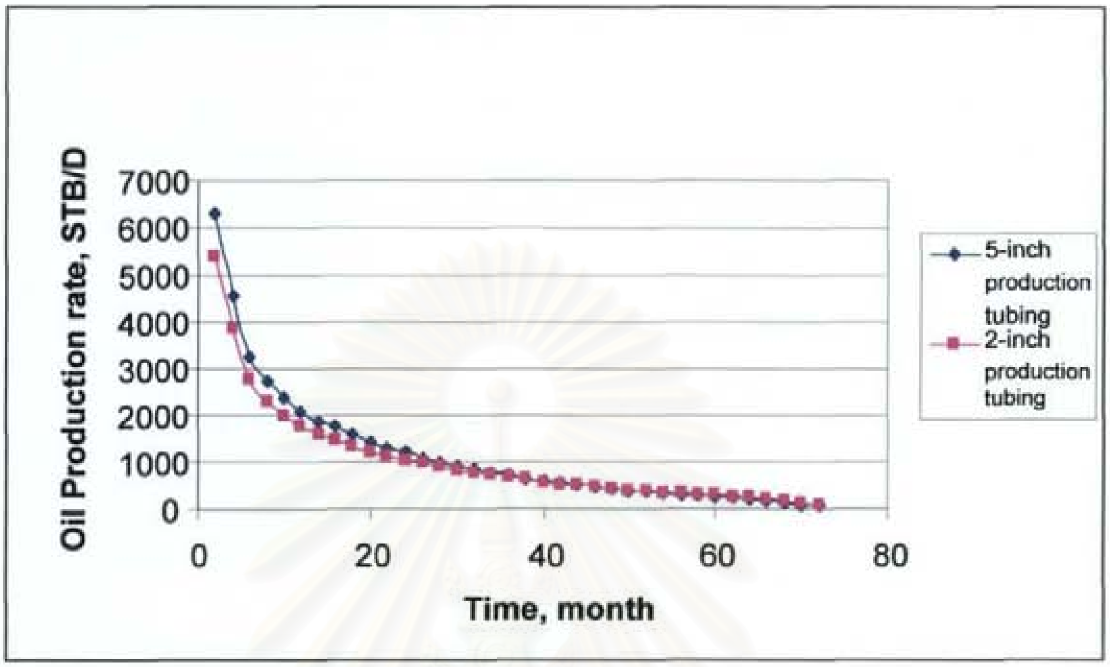

Figure 3.13: Comparison of oil production rate profile between 5-inch and 2-inch production tubing.

Effect of pipeline size

Effect of pipeline size to production rate is similar to that of the tubing size. In this case study, there is very little effect on this factor because of short pipeline distance.

3.2.2 Case 2

In this case, economic parameters are changed, in order to see more decision variable effects. Moreover, the choke size variables are set to change every 1 year. After the first four years, the choke size is kept constant. Also, the number of production wells in this case is maximum at 3 wells in order to reduce the number of possible answers.

Reservoir parameters and fluid composition of case 2 are similar to those of case 1.

Economic parameters

The differences compared to case 1 are

1) Oil and gas price

Oil price is $30 per STB.

Gas price is $7.86 per Mscf.

2) Production facility cost

Production equipment cost of oil is 540,000+5250*(Maximum oil rate, STB/day) per well.

Production equipment cost of gas is 675*(Maximum gas rate, Mscf/day) per well.

Variable inputs

Objective variables are changed from case study number 1 as follows:

1. Choke sizes are varied every 1-year period based on the following sizes: 16/64, 32/64, 48/64 and 64/64. All the wells are assumed to have the same choke size.
2. Pressure of the first separator: 100 psia, 200 psia and 300 psia.

Pressures of second and third separators are

First separator, psia	Second separator, psia	Third separator, psia
100	61	38
200	104	54
300	141	66

3. Number of wells: 1, 2, and 3.

Result and discussion

After the integrated model was run for 50 generations, 10 populations in each generation, the model had covered 500 different sets of combinations of decision variables. Each set of decision variables that yields the maximum NPV in each of the 50 generations are summarized in Table 3.5.

Table 3.5: Production parameters that provide the highest NPV in each generation.

Generation	Tubing Size	Pipeline Size	Choke1	Choke2	Choke3	Choke4	Psep	Gas inject rate	No. of well	NPV	life
	Inch	Inch					psia	Scf/day		$	year
1	4	5.5	16	64	64	64	200	100000	1	17,368,062	8
2	5.5	7	48	48	64	64	100	0	2	12,286,270	6
3	5.5	5.5	32	48	64	64	200	50000	1	12,524,867	11
4	5.5	5.5	32	32	32	32	100	0	1	11,925,792	17
5	4	7	16	32	48	48	200	0	1	17,713,347	10
6	4	4	16	32	32	32	100	0	1	12,847,544	18
7	4	5.5	48	48	48	48	100	500000	2	15,959,908	6
8	5.5	5.5	32	48	48	64	100	0	1	20,140,910	12
9	4	4	32	32	32	32	200	50000	1	14,076,841	8
10	2.875	7	32	48	48	64	300	0	1	15,356,840	5
11	5.5	7	16	32	48	48	100	50000	2	10,760,697	8
12	5.5	7	32	32	48	48	100	0	2	10,379,624	6
13	4	5.5	16	32	32	32	200	0	1	12,847,544	17
14	2.875	4	32	48	48	48	100	50000	1	17,263,385	13
15	4	5.5	48	64	64	64	100	0	3	14,573,690	3
16	4	7	48	48	64	64	100	0	1	19,257,369	10
17	2.875	7	48	48	48	64	300	0	1	17,631,330	7
18	5.5	5.5	16	48	48	48	300	100000	1	17,344,826	7
19	2.875	7	16	48	64	64	300	100000	1	17,726,931	6
20	5.5	5.5	32	32	48	48	100	0	1	16,575,392	15
21	2.875	7	64	64	64	64	100	100000	3	16,370,584	3
22	2.875	7	48	48	48	64	200	0	1	18,132,574	8
23	4	4	16	48	48	64	200	100000	1	18,642,359	8
24	4	5.5	32	64	64	64	100	50000	3	16,537,842	3
25	4	5.5	16	32	48	64	200	100000	1	19,331,526	9
26	5.5	7	48	48	48	48	100	100000	1	18,930,249	11
27	5.5	4	64	64	64	64	100	0	2	18,235,350	5
28	4	5.5	16	48	48	48	100	50000	1	18,946,309	13
29	5.5	5.5	48	64	64	64	200	0	1	19,909,974	6
30	2.875	5.5	32	48	64	64	100	0	1	20,295,210	13
31	2.875	7	16	48	48	64	100	0	1	18,946,309	17
32	4	4	32	48	64	64	200	50000	1	19,070,382	7
33	5.5	7	48	48	64	64	200	50000	3	15,370,584	4
34	4	7	32	48	48	48	100	0	1	18,263,385	13
35	2.875	5.5	16	48	48	64	100	0	1	20,853,984	11
36	2.875	7	16	64	64	64	100	50000	1	19,472,536	9
37	5.5	7	16	32	48	48	100	0	1	19,716,547	13
38	4	5.5	48	64	64	64	100	50000	2	18,235,350	5
39	4	7	32	48	48	64	100	50000	1	20,229,078	13
40	5.5	7	32	64	64	64	100	50000	1	22,125,346	11
41	5.5	5.5	16	32	48	64	100	100000	1	22,371,895	14
42	5.5	7	16	32	48	64	100	500000	1	23,496,154	13
43	2.875	7	32	48	48	64	100	500000	1	20,270,382	13
44	4	7	32	64	64	64	100	50000	1	21,671,137	12
45	4	7	32	64	64	64	100	50000	1	22,364,943	13
46	2.875	7	64	64	64	64	100	0	1	21,471,280	10
47	5.5	7	16	48	64	64	100	500000	1	21,123,456	12

Table 3.5: Production parameters that provide the highest NPV in each generation (continued).

Generation	Tubing Size	Pipeline Size	Choke1	Choke2	Choke3	Choke4	Psep	Gas inject rate	No. of well	NPV	life
48	5.5	7	16	32	48	64	100	0	1	22,110,224	12
49	4	7	16	48	48	64	100	100000	1	21,198,851	11
50	5.5	5.5	16	48	48	64	100	100000	1	22,488,723	12
Final	5.5	7	16	32	48	64	100	500000	1	23,496,154	13

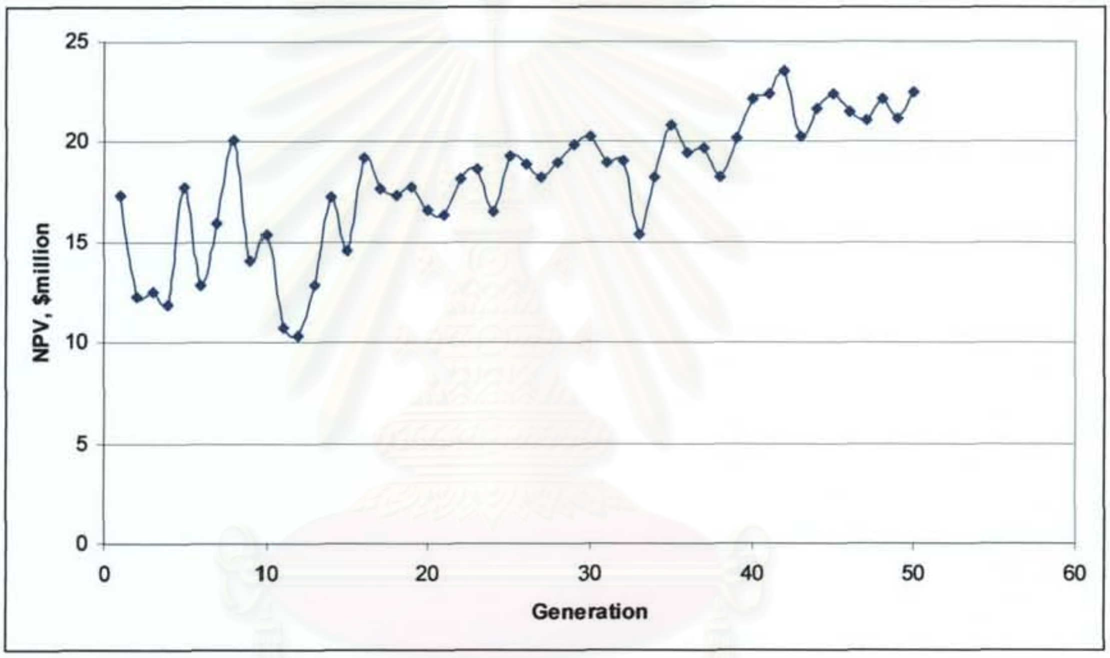

Figure 3.14: Net present value as a function of generation.

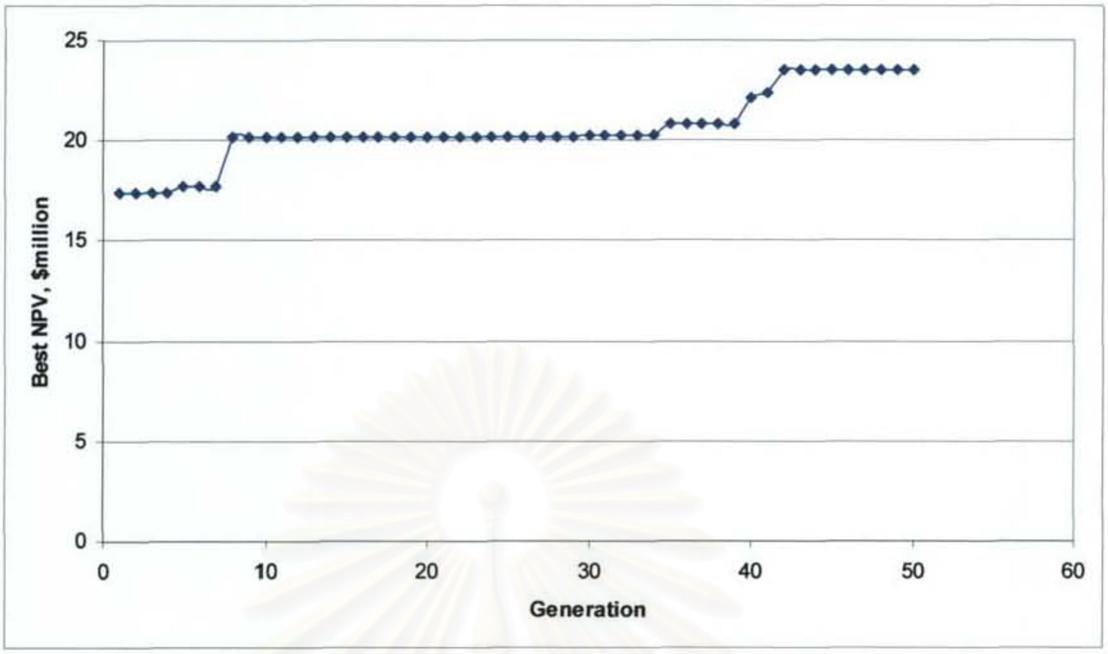

Figure 3.15: Net present value as a function of generation (keep the best answer).

From Figure 3.14, similar to the results from case 1, the value of the best NPV in the first 15 generations fluctuates. Then, it tends to increase with lesser difference when compared to that of the previous generation. Figure 3.15 shows that in the last 9 generations, NPV does not increase. The maximum NPV was found at generation 42 with NPV equal to $23,496,154. The set of decision variables in the fittest solution is shown in Table 3.6.

Table 3.6: Set of variables that gives the best answer of case 2.

Parameters	Value
Tubing Diameter, inch	5.5
Pipeline Diameter, inch	7
Choke1, by 64 inch	16
Choke2, by 64 inch	32
Choke3, by 64 inch	48
Choke4, by 64 inch	64
First separator pressure, psia	100
Injection rate, scf per day	50,000
No. of well	1
NPV, $	23,496,154

The oil and gas production rates of the best production scenario (generation 42) are shown in Figures 3.16 and 3.18, respectively. The reservoir pressure is shown in Figure 3.18.

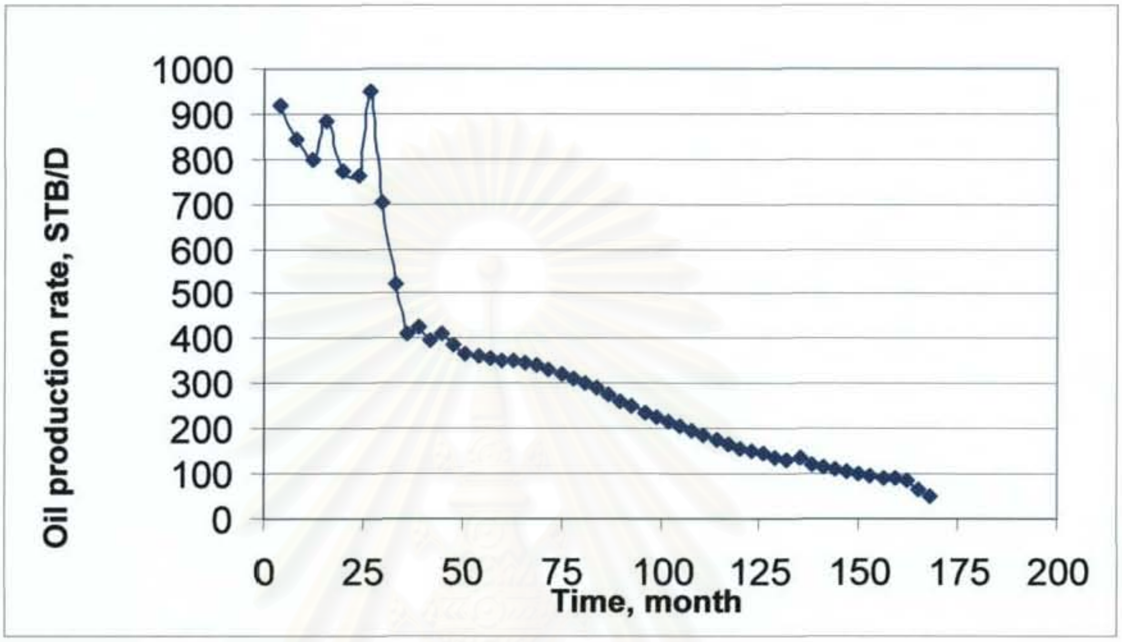

Figure 3.16: Oil production rate profile of case 2 best answer.

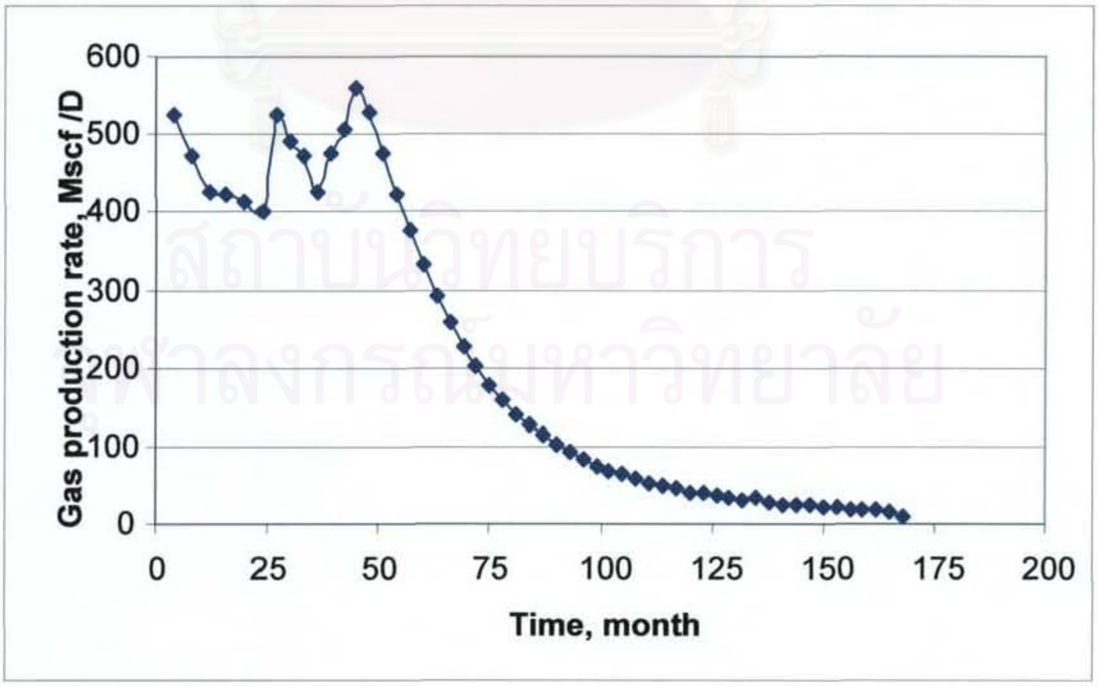

Figure 3.17: Gas production rate profile of case 2 best answer.

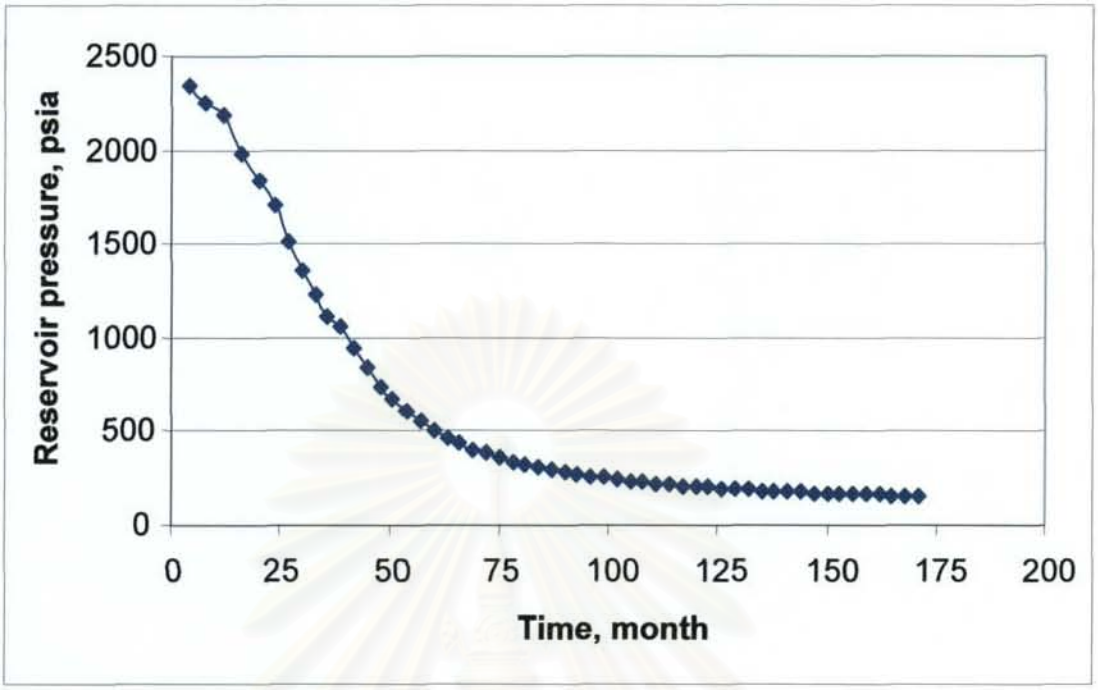

Figure 3.18: Reservoir pressure profile of case 2 best answer.

Effect of choke size

In this case study, production facility cost is adjusted such that it heavily depends on maximum oil and gas production rates.

From Figures 3.16 and 3.17, it can be seen that by adjusting the choke size, the production rate can be maintained at one rate. While the production rate of the fixed choke size, shown in Figure 3.19, in the first year starts at a very high rate and decreases very fast. This incurs a high but unnecessary cost for production facility. The cost in the case of fixed 64/64 choke size is more than that of the best answer for $7,600,510. The final NPV is lower than that of the best answer.

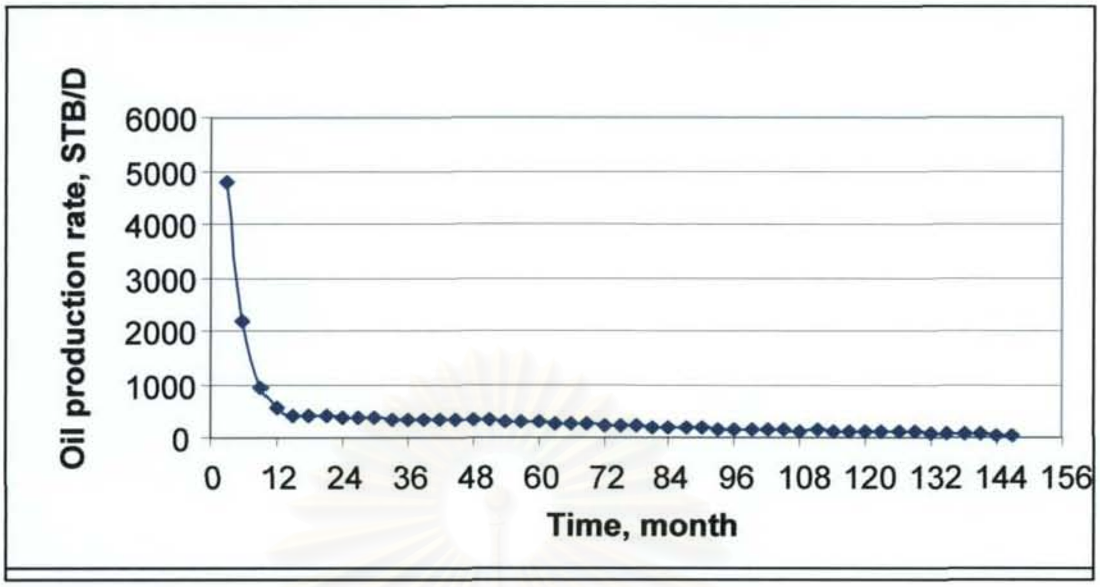

Figure 3.19: Oil production profile of producing with fixed 64 choke size.

Effect of gas injection

With gas lift, oil and gas rates are increased, as shown in Figure 3.20. The increase in production does not affect pressure across choke because the flow is subcritical. On the other hand, the higher the production rate, the faster the reservoir pressure decreases, as depicted Figure 3.21.

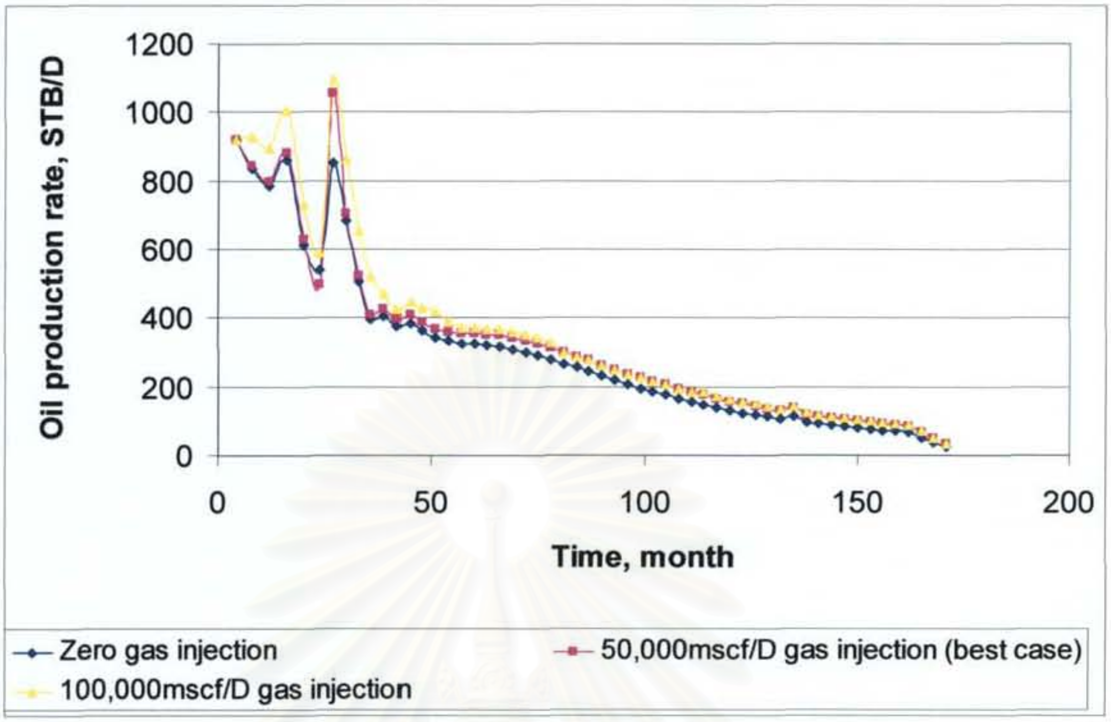

Figure 3.20: Comparison of oil production rate profile between wells with 50 Mscf per day gas injection, 100 Mscf per day and without gas injection.

