SUPRI-B

OPTIMAL SCHEDULING OF DEVELOPMENT IN AN OIL FIELD

Antonio Carlos Bittencourt de Andrade Filho

August 1994

Stanford University
Petroleum Engineering Department

Acknowledgments

I would like to express my gratitude and sincere appreciation to my research advisor Dr. Roland N. Horne for his support, guidance and encouragement throughout the course of this study.

I am grateful to PETROBRÁS for making this research at Stanford possible for me.

I appreciate the members of the SUPRI-B Industrial Affiliates Program for suggesting such a fascinating research subject and also for providing the computer resources required to complete this work.

I am also grateful to my family and colleagues of PETROBRÁS for their constant contact and encouragement during the most difficult moments.

Abstract

Field development projects involve several variables and demand different teams of professionals in the oil industry. Different software and techniques are applied to minimize investments. Sometimes it is difficult to put all these parts together in a synchronized way.

This work describes an optimizing approach that uses the current techniques of the different majors in a same procedure. The objective function consists of a cost function whose data generator is the ECLIPSE® reservoir simulator. The optimizing procedure, using the polytope search method, generates the input data file for the reservoir simulator for each polytope movement. Then, the cost function, considering the simulation results and the oil industry cost parameters, returns the cash flow analysis for the actual case. In this way, the polytope moves on an imaginary net whose nodes are based on simulator runs.

This approach requires an interface between the optimizing module and the reservoir simulator. The optimizing module consists of a main routine that generates the input for the reservoir simulator and dispatches the simulator run. The cost function reads the simulation results from a file and returns the cash flow analysis.

Because we need to define the problem and interfacing difficulties properly the following base case was considered first: when an off-shore oil field production starts declining a second field comes into production to achieve the maximum capacity of the installed facilities. For the same reasons, this work considers two parameters: the time when the second field starts producing and how long the development time lasts.

The obtained results are different from the intuitive guess showing that, for the studied case, the obvious solution can cause the unnecessary disbursement of large amounts of money.

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

An offshore oil field development involves several variables that affect the operational schedule involved in its management. Some of them are: drilling schedule, rig allocation, facilities acquisition, number and location of wells, rate of production decline, water and/or gas injection systems and, mainly, a production profile that uses the maximum capacity of the installed facilities for most of the lifetime of the field.

All these variables are usually used as input to a reservoir simulator that generates a forecast of the production profile constrained to several assumptions. In this way, the production engineer has to consider several hypotheses to achieve his the best guess for the field development problem under study. Also, each hypothesis can generate other ones, and so, generating a hypothesis tree where the main common connection is the central problem. The solution of this problem requires the effort of several people as well as computer work and physical time.

Often, in the industry, there is insufficient time or personnel to perform such a task to provide all the requirements for the field development.

Looking at this problem as an optimization problem we can reduce the time spent and the human and machine efforts. In this case, the search for the solution navigates from one hypothesis to an other in an interconnected process. As a result, we can expect to find the best option among the several considered hypotheses.

An optimization procedure requires the characterization of the function to be optimized (minimized or maximized), known as the objective function, as well as the choice of a proper optimizing method. There are two traditional types of optimizing methods: derivative-based methods, that assure a faster convergence but require evaluations of the objective function derivatives, and direct methods, that require only function evaluations but have slower convergence. Newer methods such as Genetic Algorithm and Neural Network were not considered in this work.

The complexity of predicting hydrocarbon production profiles requires the use of reservoir simulators. So, the simulator must be part of the evaluation of the objective function. Simulation also becomes a limiting factor for the derivative-based methods because of the additional cost of evaluating the derivatives of the objective function using the simulator. Direct methods do not require derivatives and therefore have an advantage.

Handling multiple variables at the same time leads to a very complex problem. To first know the problem and identify its behavior we based our investigation in a simple problem with two variables. We focused our attention on characterizing the objective function, the process of interfacing the optimization algorithm with the reservoir simulator and the choice of the most adequate optimizing method. The test problem, although simple, represents a field example of practical significance, namely the scheduling of incremental developments to maximize utilization of production facilities.