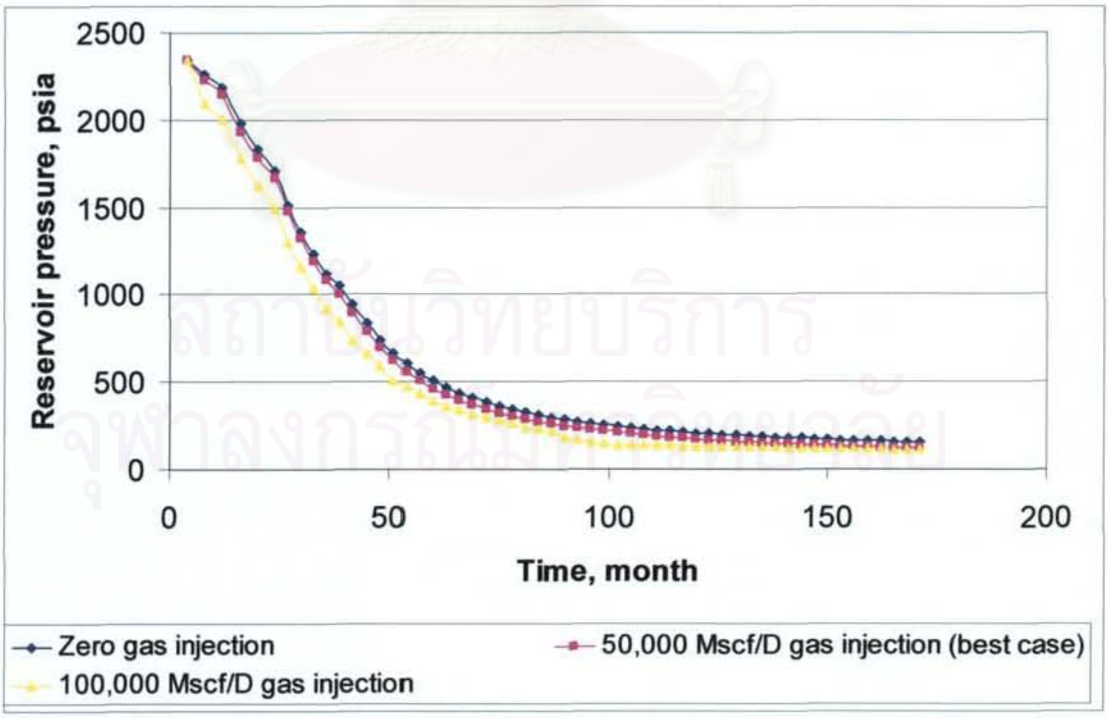

Figure 3.21: Comparison of reservoir pressure profile between wells with 50 Mscf per day gas injection, 100 Mscf per day and without gas injection.

Effect of multiple wells

In this case, oil and gas prices are lower than those of case 1. Then, multiple wells do not work well in this case. To see this effect, economic parameters in the best answer are compared to those when these are 2 production wells. From Tables 3.7 and 3.8, the costs of 2 production wells are twice the cost in the case of the best answer. This reason causes NPV of 2 production wells to be less than that of the best answer.

Table 3.7: Cash flow calculation of case 2 best answer.

Time Year	Total sale $	Drilling and completion cost, $	Operation cost, $	Facility cost, $	Net income $	Discount rate=10%	NPV $
1	8,664,184	2,800,000	1,400,000	4,621,825	-157,641	1.0000	-157,641
2	7,809,148		1,400,000		6,409,148	1.1000	5,826,498
3	7,443,059		1,400,000		6,043,059	1.2100	4,994,263
4	8,696,617		1,400,000		7,296,617	1.3310	5,482,056
5	5,314,706		1,400,000		3,914,706	1.4641	2,673,797
6	4,405,075		1,400,000		3,005,075	1.6105	1,865,915
7	3,743,269		1,400,000		2,343,269	1.7716	1,322,714
8	3,054,930		1,400,000		1,654,930	1.9487	849,241
9	2,460,559		1,400,000		1,060,559	2.1436	494,759
10	1,982,187		1,400,000		582,187	2.3579	246,904
11	1,612,878		1,400,000		212,878	2.5937	82,074
12	1,373,812	500,000	1,400,000		-526,188	2.8531	-184,426
							23,496,154

Table 3.8: Cash flow calculation of 2 production wells.

Time Year	Total sale $	Drilling and completion cost, $	Operation cost, $	Facility cost, $	Net income $	Discount rate=10%	NPV $
1	13,155,184	5,888,500	2,800,000	6,726,147	- 2,259,463	1.0000	-2,259,463
2	10,871,210		2,800,000		8,071,210	1.1000	7,337,464
3	8,200,165		2,800,000		5,400,165	1.2100	4,462,946
4	8,652,509		2,800,000		5,852,509	1.3310	4,397,077
5	6,385,004		2,800,000		3,585,004	1.4641	2,448,606
6	4,093,080	1,000,000	2,800,000		293,080	1.6105	181,980
							11,490,609

Effect of tubing size and pipeline size

Results show similar effect to Case 1.

Effect of first separator pressure

To see effect of first separator pressure, comparisons of production rates and gas oil ratios at different first separator pressures are shown in Figures 3.22 and 3.23. Sets of decision variables are as follows:

Case 2.1

1) 5.5-inch OD tubing size
2) Fixed 64/64 choke size
3) 5.5-inch OD pipeline size
4) One production well
5) Zero gas injection
6) 100 psia first separator pressure

Case 2.2

1) 5.5-inch OD tubing size

2) Fixed 64/64 choke size

3) 5.5-inch OD pipeline size

4) One production well

5) Zero gas injection

6) 200 psia first separator pressure

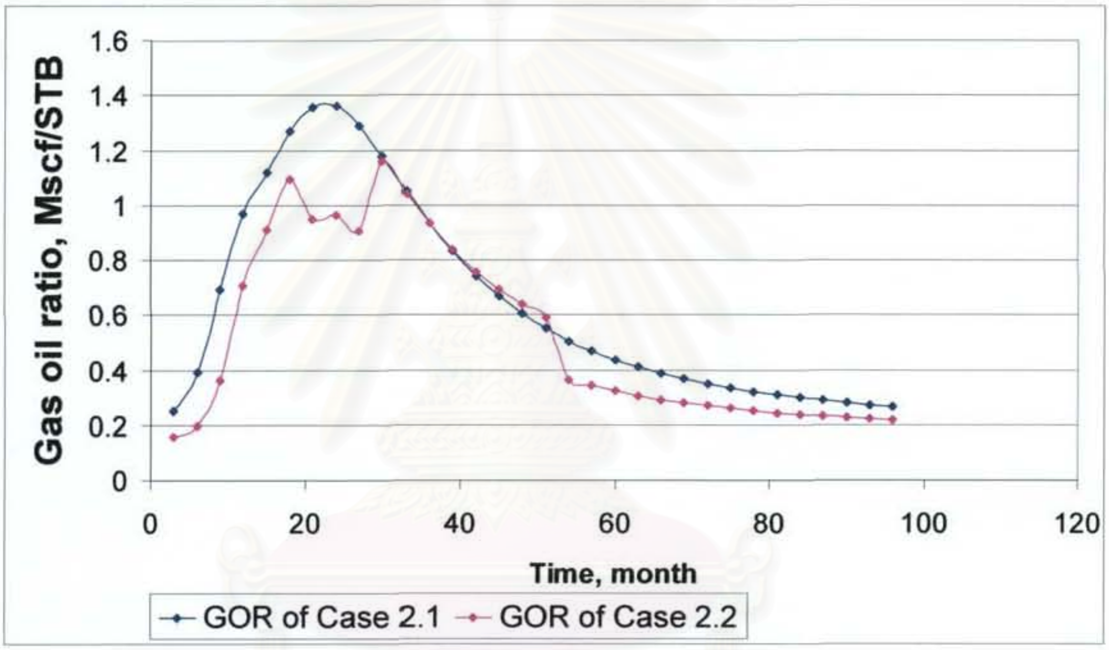

Figure 3.22: Comparison of gas oil ratio profiles between Case 2.1 and Case 2.2.

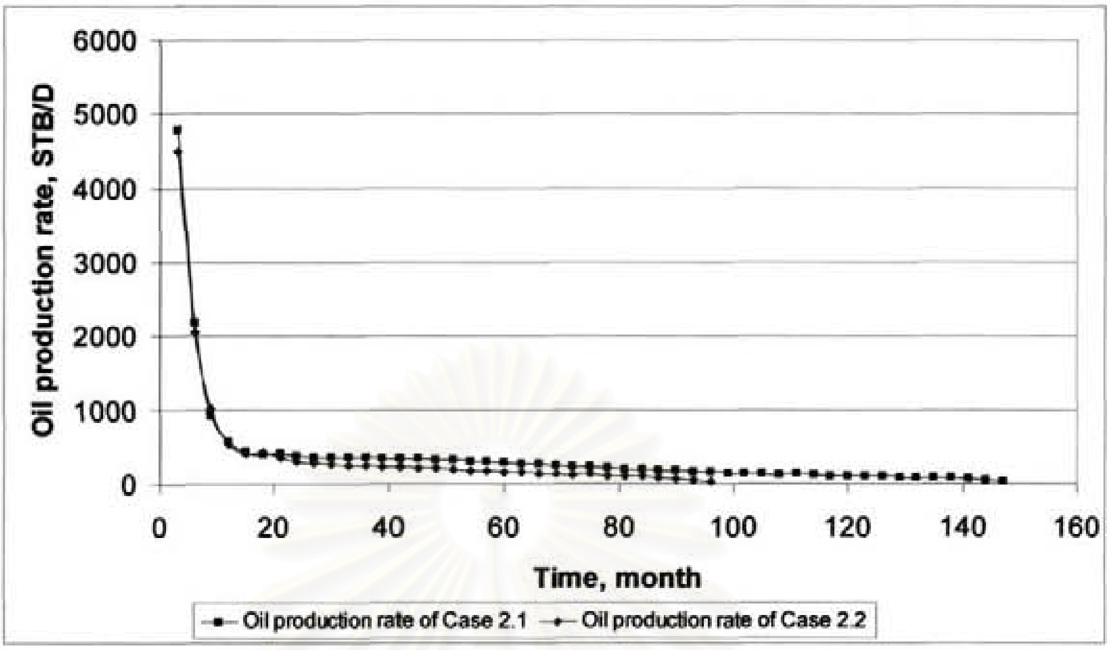

Figure 3.23: Comparison of oil production profiles between Case 2.1 and Case 2.2.

From Figures 3.22 and 3.23, gas oil ratio of Case 2.2 is less than that of Case 2.1 while oil production rate of Case 2.2 is less than that of Case 2.1. From the example, the result shows that with higher separator pressure, production gas oil ratio is lesser. Nature of hydrocarbon, gas oil ratio decreases with increasing of pressure until it reaches one pressure the gas oil ratio increases with increasing of pressure.

3.2.3 Case 3

In this case study, fluid composition is changed to be black oil composition. This is to see the effect of gas injection. The decision variable inputs and economic input are similar to those of the case 2.

The parameters that are different from those in case 1 are:

Initial reservoir pressure = 2500 psi

Reservoir fluid composition [6] is

Component C1 = 36.47%

Component C2 = 9.67%

Component C3 = 6.95%

Component i-C4 = 3.30%

Component n-C4 = 2.07%

Component i-C5 = 1.00%

Component n-C5 = 1.85%

Component C6 = 4.33%

Component C7+ = 33.29%

Molecular weight of C_{7+} is 218 lb/mole

Economic parameters

Economic parameters of case 3 are similar to those of case 2.

Variable inputs

The differences of variable inputs comparing to case 1 are:

1. Choke sizes are varied every 1-year period based on the following sizes: 16/64, 32/64, 48/64 and 64/64. All the wells are assumed to have the same choke size.
2. Pressure of the first separator: 100 psia, 200 psia and 300 psia.
3. Gas injection rate: Zero, 100000 scf/day and 200000 scf/day. All the wells are assumed to have same gas injection rate.
4. Number of wells: 1, 2, and 3.

Result and discussion

Like case 2, the integrated model was run for 50 generations, 10 populations in each generation. The model was run with 500 sets of combinations of model variables to get the fittest answer. Each set of decision variables that yield the maximum NPV in each of the 50 generations are summarized in Table 3.10.

Table 3.10: Production parameters that provide the highest NPV in each generation.

Generation	Tubing Size	Pipeline Size	Choke1	Choke2	Choke3	Choke4	Psep	Gas inject rate	No. of well	NPV	life
	Inch	Inch					psia	scf/day		$	year
1	4	5.5	32	32	48	64	100	100000	1	5,423,200	4
2	2.875	7	32	32	64	64	100	0	1	5,747,759	4
3	5.5	5.5	32	32	32	32	100	0	1	3,564,304	5
4	5.5	7	48	48	48	48	100	100000	1	5,861,974	2
5	4	7	16	32	48	48	200	0	1	3,214,758	4
6	4	4	32	32	32	32	200	100000	1	4,028,117	5
7	2.875	7	32	48	48	64	300	0	1	5,677,259	5
8	4	5.5	16	48	64	64	200	0	1	4,586,432	4
9	2.875	7	48	48	48	64	300	0	1	6,846,331	3
10	5.5	5.5	16	48	48	48	100	100000	1	4,113,820	4
11	2.875	7	16	48	64	64	100	200000	1	4,290,017	3
12	5.5	7	32	32	48	48	100	0	1	5,382,117	4
13	4	4	32	48	64	64	200	100000	1	6,849,250	3
14	5.5	5.5	32	32	32	32	100	0	1	4,320,015	4
15	4	7	48	48	64	64	100	0	1	7,032,341	3
16	2.875	5.5	16	48	48	64	100	200000	1	3,014,855	5
17	4	4	32	32	4 8	48	100	200000	1	6,932,247	3
18	4	4	32	48	64	64	200	200000	1	7,614,431	4
19	2.875	7	16	32	48	64	100	0	1	6,659,864	4
20	4	7	32	48	48	48	100	0	1	5,677,310	4
21	5.5	7	16	48	64	64	100	200000	1	4,759,921	4
22	5.5	7	32	32	48	48	100	0	1	5,382,117	4
23	5.5	5.5	32	48	64	64	200	0	1	6,541,932	3
24	5.5	7	32	32	48	48	100	200000	1	5,882,385	4
25	4	5.5	64	64	64	64	100	0	1	5,861,994	3
26	4	4	32	48	64	64	200	100000	1	6,849,250	3
27	2.875	4	32	48	48	48	100	100000	1	5,677,481	4
28	2.875	7	16	32	48	64	100	0	1	6,659,856	4
29	4	4	32	48	64	64	200	200000	1	7,614,431	4
30	2.875	7	48	48	48	64	200	200000	1	7,345,292	3
31	5.5	7	32	48	48	64	100	100000	1	6,298,015	3
32	5.5	7	32	64	64	64	100	100000	1	6,797,498	3
33	2.875	7	32	48	64	64	100	200000	1	8,532,214	3
34	4	4	32	48	64	64	200	100000	1	6,849,250	3
35	2.875	7	48	64	64	64	100	200000	1	8,324,769	3
36	4	4	32	48	64	64	200	200000	1	7,614,431	4
37	5.5	7	32	32	64	64	200	200000	1	6,351,192	4
38	2.875	5.5	48	48	64	64	100	100000	1	7,832,511	3
39	2.875	7	64	64	64	64	100	200000	1	7,590,451	3
40	4	4	16	32	48	64	100	100000	1	6,959,611	4

Table 3.10: Production parameters that provide the highest NPV in each generation (continued).

Generation	Tubing Size	Pipeline Size	Choke1	Choke2	Choke3	Choke4	Psep	Gas inject rate	No. of well	NPV	life
41	4	7	48	64	64	64	100	0	1	7,932,218	3
42	2.875	7	48	48	64	64	100	0	1	6,945,742	3
43	5.5	7	48	64	64	64	100	200000	1	10,503,214	3
44	4	4	48	48	64	64	100	200000	1	7,836,002	4
45	4	5.5	48	64	64	64	100	200000	1	9,825,350	5
46	5.5	5.5	48	64	64	64	100	200000	1	7,928,864	3
47	2.875	7	32	48	64	64	100	200000	1	8,532,214	3
48	5.5	5.5	32	48	64	64	100	200000	1	7,832,219	3
49	5	7	48	48	64	64	100	0	1	7,032,219	3
50	2.875	7	48	64	64	64	100	200000	1	8,324,769	3
Final	5.5	7	48	64	64	64	100	200000	1	10,503,214	3

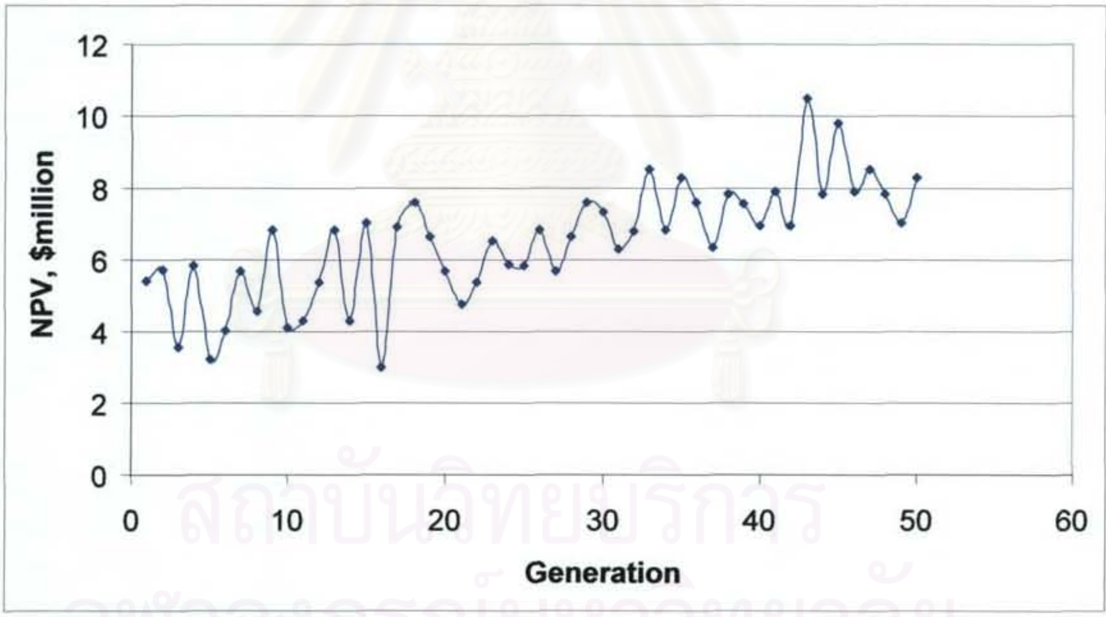

Figure 3.24: Net present value as a function of generation.

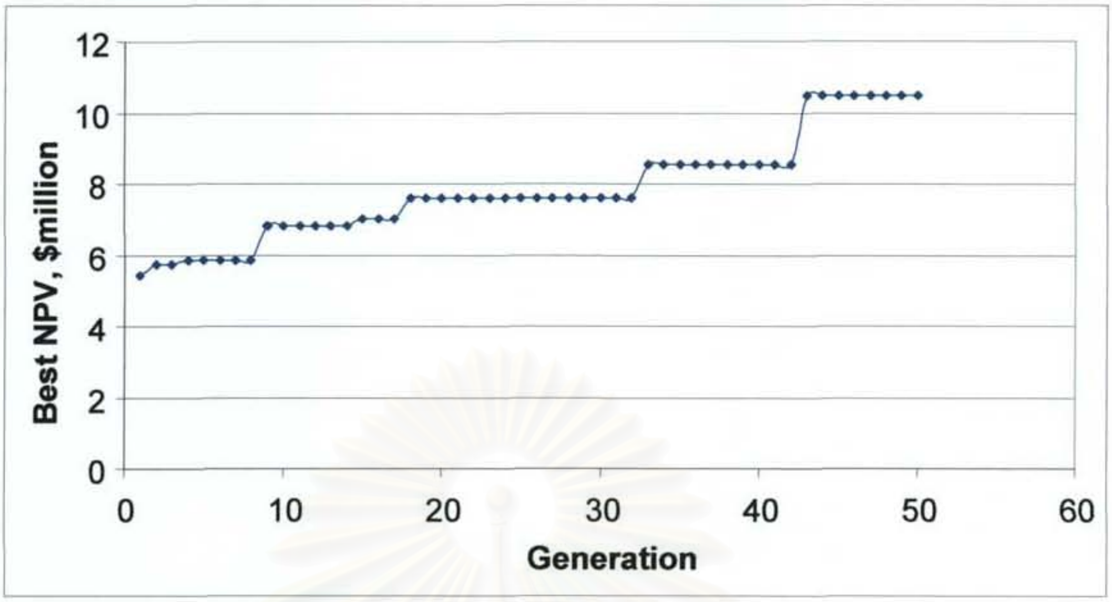

Figure 3.25: Net present value as a function of generation (keep the best answer).

From Figure 3.24, similar to the results of case 1 and 2, the value of the best NPV in the first 20 generations fluctuates. Then it tends to increase with lesser difference when compared to last generation. Figure 3.25 shows that in the last 8 generations, the NPV does not increase. The maximum NPV was equal to $10,503,214 and found in generation 43. The fittest set of decision variables is shown in Table 3.9.

Table 3.9: **Set of variables that gives the best answer of case 3.**

Parameters	Value
Tubing Diameter, inch	5.5
Pipeline Diameter, inch	7
Choke1, by 64 inch	48
Choke2, by 64 inch	64
Choke3, by 64 inch	64
Choke4, by 64 inch	64
First separator pressure, psia	100
Injection rate, scf per day	200,000
No. of well	1
NPV, $	10,503,214

The oil and gas production rates of the best production scenario (generation 43) are shown in Figures 3.26 and 3.27, respectively. Figure 3.28 shows reservoir pressure profile.

Effect of gas injection

From Figures 3.26 and 3.27, it can be clearly seen that with 200 Mscf per day of gas injection, the oil production rate is highest among zero injection and 100 Mscf per day gas injection. Moreover, from Figure 3.28, reservoir pressure of 200 Mscf per day of gas injection well is also highest. The main reason is that the reservoir fluid is dence. The dence and viscous oil causes high pressure losses along wellbore, mostly hydrostatic pressure loss. Then with gas injection, density of the fluid is less and the hydrostatic head loss is also reduced. While others effects are quite similar to those of case1 and 2.

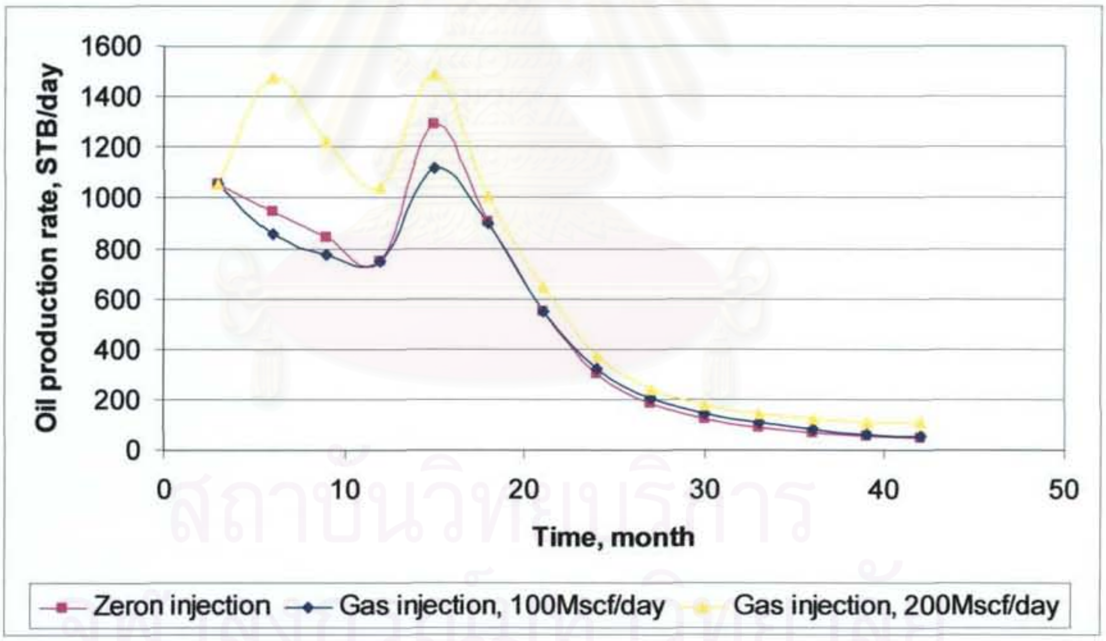

Figure 3.26: Comparison of oil production profile between zero gas injection, 100-Mscf per day gas injection and 200-Mscf per day gas injection.

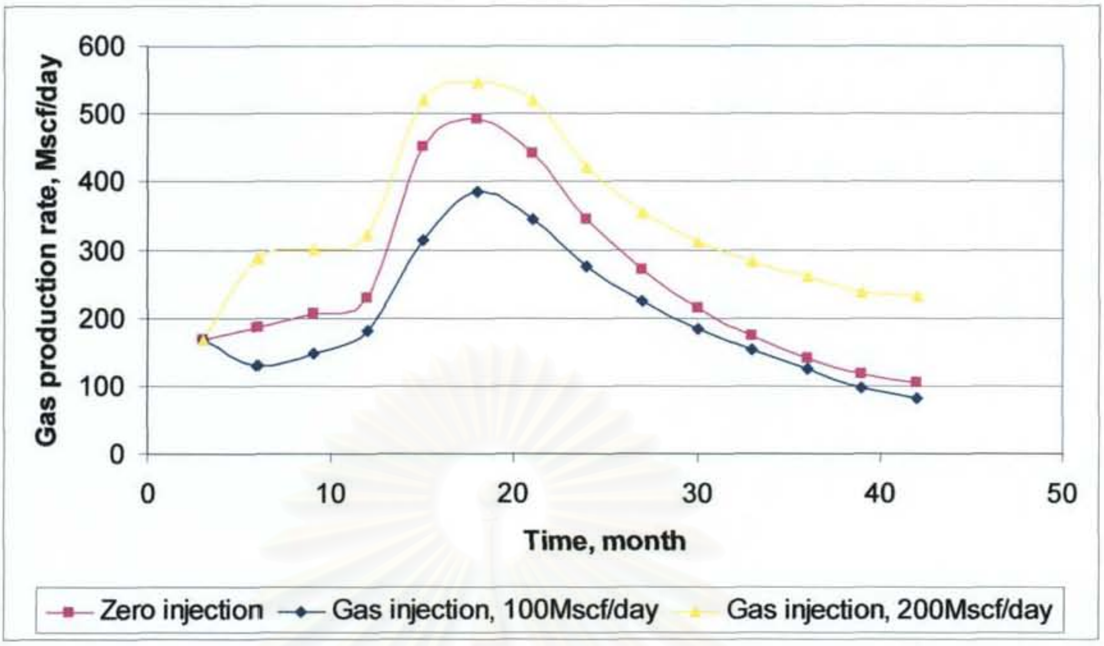

Figure 3.27: Comparison of gas production profile between zero gas injection, 100-Mscf per day gas injection and 200-Mscf per day gas injection.