2. THE BASIC PROBLEM

In order to be able to handle more complex problems we started with the following basic problem:

> *When the production of a existing reservoir declines, the maximum capacity of the installed facilities is no longer filled and we need to supply additional production from another zone/reservoir in order to achieve full capacity.*

2.1 Description

In this case we are first analyzing a simple model in order to observe the sensitivity of the response to parameters (variables) of the producing zone/reservoir that will complement the existing declining production. In the first instance, the two parameters under investigation are the time of starting the second development (t_{i2}) and the time required to develop the second zone/reservoir so that the combined output reaches Qmax, i.e., development time (dt) (see Figure 2-5). These two parameters are referred to here as the "start time" and "development time".

The development system consists of an existing production platform with 10 wells in full operation, all already paid. At a specific time t_{f1} the reservoir 1 production starts declining at a constant rate. The reservoir 2, at some other time t_{i2}, starts producing at such a rate that the maximum capacity of the platform is not exceeded.

The oil rate value and the number of wells required to reach this complementary rate are computed by the reservoir simulator. In order to have the wells producing at the proper time we need to have them already drilled and so we assumed two rigs permanently applied to this task. Considering that the rig takes about four months to drill one well we needed to plan a scheme to schedule the rig accordingly in order to drill each well.

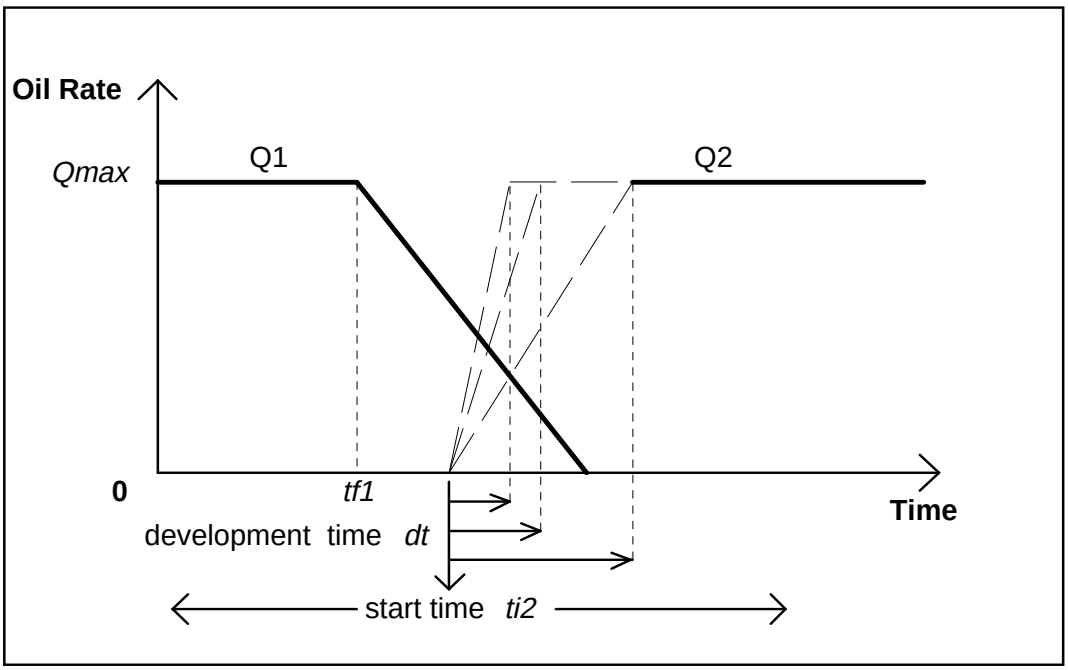

Figure 2-1

The permanent production platform for the second reservoir is assumed to be contracted two years before the first well is completely drilled. The payment is 30% down and 70% over 23 months. The facilities are also contracted one year before the production starts and paid in the same way.

2.2 Assumptions

The main assumption is to investigate only the two parameters t_{i2} and *dt*. In order to perform the analysis we had to assume a model able to represent the investments (disbursements) and income distributed through time. This model considers several operations according to reality but it is also simple enough in order to not consider other assumptions that could be relevant to the actual full-field analysis.