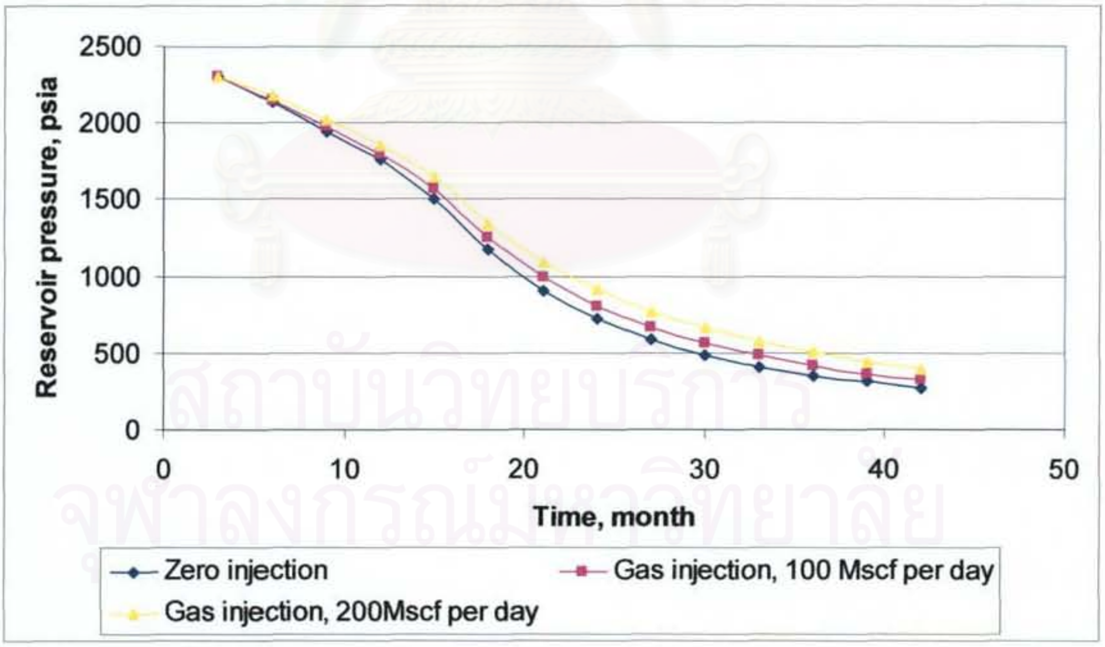

Figure 3.28: Comparison of reservoir pressure profile between zero gas injection, 100-Mscf per day gas injection and 200-Mscf per day gas injection.

3.3 Summary of model testing and case study

The integrated model was run with one set of decision variables and compared with the results to those from ECLIPSE. The maximum difference of flow rate from the integrated model is 15 percent which is considered acceptable.

Three case studies are run. The results show effects of each decision variable to the objective function. The final answer, maximum NPV, is determined by genetic algorithm. By the concept of genetic algorithm, the solution should be better with time except when the global optimum has already been reached. Anyway, since there is no convergence in genetic algorithm, stop condition depends on user criterion. Then, if the better solution is needed, more time needs to be sacrificed. The answer of each case studies might not be the best solution but it is considered satisfied with its progress and number of cases run.

CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS

The objective of this study is to optimize NPV by designing well completion and production system. The integrated model was constructed in order to find the production profile. Reservoir model, wellbore model, choke model and separator model are combined. In the reservoir model, volumetric equation is used to calculate the flow rate. Aziz, Govier and Fogarasi multiphase flow correlation is used in the wellbore model. Sachdeva *et al.* correlation is used in the choke model. In all detailed models, fluid properties are computed in fluid property model.

Then production profile is sent to economic model for NPV calculation. Overall, genetic algorithm uses a set of decision variables to be input into the integrated model. Solution of each generation does improve by genetic algorithm and it is stopped by the preset value number of generations.

4.1 Conclusions

For conclusion, with known and constant reservoir parameters; such as permeability, porosity, etc., there are 3 main topics to mention about:

1) Effect of different factors on production profile

Before discussing on the net present value which is the objective function in this study, the production profile which is the key to get the net present value needs to be discussed first.

In the three case studies, control variables which are tubing size, choke size, pipeline size, first separator pressure, amount of injected gas, and number of wells have effect on production profile in their own ways.

Tubing and pipeline size have effects on pressure loss along the flow path. With the same production rate, small tubing size causes more pressure loss. Then its flow rate is less; and the reservoir pressure reduces at a slower pace. A smaller tubing size gives a lower rate and longer life than a larger tubing size. This effect also depends on length of tubing.

In this study, choke sizes are varied into 4 sizes, 25%, 50% 75% and 100% of fully opened choke size. Choke size has similar effect to tubing size but with much more sensitively, with varied sizes in this study. The smaller the choke size is, the lower the production rate and the longer the reservoir life.

Separator pressure affects directly to gas oil ratio. In this study, the case study shows that gas oil ratio decreases with increasing of separator pressure. Anyway by nature of hydrocarbon, gas oil ratio does decrease with increasing of pressure until it reaches one specific pressure, gas oil ratio increases with increasing of pressure.

From the three case studies, gas lift does not work well with wells with high gas oil ratio or low fraction of heptanes plus. While on the third case study, with 33.29 percent of heptanes plus gas lift yields very good result.

Producing with multi production wells make production rate higher than producing with one well. The production rate is not increased linearly because of different drainage area.

2) Effect of different factors on net present value (NPV)

Besides effect on the production profile, costs affect directly to net present value. The net present value is also affected by production profile.

Effect of discount rate makes production sale in early year more valuable than the same amount of production in later years. Producing with high rate in the early year does make very high production sales in cash flow. On the other hand, high production rate causes high production facility cost; and drilling and completion cost in case of multiple production wells.

Wells with gas lift facility have higher production rate and longer life. Economically, even there is more production sale, gas lift costs more completion cost, facility cost and operation cost.

High separator pressure makes more oil sale but less sale life.

Finally from the three case studies, with different values of these economic factors, the set of studied factors that gives the maximum net present value is completely different. Then economic assumption; such as oil price, gas price, costs and discount rate, could be assumed as the most important factor on NPV.

3) Result of genetic algorithm

As discussed previously, effect of each factor can be described individually while the effect of all parameters to NPV is very difficult to be described. Then, genetic algorithm is used in order to find the optimal NPV.

From the three case studies, the genetic algorithm shows improvement of solution in each generation. It is the nature of genetic algorithm that, in the early generations, improvement of solution is quite fast and then the solution shows improvement less often. Since the solution keeps being better, there is no evidence that which solution is the optimal one. A better solution will be obtained if the genetic algorithm is run for longer period of time as long as the global optimum is not reached. However, the true global optimum is not known. Therefore, we may or may not get a better solution as we continue running the algorithm. This is the disadvantage of genetic algorithm.

In general, the number of generation to be run is preset before the simulation was run. The satisfaction of solution is considered by the improvement of the solution, number of case run, the value of the solution and available time. Finally, if a better solution is required, more time needs to be sacrificed.

4.2 Recommendations

Recommendations for future study are outlined as follows:

1) Sensitivity of economic factors

Studying on effect of economic factors will make result more adaptive to real situations.

2) Reservoir simulation

Instead of using volumetric model, reservoir simulation provides more accurate results of fluid flow in the reservoir.

3) Non-constant reservoir parameter

In real production field, reservoir parameters such as permeability is not constant.

4) Hybrid optimization method

Instead of using a single algorithm, a hybrid optimization method might give better solutions.

5) Gas injection rate as a function of oil rate

Calculating the amount of injected gas as a function of oil rate is more appropriate for gas

References

(1) Carroll, J.A. "*Multivariate Production Systems Optimization*", Master of Science Thesis, Stanford University, Stanford, California, 1990.

(2) Fujii, H. "*Multivariate Production Systems Optimization in Pipeline Networks*", Master of Science Thesis, Stanford University, Stanford, California, 1993.

(3) Palke, M. R. "*Nonlinear Optimization of Well Production Considering Gas Lift and Phase Behavior*", Master of Science Thesis, Stanford University, California, 1996.

(4) Wilson, G. M., and Deal, C. H. "*Ind. Eng. Chem. Fundam.*", 1962.

(5) Redlich, O. and Kwong, J.N.S. "*On the Thermodynamics of Solution. V-An Equation of State. Fugacities of Gaseous Solutions*" Chem. Reviews (1949).

(6) McCain, W. D. "*The Properties of Petroleum Fluids*", PennWell, Tulsa, 1990.

(7) Aziz, K., Govier, G.W., and Fogarasi, M. "Pressure *Drop in Wells Producing Oil and Gas*", J. Canadian Petro. Tech., 1972.

(8) Duns, H., Jr. and Ros, N. C. J. "*Vertical Flow of Gas and Liquid Mixtures in Wells*" Proc., 6th World Petroleum Congress, 1963, pp. 451.

(9) Sachdeva, R., Schmidt, Z., Brill, J.P., and Blais, R.M. "*Two-Phase Flow Through Chokes*", SPE 15657, presented at SPE Annual Technical Conference and Exhibition, 1986.

(10) James, P., Brill, and Hemanta, Mukherjee. "*Multiphase flow in wells*", SPE Monograph volume 17, Richardson, Texas, 1999.

(11) Campbell, J. M. "*Gas Conditioning and Processing, Vol. 2*", Campbell Petroleum Series, Norman, Oklahoma, 398pp, 1984.

(12) Chewaroungroaj, J. Advanced Petroleum Economics, unpublished class notes, Chulalongkorn University, Bangkok, 2003.

(13) Thompson, R.S., Wright, J.D. "*Oil Property Evaluation*", Thompson-Wright Associates, Golden CO, 1985.

(14) Goldberg, D. E. "*Genetic Algorithms in Search, Optimization, and Machine Learning*", Addison-Wesley Publishing, 1989.

(15) Marczyk A., Wilke C.O. "*Genetic Algorithms and Evolutionary Computation*", http://www.talkorigins.org, 2004.

(16) Amyx, J.W., Bass, D.M., and Whiting, R.L. "*Petroleum Reservoir Engineering*", New York, McGraw-Hill Book Company, 1960.

(17) Brown, K. E., and Beggs, H. D. "*The Technology of Artificial Lift Method Volume 1*", Tulsa, 1977.

(18) Economides, M.J., Hill, A.D., and Ehlig-Economides, C. "*Petroleum Production Systems*", Englewood Cliffs, New Jersey: PTR Prentice Hall, 1994.

(19) Energy Information Administration. "*Documentation of the Oil and Gas Supply Module (OGSM)*", DOE/EIA-M063, Washington, DC, January 2001

APPENDICES

APPENDIX A

Table A1: Comparing results of testing model with results from ECLIPSE.

RESULTS FROM ECLIPSE				RESULTS FROM INTEGRATED MODEL			
TIME, DAYS	Gas Production Rate, MSCF/DAY	Oil Production Rate, STB/DAY	Bottom Hole Pressure, PSIA	TIME DAYS	Gas Production Rate, MSCF/DAY	Oil Production Rate, STB/DAY	Bottom Hole Pressure, PSIA
0.5	4000.00	498.28	2875.63	2	4115.20	517.49	2728
1.0	4000.00	498.28	2857.42	4	3991.50	501.93	2649
2.0	4000.00	498.28	2838.88	6	3935.22	494.85	2589
3.6	4000.00	498.28	2814.08	8	4007.38	503.93	2530
5.2	4000.00	498.28	2790.57	10	3993.91	502.23	2477
6.1	4000.00	498.28	2777.26	12	4037.53	507.72	2430
7.0	4000.00	498.28	2763.99	14	4043.09	508.42	2388
8.6	4000.00	498.28	2741.38	16	3990.10	501.75	2351
10.1	4000.00	498.28	2719.09	18	3937.11	495.09	2318
11.7	4000.00	498.28	2697.10	20	3984.11	501.00	2288
12.8	4000.00	498.28	2680.53	22	3880.42	487.96	2260
14.0	4000.00	498.28	2664.04	24	3857.33	485.06	2233
15.5	4000.00	498.28	2642.81	26	3740.73	470.40	2210
17.0	4000.00	498.28	2621.87	28	3915.60	492.39	2188
18.5	4000.00	498.28	2601.21	30	3956.40	497.52	2167
19.8	4000.00	498.28	2584.48	32	3912.49	492.00	2148
21.0	4000.00	498.28	2567.85	34	3919.48	492.87	2129
22.5	4000.00	498.28	2547.91	36	4492.00	564.87	2111
24.0	4000.00	498.28	2528.23	38	4041.44	508.21	2094
25.4	4000.00	498.28	2508.80	40	4004.00	503.50	2079
26.7	4000.00	498.28	2491.92	42	4271.56	537.15	2063
28.0	4000.00	498.28	2475.15	44	4049.38	509.21	2049
29.4	4000.00	498.28	2456.41	46	4390.12	552.06	2035
30.9	4000.00	498.28	2437.90	48	4224.40	531.22	2021
32.3	4000.00	498.28	2419.63	50	3991.06	501.88	2007
33.7	4000.00	498.28	2402.61	52	3949.30	496.62	1994
35.0	4000.00	498.28	2385.69	54	3901.24	490.58	1981
36.4	4000.00	498.28	2368.06	56	3941.47	495.64	1967
37.8	4000.00	498.28	2350.65	58	4049.00	509.16	1954
39.2	4000.00	498.28	2333.45	60	4098.72	515.41	1942
40.6	4000.00	498.28	2316.45	62	3676.76	462.35	1928
41.3	4000.00	498.28	2307.84	64	4249.12	534.33	1912
42.0	4000.00	498.28	2299.24	66	3916.37	492.48	1878
43.4	4000.00	498.28	2282.68	68	3932.01	490.00	1842
44.7	4000.00	498.28	2266.29	70	4467.88	474.62	1831
46.1	4000.00	498.28	2250.08	72	3992.89	466.06	1809
47.4	4000.00	498.28	2234.06	74	4258.74	439.57	1799
48.2	4000.00	498.28	2224.79	76	4049.38	431.29	1761
49.0	4000.00	498.28	2215.55	78	4389.01	419.89	1755
50.3	4000.00	498.28	2199.95	80	4224.40	405.21	1724
51.7	4000.00	498.28	2184.50	82	3991.06	400.21	1697
53.0	4000.00	498.28	2169.22	84	3949.30	385.88	1676
54.3	4000.00	498.28	2154.10	86	3901.24	381.39	1663
55.1	4000.00	498.28	2144.23	88	3941.47	376.75	1655
56.0	4000.00	498.28	2134.40	90	4049.00	368.62	1644
57.3	4000.00	498.28	2119.68	92	4098.72	360.90	1641
58.6	4000.00	498.28	2105.11	94	3676.76	352.55	1634
59.9	4000.00	495.23	2088.67	96	4249.12	339.09	1628
60.5	4000.00	488.92	2080.92	98	3916.37	333.59	1608
61.7	4000.00	478.38	2067.42	100	3932.01	331.29	1580
63.0	4000.00	474.62	2052.78	102	4492.00	322.59	1562
63.6	4000.00	473.89	2044.51	104	3912.49	315.07	1545
64.9	4000.00	473.89	2044.56	106	3919.48	309.94	1526
65.5	4000.00	466.07	2027.05	108	4492.00	302.60	1508
66.1	4000.00	466.06	2027.12	110	4041.40	299.94	1490
67.4	4000.00	465.95	2026.60	112	4376.00	298.32	1463
68.0	4000.00	455.56	2007.32	114	4245.36	285.47	1445
69.0	4000.00	446.84	1991.44	116	4050.16	279.12	1425
70.0	4000.00	439.57	1979.26	118	4364.94	277.19	1416

RESULTS FROM ECLIPSE				RESULTS FROM INTEGRATED MODEL			
TIME, DAYS	Gas Production Rate, MSCF/DAY	Oil Production Rate, STB/DAY	Bottom Hole Pressure, PSIA	TIME DAYS	Gas Production Rate, MSCF/DAY	Oil Production Rate, STB/DAY	Bottom Hole Pressure, PSIA
71.2	4000.00	431.29	1964.73	120	4213.40	270.85	1399
72.5	4000.00	425.41	1950.83	122	3902.00	263.99	1381
73.7	4000.00	419.89	1937.76	124	3938.70	261.02	1366
74.9	4000.00	414.45	1925.45	126	3935.81	256.57	1342
75.9	4000.00	409.84	1915.08	128	3940.59	250.63	1316
77.0	4000.00	405.21	1904.91	130	4049.84	253.13	1300
78.2	4000.00	400.21	1893.83	132	4107.20	250.43	1283
79.3	4000.00	395.28	1882.83	134	3647.60	245.65	1267
80.5	4000.00	390.53	1871.93	136	4249.12	244.70	1250
81.6	4000.00	385.88	1861.06	138	3916.37	243.21	1242
82.8	4000.00	381.39	1850.25	140	3932.01	242.68	1207
83.4	4000.00	379.04	1844.45	142	4467.88	241.03	1199
84.0	4000.00	376.75	1838.71	144	3992.89	238.71	1158
85.1	4000.00	372.60	1828.20	146	4258.74	233.54	1136
86.3	4000.00	368.62	1817.71	148	4049.38	236.99	1126
87.4	4000.00	364.71	1807.25	150	4389.01	233.73	1130
88.5	4000.00	360.90	1796.85	152	4224.40	233.80	1106
89.6	4000.00	357.19	1786.49	154	3991.06	229.93	1079
90.3	4000.00	354.85	1779.87	156	3949.30	231.93	1069
91.0	4000.00	352.55	1773.27	158	3901.24	230.20	1055
92.1	4000.00	349.06	1763.10	160	3941.47	224.84	1026
93.2	4000.00	345.65	1752.93	162	4033.00	225.95	991
94.3	4000.00	342.33	1742.80	164	4098.72	224.56	973
95.3	4000.00	339.09	1732.72	166	3676.00	221.06	959
96.4	4000.00	335.92	1722.68	168	4249.12	222.47	920
97.2	4000.00	333.59	1715.14	170	3916.80	219.71	923
98.0	4000.00	331.29	1707.60	172	3932.01	222.10	880
99.1	4000.00	328.33	1697.72	174	4144.81	223.29	863
100.1	4000.00	325.43	1687.86	176	3884.92	226.18	852
101.1	4000.00	322.59	1678.03	178	3914.80	224.44	809
102.2	4000.00	319.82	1668.24	180	4199.00	229.08	817
103.1	4000.00	317.41	1659.57	182	4128.97	228.37	799
104.0	4000.00	315.07	1650.85	184	4163.40	231.23	790
105.0	4000.00	312.48	1641.07	186	3932.59	236.28	773
106.0	4000.00	309.94	1631.32	188	4033.84	238.47	764
107.0	4000.00	307.45	1621.61	190	4100.85	240.31	742
108.0	4000.00	305.00	1611.94	192	3680.33	244.36	714
109.0	4000.00	302.60	1602.30	194	4249.12	234.65	686
110.0	4000.00	300.28	1592.80	196	4026.25	227.07	673
111.0	4000.00	297.99	1583.29	198	3902.50	222.75	534
112.0	4000.00	295.74	1573.77	200	3819.70	216.39	511
113.0	4000.00	293.54	1564.28	202	3626.03	209.13	499
113.9	4000.00	291.38	1554.82	204	3487.56	198.07	484
114.9	4000.00	289.27	1545.40	206	3423.03	190.67	472
115.8	4000.00	287.21	1536.01	208	3294.73	184.55	462
116.8	4000.00	285.20	1526.66	210	3165.12	177.13	460
117.4	4000.00	283.93	1520.73	212	2968.06	171.75	459
118.0	4000.00	282.69	1514.80	214	2862.27	163.19	458
118.9	4000.00	280.78	1505.57	216	2737.86	157.88	458
119.9	4000.00	278.92	1496.35	218	2615.93	152.08	456
120.8	4000.00	277.09	1487.15	220	2620.00	148.26	456
121.7	4000.00	275.30	1477.97	222	2596.96	141.16	456
122.7	4000.00	273.55	1468.82	224	2487.33	135.58	455
123.6	4000.00	271.85	1459.71	226	2334.41	130.29	454
124.3	4000.00	270.53	1452.53	228	2223.85	121.66	454
125.0	4000.00	269.23	1445.35	230	2020.12	120.81	453
125.9	4000.00	267.63	1436.35	232	1945.80	117.34	452
126.8	4000.00	266.06	1427.35	234	1797.48	110.89	451
127.7	4000.00	264.53	1418.38	236	1812.68	110.28	451
128.6	4000.00	263.03	1409.44	238	1726.08	107.16	451
129.5	4000.00	261.57	1400.52	240	1720.84	101.63	451
130.3	4000.00	260.14	1391.63	242	1614.56	95.43	450
132.0	4000.00	257.50	1374.72				

RESULTS FROM ECLIPSE			
TIME, DAYS	Gas Production Rate, MSCF/DAY	Oil Production Rate, STB/DAY	Bottom Hole Pressure, PSIA
133.7	4000.00	254.87	1357.14
134.6	4000.00	253.60	1348.39
135.4	4000.00	252.36	1339.67
136.3	4000.00	251.14	1330.97
137.1	4000.00	249.96	1322.30
137.9	4000.00	248.80	1313.65
138.5	4000.00	248.08	1308.13
139.0	4000.00	247.36	1302.62
139.8	4000.00	246.27	1294.08
140.6	4000.00	245.21	1285.54
141.5	4000.00	244.17	1277.01
142.3	4000.00	243.15	1268.50
143.1	4000.00	242.16	1260.02
143.9	4000.00	241.19	1251.55
144.7	4000.00	240.25	1243.11
145.3	4000.00	239.48	1236.07
146.0	4000.00	238.72	1229.02
146.8	4000.00	237.85	1220.66
147.6	4000.00	236.99	1212.31
148.3	4000.00	236.16	1203.99
149.1	4000.00	235.35	1195.68
149.9	4000.00	234.56	1187.39
150.7	4000.00	233.80	1179.12
151.4	4000.00	233.05	1170.87
152.2	4000.00	232.33	1162.64
152.6	4000.00	231.93	1158.05
153.0	4000.00	231.54	1153.47
153.8	4000.00	230.86	1145.28
154.5	4000.00	230.20	1137.06
155.3	4000.00	229.56	1128.78
156.0	4000.00	228.93	1120.48
156.8	4000.00	228.33	1112.15
157.5	4000.00	227.74	1103.79
158.3	4000.00	227.18	1095.40
159.0	4000.00	226.63	1086.98
159.5	4000.00	226.29	1081.36
160.0	4000.00	225.95	1075.74
160.8	4000.00	225.46	1067.28
161.5	4000.00	225.00	1058.77
162.3	4000.00	224.56	1050.22
163.0	4000.00	224.13	1041.62
163.8	4000.00	223.73	1032.98
164.5	4000.00	223.36	1024.30
165.3	4000.00	223.00	1015.60
166.0	4000.00	222.67	1006.86
166.5	4000.00	222.47	1001.09
167.0	4000.00	222.27	995.32
167.8	4000.00	222.00	986.51
168.5	4000.00	221.81	977.65
169.3	4000.00	221.83	968.75
170.0	4000.00	221.95	959.85
170.8	4000.00	222.10	950.91
171.5	4000.00	222.29	941.92
172.3	4000.00	222.50	932.92
173.0	4000.00	222.74	923.86
173.5	4000.00	222.92	917.95
174.0	4000.00	223.11	912.03
174.8	4000.00	223.44	902.82
175.5	4000.00	223.80	893.53
176.3	4000.00	224.19	884.18
177.0	4000.00	224.63	874.77
177.8	4000.00	225.10	865.32

RESULTS FROM ECLIPSE			
TIME, DAYS	Gas Production Rate, MSCF/DAY	Oil Production Rate, STB/DAY	Bottom Hole Pressure, PSIA
179.3	4000.00	226.18	846.23
180.0	4000.00	226.78	836.58
180.5	4000.00	227.19	830.32
181.0	4000.00	227.63	824.06
181.8	4000.00	228.33	814.33
182.5	4000.00	229.08	804.54
183.3	4000.00	229.89	794.69
184.0	4000.00	230.74	784.78
184.8	4000.00	231.65	774.82
185.5	4000.00	232.62	764.81
186.3	4000.00	233.64	754.75
187.0	4000.00	234.72	744.64
187.5	4000.00	235.45	738.27
188.0	4000.00	236.28	731.26
188.8	4000.00	237.56	720.88
189.5	4000.00	238.89	710.70
190.3	4000.00	240.27	700.65
191.0	4000.00	241.69	690.60
191.8	4000.00	243.52	677.72
192.5	4000.00	244.95	668.16
193.3	4000.00	195.96	660.38
194.0	4000.00	234.80	649.23
194.5	4000.00	235.01	642.07
195.0	4000.00	235.65	634.89
195.8	4000.00	236.78	623.56
196.5	4000.00	237.98	612.09
197.3	4000.00	239.24	600.48
198.0	4000.00	240.57	588.72
198.8	4000.00	241.98	576.81
199.5	4000.00	243.46	564.75
200.3	4000.00	245.01	552.53
201.0	4000.00	246.65	540.13
201.5	4000.00	247.74	532.12
202.0	4000.00	248.86	524.08
202.8	4000.00	250.69	511.34
203.5	3937.84	248.68	500.00
204.3	3892.74	246.66	500.00
205.0	3813.01	242.40	500.00
205.8	3738.17	238.39	500.00
206.6	3666.01	234.50	500.00
207.4	3595.42	230.68	500.00
208.2	3528.90	227.07	500.00
209.0	3464.38	223.55	500.00
209.9	3396.63	219.84	500.00
210.7	3329.53	216.14	500.00
211.6	3263.01	212.46	500.00
212.5	3196.99	208.78	500.00
213.5	3126.78	204.98	500.00
214.4	3059.86	201.23	500.00
215.2	3004.45	198.08	500.00
216.0	2950.80	194.99	500.00
217.0	2886.34	191.25	500.00
218.0	2822.54	187.49	500.00
219.1	2759.35	183.72	500.00
220.1	2696.83	179.95	500.00
221.2	2633.65	171.75	500.00
222.1	2583.84	167.51	500.00
223.0	2535.49	164.70	500.00
224.2	2474.20	161.28	500.00
225.4	2413.39	157.88	500.00
226.6	2353.04	154.50	500.00
227.8	2293.11	151.13	500.00
228.9	2242.48	148.26	500.00

RESULTS FROM ECLIPSE			
TIME, DAYS	Gas Production Rate, MSCF/DAY	Oil Production Rate, STB/DAY	Bottom Hole Pressure, PSIA
231.3	2135.13	142.16	500.00
232.7	2077.48	138.87	500.00
234.1	2020.24	135.58	500.00
235.6	1964.44	132.36	500.00
237.0	1910.56	129.23	500.00
238.5	1854.83	125.98	500.00
240.1	1799.56	122.74	500.00
241.8	1744.71	119.52	500.00
242.9	1707.71	117.34	500.00
244.0	1671.91	115.21	500.00
245.8	1618.82	112.04	500.00
247.6	1566.13	108.87	500.00
249.3	1518.00	105.95	500.00
251.0	1471.72	103.13	500.00