The objective in performing the simplified preliminary analysis is to understand the nature of the objective function behavior, building a surface, and to fine tune the optimization algorithm without the expense of full-field simulation.

3. THE OBJECTIVE FUNCTION

The optimization procedure requires that the function to be maximized/minimized be properly defined, considering the constraints and adequately representing the real problem.

3.1 Description

In order to compare a heterogeneous range of parameters, we had to choose a reference applicable to all kind of entities regardless of their nature. The cash flow (expressed in US$) of such operations was chosen to be this reference. Hence, we are now evaluating a problem concerning the *minimization of the costs* or *maximization of the profits*.

During the search, we considered that a period in which production falls to zero must not occur. It means that the parameters are constrained to ensure continuous production of the reservoirs. Also, the maximum capacity of the facilities is a limiting factor even if, for a specific pair (t_{i2},*dt*), new wells must be shut in immediately to respect this restriction. In this case, the cost for drilling these wells in the proper time is still considered for the cash flow analysis.

Hence, we have for each pair (t_{i2},*dt*) all the values (month by month) concerning rig rental, drilling costs, facility costs, drilling platform cost and production cost that represent together the *local outcome* and the oil production income that represents the *local income*. After locating these values in the proper time scale, we correct each of them to the present value (Net Present Value), then we add them up to obtain the cash flow value (see Figure 3-1).

So, each pair of parameters (t_{i2},*dt*) is associated with a cash flow value, or:

$$C = C(t_{i2},dt) \qquad (1)$$

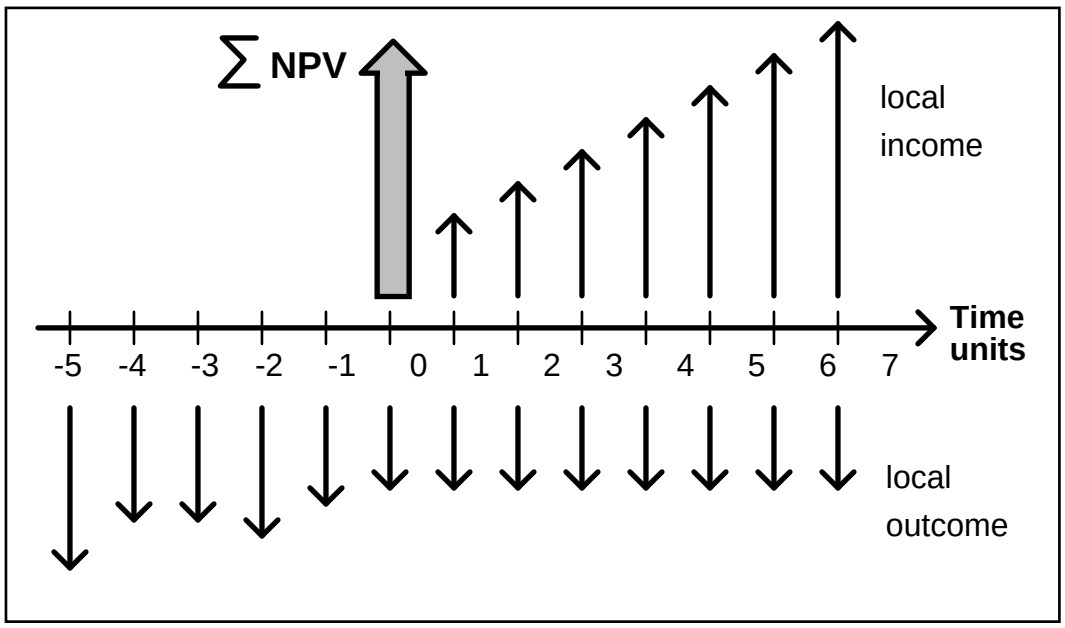

Figure 3-1

This is the objective function to be optimized. As we do not have an equation for this function due to the complexity of the problem, we had to view the surface generated by varying the parameters, so that we could define an optimization method.

The analysis is performed observing the cash flow due to the variation of t_{i2} and dt constrained to the assumptions already described.

From industry input we know most of the unitary cost values as well as the oil price (Mannarino, 1991) (see Table 3-1). From the simulation we obtain the number of wells and the oil production in a given time interval. The objective function computation then plans the rig allocation, applies associated costs and handles time requirements.