APPENDIX B

Source code of the integrated model

```
Assemblyinfo
--------------------------------------------------------------------
Imports System
Imports System.Reflection
Imports System.Runtime.InteropServices
' General Information about an assembly is controlled through the following
' set of attributes. Change these attribute values to modify the information
' associated with an assembly.
' Review the values of the assembly attributes
<Assembly: AssemblyTitle("")>
<Assembly: AssemblyDescription("")>
<Assembly: AssemblyCompany("")>
<Assembly: AssemblyProduct("")>
<Assembly: AssemblyCopyright("")>
<Assembly: AssemblyTrademark("")>
<Assembly: CLSCompliant(True)>
'The following GUID is for the ID of the typelib if this project is exposed to COM
<Assembly: Guid("864D0154-DC0E-487B-87A0-0FFFEB0C23AA")>
' Version information for an assembly consists of the following four values:
'     Major Version
'     Minor Version
'     Build Number
'     Revision
' You can specify all the values or you can default the Build and Revision Numbers
' by using the '*' as shown below:
<Assembly: AssemblyVersion("1.0.*")>
```

```
CalZsu
  Public Function Run(ByVal pathfile As PathFile, ByVal n As Integer, ByVal ZSumNp As Double, ByVal Zpi() As Double, _
  ByVal Qinj_Zinput As Double, ByVal Z_inj() As Double, ByVal Psep2 As Double, ByVal Tatm2 As Double, ByVal pc() As Double, _
  ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, ByVal inflcon As Double, _
  ByVal M() As Double, ByVal Liqstand() As Double) As Double()
    Dim ZNpi() As Double
    Dim ZNgasi() As Double
    Dim ZNp_Ngas() As Double
    Dim ZsumNp_Ngas As Double = 0
    Dim t_inj As Double = 0
    Dim O_Zu() As Double
    Dim Pg2 As Double = 0
    Dim Mg2 As Double = 0
    Dim Objf As New CompositionModel
    Objf.Run(pathfile, 0.5, n, Z_inj, Psep2, Tatm2, pc, tc, omega, R, outflcon, inflcon, M, Liqstand)
      Pg2 = Objf.GasDensity
    For ist As Integer = 0 To n - 1          Mg2 = Mg2 + Objf.y(ist) * M(ist)
    Next
    For i As Integer = 0 To n - 1
      ReDim Preserve ZNpi(i)
      ZNpi(i) = ZSumNp * Zpi(i)
      ReDim Preserve ZNgasi(i)
      t_inj = ConvertqpToNg(Qinj_Zinput, Mg2, Pg2)
      ZNgasi(i) = t_inj * Z_inj(i)
      ReDim Preserve ZNp_Ngas(i)
      ZNp_Ngas(i) = ZNpi(i) + ZNgasi(i)
      ZsumNp_Ngas = ZsumNp_Ngas + ZNp_Ngas(i)
    Next
    For i As Integer = 0 To n - 1
      ReDim Preserve O_Zu(i)
      If ZsumNp_Ngas = 0 Then
        O_Zu(i) = 0
      Else
        O_Zu(i) = ZNp_Ngas(i) / ZsumNp_Ngas
      End If
    Next
    WriteOutput(pathfile, n, O_Zu)
    Return O_Zu
  End Function
  Private Function ConvertqpToNg(ByVal Qg As Double, ByVal Mg As Double, ByVal pg As Double) As Double
    Dim t1, t2 As Double
    t1 = Qg * pg
    t2 = Mg
    If t2 = 0 Then
      Return 0
    Else
      Return t1 / t2
    End If
  End Function
  Private Function WriteOutput(ByVal pathfile As PathFile, ByVal n As Integer, ByVal oZsu() As Double)
```

```
        '------------------write file------------------
        Dim Strdata As String
        Dim ow As New WriteOutput
        Strdata = vbCrLf & "------------------------Begin Cal Zsu ---------------------" & vbCrLf
        Strdata += ow.Constrarray(n, oZsu, "Zsu")
        Strdata += "------------------------End---------------------" & vbCrLf
        Dim tempdata As String
        tempdata = ow.GetFileContents(pathfile.NewOutputFile)
        Strdata = tempdata & vbCrLf & Strdata
        ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
    End Function
End Class
```

```
Choke model
Imports System.Math
Public Class chokeModel
    Public Cv As Double
    Public Cp As Double
    Public Cl As Double
    Public X1 As Double
    Public k As Double
    Public Vg1 As Double
    Public Vg2 As Double
    Public n As Double
    Public VL As Double
    Public Yc As Double
    Public Pm2 As Double
    Public AC As Double
    Public M As Double
    Public Yu As Double
    Public Nchoke As Double
    Private Cx_Start As Double
    Private LiquidDens As Double
    Public Function runModel(ByVal pathfile As PathFile, ByVal nc As Integer, ByVal P2 As Double, _
        ByVal y As Double, ByVal temp_F As Double, ByVal GasSG As Double, ByVal OilSG As Double, _
     ByVal ChokeDiameter As Double, _
     ByVal Cd As Double, ByVal gc As Double, ByVal Zp() As Double, ByVal pc() As Double, _
     ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, ByVal inflcon As Double, _
     ByVal m() As Double, ByVal Liqstand() As Double, ByVal np As Double) As Double
        Dim O_yc As Double = 0.0
        Dim P1 As Double
        P1 = P2
        Dim YLoop As Integer = 0
        While Abs(y - O_yc) > 0.01
            O_yc = FindYc(pathfile, YLoop, nc, P2, y, temp_F, GasSG, OilSG, ChokeDiameter, Cd, gc, Zp, P1, pc, tc, omega, R, outflcon, inflcon, m,
Liqstand)
            P1 = P1 + 20
            If YLoop > 1000 Then
                Exit While
            End If
            YLoop = YLoop + 1
            ' Debug.WriteLine(Abs(y - O_yc))
        End While
        Yc = P2 / (P1 - 5)
        WriteOutPutY(pathfile, Yc)
        Dim P1_m As Double
        Dim Addnp As Integer = 100
        P1_m = P2 + 1
        Dim Cx_1 As Double = 0
        Dim Chk As Boolean = False
        Dim Loopi As Integer = 0
        While Not Chk
            Chk = FindM_New(pathfile, nc, temp_F, Zp, P1_m, P2, pc, gc, np, tc, omega, R, outflcon, inflcon, _
            m, Liqstand, ChokeDiameter, Cd, Addnp, Cx_1, Loopi)
    If Chk = False Then
            P1_m = P1_m + Addnp
            Loopi += 1
        End If
        'Debug.WriteLine(" :" & P1_m)
        End While
        WriteOutPutP1(pathfile, P1_m - Addnp)
        Return P1_m - Addnp
    End Function
#Region "FindYC"
    Private Function FindYc(ByVal pathfile As PathFile, ByVal YLoop As Integer, ByVal nc As Integer, ByVal P2 As Double, _
        ByVal y As Double, ByVal temp_F As Double, ByVal GasSG As Double, ByVal OilSG As Double, _
     ByVal ChokeDiameter As Double, _
     ByVal Cd As Double, ByVal gc As Double, ByVal Z() As Double, ByVal P1 As Double, ByVal pc() As Double, _
     ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, ByVal inflcon As Double, _
     ByVal m() As Double, ByVal Liqstand() As Double) As Double
        Dim GasDens As Double
        Dim ObjComp As New CompositionModel
        ObjComp.Run(pathfile, 0.5, nc, Z, P1, temp_F, pc, tc, omega, R, outflcon, inflcon, m, Liqstand)
        GasDens = ObjComp.GasDensity
        LiquidDens = ObjComp.oilDensity
```

```
        X1 = 1 - ObjComp.L
        Cp = 0.537
        Cv = 0.414
        Cl = 0.55
        k = Cp / Cv
        If GasDens = 0 Then
            Vg1 = 0
        Else
            Vg1 = 1 / GasDens
        End If
        VL = 1 / LiquidDens
        Vg2 = Vg1 * y ^ (-1 / k)
        n = 1 + (X1 * (Cp - Cv)) / ((X1 * Cv) + ((1 - X1) * Cl))
        Dim t1, t2, t3, t4, t5 As Double
        t1 = k / (k - 1)
        If (X1 * Vg1) = 0 Then
            t2 = 0
        Else
            t2 = ((1 - X1) * VL * (1 - y)) / (X1 * Vg1)
        End If
        t3 = n / 2
        If (X1 * Vg2) = 0 Then
            t4 = 0
        Else
            t4 = (n * (1 - X1) * VL) / (X1 * Vg2)
        End If
        If (X1 * Vg2) = 0 Then
            t5 = 0
        Else
            t5 = t3 * (((1 - X1) * VL) / (X1 * Vg2)) ^ 2
        End If
        If (t1 + t3 + t4 + t5) = 0 Then
            Yc = 0
        Else
            Yc = ((t1 + t2) / (t1 + t3 + t4 + t5)) ^ t1
        End If
        WriteOutputYc(pathfile, YLoop, nc, P2, y, temp_F, GasSG, OilSG, ChokeDiameter, Cd, gc, Z, P1, pc, tc, omega, R, outflcon, inflcon, m,
Liqstand, ObjComp.L, GasDens)
        Return Yc
    End Function
#End Region
    Private Function FindM_New(ByVal PathFile As PathFile, ByVal nc As Integer, ByVal temp_F As Double, _
ByVal Zpi() As Double, ByRef P1 As Double, _
ByVal P2 As Double, ByVal Pc() As Double, ByVal gc As Double, ByVal SumNpi As Double, _
ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
ByVal inflcon As Double, ByVal mo() As Double, ByVal Liqstand() As Double, _
ByVal ChokeDiameter As Double, ByVal Cd As Double, _
ByRef Addnp As Integer, ByRef Cx_old As Double, ByVal Loopi As Integer) As Boolean
        Dim GasDens As Double
        Dim Y As Double
        Dim ObjComp As New CompositionModel
        ObjComp.Run(PathFile, 0.5, nc, Zpi, P1, temp_F, Pc, tc, omega, R, outflcon, inflcon, mo, Liqstand)
        GasDens = ObjComp.GasDensity
        LiquidDens = ObjComp.oilDensity

        X1 = 1 - ObjComp.L
        Cp = 0.537
        Cv = 0.414
        Cl = 0.55
        k = Cp / Cv
        Vg1 = 1 / GasDens
        VL = 1 / LiquidDens
        Y = P2 / P1
        Yu = Max(Y, Yc)
        Vg2 = Vg1 * Y ^ (-1 / k)
        AC = (22 / 7) * (ChokeDiameter ^ 2) / (4 * 144)
        Pm2 = ((X1 * Vg1 * Y ^ (-1 / k)) + ((1 - X1) * VL)) ^ (-1)
        Dim t1, t2, t3, t4 As Double
        If ObjComp.L = 1 And ObjComp.V = 0 Then
            t1 = 5.615 * 22800 * (ChokeDiameter ^ 2)
            t2 = ((P1 - P2) / LiquidDens) ^ 0.5
            M = t1 * t2
        Else
            t1 = 86400 * AC * Cd
            t2 = 2 * gc * 144 * P1 * (Pm2 ^ 2)
            t3 = ((1 - X1) * (1 - Yu)) / LiquidDens
            t4 = ((X1 * k) / (k - 1)) * (Vg1 - (Y * Vg2))
            M = t1 * Sqrt(t2 * (t3 + t4))
        End If
        Dim i As Integer
        Dim Fmw As Double
        For i = 0 To nc - 1
            Fmw = Fmw + (Zpi(i) * mo(i))
        Next
```

```
        Nchoke = M / Fmw
        Dim chk As Boolean
        Dim schk As String
        Dim Cx As Double
        Cx = Abs(Nchoke - SumNpi)
        If Cx < 1 Then
            chk = True
            schk = "yes"
        Else
            If Loopi = 0 Then
                Cx_Start = Cx
                Cx_old = Cx
                chk = False
            Else
                If Cx > Cx_old Then
                    If Addnp > 50 Then
                        P1 = P1 - (Addnp * 2)
                        Cx_old = Cx_Start
                        Addnp = Addnp - 10
                        chk = False
                        schk = "No"
                    ElseIf Addnp > 10 Then
                        P1 = P1 - (Addnp * 2)
                        Cx_old = Cx_Start
                        Addnp = Addnp - 2
                        chk = False
                        schk = "No"
                    ElseIf Addnp <= 10 And Addnp > 1 Then
                        P1 = P1 - (Addnp * 2)
                        Cx_old = Cx_Start
                        Addnp = Addnp - 1
                        chk = False
                        schk = "No"
                    ElseIf Addnp <= 1 Then
                        P1 = P1 - Addnp
                        chk = True
                        schk = "yes"
                    End If
                Else
                    Cx_Start = Cx_old
                    Cx_old = Cx
                    chk = False
                    schk = "No"
                End If
            End If
        End If
        WriteOutputM(PathFile, nc, temp_F, Zpi, P1, P2, Pc, gc, SumNpi, tc, omega, R, outflcon, inflcon, _
            mo, Liqstand, ChokeDiameter, Cd, Addnp, Cx_old, Loopi, Y, Fmw, schk)
        Return chk
    End Function
#Region "Writeoutput"
    Private Function WriteOutPutY(ByVal PathFile As PathFile, ByVal Yc1 As Double)
        Dim Strdata As String
        Dim ow As New WriteOutput
        Strdata = vbCrLf & "Output Yc=" & CStr(Yc1) & vbCrLf
        Dim tempdata As String
        tempdata = ow.GetFileContents(PathFile.NewOutputFile)
        Strdata = tempdata & vbCrLf & Strdata
        ow.SaveTextToFile(Strdata, PathFile.NewOutputFile)
    End Function
    Private Function WriteOutPutP1(ByVal PathFile As PathFile, ByVal P1 As Double)
        Dim Strdata As String
        Dim ow As New WriteOutput
        Strdata = vbCrLf & "Output Choke Model P1=" & CStr(P1) & vbCrLf
        Dim tempdata As String
        tempdata = ow.GetFileContents(PathFile.NewOutputFile)
        Strdata = tempdata & vbCrLf & Strdata
        ow.SaveTextToFile(Strdata, PathFile.NewOutputFile)
    End Function
    Private Function WriteOutputYc(ByVal pathfile As PathFile, ByVal YLoop As Integer, ByVal nc As Integer, ByVal P2 As Double, _
        ByVal y As Double, ByVal temp_F As Double, ByVal GasSG As Double, ByVal OilSG As Double, _
      ByVal ChokeDiameter As Double, _
      ByVal Cd As Double, ByVal gc As Double, ByVal Z() As Double, ByVal P1 As Double, ByVal pc() As Double, _
      ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, ByVal inflcon As Double, _
      ByVal m() As Double, ByVal Liqstand() As Double, ByVal L As Double, ByVal GasDens As Double)
        "------------------write file------------------
        Dim Strdata As String
        Dim ow As New WriteOutput
        Strdata = vbCrLf & "----------------------- Begin Choke Model (FindYc)Loop=" & CStr(YLoop) & " - -----------" & vbCrLf
        Strdata += "---------------------Input ---------------------" & vbCrLf
        Strdata += "P1=" & CStr(P1) & vbCrLf
        Strdata += "P2=" & CStr(P2) & vbCrLf
        Strdata += "l=" & CStr(L) & vbCrLf
        Strdata += "LiquidDens=" & CStr(LiquidDens) & vbCrLf
```

```
    Strdata += "GasDens=" & CStr(GasDens) & vbCrLf
    Strdata += "y=" & CStr(y) & vbCrLf
    Strdata += "Temp=" & CStr(temp_F) & vbCrLf
    Strdata += "GasSG=" & CStr(GasSG) & vbCrLf
    Strdata += "OilSG=" & CStr(OilSG) & vbCrLf
    Strdata += "ChokeDiameter=" & CStr(ChokeDiameter) & vbCrLf
    Strdata += "Cd=" & CStr(Cd) & vbCrLf
    Strdata += "gc=" & CStr(gc) & vbCrLf
    Strdata += ow.Constrarray(nc, Z, "z")
    Strdata += "----------------------output----------------------" & vbCrLf
    Strdata += "cp=" & CStr(Cp) & vbCrLf
    Strdata += "cv=" & CStr(Cv) & vbCrLf
    Strdata += "cl=" & CStr(Cl) & vbCrLf
    Strdata += "x1=" & CStr(X1) & vbCrLf
    Strdata += "k=" & CStr(k) & vbCrLf
    Strdata += "n=" & CStr(n) & vbCrLf
    Strdata += "vg1=" & CStr(Vg1) & vbCrLf
    Strdata += "vg2=" & CStr(Vg2) & vbCrLf
    Strdata += "yc=" & CStr(Yc) & vbCrLf
    Strdata += "------------------------ End  Choke Model (FindYc)----------------------" & vbCrLf
    Dim tempdata As String
    tempdata = ow.GetFileContents(pathfile.NewOutputFile)
    Strdata = tempdata & vbCrLf & Strdata
    ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
  End Function
  Private Function WriteOutputM(ByVal PathFile As PathFile, ByVal nc As Integer, ByVal temp_F As Double, _
ByVal Zpi() As Double, ByRef P1 As Double, _
ByVal P2 As Double, ByVal Pc() As Double, ByVal gc As Double, ByVal SumNpi As Double, _
ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
ByVal inflcon As Double, ByVal mo() As Double, ByVal Liqstand() As Double, _
ByVal ChokeDiameter As Double, ByVal Cd As Double, _
ByRef Addnp As Integer, ByRef Cx_old As Double, ByVal Loopi As Integer, ByVal Y As Double, ByVal Fmw As Double, _
ByVal schk As String)
    ''------------------write file------------------
    Dim Strdata As String
    Dim ow As New WriteOutput
    Strdata = vbCrLf & "------------------------ Begin Massflowrate ----------------------" & vbCrLf
    Strdata += "------------------------Input------------------------" & vbCrLf
    Strdata += "P1=" & CStr(P1) & vbCrLf
    Strdata += "P2=" & CStr(P2) & vbCrLf
    Strdata += "X1=" & CStr(X1) & vbCrLf
    Strdata += "Pl=" & CStr(LiquidDens) & vbCrLf
    Strdata += "Vg1=" & CStr(Vg1) & vbCrLf
    Strdata += "Gc=" & CStr(gc) & vbCrLf
    Strdata += "k=" & CStr(k) & vbCrLf
    Strdata += "yc=" & CStr(Yc) & vbCrLf
    Strdata += "yu=" & CStr(Yu) & vbCrLf
    Strdata += "Ac=" & CStr(AC) & vbCrLf
    Strdata += "Cd=" & CStr(Cd) & vbCrLf
    Strdata += "y=" & CStr(Y) & vbCrLf
    Strdata += "Vl=" & CStr(VL) & vbCrLf
    Strdata += "gc=" & CStr(gc) & vbCrLf
    Strdata += "SumNpi=" & CStr(SumNpi) & vbCrLf
    Strdata += "------------------------output------------------------" & vbCrLf
    Strdata += "Vg2=" & CStr(Vg2) & vbCrLf
    Strdata += "Pm2=" & CStr(Pm2) & vbCrLf
    Strdata += "M=" & CStr(M) & vbCrLf
    Strdata += "Fluid molacelar weight=" & CStr(Fmw) & vbCrLf
    Strdata += "nchoke=" & CStr(Nchoke) & vbCrLf
    Strdata += "check=" & CStr(schk) & vbCrLf
    Strdata += "------------------------ End Massflowrate ----------------------" & vbCrLf
    Dim tempdata As String
    tempdata = ow.GetFileContents(PathFile.NewOutputFile)
    Strdata = tempdata & vbCrLf & Strdata
    ow.SaveTextToFile(Strdata, PathFile.NewOutputFile)
  End Function
#End Region
End Class
```

```
Composition model
Imports System.Math
Public Class CompositionModel
  '--output
  Public L As Double
  Public x() As Double
  Public y() As Double
  Public vl2 As Double
  Public vv2 As Double
  Public K() As Double
  Public a As Double
  Public b As Double
  Public ai() As Double
  Public bi() As Double
  Public aij(,) As Double
```

```
Public V As Double
Public Z_cubic As Double
Public GasDensity, oilDensity As Double
Private fl() As Double
Private fv() As Double
Public Function Run(ByVal pathfile As PathFile, ByVal il As Double, ByVal n As Integer, ByVal z() As Double, _
ByVal pressure As Double, ByVal temperature_R As Double, ByVal pc() As Double, ByVal tc() As Double, ByVal omega() As Double, _
ByVal R As Double, _
 ByVal outflcon As Double, ByVal inflcon As Double, ByVal m() As Double, ByVal Liqstand() As Double)
  Dim x1, x2, x3, x4, p, t As Double
  Dim i, bob As Integer
  Dim ferror, errortol As Double
  p = pressure
  t = temperature_R
  L = il
  If (L < 0.0 Or L > 1) Then L = 0.5
  ferror = 1
  vv2 = 0
  vl2 = 0
  errortol = outflcon
  bob = 1
  For i = 0 To n - 1
    x1 = pc(i) / p
    x2 = 1 + omega(i)
    x3 = 1 - tc(i) / t
    x4 = Exp(5.37 * x2 * x3)
    ReDim Preserve K(i)
    K(i) = New Double
    K(i) = x1 * x4
  Next
  While bob < 50
    FlashCalcution2(L, n, z, K, inflcon)

    If L = 0 Then
      Call rkvarinit(n, y, pc, tc, a, b, ai, bi, aij, R)
      Call rkvolume(n, p, t, x1, x2, x3, R, a, b)
      vv2 = Max(x1, x2)
      vv2 = Max(vv2, x3)
      Z_cubic = vv2
      Call specificvolume(vv2, n, m, y, R, t, p)
      Call fugacity(n, y, fv, t, p, vv2, a, b, ai, bi, R)
      vl2 = 0
    End If
    If L = 1 Then
      Call rkvarinit(n, x, pc, tc, a, b, ai, bi, aij, R)
      Call rkvolume(n, p, t, x1, x2, x3, R, a, b)
      vl2 = Max(x1, x2)
      vl2 = Max(vl2, x3)
      Z_cubic = vl2
      Call specificvolume(Z_cubic, n, m, y, R, t, p)
      Call fugacity(n, x, fl, t, p, vl2, a, b, ai, bi, R)
      vv2 = 0
    End If
    If L > 0 And L < 1 Then
      Call rkvarinit(n, y, pc, tc, a, b, ai, bi, aij, R)
      Call rkvolume(n, p, t, x1, x2, x3, R, a, b)
      vv2 = Max(x1, x2)
      vv2 = Max(vv2, x3)
      Z_cubic = vv2
      Call specificvolume(Z_cubic, n, m, y, R, t, p)
      Call fugacity(n, y, fv, t, p, Z_cubic, a, b, ai, bi, R)
      Call rkvarinit(n, x, pc, tc, a, b, ai, bi, aij, R)
      Call rkvolume(n, p, t, x1, x2, x3, R, a, b)
      If (x2 = 0) Then x2 = 9999999.99
      If (x3 = 0) Then x3 = 9999999.99
      vl2 = Min(x1, x2)
      vl2 = Min(vl2, x3)
      Call fugacity(n, x, fl, t, p, vl2, a, b, ai, bi, R)
    End If
    'ferror = 0.0
    'For i = 0 To n - 1
    '  ferror = ferror + Abs(fl(i) - fv(i))
    'Next
    'If (ferror < errortol) Then Exit While
    'If (L >= 1 Or L <= 0) Then Exit While
    'For i = 0 To n - 1
    '  K(i) = K(i) + (K(i) * fl(i) / fv(i) - K(i))
    'Next
    'bob = bob + 1
    bob = 100
  End While
  Dim objden As New oildens
  oilDensity = objden.run(n, Liqstand, x, pressure, temperature_R, m, 50)
  WriteOutput(pathfile, L, n, z, pressure, temperature_R, pc, tc, omega, R, outflcon, inflcon, m, Liqstand)
```

```
    End Function
#Region "Flash"
    Private Function FlashCalcution2(ByVal li As Double, ByVal n As Integer, ByVal z() As Double, ByVal k() As Double, ByVal inflcon As
Double) As Boolean
        Dim L_llimit, L_ulimit, L_mid As Double
        Dim SumXminusY As Double = 0.0
      If (calc_sumXminusY(0.0, n, z, k) <= 0.0) Then L_llimit = 0.0
        If (calc_sumXminusY(0.0, n, z, k) > 0.0) Then L_ulimit = 0.0
        If (calc_sumXminusY(1.0, n, z, k) <= 0.0) Then L_llimit = 1.0
        If (calc_sumXminusY(1.0, n, z, k) > 0.0) Then L_ulimit = 1.0
        If (calc_sumXminusY(L_llimit, n, z, k) * calc_sumXminusY(L_ulimit, n, z, k) < 0.0) Then
            ' cout << "Limit --> Correct" << endl;
        Else
            'cout << "Limit --> In-Correct" << endl;
        End If
        For i As Integer = 0 To 100
            L_mid = 0.5 * (L_llimit + L_ulimit)
            If (calc_sumXminusY(L_mid, n, z, k)) >= 0.0 Then
                L_ulimit = L_mid
            ElseIf (calc_sumXminusY(L_mid, n, z, k) <= 0.0) Then
                L_llimit = L_mid
            End If
        Next
        L = L_mid
        V = 1.0 - L
      End Function
    Private Function calc_sumXminusY(ByVal Lin As Double, ByVal n As Integer, ByVal z() As Double, ByVal k() As Double) As Double
        Dim sumout As Double = 0.0
        For i As Integer = 0 To n - 1
            ReDim Preserve x(i)
            x(i) = New Double
            x(i) = calc_x(Lin, i, z, k)
      ReDim Preserve y(i)
            y(i) = New Double
            y(i) = calc_y(Lin, i, z, k)
            sumout = sumout + (x(i) - y(i))
        Next
        Return sumout
    End Function
    Private Function calc_x(ByVal Lin As Double, ByVal i As Integer, ByVal z() As Double, ByVal k() As Double) As Double
        Return z(i) / (Lin + ((1.0 - Lin) * k(i)))
    End Function
    Private Function calc_y(ByVal Lin As Double, ByVal i As Integer, ByVal z() As Double, ByVal k() As Double) As Double
        Return k(i) * z(i) / (Lin + ((1.0 - Lin) * k(i)))
    End Function
#End Region
    Private Function specificvolume(ByVal z As Double, ByVal n As Integer, ByVal m() As Double, ByVal y() As Double, ByVal r As Double,
ByVal t As Double, ByVal p As Double)
        Dim i As Integer
        Dim va As Double
        Dim mw As Double
        Dim vol As Double
        For i = 0 To n - 1
            mw = mw + m(i) * y(i)
        Next
        va = (z * r * t) / p
        If mw = 0 Then
            vol = 0
        Else
            vol = va / mw
        End If
        vol = va / mw
        If va = 0 And mw = 0 Then
            GasDensity = 0
        Else
            GasDensity = 1 / vol
        End If
    End Function
    Private Function rkvarinit(ByVal n As Integer, ByVal zsub() As Double, ByVal pc() As Double, ByVal tc() As Double, _
     ByRef a As Double, ByRef b As Double, _
     ByRef ai() As Double, ByRef bi() As Double, ByRef aij(,) As Double, ByVal r As Double)
        Dim i, j As Integer
        a = 0
        b = 0
        For i = 0 To n - 1
            ReDim Preserve ai(i)
            ai(i) = New Double
            ai(i) = 0.42748 * r * r * tc(i) * tc(i) * tc(i) / (Sqrt(tc(i)) * pc(i))
            ReDim Preserve bi(i)
            bi(i) = New Double
            bi(i) = 0.08664 * r * tc(i) / pc(i)
        Next
        ReDim Preserve aij(n - 1, n - 1)
        For i = 0 To n - 1
```

```
            For j = 0 To n - 1
                Dim ii As Integer = i '(i + 1) - 1
                Dim jj As Integer = j '4 + (j + 1)
                aij(ii, jj) = New Double
                aij(ii, jj) = Sqrt(ai(i) * ai(j))
                a = a + zsub(i) * zsub(j) * aij(ii, jj)
            Next
            b = b + zsub(i) * bi(i)
        Next
    End Function