Rig allocation requires us to keep track of the already allocated rig and to perform an economic analysis to check what is cheaper: to drill one well just after the other or to pay the rig mobilization cost for each well to be drilled, if necessary. The resulting values are then placed in the proper time scale.

A schedule with complete operation requirements is obtained from this procedure, allowing us to evaluate the cash flow.

This procedure requires that the cash flow function interacts with the simulator, generating the proper input file, dispatching the simulator run and recovering its results when the run ends. In this way, the values obtained from the simulation run could be considered in evaluating costs and placing them in the correct time scale.

Description	Value	Units
Maximum Simulated Period	504.00	months
Field 1 Decline Starting Time	144.00	months
Field 1 Zero Production Time	348.00	months
Interest per Year	10	%
Maximum Production Capacity	15000.00	BPD
Maximum Oil Rate per Well	1500.00	BPD
Oil Barrel Price	15.00	US$/BBL
Platform Cost	100000000.00	US$ / 15 wells
Rig Cost	600000.00	US$ / month
Rig Mobilization Cost	1000000.00	US$
Rig Capacity	4	wells
Offshore Well Drilling Cost	1400000.00	US$ / well / month
Facilities Initial Cost	400.00	US$ / BPD
Overhead Factor	15	%
Insurance Cost	12500.00	per month
Factor f (transport)	0.13	[fractional]
Operational Cost per Well	5000.00	US$ / month
Handling Cost	1.00	US$ / BBL

Table 3-1

3.2 The Interface with the Simulator

The ECLIPSE® simulator, from ECL, was used because of its ability to handle a well drilling queue. So, the objective function did not need to handle the parameter NUMBER OF WELLS to achieve the complementary oil production. ECLIPSE® can handle it easily and in a very convenient way.

In order to speed up the interface task a restart file with the initialization step is required. Another file is also required, a user supplied file called ROOT, that refers to the restart file, sets output flags, defines wells and production constraints but does not contain the production time schedule. This file is read by the optimizing program. The production profile reproducing the actual parameter *dt*, *development time*, following the simulator syntax, is then appended to the ROOT file generating the TRIAL.DATA file which is used as input file for ECLIPSE®. The other parameter t_{i2}, *starting time*, is considered later when the resulting predicted production profile along with the number of wells is loaded into a table. This table is an array consisting of three columns and as many as necessary rows. Each row is a time information entry so, for each row, the first column is the time value in months, the second one is the oil rate value in BPD and the third one is the number of wells. The first entry in time is shifted by *starting time* relative to

the first array element. When the oil rate and number of wells are needed for a certain time then the table values corresponding to this time level are returned.

The optimizing program, after generating the ECLIPSE® input file, dispatches the simulator run. After the run ends the results contained in the Unified Formatted File (.FUNSMRY file), an ASCII file, are read into the optimizing program (see Figure 3-2). The format of this file is described in Section 4. Then, the objective function algorithm distributes the costs and income along the time and returns the calculated cash flow value for the pair(t_{i2},dt).