    Private Function rkvolume(ByVal n As Integer, ByVal p As Double, ByVal t As Double, ByRef x1 As Double, _
    ByRef x2 As Double, ByRef x3 As Double, ByVal r As Double, ByVal a As Double, ByVal b As Double)
        Dim kcu, pcu, qcu, rcu As Double
        Dim roots As Integer
        'pcu = -r * t / p
        'qcu = (1 / p) * ((a / Sqrt(t)) - b * r * t - p * b * b)
        'rcu = -a * b / (p * Sqrt(t))
        kcu = ((r * t) / p) ^ 3
        pcu = -(((r * t) / p) ^ 3) / kcu
        qcu = (r * t / (p ^ 2)) * ((a / Sqrt(t)) - (b * r * t) - (p * (b ^ 2)))
        qcu = qcu / kcu
        rcu = -(a * b) / (p * Sqrt(t))
        rcu = rcu / kcu
        Call Cubic(pcu, qcu, rcu, roots, x1, x2, x3)
    End Function
    Private Function fugacity(ByVal n As Integer, ByVal zsub() As Double, ByRef f() As Double, _
    ByVal t As Double, ByVal p As Double, ByVal z As Double, ByVal a As Double, ByVal b As Double, ByVal ai() As Double, ByVal bi() As
Double, ByVal r As Double)
        Dim i As Integer
        Dim x1, x2, x3, x4, x5 As Double
        ''z = (p * V) / (r * t)
        For i = 0 To n - 1
            x1 = bi(i) * (z - 1) / b
            x2 = Log(z * (1 - b / V))
            x3 = 1 / (b * r * (t ^ 1.5)) 'Sqrt(t) / (b * r * t * t)
            x4 = (a * bi(i) / b) - 2 * Sqrt(a * ai(i))
            x5 = Log(1 + b / V)
            ReDim Preserve f(i)
            f(i) = New Double
            f(i) = x1 - x2 + x3 * x4 * x5
            f(i) = Exp(f(i))
            f(i) = f(i) * p * zsub(i)
        Next
    End Function
#Region "Cubic"
    Private Function Cubic(ByVal p As Double, ByVal q As Double, ByVal r As Double, ByRef numroots As Integer, _
        ByRef x1 As Double, ByRef x2 As Double, ByRef x3 As Double)
        Dim alpha, beta, checker, a, b, phi As Double
        alpha = (3 * q - (p ^ 2)) / 3
        beta = 0.0740741 * (p * p * p) - 0.333333333 * p * q + r
        checker = ((beta * beta) / 4) + ((alpha * alpha * alpha) / 27)
        If (Abs(checker) < 0.000001) Then
            numroots = 2
            a = (-beta / 2) ^ (1 / 3)
            x1 = 2 * a - (p - 3)
            x2 = -a - (p / 3)
            x3 = x2
        ElseIf (checker < 0) Then
            numroots = 3
            phi = Acos(-(beta / 2) * Sqrt(-27 / (alpha * alpha * alpha)))
            x1 = 1.154701 * Sqrt(-alpha) * Cos(phi / 3) - (p / 3)
            x2 = 1.154701 * Sqrt(-alpha) * Cos((phi / 3) + (0.666666 * PI)) - (p / 3)
            x3 = 1.154701 * Sqrt(-alpha) * Cos((phi / 3) + (1.33333 * PI)) - (p / 3)
        ElseIf (checker > 0) Then
            numroots = 1
            a = -beta / 2 + Sqrt(checker)
            b = -beta / 2 - Sqrt(checker)
            If (a < 0.0) Then a = -((-a) ^ (1 / 3))
            If (a > 0.0) Then a = (a ^ (1 / 3))
            If (b < 0.0) Then b = -((-b) ^ (1 / 3))
            If (b > 0.0) Then b = (b ^ (1 / 3))
            x1 = -(p / 3) + a + b
            x2 = 0
            x3 = x2
        End If
    End Function
#End Region
      Private Function WriteOutput(ByVal pathfile As PathFile, ByVal il As Double, ByVal n As Integer, ByVal z() As Double, _
    ByVal pressure As Double, ByVal temperature_R As Double, ByVal pc() As Double, ByVal tc() As Double, ByVal omega() As Double, _
    ByVal R As Double, _
     ByVal outflcon As Double, ByVal inflcon As Double, ByVal m() As Double, ByVal Liqstand() As Double)
     ''-----------------write file-----------------
        Dim Strdata As String
        Dim ow As New WriteOutput
```

```
        Strdata = vbCrLf & "-----------------------Begin Composition Model --------------------" & vbCrLf
        Strdata += "----------------------- Input----------------------- " & vbCrLf
        Strdata += ow.Constrarray(n, z, "zi")
        Strdata += "P=" & CStr(pressure) & vbCrLf
        Strdata += "T=" & CStr(temperature_R) & vbCrLf
        Strdata += ow.Constrarray(n, tc, "Tc")
        Strdata += ow.Constrarray(n, pc, "Pc")
        Strdata += ow.Constrarray(n, omega, "omega")
        Strdata += ow.Constrarray(n, m, "M")
        Strdata += ow.Constrarray(n, Liqstand, "Liqstand")
        Strdata += "R=" & CStr(R) & vbCrLf
        Strdata += "Outflcon=" & CStr(outflcon) & vbCrLf
        Strdata += "inflcon=" & CStr(inflcon) & vbCrLf
        Strdata += "----------------------- output ----------------------- " & vbCrLf
        Strdata += ow.Constrarray(n, K, "K")
        Strdata += "L=" & CStr(L) & vbCrLf
        Strdata += "V=" & CStr(V) & vbCrLf
        Strdata += "Z_cubic=" & CStr(Z_cubic) & vbCrLf
        Strdata += ow.Constrarray(n, x, "X")
        Strdata += ow.Constrarray(n, y, "Y")
        Strdata += ow.Constrarray(n, ai, "ai")
        Strdata += ow.Constrarray(n, bi, "bi")
        Strdata += "am=" & CStr(a) & vbCrLf
        Strdata += "bm=" & CStr(b) & vbCrLf
        Strdata += "gas density=" & CStr(GasDensity) & vbCrLf
        Strdata += "oil density=" & CStr(oilDensity) & vbCrLf
        Strdata += "----------------------- End Composition Model -----------------------------------------"
        Dim tempdata As String
        tempdata = ow.GetFileContents(pathfile.NewOutputFile)
        Strdata = tempdata & vbCrLf & Strdata
        ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
    End Function
End Class
```

```
Form1
Public Class Form1
    Inherits System.Windows.Forms.Form
#Region " Windows Form Designer generated code "
    Public Sub New()
        MyBase.New()
        'This call is required by the Windows Form Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call
    End Sub
    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
            End If
        End If
        MyBase.Dispose(disposing)
    End Sub
    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
    Friend WithEvents btnBrowse As System.Windows.Forms.Button
    Friend WithEvents txtExcelPath As System.Windows.Forms.TextBox
    Friend WithEvents GroupBox2 As System.Windows.Forms.GroupBox
    Friend WithEvents btnSave As System.Windows.Forms.Button
    Friend WithEvents txtOutFile As System.Windows.Forms.TextBox
    Friend WithEvents sveFile As System.Windows.Forms.SaveFileDialog
    Friend WithEvents opnFile As System.Windows.Forms.OpenFileDialog
    Friend WithEvents btnClose As System.Windows.Forms.Button
    Friend WithEvents Button7 As System.Windows.Forms.Button
    Friend WithEvents UltraGrid1 As Infragistics.Win.UltraWinGrid.UltraGrid
    Friend WithEvents UltraGridExcelExporter1 As Infragistics.Win.UltraWinGrid.ExcelExport.UltraGridExcelExporter
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Dim Appearance1 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance2 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance3 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance4 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance5 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance6 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance7 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance8 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance9 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance10 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance11 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Dim Appearance12 As Infragistics.Win.Appearance = New Infragistics.Win.Appearance
        Me.GroupBox1 = New System.Windows.Forms.GroupBox
```

```
Me.btnBrowse = New System.Windows.Forms.Button
Me.txtExcelPath = New System.Windows.Forms.TextBox
Me.GroupBox2 = New System.Windows.Forms.GroupBox
Me.btnSave = New System.Windows.Forms.Button
Me.txtOutFile = New System.Windows.Forms.TextBox
Me.sveFile = New System.Windows.Forms.SaveFileDialog
Me.opnFile = New System.Windows.Forms.OpenFileDialog
Me.btnClose = New System.Windows.Forms.Button
Me.Button7 = New System.Windows.Forms.Button
Me.UltraGrid1 = New Infragistics.Win.UltraWinGrid.UltraGrid
Me.UltraGridExcelExporter1 = New Infragistics.Win.UltraWinGrid.ExcelExport.UltraGridExcelExporter
Me.GroupBox1.SuspendLayout()
Me.GroupBox2.SuspendLayout()
CType(Me.UltraGrid1, System.ComponentModel.ISupportInitialize).BeginInit()
Me.SuspendLayout()
'     'GroupBox1
Me.GroupBox1.Controls.Add(Me.btnBrowse)
Me.GroupBox1.Controls.Add(Me.txtExcelPath)
Me.GroupBox1.Location = New System.Drawing.Point(24, 32)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(424, 72)
Me.GroupBox1.TabIndex = 7
Me.GroupBox1.TabStop = False
Me.GroupBox1.Text = "Input File"
'   'btnBrowse     '
Me.btnBrowse.Location = New System.Drawing.Point(336, 29)
Me.btnBrowse.Name = "btnBrowse"
Me.btnBrowse.Size = New System.Drawing.Size(72, 20)
Me.btnBrowse.TabIndex = 3
Me.btnBrowse.Text = "Browse"     '
'txtExcelPath     '
Me.txtExcelPath.Location = New System.Drawing.Point(8, 29)
Me.txtExcelPath.Name = "txtExcelPath"
Me.txtExcelPath.Size = New System.Drawing.Size(320, 20)
Me.txtExcelPath.TabIndex = 2
Me.txtExcelPath.Text = ""     '
'GroupBox2     '
Me.GroupBox2.Controls.Add(Me.btnSave)
Me.GroupBox2.Controls.Add(Me.txtOutFile)
Me.GroupBox2.Location = New System.Drawing.Point(24, 128)
Me.GroupBox2.Name = "GroupBox2"
Me.GroupBox2.Size = New System.Drawing.Size(424, 64)
Me.GroupBox2.TabIndex = 8
Me.GroupBox2.TabStop = False
Me.GroupBox2.Text = " Output File :"
'     'btnSave     '
Me.btnSave.Location = New System.Drawing.Point(336, 24)
Me.btnSave.Name = "btnSave"
Me.btnSave.Size = New System.Drawing.Size(72, 20)
Me.btnSave.TabIndex = 6
Me.btnSave.Text = "Browse"
'     'txtOutFile     '
Me.txtOutFile.Location = New System.Drawing.Point(16, 24)
Me.txtOutFile.Name = "txtOutFile"
Me.txtOutFile.Size = New System.Drawing.Size(312, 20)
Me.txtOutFile.TabIndex = 5
Me.txtOutFile.Text = ""
'     'sveFile     '
Me.sveFile.DefaultExt = "txt"
Me.sveFile.Filter = "text file (*.txt)|*.txt"
'     'btnClose     '
Me.btnClose.Location = New System.Drawing.Point(240, 208)
Me.btnClose.Name = "btnClose"
Me.btnClose.Size = New System.Drawing.Size(80, 20)
Me.btnClose.TabIndex = 10
Me.btnClose.Text = "Close"
'     'Button7     '
Me.Button7.Location = New System.Drawing.Point(136, 208)
Me.Button7.Name = "Button7"
Me.Button7.TabIndex = 18
Me.Button7.Text = "Run"
'     'UltraGrid1     '
Appearance1.BackColor = System.Drawing.SystemColors.Window
Appearance1.BorderColor = System.Drawing.SystemColors.InactiveCaption
Me.UltraGrid1.DisplayLayout.Appearance = Appearance1
Me.UltraGrid1.DisplayLayout.BorderStyle = Infragistics.Win.UIElementBorderStyle.Solid
Me.UltraGrid1.DisplayLayout.CaptionVisible = Infragistics.Win.DefaultableBoolean.False
Appearance2.BackColor = System.Drawing.SystemColors.ActiveBorder
Appearance2.BackColor2 = System.Drawing.SystemColors.ControlDark
Appearance2.BackGradientStyle = Infragistics.Win.GradientStyle.Vertical
Appearance2.BorderColor = System.Drawing.SystemColors.Window
Me.UltraGrid1.DisplayLayout.GroupByBox.Appearance = Appearance2
Appearance3.ForeColor = System.Drawing.SystemColors.GrayText
Me.UltraGrid1.DisplayLayout.GroupByBox.BandLabelAppearance = Appearance3
```

```
        Me.UltraGrid1.DisplayLayout.GroupByBox.BorderStyle = Infragistics.Win.UIElementBorderStyle.Solid
        Appearance4.BackColor = System.Drawing.SystemColors.ControlLightLight
        Appearance4.BackColor2 = System.Drawing.SystemColors.Control
        Appearance4.BackGradientStyle = Infragistics.Win.GradientStyle.Horizontal
        Appearance4.ForeColor = System.Drawing.SystemColors.GrayText
        Me.UltraGrid1.DisplayLayout.GroupByBox.PromptAppearance = Appearance4
        Me.UltraGrid1.DisplayLayout.MaxColScrollRegions = 1
        Me.UltraGrid1.DisplayLayout.MaxRowScrollRegions = 1
        Appearance5.BackColor = System.Drawing.SystemColors.Window
        Appearance5.ForeColor = System.Drawing.SystemColors.ControlText
        Me.UltraGrid1.DisplayLayout.Override.ActiveCellAppearance = Appearance5
        Appearance6.BackColor = System.Drawing.SystemColors.Highlight
        Appearance6.ForeColor = System.Drawing.SystemColors.HighlightText
        Me.UltraGrid1.DisplayLayout.Override.ActiveRowAppearance = Appearance6
        Me.UltraGrid1.DisplayLayout.Override.BorderStyleCell = Infragistics.Win.UIElementBorderStyle.Dotted
        Me.UltraGrid1.DisplayLayout.Override.BorderStyleRow = Infragistics.Win.UIElementBorderStyle.Dotted
        Appearance7.BackColor = System.Drawing.SystemColors.Window
        Me.UltraGrid1.DisplayLayout.Override.CardAreaAppearance = Appearance7
        Appearance8.BorderColor = System.Drawing.Color.Silver
        Appearance8.TextTrimming = Infragistics.Win.TextTrimming.EllipsisCharacter
        Me.UltraGrid1.DisplayLayout.Override.CellAppearance = Appearance8
        Me.UltraGrid1.DisplayLayout.Override.CellClickAction = Infragistics.Win.UltraWinGrid.CellClickAction.EditAndSelectText
        Me.UltraGrid1.DisplayLayout.Override.CellPadding = 0
        Appearance9.BackColor = System.Drawing.SystemColors.Control
        Appearance9.BackColor2 = System.Drawing.SystemColors.ControlDark
        Appearance9.BackGradientAlignment = Infragistics.Win.GradientAlignment.Element
        Appearance9.BackGradientStyle = Infragistics.Win.GradientStyle.Horizontal
        Appearance9.BorderColor = System.Drawing.SystemColors.Window
        Me.UltraGrid1.DisplayLayout.Override.GroupByRowAppearance = Appearance9
        Appearance10.TextHAlign = Infragistics.Win.HAlign.Left
        Me.UltraGrid1.DisplayLayout.Override.HeaderAppearance = Appearance10
        Me.UltraGrid1.DisplayLayout.Override.HeaderClickAction = Infragistics.Win.UltraWinGrid.HeaderClickAction.SortMulti
        Me.UltraGrid1.DisplayLayout.Override.HeaderStyle = Infragistics.Win.HeaderStyle.WindowsXPCommand
        Appearance11.BackColor = System.Drawing.SystemColors.Window
        Appearance11.BorderColor = System.Drawing.Color.Silver
        Me.UltraGrid1.DisplayLayout.Override.RowAppearance = Appearance11
        Me.UltraGrid1.DisplayLayout.Override.RowSelectors = Infragistics.Win.DefaultableBoolean.False
        Appearance12.BackColor = System.Drawing.SystemColors.ControlLight
        Me.UltraGrid1.DisplayLayout.Override.TemplateAddRowAppearance = Appearance12
        Me.UltraGrid1.DisplayLayout.ScrollBounds = Infragistics.Win.UltraWinGrid.ScrollBounds.ScrollToFill
        Me.UltraGrid1.DisplayLayout.ScrollStyle = Infragistics.Win.UltraWinGrid.ScrollStyle.Immediate
        Me.UltraGrid1.DisplayLayout.ViewStyleBand = Infragistics.Win.UltraWinGrid.ViewStyleBand.OutlookGroupBy
        Me.UltraGrid1.Location = New System.Drawing.Point(8, 248)
        Me.UltraGrid1.Name = "UltraGrid1"
        Me.UltraGrid1.Size = New System.Drawing.Size(0, 0)
        Me.UltraGrid1.TabIndex = 19
        Me.UltraGrid1.Text = "UltraGrid1"
        '
        'Form1
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(512, 286)
        Me.Controls.Add(Me.UltraGrid1)
        Me.Controls.Add(Me.Button7)
        Me.Controls.Add(Me.btnClose)
        Me.Controls.Add(Me.GroupBox2)
        Me.Controls.Add(Me.GroupBox1)
        Me.Name = "Form1"
        Me.Text = "Form1"
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox2.ResumeLayout(False)
        CType(Me.UltraGrid1, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)
    End Sub
#End Region
    Public maxper As Integer = 20
    Private Sub btnBrowse_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnBrowse.Click
        Try
            opnFile.Filter = "Excel Files (*.xls)|*.xls"
            opnFile.ShowDialog()
            txtExcelPath.Text = opnFile.FileName
        Catch ex As Exception
        End Try
    End Sub
    Private Sub btnSave_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSave.Click
        Try
            sveFile.ShowDialog()
            txtOutFile.Text = sveFile.FileName
        Catch ex As Exception
        End Try
    End Sub
    Private Sub btnClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClose.Click
        End
    End Sub
    Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button7.Click
        Dim o As New PathFile
```

```
    Try
      If txtOutFile.Text = "" Then
        Err.Raise(9999, , "Text Output file")
      ElseIf txtExcelPath.Text = "" Then
        Err.Raise(9999, , "Excel File")
      End If
      o.OutputFile = txtOutFile.Text
      o.InputFile = txtExcelPath.Text
    Catch ex As Exception
      MessageBox.Show(ex.Message)
    End Try
    Dim rds As New DataSet
    Dim obj As New ngf
    If obj.RunMainProgram(o, UltraGrid1, UltraGridExcelExporter1) Then
      'rds = obj.ds
      'Try
      '  Dim outputfile As String
      '  outputfile = Mid(o.OutputFile, 1, o.OutputFile.Length - 4) & ".xls"
      '  If Not IsNothing(rds) Then
      '    UltraGrid1.DataSource = rds.Tables(0)
      '    Me.UltraGridExcelExporter1.Export(Me.UltraGrid1, outputfile)
      '  End If
      'Catch ex As Exception
      'End Try
      MessageBox.Show("Complete")
    Else
      MessageBox.Show("Error")
    End If
  End Sub
End Class
```

```
Load input file
Imports System.Math
Public Class LoadInputFile
  "Composition model"
  Public z() As Double 'composition
  Public pc() As Double 'CRITICAL P
  Public tc() As Double 'CRITICAL T
  Public m() As Double ''MOLECULAR WEIGHT
  Public omega() As Double 'Accentric factor
  Public liqstand() As Double  'liquid density @standard condition
  Public n As Integer
  Public Pressure As Double
  Public temperature As Double
  Public outflcon As Double
  Public inflcon As Double
  Public R As Double
  Public Area As Double
  Public Pipeline_Length As Double
  Public Tubing_Length As Double
  Public Qinj_input, qgaban, qoaban As Double
  Public numberofwell As Double
  "Reservoir Model"
  Public sor As Double
  Public sgr As Double
  Public noil As Double
  Public ngas As Double
  Public porosity As Double
  Public k As Double
  Public h As Double
  Public Uoil As Double
  Public Ugas As Double
  Public re As Double
  Public rw As Double
  Public dt As Double
  "Tubing Model ""
  'ChokeModel
  '---temp--------
  Public x() As Double
  Public y() As Double
  Public L As Double
  Public V As Double
  Public Oildensity As Double
  Public GasDensity As Double
  Public OilViscosity As Double
  Public GasViscosity As Double
  Public Pbar As Double
  Public Pwf As Double
  Public qo As Double
  Public qg As Double
  Public A As Double
  Public D As Double
  Public A_tub As Double
```

```
Public D_tub As Double
Public g As Double
Public DelL As Double
Public e As Double
Public liquidvis As Double
Public P2 As Double
Public yc As Double
Public temp As Double
Public GasSG As Double
Public OilSG As Double
Public ChokeDiameter As Double
Public Cd, gc As Double
Public X1, Pl, Vg1, Yu, Ac, Vl, SumNpi As Double
Public Psep1, psep2, psep3, patm, P1, Tatm, Tatm2 As Double
Public InjectionPoint As Double
'-----------
 Private Function ExcelConnect(ByVal iopath As PathFile) As System.Data.DataSet
  Dim mConnect As System.Data.OleDb.OleDbConnection
  Try
     ' ReserveChar = ConfigurationSettings.AppSettings("ReserveChar")  ' ConfigurationSettings.AppSettings("ReserveWord")
     'Fetch Data from excel
     Dim dtSet As System.Data.DataSet
     'Dim mSheetName As String
     dtSet = New System.Data.DataSet
     Dim mCommand As System.Data.OleDb.OleDbDataAdapter
     mConnect = New System.Data.OleDb.OleDbConnection("provider=Microsoft.Jet.OLEDB.4.0; " & _
             "data source='" & iopath.InputFile & " '; " & "Extended Properties=Excel 8.0;")
           mCommand = New System.Data.OleDb.OleDbDataAdapter("select * from [input$]", mConnect) '
     mCommand.Fill(dtSet, "input")
     mConnect.Close()
     Return (dtSet)
  Catch ex As Exception
     MessageBox.Show(ex.Message)
  End Try
End Function
Public Function ReadInputMain(ByVal Pathfile As PathFile)
  Dim ds As New System.Data.DataSet
  Dim i As Integer = 0
  ds = ExcelConnect(Pathfile)
  Dim _tableName As String = "input"
  If Not IsNothing(ds) Then
     For i = 0 To ds.Tables(_tableName).Rows.Count - 1
       ReDim Preserve z(i)
       z(i) = New Double
       z(i) = ds.Tables(_tableName).Rows(i).Item("zi").ToString
       ReDim Preserve pc(i)
       pc(i) = New Double
       pc(i) = Val(ds.Tables(_tableName).Rows(i).Item("pci").ToString)
       ReDim Preserve tc(i)
       tc(i) = New Double
       tc(i) = Val(ds.Tables(_tableName).Rows(i).Item("tci").ToString)
       ReDim Preserve omega(i)
       omega(i) = New Double
       omega(i) = ds.Tables(_tableName).Rows(i).Item("omega").ToString
       ReDim Preserve m(i)
       m(i) = New Double
       m(i) = ds.Tables(_tableName).Rows(i).Item("MolecularWeight").ToString
       ReDim Preserve liqstand(i)
       liqstand(i) = New Double
       liqstand(i) = ds.Tables(_tableName).Rows(i).Item("Liquidstand").ToString
     Next
     Dim tb As Double
     Dim c7sg As Double
     c7sg = Val(ds.Tables(_tableName).Rows(0).Item("c7Specificgravity").ToString)
     tb = (4.5579 * (m(6) ^ 0.15178) * (c7sg ^ 0.15427)) ^ 3
     Dim tc7, pc7 As Double
     tc7 = 341.7 + (811 * c7sg) + (0.4244 + (0.1174 * c7sg)) * tb + (((0.4669 - (3.2623 * c7sg)) * 10 ^ 5) / tb)
     tc(6) = tc7
     Dim PCA, PCB, PCC As Double
     PCA = 8.3634 - (0.0566 / c7sg) - ((0.24244 + (2.2898 / c7sg) + (0.11875 / (c7sg) ^ 2)) * 0.001 * tb)
     PCB = (1.4685 + (3.648 / c7sg) + (0.47227 / (c7sg) ^ 2)) * ((0.1) ^ 7) * (tb ^ 2)
     PCC = (0.42019 + (1.6977 / (c7sg ^ 2))) * (0.1 ^ 10) * (tb ^ 3)
     pc7 = Exp(PCA + PCB - PCC)
     pc(6) = pc7
     temperature = ds.Tables(_tableName).Rows(0).Item("temperature").ToString
     Pressure = ds.Tables(_tableName).Rows(0).Item("pressure").ToString
     inflcon = ds.Tables(_tableName).Rows(0).Item("inflcon").ToString
     outflcon = ds.Tables(_tableName).Rows(0).Item("outflcon").ToString
     n = z.Length
     R = ds.Tables(_tableName).Rows(0).Item("R").ToString
     sgr = Val(ds.Tables(_tableName).Rows(0).Item("sgr").ToString)
     sor = Val(ds.Tables(_tableName).Rows(0).Item("sor").ToString)
     noil = Val(ds.Tables(_tableName).Rows(0).Item("noil").ToString)
     ngas = Val(ds.Tables(_tableName).Rows(0).Item("ngas").ToString)
```

```
            porosity = Val(ds.Tables(_tableName).Rows(0).Item("porosity").ToString)
            k = Val(ds.Tables(_tableName).Rows(0).Item("k").ToString)
            h = Val(ds.Tables(_tableName).Rows(0).Item("h").ToString)
            re = Val(ds.Tables(_tableName).Rows(0).Item("re").ToString)
            rw = Val(ds.Tables(_tableName).Rows(0).Item("rw").ToString)
            dt = Val(ds.Tables(_tableName).Rows(0).Item("dt").ToString)
            Pbar = Val(ds.Tables(_tableName).Rows(0).Item("pbar").ToString)
            Pwf = Val(ds.Tables(_tableName).Rows(0).Item("pwf").ToString)
            Psep1 = ds.Tables(_tableName).Rows(0).Item("psep1").ToString
            psep2 = ds.Tables(_tableName).Rows(0).Item("psep2").ToString
            psep3 = ds.Tables(_tableName).Rows(0).Item("psep3").ToString
            Tatm = ds.Tables(_tableName).Rows(0).Item("Tatm").ToString
            Tatm2 = ds.Tables(_tableName).Rows(0).Item("Tatm2").ToString
            GasSG = Val(ds.Tables(_tableName).Rows(0).Item("GasSG").ToString)
            OilSG = Val(ds.Tables(_tableName).Rows(0).Item("OilSG").ToString)
            ChokeDiameter = Val(ds.Tables(_tableName).Rows(0).Item("ChokeDiameter").ToString)
            Cd = Val(ds.Tables(_tableName).Rows(0).Item("Cd").ToString)
            gc = Val(ds.Tables(_tableName).Rows(0).Item("gc").ToString)
            A = ds.Tables(_tableName).Rows(0).Item("A").ToString