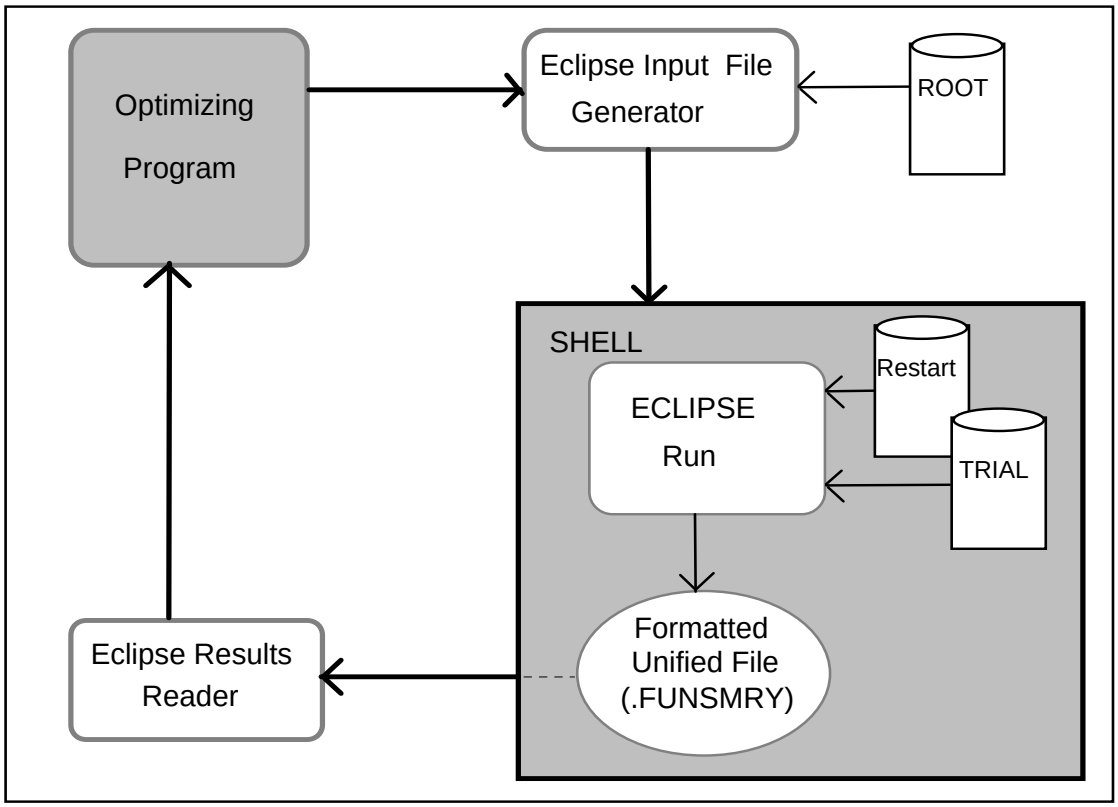

Figure 3-2

The objective function routine is then ready to perform the same task for new values of (t_{i2},dt) requested by the optimizing method.

4. THE OPTIMIZING METHOD

As we have already noted the actual problem is a multidimensional problem, although only two parameters were considered in this work. Several methods are available to handle this kind of problem.

There are two categories of searching methods: derivative-based methods and direct methods. Both of them have advantages for specific problems. The choice between these categories is based on the problem type and the limitation concerning the characterization of the objective function. We know that the simulation run must be part of this function and so, all the derivative-based methods are discarded because of the high cost of calculating derivatives using the simulator. Direct methods, where only function evaluations are required, were chosen.

The main direct methods available are the polytope search method, the genetic algorithm and the artificial neural network method.

Our preliminary approach to this problem was based on the polytope search method because of its robustness and adaptability.

The surface generated by the objective function routine (see Figures 4-1 and 4-2), by varying parameters t_{i2} and *dt*, appears to be reasonably smooth. The assumption of a constant rate decline of Field 1 production strongly affects this behavior. For a more complex case we expect a rough surface, with local maxima/minima causing bumps on the surface.

In order to generate the whole surface 60 ECLIPSE® runs were required, each with a different *development time* value. Each production profile curve was then displaced along the *starting time* axis in order to cover all cases. The curve resolution is six months.

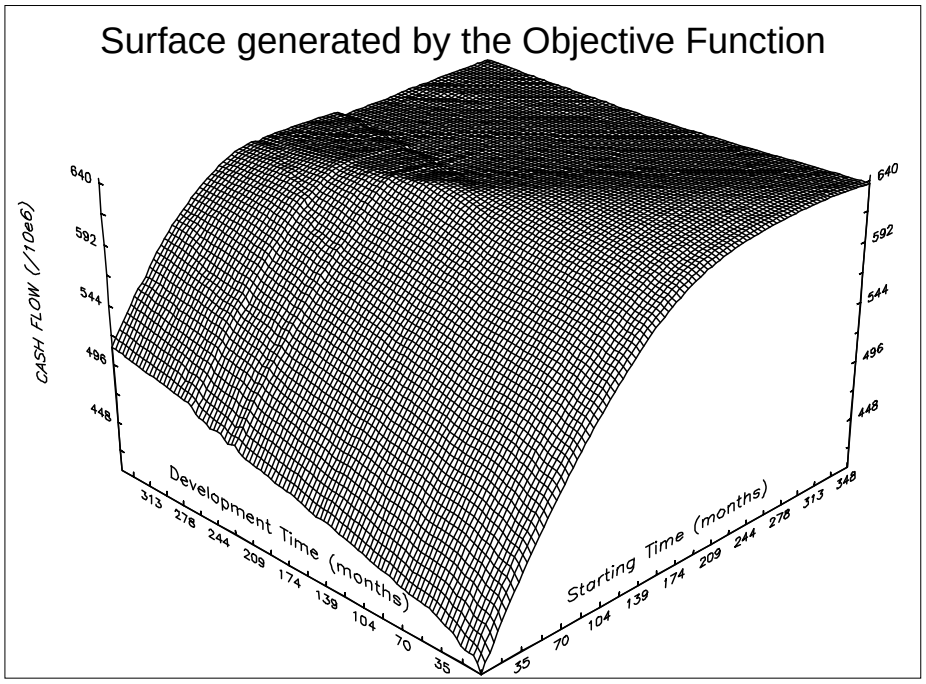

Figure 4-1

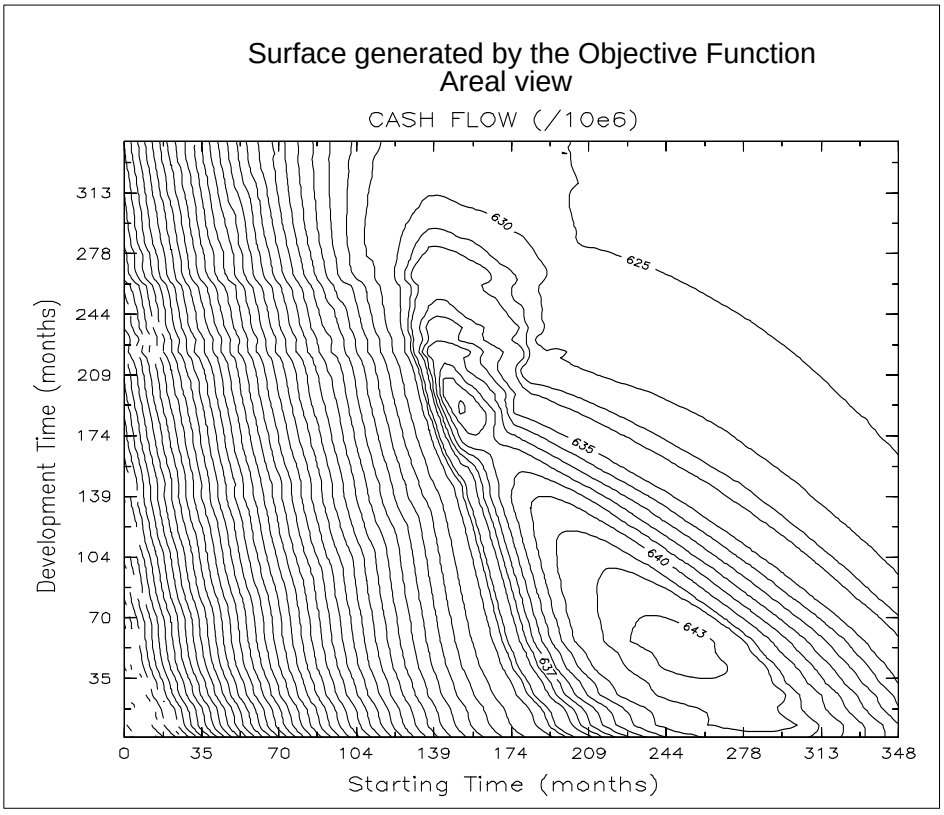

Figure 4-2

4.1 The Polytope Search Method

This method is due to Nelder and Mead. It is based on the movement of a geometrical entity towards the optimal point.

The polytope is a geometrical figure consisting, in n dimensions, of n+1 nodes or vertices and all their interconnecting segments, polygonal faces, etc. This figure is not necessarily regular but must not degenerate, i.e., must enclose a finite inner n-dimensional volume.