            D = Val(ds.Tables(_tableName).Rows(0).Item("D").ToString)
            A_tub = ds.Tables(_tableName).Rows(0).Item("A_tub").ToString
            D_tub = Val(ds.Tables(_tableName).Rows(0).Item("D_tub").ToString)
            g = Val(ds.Tables(_tableName).Rows(0).Item("G").ToString)
            DelL = Val(ds.Tables(_tableName).Rows(0).Item("delL").ToString)
            e = Val(ds.Tables(_tableName).Rows(0).Item("e").ToString)
            g = Val(ds.Tables(_tableName).Rows(0).Item("G").ToString)
            DelL = Val(ds.Tables(_tableName).Rows(0).Item("delL").ToString)
            e = Val(ds.Tables(_tableName).Rows(0).Item("e").ToString)
            Pipeline_Length = Val(ds.Tables(_tableName).Rows(0).Item("Pipeline_length").ToString)
            Tubing_Length = Val(ds.Tables(_tableName).Rows(0).Item("Tubing_Length").ToString)
            InjectionPoint = Val(ds.Tables(_tableName).Rows(0).Item("InjectionPoint").ToString)
            Qinj_input = Val(ds.Tables(_tableName).Rows(0).Item("Qinj_input").ToString)
            qoaban = Val(ds.Tables(_tableName).Rows(0).Item("qoaban").ToString)
            qgaban = Val(ds.Tables(_tableName).Rows(0).Item("qgaban").ToString)
            numberofwell = Val(ds.Tables(_tableName).Rows(0).Item("numberofwell").ToString)
            Area = Val(ds.Tables(_tableName).Rows(0).Item("area").ToString)
        End If
    End Function
End Class
```

```
Ngf
Imports System.Math
Imports System
Imports System.Data
Public Class ngf
    ''Main Input
    Public n As Integer
    Public z() As Double
    Public Pressure As Double
    Public Temperature As Double
    Public pc() As Double
    Public tc() As Double
    Public omega() As Double
    Public R As Double
    Public liqstand() As Double
    Public inflcon As Double
    Public outflcon As Double
    Public m() As Double
    Public porosity As Double
    Public sgr As Double
    Public sor As Double
    Public noil As Double
    Public ngas As Double
    Public k As Double
    Public h As Double
    Public re As Double
    Public rw As Double
    Public Pbar As Double
    Public Pwf As Double
    Public dt As Double
    Public numberofwell As Double
    '---for pipeline
    Public A As Double
    Public D As Double
    Public g As Double
    Public DelL As Double
    Public Psep1, psep2, psep3 As Double
    Public Tatm, Tatm2 As Double
    Public Pipeline_length As Double
    Public e As Double
    '---for choke --
    Public GasSG, OilSG, ChokeDiameter, Cd, GC As Double
    '--for tubing --
```

```
    Public tubing_length As Double
    Public InjectionPoint As Double
    Public A_tub As Double
    Public D_tub As Double
    '--for separator ---
    Public Qinj_input, qgaban, qoaban As Double
    'Main Output
    Public qg_out, qo_out As Double
    Public Np As Double
    Public Ninj As Double
    Public Zinj() As Double
    Public Zp() As Double
    Public t As Double
    Public P2 As Double
    Public P1 As Double
    Public Pres As Double
    Public Zi_new() As Double
    Public TimeStep As Integer
    Public New_Pwf As Double
    Public Old_Pwf As Double
    Public Zsu() As Double
    Public New_Pres As Double
    Public Qinj_Out As Double
    Public Area As Double
    Dim OldPathfile As New PathFile
    Dim PwfLoop As Integer
    Public ds As New DataSet
    Dim dst As System.Data.DataTable
#Region "MainProgram"
    Public Function RunMainProgram(ByVal Pathfile As PathFile, _
   ByVal UltraGrid1 As Infragistics.Win.UltraWinGrid.UltraGrid, _
ByVal UltraGridExcelExporter1 As Infragistics.Win.UltraWinGrid.ExcelExport.UltraGridExcelExporter) As Boolean
        OldPathfile = Pathfile
        Try
            Dim o1 As New LoadInputFile
            o1.ReadInputMain(Pathfile)
            n = o1.n
            z = o1.z
            Pressure = o1.Pressure
            Temperature = o1.temperature
            pc = o1.pc
            tc = o1.tc
            R = o1.R
            omega = o1.omega
            liqstand = o1.liqstand
            outflcon = o1.outflcon
            inflcon = o1.inflcon
            m = o1.m
            porosity = o1.porosity
            sgr = o1.sgr
            sor = o1.sor
            noil = o1.noil
            ngas = o1.ngas
            k = o1.k
            h = o1.h
            re = o1.re
            rw = o1.rw
            Pbar = o1.Pbar
            Pwf = o1.Pwf
            dt = o1.dt
            A = o1.A
            D = o1.D
            A_tub = o1.A_tub
            D_tub = o1.D_tub
            g = o1.g
            DelL = o1.DelL
            e = o1.e
            Psep1 = o1.Psep1
            psep2 = o1.psep2
            psep3 = o1.psep3
            Tatm = o1.Tatm
            Tatm2 = o1.Tatm2
            Pipeline_length = o1.Pipeline_Length
            tubing_length = o1.Tubing_Length
            InjectionPoint = o1.InjectionPoint
            GasSG = o1.GasSG
            OilSG = o1.OilSG
            ChokeDiameter = o1.ChokeDiameter
            Cd = o1.Cd
            GC = o1.gc
            qgaban = o1.qgaban
            qoaban = o1.qoaban
            Area = o1.Area
            numberofwell = o1.numberofwell
```

```
Qinj_input = o1.Qinj_input
Ninj = 0
TimeStep = 1
PwfLoop = 1
Dim ChkPwf As Boolean = False
Dim ChkPabandon As Boolean = False
Dim AddnPwf As Integer = 20
Dim Start_Pwf As Double
Dim New_Error, Old_Error As Double
Dim GetStart As Boolean = True
Dim GetOld_Error As Boolean = True
Dim OutPutPWF As Double
ds = CreateEventDS()
dst = ds.Tables(0)
Dim ChkSign As Boolean = True "+ = true , - = false
Old_Error = 0
Start_Pwf = 0
Old_Pwf = Pwf
While Not ChkPabandon
  While Not ChkPwf
    If Model1_3(Pathfile) Then
      If MainTubingmodel(Pathfile) Then "--->Pwf
        Debug.WriteLine("TimeStep=" & TimeStep & " :Old_PWf=" & Old_Pwf & " :Pwf=" & Pwf & " :error=" & Abs(Old_Pwf - Pwf)
& " :PwfLoop=" & PwfLoop)
        If TimeStep > 1 Then
          Dim Rst2 As Double
          Rst2 = Pwf - Old_Pwf
          If Rst2 >= 0 And Not GetOld_Error Then
            OutPutPWF = Old_Pwf + AddnPwf
            ChkPwf = True
          End If
          If Rst2 >= 0 And GetOld_Error Then
            Old_Pwf = Old_Pwf + AddnPwf + 1
            AddnPwf = 1
            GetOld_Error = False
          End If
          Old_Pwf = Old_Pwf - AddnPwf
        Else
          If Not GetOld_Error Then
            New_Error = Abs(Old_Pwf - Pwf)
            If New_Error > Old_Error Then
              OutPutPWF = Old_Pwf - AddnPwf
              ChkPwf = True
            Else
              Old_Error = New_Error
            End If
          End If
          If Not GetStart And GetOld_Error Then
            Old_Error = Abs(Old_Pwf - Pwf)
            GetOld_Error = False
          End If
          Dim Rst As Integer
          Rst = Pwf - Old_Pwf
          If Rst < 0 Then
            ChkSign = False
          End If
          If Not ChkSign And GetStart Then
            Start_Pwf = Old_Pwf - AddnPwf
            AddnPwf = 1
            Old_Pwf = Start_Pwf - 1
            GetStart = False
          End If
          Old_Pwf = Old_Pwf + AddnPwf
        End If
        PwfLoop = PwfLoop + 1
        Pwf = Old_Pwf
      End If
    End If
  End While
  Pwf = OutPutPWF
  Old_Pwf = Pwf
  GetStart = True
  GetOld_Error = True
  AddnPwf = 10
  If ModelUpdatePres(Pathfile) Then
    If qo_out < qoaban AndAlso qg_out < qgaban Then
      ChkPabandon = True
      WriteOutput(Pathfile)
      WriteExcel(Pathfile, UltraGrid1, UltraGridExcelExporter1)
    Else
      WriteOutput(Pathfile)
      WriteExcel(Pathfile, UltraGrid1, UltraGridExcelExporter1)
      If Pwf > Pbar Then
        Old_Pwf = Pbar - 10
```

```
                Pwf = Pbar - 10
              End If
              z = Zi_new
              ChkPwf = False
              PwfLoop = 1
              TimeStep = TimeStep + 1
            End If
          End If
        End While
        Return True
      Catch ex As Exception
        Return False
      End Try
    End Function
    Private Function WriteExcel(ByVal Pathfile As PathFile, ByVal UltraGrid1 As Infragistics.Win.UltraWinGrid.UltraGrid, _
ByVal UltraGridExcelExporter1 As Infragistics.Win.UltraWinGrid.ExcelExport.UltraGridExcelExporter) As Boolean
      Try
        Dim outputfile As String
        outputfile = Mid(Pathfile.OutputFile, 1, Pathfile.OutputFile.Length - 4) & "_TimeStep" & CStr(TimeStep) & ".xls"
        If Not IsNothing(ds) Then
          UltraGrid1.DataSource = ds.Tables(0)
          UltraGridExcelExporter1.Export(UltraGrid1, outputfile)
        End If
      Catch ex As Exception
      End Try
    End Function
    Private Function Model1_3(ByVal Pathfile As PathFile) As Boolean
      Dim Chk As Boolean = False
      If MainReservoirModel(Pathfile) Then "'-->zp,np,Pres,zi
        If CalZsu(Pathfile) Then "'--->Zsu
          If MainPipelinemodel(Pathfile) Then "'-->P2
            If MainChokemodel(Pathfile) Then            '--->P1
              Chk = True
            End If
          End If
        End If
      End If
      Return Chk
    End Function
    Private Function ModelUpdatePres(ByVal pathfile As PathFile) As Boolean
      Dim chk As Boolean = False
      If MainUpdatePres(pathfile) Then "'-->Pressure , zi
        If Mainseparatormodel(pathfile) Then "'-->Ninj,Zsu
                chk = True
        End If
      End If
      Return chk
    End Function
    Private Function MainReservoirModel(ByVal Pathfile As PathFile) As Boolean
      Try
        Dim obj1 As New ReservoirModel
        Dim OutputFile As String
        OutputFile = Left(Pathfile.OutputFile, Pathfile.OutputFile.Length - 4)
        OutputFile = OutputFile & "_ReservoirModel_Timestep_" & CStr(TimeStep) & "_PwfLoop=" & PwfLoop & ".txt"
        Pathfile.NewOutputFile = OutputFile
        obj1.runModel(Pathfile, n, z, Pressure, Temperature, pc, tc, _
        omega, R, liqstand, outflcon, inflcon, m, porosity, sgr, _
        sor, noil, ngas, k, h, re, rw, Pbar, Pwf, dt, numberofwell, Area)
        Np = obj1.Np
        Zp = obj1.Zp
        Pres = obj1.Pres
        Zi_new = obj1.Zi
        Return True
      Catch ex As Exception
        Return False
      End Try
    End Function
    Private Function MainPipelinemodel(ByVal Pathfile As PathFile) As Boolean
      Try
        Dim OutputFile As String
        OutputFile = Left(Pathfile.OutputFile, Pathfile.OutputFile.Length - 4)
        OutputFile = OutputFile & "_Pipelinemodel_Timestep_" & CStr(TimeStep) & "_PwfLoop=" & PwfLoop & ".txt"
        Pathfile.NewOutputFile = OutputFile
        Dim obj3 As New PipelineModel
        If TimeStep = 1 Then
          P2 = obj3.runModel(Pathfile, n, Np + Ninj, A, D, g, DelL, e, liqstand, Zp, Psep1, Tatm, pc, tc, omega, R, outflcon, inflcon, m,
Pipeline_length)
        Else
          P2 = obj3.runModel(Pathfile, n, Np + Ninj, A, D, g, DelL, e, liqstand, Zsu, Psep1, Tatm, pc, tc, omega, R, outflcon, inflcon, m,
Pipeline_length)
        End If
        Return True
      Catch ex As Exception
        Return False
```

```
    End Try
  End Function
  Private Function MainChokemodel(ByVal Pathfile As PathFile) As Boolean
    Try
      Dim OutputFile As String
      OutputFile = Left(Pathfile.OutputFile, Pathfile.OutputFile.Length - 4)
      OutputFile = OutputFile & "_Chokemodel_Timestep_" & CStr(TimeStep) & "_PwfLoop=" & PwfLoop & ".txt"
      Pathfile.NewOutputFile = OutputFile
      Dim obj4 As New chokeModel
      Dim y As Double
      y = 0.5
      If TimeStep = 1 Then
        P1 = obj4.runModel(Pathfile, n, P2, y, Tatm, GasSG, OilSG, ChokeDiameter, Cd, GC, Zp, pc, tc, omega, R, outflcon, inflcon, m, liqstand,
Np + Ninj)
      Else
        P1 = obj4.runModel(Pathfile, n, P2, y, Tatm, GasSG, OilSG, ChokeDiameter, Cd, GC, Zsu, pc, tc, omega, R, outflcon, inflcon, m,
liqstand, Np + Ninj)
      End If
      Return True
    Catch ex As Exception
      Return False
    End Try
  End Function
  Private Function MainTubingmodel(ByVal Pathfile As PathFile) As Boolean
    Try
      Dim OutputFile As String
      OutputFile = Left(Pathfile.OutputFile, Pathfile.OutputFile.Length - 4)
      OutputFile = OutputFile & "_Tubingmodel_Timestep_" & CStr(TimeStep) & "_PwfLoop=" & PwfLoop & ".txt"
      Pathfile.NewOutputFile = OutputFile
      Dim obj5 As New TubingModel
      If TimeStep = 1 Then
        Pwf = obj5.runmodel(Pathfile, n, A_tub, D_tub, g, DelL, e, Np, Zp, Np + Ninj, Zp, P1, Temperature, pc, tc, omega, R, outflcon, inflcon,
m, liqstand, tubing_length, InjectionPoint)
      Else
        Pwf = obj5.runmodel(Pathfile, n, A_tub, D_tub, g, DelL, e, Np, Zp, Np + Ninj, Zsu, P1, Temperature, pc, tc, omega, R, outflcon, inflcon,
m, liqstand, tubing_length, InjectionPoint)
      End If
      New_Pwf = Pwf
      Return True
    Catch ex As Exception
      Return False
    End Try
  End Function
  Private Function Mainseparatormodel(ByVal Pathfile As PathFile) As Boolean
    Try
      Dim OutputFile As String
      OutputFile = Left(Pathfile.OutputFile, Pathfile.OutputFile.Length - 4)
      OutputFile = OutputFile & "_SeparatorModel_Timestep_" & CStr(TimeStep) & ".txt"
      Pathfile.NewOutputFile = OutputFile
      Dim obj6 As New SeparatorModel
      obj6.runMainModel(Pathfile, n, Psep1, psep2, psep3, Np + Ninj, Zp, Tatm2, pc, tc, omega, R, outflcon, inflcon, m, Qinj_input, liqstand,
numberofwell, Zsu, TimeStep)
      Ninj = obj6.Ninj
      Zinj = obj6.Zinj
      qo_out = obj6.qo_out
      qg_out = obj6.qg_out
      Qinj_Out = obj6.OutQinj
      Return True
    Catch ex As Exception
      Return False
    End Try
  End Function
  Private Function MainUpdatePres(ByVal Pathfile As PathFile) As Boolean
    Try
      Dim OutputFile As String
      OutputFile = Left(Pathfile.OutputFile, Pathfile.OutputFile.Length - 4)
      OutputFile = OutputFile & "_UpdatePres_Timestep_" & CStr(TimeStep) & ".txt"
      Pathfile.NewOutputFile = OutputFile
      Dim obj8 As New CompositionModel
      Dim Chk As Boolean = True
      Dim Pres_new As Double
      Dim Pr, OutPutPr As Double
      Pr = Pressure
      Dim old_Error, new_Error As Double
      Dim GetOError As Boolean = True
      While Chk
        obj8.Run(Pathfile, 0.5, n, Zi_new, Pr, Temperature, pc, tc, omega, R, outflcon, inflcon, m, liqstand)
        Pres_new = (obj8.L * obj8.oilDensity) + (obj8.V * obj8.GasDensity)
        If Not GetOError Then
          new_Error = Abs(Pres_new - Pres)
          If new_Error > old_Error Then
            OutPutPr = Pr + 1
            Chk = False
          Else
```

```
                old_Error = new_Error
            End If
          End If
          If GetOError Then
            old_Error = Abs(Pres_new - Pres)
            GetOError = False
          End If
          Pr = Pr - 1
        End While
        Pressure = OutPutPr
        Pbar = OutPutPr
        Return True
      Catch ex As Exception
        Return False
      End Try
    End Function
    Private Function CalZsu(ByVal pathfile As PathFile) As Boolean
      Try
        Dim OutputFile As String
        OutputFile = Left(pathfile.OutputFile, pathfile.OutputFile.Length - 4)
        OutputFile = OutputFile & "_CalZsu_Timestep_" & CStr(TimeStep) & ".txt"
        pathfile.NewOutputFile = OutputFile
        Dim obj10 As New CalZsu
        If TimeStep > 1 Then
          Zsu = obj10.Run(pathfile, n, Np, Zp, Qinj_input, Zinj, psep2, Tatm2, pc, tc, omega, R, outflcon, inflcon, m, liqstand)
        End If
        Return True
      Catch ex As Exception
        Return False
      End Try
    End Function
#End Region
#Region "Writeoutput"
    Public Function WriteOutput(ByVal pathfile As PathFile)
      Dim OutputFile As String
      Try
        OutputFile = Left(pathfile.OutputFile, pathfile.OutputFile.Length - 4)
        OutputFile = OutputFile & "_Main" & ".txt"
        pathfile.NewOutputFile = OutputFile
      Dim ow As New WriteOutput
      Dim Strdata As String = ""
      Strdata += "---------------------- Main Program TimeStep=" & CStr(TimeStep)
      Strdata += "----------------------" & vbCrLf
      Strdata += "Pressure=" & CStr(Pressure) & vbCrLf
      Strdata += "P1=" & CStr(P1) & vbCrLf
      Strdata += "Pwf=" & CStr(Pwf) & vbCrLf
      Strdata += "qoatm=" & CStr(qo_out * numberofwell) & vbCrLf
      Strdata += "qgatm=" & CStr(qg_out * numberofwell) & vbCrLf
      Strdata += "qinj=" & CStr(Qinj_Out) & vbCrLf
      Strdata += "-----------------------------------------------" & vbCrLf
      Dim tempdata As String
      tempdata = ow.GetFileContents(pathfile.NewOutputFile)
      Strdata = tempdata & vbCrLf & Strdata
      ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
      Catch ex As Exception
      End Try
      Try
        CreatRow(dst, TimeStep, Pressure, P1, Pwf, qo_out * numberofwell, qg_out * numberofwell, Qinj_Out)
      Catch ex As Exception
      End Try
    End Function
#End Region
    Private Sub CreatRow(ByRef dt As System.Data.DataTable, ByVal otp As Integer, _
ByVal oPr As Double, ByVal oP1 As Double, ByVal oPwf As Double, ByVal oqotam As Double, ByVal oqgatm As Double, _
ByVal qinj As Double)
      Dim dr As DataRow = dt.NewRow
      dr("TimeStep") = otp
      dr("Pressure") = oPr
      dr("P1") = oP1
      dr("Pwf") = oPwf
      dr("qoatm") = oqotam
      dr("qgatm") = oqgatm
      dr("qinj") = qinj
      dt.Rows.Add(dr)
    End Sub
    Private Function CreateEventDS() As DataSet
      Dim dsset1 As New DataSet
      Dim dtTable1 As New System.Data.DataTable
      With dtTable1.Columns
        .Add("TimeStep")
        .Add("Pressure")
        .Add("P1")
        .Add("Pwf")
        .Add("qoatm")
```

```
      .Add("qgatm")
      .Add("qinj")
    End With
    dsset1.Tables.Add(dtTable1)
    Return dsset1
  End Function
End Class
```

```
Oildensity
Imports System.Math
Public Class oildens
  Private Liqstand() As Double
  Public density As Double "liqdens
  Public Function run(ByVal nc As Integer, ByVal Liqstand_i() As Double, ByVal comp() As Double, ByVal p As Double, _
  ByVal temperature_R As Double, ByVal m() As Double, ByVal liqdens As Double) As Double
    Dim diff, ma, mwap As Double      Dim ferror, olddensity, numer, delrowp, delrowt As Double      Dim i, j As Integer
    Dim x1, x2, x3, x4, x5 As Double
    Liqstand = Liqstand_i      density = liqdens      ma = 0      diff = 0.01      For i = 0 To nc - 1      ma = ma + comp(i) * m(i)
    Next
       ferror = 0.001 If density < 0.5 Then density = 0.77      Dim Lp As Boolean = True      While Lp      olddensity = density
      If (Abs(m(0) - 16.043) < diff) Then
        Liqstand(0) = 0.312 + density * 0.45  "'paper is 0.45 code is 28.0665
        Liqstand(0) = Liqstand(0)
      End If
      If (Abs(m(1) - 30.07) < diff) Then
        Liqstand(1) = 15.3 + 0.3167 * density           Liqstand(1) = Liqstand(1)
      End If
      numer = 0.0
      For i = 0 To nc - 1
        numer = numer + comp(i) * m(i) / (Liqstand(i))
      Next
      If ma = 0 And numer = 0 Then           density = 0
      Else          density = (ma / numer)
      End If
      If (Abs(density - olddensity) <= ferror) Then Lp = False
    End While
    Dim Psta As Double      Psta = density      'density = density
    delrowp = (0.167 + 16.181 * (10 ^ (-0.0425 * density))) * (p / 1000)
    delrowp = delrowp - 0.01 * (0.299 + 263 * (10 ^ (-0.0603 * density))) * (p / 1000) ^ 2
    If density = 0 Then
      x1 = 0
    Else
      x1 = density ^ (-0.951)
    End If
    x2 = temperature_R - 520
    x2 = Max(0.0, x2)
    x3 = x2 ^ 0.938
    delrowt = (0.00302 + 1.505 * (x1)) * (x3)
    delrowt = delrowt - (0.0216 - 0.0233 * (10 ^ (-0.0161 * density))) * ((x2) ^ (0.475))
    density = Psta + delrowp - delrowt
    Return density
  End Function
End Class
```

```
Pathfile
Public Class PathFile
  Public InputFile As String
  Public OutputFile As String
  Public NewOutputFile As String
End Class
```

```
Regimes
Imports System.Math
Public Class Regimes
  Public Vb As Double
  Public Vt As Double
  Public EL As Double
  Public DelPH As Double
  Public Re As Double
  Public fmoody As Double
  Public DelPf As Double
  Private Vmsl, Vmsg As Double
  #Region "Tubing"
  Public Function Bubble(ByVal pathfile As PathFile, ByVal Vm As Double, ByVal Vsg As Double, ByVal Pl As Double, _
  ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
    'Dim fmoody As Double
    Vb = 1.41 * ((g * 95 * (Pl - Pg)) / (Pl ^ 2)) ^ (0.25)
    Vt = (1.2 * Vm) + Vb
    EL = 1 - (Vsg / Vt)
    DelPH = (delL / 144) * ((g / g) * ((Pl * EL) + (1 - EL) * Pg))
    Re = ((D * Vm * Pl) / liquidvis) * 1448
```

```
        fmoody = moody(E, D, Re)
        DelPf = (2 * fmoody * Vm * Vm * Pl * delL) / (144 * g * D)
      WriteOutput(pathfile, "Bubble", Vm, Vsg, Pl, Pg, D, g, delL, E, liquidvis, 0, 0)
    End Function
    Public Function Slug(ByVal pathfile As PathFile, ByVal Vsg As Double, ByVal Vm As Double, ByVal Pl As Double, _
    ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        'Dim fmoody As Double
        Vb = 0.345 * ((D * g * (Pl - Pg)) / (Pl)) ^ (0.5)
        Vt = 1.2 * Vm + Vb
        EL = 1 - (Vsg / Vm)
        Re = (((D * Vm) * Pl) * 1448) / liquidvis
        DelPH = (delL / 144) * ((g / g) * ((Pl * EL) + (1 - EL) * Pg))
        'fmoody = moody2(E, D, Re, 0.01)
        fmoody = moody(E, D, Re)
        DelPf = (2 * fmoody * Vm * Vm * Pl * delL * (EL)) / (144 * g * D)
        'DelPf = 0.001295 * fmoody * Vm * Vm * Pl * delL * EL / D
        WriteOutput(pathfile, "Slug", Vm, Vsg, Pl, Pg, D, g, delL, E, liquidvis, 0, 0)
    End Function
    Public Function AnnularMist(ByVal pathfile As PathFile, ByVal Vsl As Double, ByVal Vm As Double, ByVal Vsg As Double, _
    ByVal Pl As Double, _
      ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        'Dim fmoody As Double
        EL = Vsl / Vm
        DelPH = (delL / 144) * ((g / g) * ((Pl * EL) + (1 - EL) * Pg))
        fmoody = moody(E, D, Re)
        DelPf = (2 * fmoody * Vsg * Vsg * Pg * delL) / (144 * g * D)
        WriteOutput(pathfile, "AnnularMist", Vm, Vsg, Pl, Pg, D, g, delL, E, liquidvis, 0, 0)
    End Function
    Public Function Froth(ByVal pathfile As PathFile, ByVal Vsg As Double, ByVal Vm As Double, ByVal Vmsl As Double, ByVal Vmsg As
Double, ByVal Pl As Double, _
    ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        Re = ((D * Vsg * Pg) / liquidvis) * 1448
        Dim delpf1, delpf2, vsg2, vsg3, fmoody As Double
        Dim EEl As Double
        EEl = 1 - (Vsg / Vm)
        fmoody = moody(E, D, Re)
        delpf1 = (2 * fmoody * Vm * Vm * Pl * delL * (EEl)) / (144 * g * D)
        delpf2 = (2 * fmoody * Vsg * Vsg * Pg * delL) / (144 * g * D)
        vsg2 = ((Vmsl / 0.263) + 8.6) / Vmsg
        vsg3 = (((100 * Vmsl) ^ -0.152) * 70) / Vmsg
        DelPf = (delpf2 - delpf1) * ((Vsg - vsg2) / (vsg3 - vsg2)) + delpf1
        WriteOutput(pathfile, "Froth", Vm, Vsg, Pl, Pg, D, g, delL, E, liquidvis, 0, 0)
    End Function
#End Region
#Region "pipeline"
    Public Function Bubble_p(ByVal pathfile As PathFile, ByVal Vm As Double, ByVal Pl As Double, _
     ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        Re = ((D * Vm * Pl) / liquidvis) * 1448
        fmoody = moody(E, D, Re)
        DelPf = (2 * fmoody * Vm * Vm * Pl * delL) / (144 * g * D)
        WriteOutput(pathfile, "Bubble", Vm, 0, Pl, 0, D, g, delL, E, liquidvis, 0, 0)
    End Function
    Public Function Slug_p(ByVal pathfile As PathFile, ByVal ql As Double, ByVal qg As Double, ByVal Vm As Double, ByVal Pl As Double, _
     ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        'Dim fmoody As Double