The polytope movement consists of a series of reflections of the worst point about the centroid of the remaining n nodes. In this way, the polytope flips about the centroid towards the optimum.

All n+1 points together with their function values are required to start the method. For each iteration, the function values are sorted in a descending order so that,

$$F_{n+1} > F_n > ,....., F_2 > F_1 \qquad (2)$$

Among the n+1 function values, the worst point x_{n+1} is taken as being the initial starting point $\mathbf{P}_o$, then the other n points can be expressed as:

$$\mathbf{P}_i = \mathbf{P}_o + \lambda_i \mathbf{e}_i \qquad (3)$$

where $\mathbf{e}_i$ are n unit vectors and λ_i are the problem characteristic length scale constants for each vector direction. In this way, the n other points define vector directions that span the n-dimensional vector space.

The worst point x_{n+1} is then replaced with a new point that is merely the reflection of x_{n+1} about the centroid c of the remaining n points (Gill, Murray, Wright, 1992). This movement can be represented mathematically as:

$$c = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_i \qquad (4)$$

$$x_{new} = c + \alpha(c - x_{n+1}) \qquad (5)$$

where α is called the reflection coefficient, which is always positive. This step is constructed to conserve the volume of the polytope (α =1), avoiding its degeneracy.

Three cases can occur when evaluating the new point (see Figure4-3):

1. $F_n > F_{new} > F_1$
   The new point has the function value between the best and the worst points.

   So, accept the new point and go to the next iteration.

2. $F_{new} > F_n$
   The new point is the new worst point.
   Try $\alpha < 1$ and guess a new point. If successful, accept the contraction factor, i.e., the new volume, and go to the next iteration. Otherwise, try a smaller α.

3. $F_{new} < F_1$
   The new point is the new best point.
   Try to expand the polytope using $\alpha > 1$. If successful, accept the expansion factor. Otherwise, use the original x_{new} for the next iteration

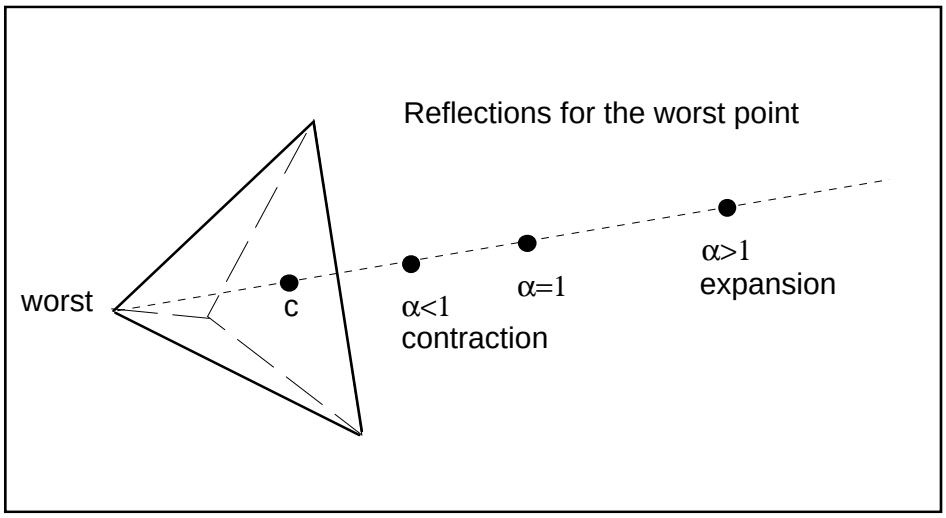

Figure 4-3

Termination criteria can be delicate in any multidimensional minimization routine (Press et al, 1992). We cannot set a tolerance for a single independent variable. All the variables must be considered. Here, we appeal again to the objective function, the only way we have to relate all variables. In our case, the polytope search stops when the following criterion is satisfied (Press et al, 1992):

$$\frac{|F_{best} - F_{worst}|}{\frac{|F_{best}| + |F_{worst}|}{2}} < \varepsilon \qquad (6)$$

where ε is a given tolerance.

5. THE PROGRAM ALGORITHM

The developed algorithm to solve this problem assembles the procedures described in the previous sections.

Figure 5-1 shows the implemented algorithm.