        EL = ql / (ql + qg)
        Re = (((D * Vm) * Pl) * 1448) / liquidvis
        DelPH = (delL / 144) * ((g / g) * ((Pl * EL) + (1 - EL) * Pg))
        fmoody = moody(E, D, Re)
        DelPf = (2 * fmoody * Vm * Vm * Pl * delL * (EL)) / (144 * g * D)
        WriteOutput(pathfile, "Slug", Vm, 0, Pl, Pg, D, g, delL, E, liquidvis, ql, qg)
    End Function
    Public Function AnnularMist_p(ByVal pathfile As PathFile, ByVal Vsg As Double, _
      ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        Re = (D * Vsg * Pg) / liquidvis * 1448
        fmoody = moody(E, D, Re)
        DelPf = (2 * fmoody * Vsg * Vsg * Pg * delL) / (144 * g * D)
        WriteOutput(pathfile, "AnnularMist", 0, Vsg, 0, Pg, D, g, delL, E, liquidvis, 0, 0)
    End Function
    Public Function Froth_p(ByVal pathfile As PathFile, ByVal ql As Double, ByVal qg As Double, _
    ByVal Vsg As Double, ByVal Vm As Double, ByVal Vmsl As Double, ByVal Vmsg As Double, ByVal Pl As Double, _
    ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, _
ByVal liquidvis As Double)
        EL = ql / (ql + qg)
        Re = ((D * Vsg * Pg) / liquidvis) * 1448
        Dim delpf1, delpf2, vsg2, vsg3, fmoody As Double
        Dim EEl As Double
        fmoody = moody(E, D, Re)
        delpf1 = (2 * fmoody * Vm * Vm * Pl * delL * (EL)) / (144 * g * D)
        delpf2 = (2 * fmoody * Vsg * Vsg * Pg * delL) / (144 * g * D)
        vsg2 = ((Vmsl / 0.263) + 8.6) / Vmsg
        vsg3 = (((100 * Vmsl) ^ -0.152) * 70) / Vmsg
        DelPf = (delpf2 - delpf1) * ((Vsg - vsg2) / (vsg3 - vsg2)) + delpf1
        Me.Vmsl = Vmsl
```

```
        Me.Vmsg = Vmsg
        WriteOutput(pathfile, "Froth", Vm, Vsg, Pl, Pg, D, g, delL, E, liquidvis, ql, qg)
    End Function
#End Region
    Private Function WriteOutput(ByVal pathfile As PathFile, ByVal Type As String, ByVal Vm As Double, ByVal Vsg As Double, ByVal Pl As Double, _ ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double, ByVal ql As Double, ByVal qg As Double)
        '--------writefile----------
        '------------------write file------------------
        Dim Strdata As String
        Dim ow As New WriteOutput
        Strdata = vbCrLf & "------------------------Begin " & Type & " Flow Regime --------------------" & vbCrLf
        Strdata += "----------------------Input-----------------------" & vbCrLf
        Strdata += "Vm =" & CStr(Vm) & vbCrLf
        Strdata += "Vsg=" & CStr(Vsg) & vbCrLf
        Strdata += "Pl=" & CStr(Pl) & vbCrLf
        Strdata += "Pg =" & CStr(Pg) & vbCrLf
        Strdata += "D" & CStr(D) & vbCrLf
        Strdata += "g=" & CStr(g) & vbCrLf
        Strdata += "delL=" & CStr(delL) & vbCrLf
        Strdata += "E=" & CStr(E) & vbCrLf
        Strdata += "liquidvis=" & CStr(liquidvis) & vbCrLf
        Strdata += "ql=" & CStr(ql) & vbCrLf
        Strdata += "qg =" & CStr(qg) & vbCrLf
        Strdata += "Vmsl=" & CStr(Vmsl) & vbCrLf
        Strdata += "Vmsg =" & CStr(Vmsg) & vbCrLf
        Strdata += "------------------------output------------------------" & vbCrLf
        Strdata += "Vb=" & CStr(Vb) & vbCrLf
        Strdata += "Vt=" & CStr(Vt) & vbCrLf
        Strdata += "EL=" & CStr(EL) & vbCrLf
        Strdata += "DelPH=" & CStr(DelPH) & vbCrLf
        Strdata += "Re=" & CStr(Re) & vbCrLf
        Strdata += "Ff=" & CStr(fmoody) & vbCrLf
        Strdata += "DelPf=" & CStr(DelPf) & vbCrLf
        Strdata += "------------------------End Regime --------------------" & vbCrLf
        Dim tempdata As String
        tempdata = ow.GetFileContents(pathfile.NewOutputFile)
        Strdata = tempdata & vbCrLf & Strdata
        ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
    End Function
    Public Function moody(ByVal epsilon As Double, ByVal diameter As Double, ByVal Nre As Double) As Double
        Dim moodyold As Double
        Dim imoody, imoody2, err1, err2, derr As Double
        Dim i As Integer
        moodyold = 0.01
        imoody = moodyold
        i = 1
        Dim chkl As Boolean = True
        Dim t1, t2, t3 As Double
        While chkl
            moodyold = imoody
            t1 = (2 * epsilon) / diameter
            t2 = 18.7 / (Nre * Sqrt(moodyold))
            t3 = 18.7 / (Nre * Sqrt(moodyold * 1.00001))
            imoody = 1 / ((1.74 - 2 * Log10(t1 + t2)) ^ 2)
            imoody2 = 1 / ((1.74 - 2 * Log10(t1 + t3)) ^ 2)
            err1 = imoody - moodyold
            err2 = imoody2 - moodyold * 1.00001
            derr = (err2 - err1) / (0.00001 * moodyold)
            i = i + 1
            imoody = moodyold - err1 / derr
            imoody = Max(10 ^ -7, imoody)
            If (Abs(imoody - moodyold) > 0.001) And i < 10 Then
                chkl = True
            ElseIf (i > 9) Then
                imoody = 0.01
                chkl = False
            Else
                chkl = False
            End If
        End While
        Return imoody
    End Function
    Private Function moody1(ByVal epsilon As Double, ByVal diameter As Double, ByVal Nre As Double) As Double
        Dim moodyold As Double
        Dim imoody, imoody1, imoody2, err1, err2, derr As Double
        Dim i As Integer
        moodyold = 0.0005
        imoody = moodyold
        i = 1
        Dim chkl As Boolean = True
        Dim t1, t2, t3 As Double
        While chkl
            'moodyold = imoody
```

```
        t1 = (2 * epsilon) / diameter
        t2 = 18.7 / (Nre * Sqrt(moodyold))
        t3 = 18.7 / (Nre * Sqrt(moodyold * 1.00001))        imoody = 1.74 - (2 * Log10(t1 + t2))        imoody2 = 1 / moodyold
        err1 = imoody - imoody2        i = i + 1        Debug.WriteLine(moodyold)        If Abs(err1) > 0.00001 Then        If i > 100000000
Then
            chkl = False        Else        chkl = True        moodyold += moodyold * 10 ^ -5
          End If
        Else
          chkl = False        End If        End While        Return moodyold ^ 2 / 4    End Function
    Private Function moody2(ByVal epsilon As Double, ByVal diameter As Double, ByVal Nre As Double, ByVal moodyold As Double) As
Double
      Dim imoody, imoody2, err1, err2, derr As Double        Dim i As Integer        imoody = moodyold        i = 1        Dim chkl As Boolean = True
      Dim t1, t2, t3 As Double        While chkl        moodyold = imoody        t1 = (2 * epsilon) / diameter        t2 = 18.7 / (Nre *
Sqrt(moodyold))
        t3 = 18.7 / (Nre * Sqrt(moodyold * 1.00001))        imoody = 1 / ((1.74 - 2 * Log10(t1 + t2)) ^ 2)        imoody2 = 1 / ((1.74 - 2 *
Log10(t1 + t3)) ^ 2)
        err1 = imoody - moodyold        err2 = imoody2 - moodyold * 1.00001        derr = (err2 - err1) / (0.00001 * moodyold)
        i = i + 1        imoody = moodyold - (err1 / derr)        imoody = Max(10 ^ -7, imoody)        If (Abs(imoody - moodyold) > 0.001) And
i < 10 Then
          chkl = True        ElseIf (i > 9) Then        imoody = 0.01        chkl = False        Else        chkl = False        End If
      End While        Return imoody    End Function
End Class
```

```
Pipeline model
Imports System.Math
Public Class PipelineModel
    Public Vsl As Double
    Public Vsg As Double
    Public Vm As Double
    Public Vmsl As Double
    Public Vmsg As Double
    Public Vb As Double
    Public Vt As Double
    Public EL As Double
    Public DelPH As Double
    Public Re As Double
    Public DelPf As Double
    Public fmoody As Double
    Private Pl, Pg, ql, qg, liquidvis As Double
    Public Function runModel(ByVal pathfile As PathFile, ByVal n As Integer, ByVal Np As Double, _
      ByVal A As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal e As Double, _
    ByVal Liqstand() As Double, ByVal zp() As Double, ByVal Psep1 As Double, ByVal Tatm As Double, _
    ByVal pc() As Double, ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
    ByVal inflcon As Double, ByVal m() As Double, ByVal pipeline_length As Double) As Double
        Dim P2 As Double = 0.0
        Dim pipeline_length_1 As Double
        pipeline_length_1 = pipeline_length
        Dim nL As Integer
        Dim pipeline_length_2 As Double
        P2 = Psep1
        If pipeline_length_1 >= 100 Then
          nL = Fix(pipeline_length_1 / 100)
          For i As Integer = 1 To nL
            P2 = FindP2(pathfile, n, Np, A, D, g, delL, e, Liqstand, zp, P2, Tatm, pc, tc, omega, R, outflcon, inflcon, m, 100)
          Next
          pipeline_length_2 = pipeline_length_1 - (100 * nL)
          If pipeline_length_2 > 0 Then
            P2 = FindP2(pathfile, n, Np, A, D, g, delL, e, Liqstand, zp, P2, Tatm, pc, tc, omega, R, outflcon, inflcon, m, pipeline_length_2)
          End If
        Else
          P2 = FindP2(pathfile, n, Np, A, D, g, delL, e, Liqstand, zp, P2, Tatm, pc, tc, omega, R, outflcon, inflcon, m, pipeline_length_1)
        End If
        WriteOutput(pathfile, n, Np, A, D, g, delL, e, Liqstand, zp, Psep1, Tatm, pc, tc, omega, R, outflcon, inflcon, m, pipeline_length, P2)
        Return P2
    End Function
    Private Function FindP2(ByVal pathfile As PathFile, ByVal n As Integer, ByVal Np As Double, _
      ByVal A As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal e As Double, _
    ByVal Liqstand() As Double, ByVal zp() As Double, ByVal Psep1 As Double, ByVal Tatm As Double, _
    ByVal pc() As Double, ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
    ByVal inflcon As Double, ByVal m() As Double, ByVal pipeline_length As Double) As Double
        '--composition model---'
        Dim oilDensity, GasDensity, gasViscosity As Double
        Dim X(), Y() As Double
        Dim L, V As Double
        Dim obj1 As New CompositionModel
        obj1.Run(pathfile, 0.5, n, zp, Psep1, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand)
        GasDensity = obj1.GasDensity
        X = obj1.x
        Y = obj1.y
        L = obj1.L
        V = obj1.V
        oilDensity = obj1.oilDensity
        Dim viscman As New viscman
```

```
    liquidvis = visman.run(pathfile, n, Tatm, X, m, 1, oilDensity)
    gasViscosity = visman.run(pathfile, n, Tatm, Y, m, 0, GasDensity)
    '==========='
    Dim mo, mg As Double
    Dim i As Integer
    For i = 0 To n - 1
      mo = mo + X(i) * m(i)
    Next
    For i = 0 To n - 1
      mg = mg + Y(i) * m(i)
    Next
    Pl = oilDensity
    Pg = GasDensity
    Dim Npo, Npg As Double
    Npo = Np * L
    Npg = Np * V
    ql = ConvertNpToql(Npo, mo, Pl)
    qg = ConvertNpToqg(Npg, mg, Pg)
    Vsl = ql / A
    Vsg = qg / A
    Vm = Vsl + Vsg
    Vmsl = Vsl * (((Pl * 72) / (63.37 * 50)) ^ (0.25))
    Vmsg = Vsg * ((Pg) / 0.078) ^ (0.33) * ((Pl * 72) / (62.37 * 50)) ^ 0.25
    Dim b1 As Double      Dim b2 As Double      Dim b3 As Double
    b1 = ((100 * Vmsl) ^ 0.17211) / 1.96
    b2 = (Vmsl / 0.263) + 8.6
    b3 = 70 * ((100 * Vmsl) ^ -0.152)
    If Vmsl > 4 Then
      If Vmsg < b1 Then
        Call Bubble(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
      ElseIf Vmsg >= b1 Then
        If Vmsg < 26.5 Then
          Call Slug(pathfile, ql, qg, Pl, Pg, D, g, delL, e, liquidvis)
        ElseIf Vmsg >= 26.5 Then
          Call AnnularMist(pathfile, Pg, D, g, delL, e, liquidvis)
        End If
      End If
    ElseIf Vmsl <= 4 Then
      If Vmsg < b1 Then
        Call Bubble(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
      ElseIf Vmsg >= b1 Then
        If Vmsg < b2 Then
          Call Slug(pathfile, ql, qg, Pl, Pg, D, g, delL, e, liquidvis)
        ElseIf Vmsg >= b2 Then
          If Vmsg < b3 Then
            Call Froth(pathfile, ql, qg, Pl, Pg, D, g, delL, e, liquidvis)
          ElseIf Vmsg >= b3 Then
            Call AnnularMist(pathfile, Pg, D, g, delL, e, liquidvis)
          End If
        End If
      End If
    End If
    Return Psep1 + (DelPf * pipeline_length)
  End Function
  Private Function ConvertNpToql(ByVal npo As Double, ByVal Mo As Double, ByVal po As Double) As Double
    Dim t1, t2 As Double
    t1 = npo * Mo
    t2 = 86400 * po
    Return t1 / t2
  End Function    Private Function ConvertNpToqg(ByVal npg As Double, ByVal Mg As Double, ByVal pg As Double) As Double
    Dim t1, t2 As Double
    t1 = npg * Mg
    t2 = 86400 * pg
    Return t1 / t2
  End Function
  Private Function WriteOutput(ByVal pathfile As PathFile, ByVal n As Integer, ByVal Np As Double, _
   ByVal A As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal e As Double, _
 ByVal Liqstand() As Double, ByVal zp() As Double, ByVal Psep1 As Double, ByVal Tatm As Double, _
 ByVal pc() As Double, ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
 ByVal inflcon As Double, ByVal m() As Double, ByVal pipeline_length As Double, ByVal P2 As Double)
    "-----------------write file-----------------
    Dim Strdata As String
    Dim ow As New WriteOutput
    Strdata = vbCrLf & "------------------------Begin Pipeline Model ---------------------" & vbCrLf
    Strdata += "---------------------- Input---------------------- " & vbCrLf
    Strdata += "qo=" & CStr(ql) & vbCrLf
    Strdata += "qg=" & CStr(qg) & vbCrLf
    Strdata += "A=" & CStr(A) & vbCrLf
    Strdata += "OilDensity(pl)=" & CStr(Pl) & vbCrLf
    Strdata += "GasDensity(pg)" & CStr(Pg) & vbCrLf
    Strdata += "D=" & CStr(D) & vbCrLf
    Strdata += "g=" & CStr(g) & vbCrLf
    Strdata += "delL=" & CStr(delL) & vbCrLf
    Strdata += "E=" & CStr(e) & vbCrLf
```

```
        Strdata += "liquidvis=" & CStr(liquidvis) & vbCrLf
        Strdata += "pipeline_length=" & CStr(pipeline_length) & vbCrLf
        Strdata += "------------------------ output------------------------- " & vbCrLf
        Strdata += "Vm=" & CStr(Vm) & vbCrLf
        Strdata += "Vmsl=" & CStr(Vmsl) & vbCrLf
        Strdata += "Vmsg=" & CStr(Vmsg) & vbCrLf
        Strdata += "EL=" & CStr(EL) & vbCrLf
        Strdata += "DelPH=" & CStr(DelPH) & vbCrLf
        Strdata += "Re=" & CStr(Re) & vbCrLf
        Strdata += "Ff=" & CStr(fmoody) & vbCrLf
        Strdata += "DelPf=" & CStr(DelPf) & vbCrLf
        Strdata += "P2=" & CStr(P2) & vbCrLf
        Strdata += "------------------------End Pipeline Model ---------------------" & vbCrLf
        Dim tempdata As String
        tempdata = ow.GetFileContents(pathfile.NewOutputFile)
        Strdata = tempdata & vbCrLf & Strdata
        ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
    End Function
#Region "Regimes"
    Private Function OutputRegimes(ByVal data As Regimes)
        EL = data.EL
        Re = data.Re
        DelPH = data.DelPH
        DelPf = data.DelPf
        fmoody = data.fmoody
    End Function
    Private Function Bubble(ByVal pathfile As PathFile, ByVal Pl As Double, _
 ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        Dim ObjR As New Regimes
        ObjR.Bubble_p(pathfile, Vm, Pl, D, g, delL, E, liquidvis)
        OutputRegimes(ObjR)
    End Function
    Private Function Slug(ByVal pathfile As PathFile, ByVal ql As Double, ByVal qg As Double, ByVal Pl As Double, ByVal Pg As Double, _
     ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        Dim ObjR As New Regimes
        ObjR.Slug_p(pathfile, ql, qg, Vm, Pl, Pg, D, g, delL, E, liquidvis)
        OutputRegimes(ObjR)
    End Function
    Private Function AnnularMist(ByVal pathfile As PathFile, ByVal Pg As Double, ByVal D As Double, ByVal g As Double, _
    ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        'Dim fmoody As Double
        Dim ObjR As New Regimes
        ObjR.AnnularMist_p(pathfile, Vsg, Pg, D, g, delL, E, liquidvis)
        OutputRegimes(ObjR)
    End Function
    Private Function Froth(ByVal pathfile As PathFile, ByVal ql As Double, ByVal qg As Double, ByVal Pl As Double, _
    ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
        Dim ObjR As New Regimes
        ObjR.Froth_p(pathfile, ql, qg, Vsg, Vm, Vmsl, Vmsg, Pl, Pg, D, g, delL, E, liquidvis)
        OutputRegimes(ObjR)
    End Function
#End Region
End Class
---------------------------------------------------------------------------------------------
Reservoir model
Imports System.Math
Public Class ReservoirModel
    Public So As Double
    Public Sg As Double
    Public Kro As Double
    Public Krg As Double
    Public qo As Double
    Public qg As Double
    Public Npo As Double
    Public Npg As Double
    Public Np As Double
    Public Zp() As Double
    Public Nk As Double
    Public Nki() As Double
    Public Zi() As Double
    Public Pres As Double
    Public SumNpi As Double
    Public Mo, Mg As Double
    Public po, pg As Double
    Public Pres_old As Double
    Public Function runModel(ByVal pathfile As PathFile, _
    ByVal n As Integer, ByVal z() As Double, _
    ByVal pressure As Double, ByVal temperature_R As Double, ByVal pc() As Double, _
    ByVal tc() As Double, ByVal omega() As Double, _
    ByVal R As Double, ByVal Liqstand() As Double, _
     ByVal outflcon As Double, ByVal inflcon As Double, ByVal m() As Double, _
      ByVal porosity As Double, ByVal sgr As Double, ByVal sor As Double, _
      ByVal noil As Double, ByVal ngas As Double, _
       ByVal k As Double, ByVal h As Double, ByVal re As Double, ByVal rw As Double, _
```

```
ByVal pbar As Double, ByVal pwf As Double, ByVal dt As Double, ByVal numberofwell As Double, ByVal Area As Double)
"--composition model---"
Dim oilDensity, GasDensity, oilViscosity, gasViscosity As Double
Dim X(), Y() As Double
Dim L, V As Double
Dim obj1 As New CompositionModel
obj1.Run(pathfile, 0.5, n, z, pressure, temperature_R, pc, tc, omega, R, outflcon, inflcon, m, Liqstand)
GasDensity = obj1.GasDensity
X = obj1.x
Y = obj1.y
L = obj1.L
V = obj1.V
oilDensity = obj1.oilDensity
Dim viscman As New viscman
oilViscosity = viscman.run(pathfile, n, temperature_R, X, m, 1, oilDensity)
gasViscosity = viscman.run(pathfile, n, temperature_R, Y, m, 0, GasDensity)
"--end composition model---"
Dim po, pg, uoil, ugas As Double
po = oilDensity
pg = GasDensity
uoil = oilViscosity
ugas = gasViscosity
"-------parameter ----------------
Dim i As Integer
For i = 0 To n - 1
   Mo = Mo + X(i) * m(i)
Next
For i = 0 To n - 1
   Mg = Mg + Y(i) * m(i)
Next
Dim mt1, mt2 As Double
mt1 = (Mo * L) / po
mt2 = (Mg * (1 - L)) / pg
So = mt1 / (mt1 + mt2)
Sg = 1 - So
Kro = ((So - sor) / (1 - sor - sgr)) ^ noil
Krg = ((Sg - sgr) / (1 - sor - sgr)) ^ ngas
"---------------------------
"----flowrate---------------
qo = (0.00708 * k * Kro * h / uoil) * ((pbar - pwf) / (Log(re / rw) - 0.75))
qg = (0.00708 * k * Krg * h / ugas) * ((pbar - pwf) / (Log(re / rw) - 0.75))
Npo = (5.615 * qo * po) / Mo
Npg = (5.615 * qg * pg) / Mg
Np = Npo + Npg
For i = 0 To n - 1
   ReDim Preserve Zp(i)
   Zp(i) = New Double
   Zp(i) = ((Npo * X(i)) + (Npg * Y(i))) / Np
Next
"--------------------
"------NK-----------------
Dim Mres As Double
Pres_old = (L * po) + (V * pg)
For i = 0 To n - 1
   Mres = Mres + z(i) * m(i)
Next
Nk = (Area * Pres_old * h * porosity) / Mres
"-------Nki---------------------
For i = 0 To n - 1
   ReDim Preserve Nki(i)
   Nki(i) = New Double
   Nki(i) = z(i) * Nk
Next
"---- Product Mole ------------
Dim prm() As Double
For i = 0 To n - 1
   ReDim Preserve prm(i)
   prm(i) = New Double
   prm(i) = Np * Zp(i) * dt
Next
"---total product mass------
SumNpi = 0
For i = 0 To n - 1
   SumNpi = SumNpi + prm(i)
Next
"---new reservoir mole------
Dim Nki_1() As Double
If numberofwell > 1 Then
   For i = 0 To n - 1
      ReDim Preserve Nki_1(i)
      Nki_1(i) = New Double
      Nki_1(i) = Nki(i) - (numberofwell * prm(i))
   Next
Else
```

```
    For i = 0 To n - 1
      ReDim Preserve Nki_1(i)
      Nki_1(i) = New Double
      Nki_1(i) = Nki(i) - prm(i)
    Next
  End If
  "---total reservoir mass -------
  Dim SumNki_1 As Double = 0
  For i = 0 To n - 1
    SumNki_1 = SumNki_1 + Nki_1(i)
  Next
  "----new reservoir composition
  For i = 0 To n - 1
    ReDim Preserve Zi(i)
    Zi(i) = New Double
    Zi(i) = Nki_1(i) / SumNki_1
  Next
  Dim Mres_new() As Double
  For i = 0 To n - 1
    ReDim Preserve Mres_new(i)
    Mres_new(i) = New Double
    Mres_new(i) = Zi(i) * m(i)
  Next
  Dim summres_new As Double = 0
  For i = 0 To n - 1
    summres_new = summres_new + Mres_new(i)
  Next
  Pres = (SumNki_1 * summres_new) / (Area * h * porosity)
  WriteOutPut(pathfile, n, z, pressure, temperature_R, pc, tc, _
    omega, R, Liqstand, outflcon, inflcon, m, porosity, sgr, _
    sor, noil, ngas, k, h, re, rw, pbar, pwf, dt, L, V, uoil, ugas, X, Y, prm, Nki_1, Mres_new, SumNki_1, Area)
End Function
Private Function WriteOutPut(ByVal pathfile As PathFile, _
ByVal n As Integer, ByVal z() As Double, _
ByVal pressure As Double, ByVal temperature_R As Double, ByVal pc() As Double, _
ByVal tc() As Double, ByVal omega() As Double, _
ByVal R As Double, ByVal Liqstand() As Double, _
 ByVal outflcon As Double, ByVal inflcon As Double, ByVal m() As Double, _
  ByVal porosity As Double, ByVal sgr As Double, ByVal sor As Double, _
  ByVal noil As Double, ByVal ngas As Double, _
   ByVal k As Double, ByVal h As Double, ByVal re As Double, ByVal rw As Double, _
   ByVal pbar As Double, ByVal pwf As Double, ByVal dt As Double, ByVal L As Double, ByVal V As Double, _
   ByVal uoil As Double, ByVal ugas As Double, ByVal x() As Double, ByVal y() As Double, _
   ByVal prm() As Double, ByVal Nki_1() As Double, ByVal Mres_new() As Double, ByVal sumnki_1 As Double, ByVal Area As Double)
   "------------------write file------------------
   Dim Strdata As String
   Dim ow As New WriteOutput
   Strdata = vbCrLf & "------------------------Begin Reservoir Model ---------------------" & vbCrLf
   Strdata += "----------------------- Input----------------------- " & vbCrLf
   Strdata += "L=" & CStr(L) & vbCrLf
   Strdata += "V=" & CStr(V) & vbCrLf
   Strdata += "porosity=" & CStr(porosity) & vbCrLf
   Strdata += "sgr=" & CStr(sgr) & vbCrLf
   Strdata += "sor=" & CStr(sor) & vbCrLf
   Strdata += "noil=" & CStr(noil) & vbCrLf
   Strdata += "ngas=" & CStr(ngas) & vbCrLf
   Strdata += "oilDensity=" & CStr(po) & vbCrLf
   Strdata += "GasDensity=" & CStr(pg) & vbCrLf
   Strdata += "k=" & CStr(k) & vbCrLf
   Strdata += "h=" & CStr(h) & vbCrLf
   Strdata += "oilViscosity=" & CStr(uoil) & vbCrLf
   Strdata += "gasViscosity=" & CStr(ugas) & vbCrLf
   Strdata += "re=" & CStr(re) & vbCrLf
   Strdata += "rw=" & CStr(rw) & vbCrLf
   Strdata += "pbar=" & CStr(pbar) & vbCrLf
   Strdata += "pwf=" & CStr(pwf) & vbCrLf
   Strdata += ow.Constrarray(n, z, "z")
   Strdata += ow.Constrarray(n, x, "x")
   Strdata += ow.Constrarray(n, y, "y")
   Strdata += ow.Constrarray(n, m, "m")
   Strdata += "Pres=" & CStr(Pres_old) & vbCrLf
   Strdata += "area=" & CStr(Area) & vbCrLf
   Strdata += "---------------------- output----------------------- " & vbCrLf
   Strdata += "Mo=" & CStr(Mo) & vbCrLf
   Strdata += "Mg=" & CStr(Mg) & vbCrLf
   Strdata += "So=" & CStr(So) & vbCrLf
   Strdata += "Sg=" & CStr(Sg) & vbCrLf
   Strdata += "Kro=" & CStr(Kro) & vbCrLf
   Strdata += "Krg=" & CStr(Krg) & vbCrLf
   Strdata += "qo=" & CStr(qo) & vbCrLf
   Strdata += "qg=" & CStr(qg) & vbCrLf
   Strdata += "Npo=" & CStr(Npo) & vbCrLf
   Strdata += "Npg=" & CStr(Npg) & vbCrLf
   Strdata += "Np=" & CStr(Np) & vbCrLf
```

```
    Strdata += ow.Constrarray(n, Zp, "zp")
    Strdata += "Nk=" & CStr(Nk) & vbCrLf
    Strdata += ow.Constrarray(n, Nki, "Initial Mole(Nki)")
    Strdata += ow.Constrarray(n, prm, "produced mole")
    Strdata += "total product mass(sumNpi)=" & CStr(SumNpi) & vbCrLf
    Strdata += ow.Constrarray(n, Nki_1, "new reservoir mole")
    Strdata += "total reservoir mass(sumNpi+1)=" & CStr(sumnki_1) & vbCrLf
    Strdata += ow.Constrarray(n, Zi, "new reservoir composition")
    Strdata += ow.Constrarray(n, Mres_new, "New reservoir Molecular weight")
    Strdata += "new reservoir density(Pres)=" & CStr(Pres) & vbCrLf
    Strdata += vbCrLf & "------------------------- End Reservoir Model ---------------------" & vbCrLf
    Dim tempdata As String      tempdata = ow.GetFileContents(pathfile.NewOutputFile)
    Strdata = tempdata & vbCrLf & Strdata
    ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
  End Function
End Class
```

```
Separator model
Public Class SeparatorModel
  Public L1 As Double   Public X1() As Double   Public Y1() As Double   Public L2 As Double   Public X2() As Double   Public Y2() As
Double
  Public L3 As Double   Public X3() As Double   Public Y3() As Double   Public Lst As Double   Public Xst() As Double   Public Yst() As
Double
  Public Patm As Double   Public OilRate1 As Double   Public GasRate1 As Double   Public OilRate2 As Double   Public GasRate2 As Double
  Public OilRate3 As Double   Public GasRate3 As Double   Public OilRateAtm As Double   Public GasRateAtm As Double   Public Ninj As
Double
  Public qo_out, qg_out As Double   Public OutQinj As Double   Public Zinj() As Double
  Public Function runMainModel(ByVal pathfile As PathFile, ByVal n As Integer, ByVal Psep1 As Double, _
ByVal Psep2 As Double, ByVal Psep3 As Double, ByVal SumNpi As Double, ByVal Zpi() As Double, _
ByVal temperature_R As Double, ByVal pc() As Double, ByVal tc() As Double, _
ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
ByVal inflcon As Double, ByVal M() As Double, ByVal Qinj_input As Double, ByVal Liqstand() As Double, _
 ByVal numberofwell As Double, ByVal Zsu() As Double, ByVal Timestep As Integer)
    Dim mg1, mg2, mg3, mgst As Double     Dim Pg1, Pg2, Pg3, Pgst As Double     Patm = 14.7     '---------1--------------     Dim Z_input()
As Double     If Timestep = 1 Then     Z_input = Zpi     Else     Z_input = Zsu     End If
    Dim Objf As New CompositionModel
    Objf.Run(pathfile, 0.5, n, Z_input, Psep1, temperature_R, pc, tc, omega, R, outflcon, inflcon, M, Liqstand)
    L1 = Objf.L     X1 = Objf.x     Y1 = Objf.y     OilRate1 = SumNpi * L1     GasRate1 = SumNpi * (1 - L1)     Pg1 = Objf.GasDensi
    For ist As Integer = 0 To n - 1     mg1 = mg1 + Y1(ist) * M(ist)     Next     '---------2--------------     Objf = New CompositionModel
    Objf.Run(pathfile, 0.5, n, X1, Psep2, temperature_R, pc, tc, omega, R, outflcon, inflcon, M, Liqstand)
    L2 = Objf.L     X2 = Objf.x     Y2 = Objf.y
    OilRate2 = SumNpi * L1 * L2     GasRate2 = SumNpi * L1 * (1 - L2)     Pg2 = Objf.GasDensity
    For ist As Integer = 0 To n - 1     mg2 = mg2 + Y2(ist) * M(ist)     Next     '---------3--------------
    Objf = New CompositionModel
    Objf.Run(pathfile, 0.5, n, X2, Psep3, temperature_R, pc, tc, omega, R, outflcon, inflcon, M, Liqstand)     L3 = Objf.L     X3 = Objf.x
Y3 = Objf.y     OilRate3 = SumNpi * L1 * L2 * L3     GasRate3 = SumNpi * L1 * L2 * (1 - L3)     Pg3 = Objf.GasDensity
    For ist As Integer = 0 To n - 1     mg3 = mg3 + Y3(ist) * M(ist)     Next     Dim Mlst As Double     Dim Plst As Double
    '---------Lst--------------     Objf = New CompositionModel
    Objf.Run(pathfile, 0.5, n, X3, Patm, temperature_R, pc, tc, omega, R, outflcon, inflcon, M, Liqstand)     Lst = Objf.L     Xst = Objf.x
Yst = Objf.y     Pgst = Objf.GasDensity     OilRateAtm = SumNpi * L1 * L2 * L3 * Lst
    GasRateAtm = SumNpi * (1 - L1) + SumNpi * L1 * (1 - L2) + SumNpi * L1 * L2 * (1 - L3)
    For ist As Integer = 0 To n - 1     Mlst = Mlst + Xst(ist) * M(ist)     Next     Plst = Objf.oilDensity     For ist As Integer = 0 To n - 1
      mgst = mgst + Yst(ist) * M(ist)     Next     Dim Ngas() As Double     Dim Yfinal() As Double     Dim SumNgas As Double = 0
    For i As Integer = 0 To n - 1
      Dim T1, T2, T3, T4, T5, T6 As Double
      T1 = SumNpi * (1 - L1)     T2 = Y1(i)     T3 = SumNpi * L1 * (1 - L2)     T4 = Y2(i)     T5 = SumNpi * L1 * L2 * (1 - L3)
      T6 = Y3(i)     ReDim Preserve Ngas(i)     Ngas(i) = T1 * T2 + T3 * T4 + T5 * T6     SumNgas = SumNgas + Ngas(i)
    Next
    For i As Integer = 0 To n - 1     ReDim Preserve Yfinal(i)     Yfinal(i) = Ngas(i) / SumNgas     Next
    Dim Noil() As Double     Dim SumNoil As Double = 0     For i As Integer = 0 To n - 1     Dim Tx As Double
      Tx = SumNpi * L1 * L2 * L3 * Lst     ReDim Preserve Noil(i)     Noil(i) = Tx * Xst(i)
    Next
    Dim Nsum() As Double     Dim SumNsum As Double = 0     For i As Integer = 0 To n - 1     ReDim Preserve Nsum(i)
      Nsum(i) = Ngas(i) * Noil(i)     SumNsum = SumNsum + Nsum(i)
    Next     For i As Integer = 0 To n - 1     ReDim Preserve Zinj(i)     Zinj(i) = Yfinal(i)     Next
    Dim qoatm As Double     qoatm = ConvertNpToqg(OilRateAtm, Mlst, Plst) / 5.615     Dim qgatm As Double
    Dim q1, q2, q3, qst As Double     q1 = ConvertNpToqg(GasRate1, mg1, Pg1)
    q2 = ConvertNpToqg(GasRate2, mg2, Pg2)     q3 = ConvertNpToqg(GasRate3, mg3, Pg3)     qgatm = q1 + q2 + q3
    If qgatm < Qinj_input Then     Ninj = GasRateAtm     OutQinj = qgatm     Else     Ninj = ConvertqpToNg(Qinj_input, mg2, Pg2)
      OutQinj = Qinj_input     End If     qo_out = qoatm     qg_out = qgatm
    WriteOutput(pathfile, n, Psep1, Psep2, Psep3, SumNpi, Zpi, temperature_R, pc, tc, omega, R, outflcon, inflcon, M, Qinj_input, Liqstand, Ngas,
Yfinal, Noil, Zinj, Zsu, OutQinj, numberofwell * qgatm, numberofwell * qoatm)
  End Function
  Private Function ConvertNpToqg(ByVal npg As Double, ByVal Mg As Double, ByVal pg As Double) As Double
    Dim t1, t2 As Double     t1 = npg * Mg     t2 = pg     Return t1 / t2   End Function
  Private Function ConvertqpToNg(ByVal Qg As Double, ByVal Mg As Double, ByVal pg As Double) As Double
    Dim t1, t2 As Double     t1 = Qg * pg     t2 = Mg     Return t1 / t2
  End Function
  Private Function WriteOutput(ByVal pathfile As PathFile, ByVal n As Integer, ByVal Psep1 As Double, _
ByVal Psep2 As Double, ByVal Psep3 As Double, ByVal SumNpi As Double, ByVal Zpi() As Double, _
ByVal temperature_R As Double, ByVal pc() As Double, ByVal tc() As Double, _
ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
ByVal inflcon As Double, ByVal M() As Double, ByVal Ninj_input As Double, ByVal Liqstand() As Double, _
ByVal Ngas() As Double, ByVal Yfinal() As Double, ByVal Noil() As Double, ByVal Zinj() As Double, _
```

```
ByVal Zsu() As Double, ByVal out_Qinj As Double, ByVal Qgatm As Double, ByVal Qoatm As Double)
    "------------------write file------------------
    Dim Strdata As String    Dim ow As New WriteOutput    Strdata = vbCrLf & "-------------------------Begin Separator Model --------------------
---" & vbCrLf    Strdata += "-------------------------Input-------------------------" & vbCrLf
    Strdata += "Psep1=" & CStr(Psep1) & vbCrLf    Strdata += "Psep2=" & CStr(Psep2) & vbCrLf
    Strdata += "Psep3=" & CStr(Psep3) & vbCrLf    Strdata += "Patm=" & CStr(Patm) & vbCrLf
    Strdata += "Qinj_input=" & CStr(Ninj_input) & vbCrLf    Strdata += "Np=" & CStr(SumNpi) & vbCrLf
    Strdata += ow.Constrarray(n, Zpi, "Zpi")    Strdata += ow.Constrarray(n, Zsu, "Zsu")
    Strdata += "-------------------------output-------------------------" & vbCrLf    Strdata += "L1=" & CStr(L1) & vbCrLf
    Strdata += ow.Constrarray(n, X1, "X1")    Strdata += ow.Constrarray(n, Y1, "Y1")
    Strdata += "OilRate1=" & CStr(OilRate1) & vbCrLf    Strdata += "GasRate1=" & CStr(GasRate1) & vbCrLf
    Strdata += "L2=" & CStr(L2) & vbCrLf    Strdata += ow.Constrarray(n, X2, "X2")
    Strdata += ow.Constrarray(n, Y2, "Y2")    Strdata += "OilRate2=" & CStr(OilRate2) & vbCrLf
    Strdata += "GasRate2=" & CStr(GasRate2) & vbCrLf    Strdata += "L3=" & CStr(L3) & vbCrLf
    Strdata += ow.Constrarray(n, X3, "X3")    Strdata += ow.Constrarray(n, Y3, "Y3")
    Strdata += "OilRate3=" & CStr(OilRate3) & vbCrLf    Strdata += "GasRate3=" & CStr(GasRate3) & vbCrLf
    Strdata += "Lst=" & CStr(Lst) & vbCrLf    Strdata += ow.Constrarray(n, Xst, "Xst")
    Strdata += ow.Constrarray(n, Yst, "Yst")    Strdata += "OilRateAtm=" & CStr(OilRateAtm) & vbCrLf
    Strdata += "GasRateAtm=" & CStr(GasRateAtm) & vbCrLf    Strdata += ow.Constrarray(n, Ngas, "Ngas")
    Strdata += ow.Constrarray(n, Yfinal, "Yfinal"    Strdata += ow.Constrarray(n, Noil, "Noil")
    Strdata += ow.Constrarray(n, Zinj, "Zinj")    Strdata += "out_Qinj=" & CStr(out_Qinj) & vbCrLf
    Strdata += "qgatm=" & CStr(Qgatm) & vbCrLf    Strdata += "qoatm=" & CStr(Qoatm) & vbCrLf
    Strdata += "-------------------------End Separator Model --------------------" & vbCrLf
    Dim tempdata As String
    tempdata = ow.GetFileContents(pathfile.NewOutputFile)
    Strdata = tempdata & vbCrLf & Strdata
    ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
  End Function
End Class
------------------------------------------------------------------------------------------
write output vb
Imports System.Text
Imports System.IO
Public Class WriteOutput
  Public Function GetFileContents(ByVal FullPath As String, _
     Optional ByRef ErrInfo As String = "") As String
    Dim strContents As String
    Dim objReader As StreamReader
    Try
      objReader = New StreamReader(FullPath)
      strContents = objReader.ReadToEnd()
      objReader.Close()
      Return strContents
    Catch Ex As Exception
      ErrInfo = Ex.Message
    End Try
  End Function
  Public Function SaveTextToFile(ByVal strData As String, _
   ByVal FullPath As String, _
    Optional ByVal ErrInfo As String = "") As Boolean
    Dim Contents As String
    Dim bAns As Boolean = False
    Dim objReader As StreamWriter
    Try
      objReader = New StreamWriter(FullPath)    objReader.Write(strData)    objReader.Close()    bAns = True
    Catch Ex As Exception    ErrInfo = Ex.Message
    End Try
    Return bAns
  End Function
  Public Function Constrarray(ByVal n As Integer, ByVal input() As Double, ByVal name As String) As String
    Dim i As Int16    Dim outstr As String = ""    If Not IsNothing(input) Then
      For i = 0 To n - 1
        outstr = outstr + name + CStr(i + 1) + "=" + CStr(input(i)) & vbCrLf    Next
    End If
    Return outstr
  End Function
End Class
```

```
Viscman
Imports System.Math
Public Class viscman
  Public visc As Double
  "lt =0 > gas
  "lt=1>oil
  Public Function run(ByVal pathfile As PathFile, ByVal nc As Integer, ByVal temperature_R As Double, _
  ByVal comp() As Double, ByVal m() As Double, _
  ByVal lt As Integer, ByVal density As Double) As Double
    Dim tden, t As Double
    Dim ma As Double
    Dim i As Integer
    Dim grav, a, b, c As Double
    Dim yo As Double
    tden = density
```

```
      t = temperature_R
      For i = 0 To nc - 1
        ma = ma + comp(i) * m(i)
      Next
      If lt = 1 Then "oil
        "      T = F
        "density= Liquid Density
        yo = density / 62.37
        grav = (141.5 / yo) - 131.5
        If grav > 58 Then grav = (5 * 58 + grav) / 6
        visc = 10 ^ (10 ^ (1.8653 - 0.025086 * grav - 0.5644 * Log10(t - 460))) - 1
      End If
      If lt = 0 Then "gas
        "T=R
        "density=
        density = density * 0.01602
        a = (9.379 + 0.01607 * ma) * (t ^ 1.5) / (209.2 + 19.26 * ma + t)
        'a = (t * t * (9.379 + 0.01607 * ma)) / (209.2 + 19.26 * ma + t)
        'a = a / Sqrt(t)
        b = 3.448 + (986.4 / t) + (ma * 0.01009)
        c = 2.447 - 0.2224 * b
        visc = a * (10 ^ -4) * Exp(b * (density ^ c))
      End If
      WriteOutput(lt, visc, pathfile, t, comp, m, nc, tden)
      Return visc
    End Function
    Public Function WriteOutput(ByVal lt As Integer, ByVal visc As Double, ByVal pathfile As PathFile, ByVal t As Double, _
    ByVal comp() As Double, ByVal m() As Double, ByVal n As Integer, ByVal density As Double)
      Dim Strdata As String      Dim ow As New WriteOutput
      Strdata = vbCrLf & "------------------------ Viscosity --------------------" & vbCrLf
      Strdata += "-Input-" & vbCrLf
      Strdata += "T=" & CStr(t) & vbCrLf
      Strdata += ow.Constrarray(n, comp, "comp")
      Strdata += ow.Constrarray(n, m, "m")
      Strdata += "density=" & CStr(density) & vbCrLf
      Strdata += "-output-" & vbCrLf
      If lt = 0 Then
        Strdata += "Gas viscosity=" & CStr(visc) & vbCrLf      Else
        Strdata += "Oil viscosity=" & CStr(visc) & vbCrLf      End If
      Dim tempdata As String
      tempdata = ow.GetFileContents(pathfile.NewOutputFile)
      Strdata = tempdata & vbCrLf & Strdata
      ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
    End Function
End Class
```

```
Tubing model
Imports System.Math
Public Class TubingModel
    Public Vsl As Double   Public Vsg As Double   Public Vm As Double   Public Vmsl As Double   Public Vmsg As Double
    Public Vb As Double   Public Vt As Double   Public EL As Double   Public DelPH As Double   Public Re As Double
    Public DelPf As Double   Public fmoody As Double   Private Pl, Pg, ql, qg, liquidvis As Double
    Public Function runmodel(ByVal pathfile As PathFile, ByVal n As Integer, _
      ByVal A As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal e As Double, _
ByVal Np As Double, ByVal zp() As Double, ByVal Np_inj As Double, ByVal zp_inj() As Double, ByVal P1 As Double, ByVal Tatm As Double,
_
    ByVal pc() As Double, ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
    ByVal inflcon As Double, ByVal m() As Double, ByVal Liqstand() As Double, ByVal Tubing_length As Double, ByVal InjectionPoint As
Double)
      Dim Pwf As Double = 0.0
      Dim Tubing_length_1 As Double
      Tubing_length_1 = InjectionPoint
      Dim nL As Integer
      Dim Tubing_length_2 As Double
      Pwf = P1
      If Tubing_length_1 >= 100 Then
        nL = Fix(Tubing_length_1 / 100)
        For i As Integer = 1 To nL
          Pwf = CalbyLength(pathfile, n, Np_inj, A, D, g, delL, e, zp_inj, Pwf, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand, 100)
        Next
        Tubing_length_2 = Tubing_length_1 - (100 * nL)
        If Tubing_length_2 > 0 Then
          Pwf = CalbyLength(pathfile, n, Np_inj, A, D, g, delL, e, zp_inj, Pwf, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand,
Tubing_length_2)
        End If
      Else
        Pwf = CalbyLength(pathfile, n, Np_inj, A, D, g, delL, e, zp_inj, Pwf, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand,
Tubing_length_1)
      End If
      Dim Tubing_length_3 As Double = Tubing_length - InjectionPoint
      Dim Tubing_length_4 As Double
      If Tubing_length_3 >= 100 Then
        nL = Fix(Tubing_length_3 / 100)
```

```
        For i As Integer = 1 To nL
            Pwf = CalbyLength(pathfile, n, Np, A, D, g, delL, e, zp, Pwf, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand, 100)
        Next
        Tubing_length_4 = Tubing_length_3 - (100 * nL)
        If Tubing_length_4 > 0 Then
            Pwf = CalbyLength(pathfile, n, Np, A, D, g, delL, e, zp, Pwf, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand, Tubing_length_4)
        End If
    Else
        Pwf = CalbyLength(pathfile, n, Np, A, D, g, delL, e, zp, Pwf, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand, Tubing_length_3)
    End If
    WriteOutput(pathfile, n, A, D, g, delL, e, Np, zp, Np_inj, zp_inj, P1, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand, Tubing_length,
InjectionPoint, Pwf)
    Return Pwf
  End Function
  Private Function CalbyLength(ByVal pathfile As PathFile, ByVal n As Integer, ByVal Np As Double, _
    ByVal A As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal e As Double, _
ByVal zp() As Double, ByVal P1 As Double, ByVal Tatm As Double, _
  ByVal pc() As Double, ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
  ByVal inflcon As Double, ByVal m() As Double, ByVal Liqstand() As Double, ByVal Tubing_length As Double) As Double
    '---composition model---'
    Dim oilDensity, GasDensity, gasViscosity As Double
    Dim X(), Y() As Double
    Dim L, V As Double
    Dim obj1 As New CompositionModel
    obj1.Run(pathfile, 0.5, n, zp, P1, Tatm, pc, tc, omega, R, outflcon, inflcon, m, Liqstand)
    GasDensity = obj1.GasDensity
    X = obj1.x      Y = obj1.y      L = obj1.L      V = obj1.V
    oilDensity = obj1.oilDensity
    Dim viscman As New viscman
    liquidvis = viscman.run(pathfile, n, Tatm, X, m, 1, oilDensity)
    gasViscosity = viscman.run(pathfile, n, Tatm, Y, m, 0, GasDensity)
    '=====_======'
    Dim mo, mg As Double      Dim i As Integer      For i = 0 To n - 1      mo = mo + X(i) * m(i)      Next
    For i = 0 To n - 1      mg = mg + Y(i) * m(i)      Nex
    Pl = oilDensity
    Pg = GasDensity
    Dim Npo, Npg As Double
    Npo = Np * L
    Npg = Np * V
    ql = ConvertNpToql(Npo, mo, Pl)
    qg = ConvertNpToqg(Npg, mg, Pg)
    Vsl = ql / A
    Vsg = qg / A
    Vm = Vsl + Vsg
    Vmsl = Vsl * (((Pl * 72) / (63.37 * 50)) ^ (0.25))
    Vmsg = Vsg * ((Pg) / 0.078) ^ (0.33) * ((Pl * 72) / (62.37 * 50)) ^ 0.25
    Dim b1 As Double
    Dim b2 As Double
    Dim b3 As Double
    b1 = ((100 * Vmsl) ^ 0.17211) / 1.96
    b2 = (Vmsl / 0.263) + 8.6
    b3 = 70 * ((100 * Vmsl) ^ -0.152)
    If Vmsl > 4 Then
        If Vmsg < b1 Then
            Call Bubble(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
        ElseIf Vmsg >= b1 Then
            If Vmsg < 26.5 Then
                Call Slug(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
            ElseIf Vmsg >= 26.5 Then
                Call AnnularMist(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
            End If
        End If
    ElseIf Vmsl <= 4 Then
        If Vmsg < b1 Then
            Call Bubble(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
        ElseIf Vmsg >= b1 Then
            If Vmsg < b2 Then
                Call Slug(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
            ElseIf Vmsg >= b2 Then
                If Vmsg < b3 Then
                    Call Froth(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
                ElseIf Vmsg >= b3 Then
                    Call AnnularMist(pathfile, Pl, Pg, D, g, delL, e, liquidvis)
                End If
            End If
        End If
    End If
    Return P1 + (DelPf * Tubing_length) + (DelPH * Tubing_length)
  End Function
  Private Function WriteOutput(ByVal pathfile As PathFile, ByVal n As Integer, _
    ByVal A As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal e As Double, _
ByVal Np As Double, ByVal zp() As Double, ByVal Np_inj As Double, ByVal zp_inj() As Double, ByVal P1 As Double, ByVal Tatm As Double,
_ ByVal pc() As Double, ByVal tc() As Double, ByVal omega() As Double, ByVal R As Double, ByVal outflcon As Double, _
```

```
  ByVal inflcon As Double, ByVal m() As Double, ByVal Liqstand() As Double, ByVal Tubing_length As Double, ByVal InjectionPoint As
Double, ByVal Pwf As Double)
      "--------writefile---------
      "-----------------write file------------------
      Dim Strdata As String
      Dim ow As New WriteOutput
      Strdata = vbCrLf & "---------------------------Begin Tubing Model --------------------" & vbCrLf
      Strdata += "-----------------------Input-------------------------" & vbCrLf
      Strdata += "ql=" & CStr(ql) & vbCrLf
      Strdata += "qg=" & CStr(qg) & vbCrLf
      Strdata += "A=" & CStr(A) & vbCrLf
      Strdata += "OilDensity(pl)=" & CStr(Pl) & vbCrLf
      Strdata += "GasDensity(pg)" & CStr(Pg) & vbCrLf
      Strdata += "D=" & CStr(D) & vbCrLf
      Strdata += "g=" & CStr(g) & vbCrLf
      Strdata += "delL=" & CStr(delL) & vbCrLf
      Strdata += "E=" & CStr(e) & vbCrLf
      Strdata += "liquidvis=" & CStr(liquidvis) & vbCrLf
      Strdata += ow.Constrarray(n, zp, "zp")
      Strdata += ow.Constrarray(n, zp_inj, "Zsu")
      Strdata += "Np=" & CStr(Np) & vbCrLf
      Strdata += "Np_inj=" & CStr(Np_inj) & vbCrLf
      Strdata += "InjectionPoint=" & CStr(InjectionPoint) & vbCrLf
      Strdata += "Tubing_length=" & CStr(Tubing_length) & vbCrLf
      Strdata += "-----------------------output-----------------------" & vbCrLf
      Strdata += "Vsl=" & CStr(Vsl) & vbCrLf
      Strdata += "Vsg=" & CStr(Vsg) & vbCrLf
      Strdata += "Vm=" & CStr(Vm) & vbCrLf
      Strdata += "Vmsl=" & CStr(Vmsl) & vbCrLf
      Strdata += "Vmsg=" & CStr(Vmsg) & vbCrLf
      Strdata += "Vb=" & CStr(Vb) & vbCrLf
      Strdata += "Vt=" & CStr(Vt) & vbCrLf
      Strdata += "EL=" & CStr(EL) & vbCrLf
      Strdata += "DelPH=" & CStr(DelPH) & vbCrLf
      Strdata += "Re=" & CStr(Re) & vbCrLf
      Strdata += "Ff=" & CStr(fmoody) & vbCrLf
      Strdata += "DelPf=" & CStr(DelPf) & vbCrLf
      Strdata += "Pwf=" & CStr(Pwf) & vbCrLf
      Strdata += "-------------------------End Tubing Model  --------------------" & vbCrLf
      Dim tempdata As String
      tempdata = ow.GetFileContents(pathfile.NewOutputFile)
      Strdata = tempdata & vbCrLf & Strdata
      ow.SaveTextToFile(Strdata, pathfile.NewOutputFile)
   End Function
   Private Function ConvertNpToql(ByVal npo As Double, ByVal Mo As Double, ByVal po As Double) As Double
      Dim t1, t2 As Double       t1 = npo * Mo       t2 = 86400 * po       Return t1 / t2
   End Function
   Private Function ConvertNpToqg(ByVal npg As Double, ByVal Mg As Double, ByVal pg As Double) As Double
      Dim t1, t2 As Double       t1 = npg * Mg       t2 = 86400 * pg       Return t1 / t2    End Function
   Private Function OutputRegimes(ByVal data As Regimes)
      Vb = data.Vb
 Vt = data.Vt
EL = data.EL
Re = data.Re       DelPH = data.DelPH
DelPf = data.DelPf       fmoody = data.fmoody
   End Functio
#Region "Regimes"
   Private Function Bubble(ByVal pathfile As PathFile, ByVal Pl As Double, _
   ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
      Dim ObjR As New Regimes
ObjR.Bubble(PathFile, Vm, Vsg, Pl, Pg, D, g, delL, E, liquidvis)
OutputRegimes(ObjR)
 End Function
   Private Function Slug(ByVal pathfile As PathFile, ByVal Pl As Double, _
   ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
      Dim ObjR As New Regimes
ObjR.Slug(pathfile, Vsg, Vm, Pl, Pg, D, g, delL, E, liquidvis)
OutputRegimes(ObjR)
End Function
   Private Function AnnularMist(ByVal pathfile As PathFile, ByVal Pl As Double, _
     ByVal Pg As Double, ByVal D As Double, ByVal g As Double, ByVal delL As Double, ByVal E As Double, ByVal liquidvis As Double)
      'Dim fmoody As Double
 Dim ObjR As New Regimes
ObjR.AnnularMist(pathfile, Vsl, Vm, Vsg, Pl, Pg, D, g, delL, E, liquidvis)
      OutputRegimes(ObjR)
End Function
   Private Function Froth(ByVal pathfile As PathFile, ByVal Pl As Double, _
   ByVal Pg As Double, ByVal D
As Double, ByVal g As Double, ByVal delL
As Double, ByVal E As Double, ByVal liquidvis As Double)
      Dim ObjR As New Regimes
ObjR.Froth(pathfile, Vsg, Vm, Vmsl, Vmsg, Pl, Pg, D, g, delL, E, liquidvis)
      OutputRegimes(ObjR)
End Function#End RegionEnd Class
```

VITAE

Nipon Tantayopin was born on October 03, 1981 in Bangkok, Thailand. He received his B. Eng. in Mechanical Engineering from Faculty of Engineering, Chulalongkorn University in 2002. In 2003, he continued his study in Master of Petroleum Engineering program at Department of Mining and Petroleum Engineering, Faculty of Engineering, Chulalongkorn University. After he finished the course work in 2005, he has worked for Baker Atlas, Baker Hughes (Thailand) Co., Ltd., in position of field engineer.