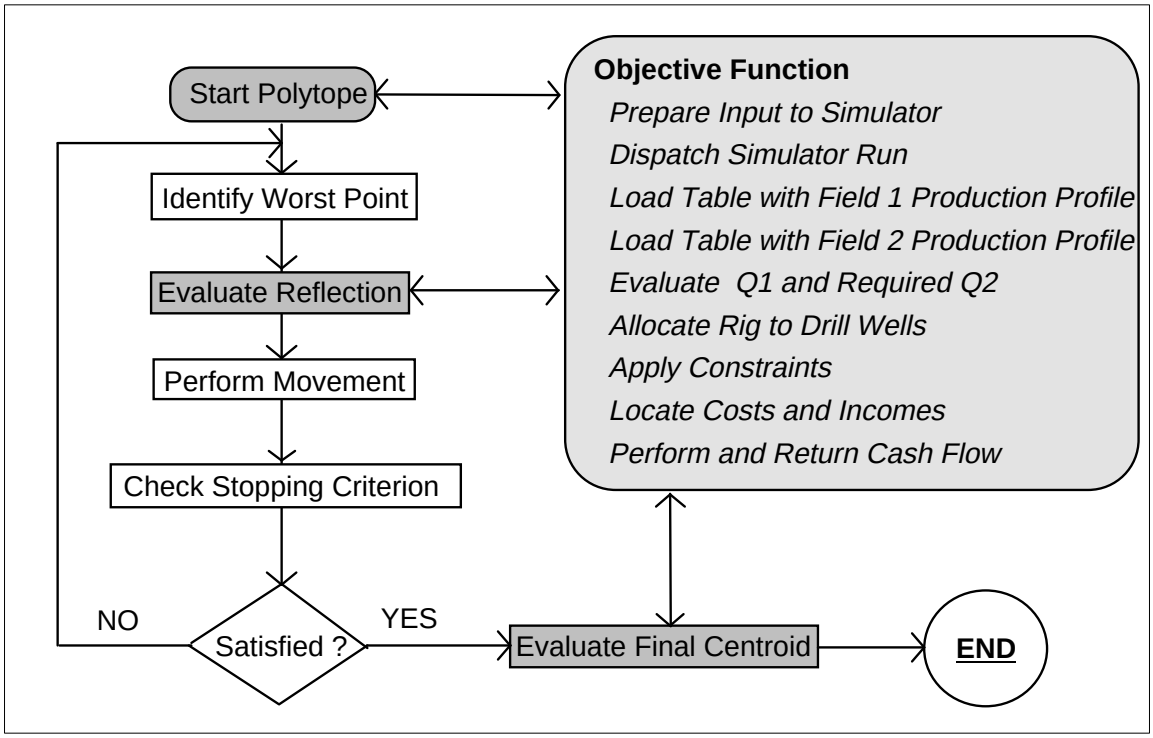

Figure 5-1

The computer code was written in ANSI C and uses the polytope method implemented in the Numerical Recipes Software package (UNIX, version 2.02) installed on a DEC machine model 3000 AXP/400, based on a RISC processor.

The starting polytope size is user supplied as well as ε, the tolerance value, to be used in the stopping criterion. Practical values for ε lies between 10^{-3} and 10^{-6}. Numerical Recipes Software library provides the routine *amoeba.c* that is a well tested routine for the polytope

method and parameters like contraction and expansion factors, number of trials for expansion and contraction and others are internally defined.

The program source code listing, libraries listing and routine definitions are listed in Appendix 1.

The results obtained from ECLIPSE® are read from an ASCII file, with all time step summaries. This file is known as a Unified Formatted File (ECLIPSE® USER'S MANUAL, 1993). According to File Handling Section (ECLIPSE® USER'S MANUAL, 1993) its format is specified as follows:.

> File internal format
>
> All ECLIPSE files consist of a series of data blocks each headed by a descriptor record. The block descriptor record comprises
>
> (a) An 8-character keyword which identifies the data in the block.
>
> (b) An integer (4 byte) containing the number of elements in the block.
>
> (c) A 4-character field specifying the type of the data in the block.
>
> Permitted values are
>
> 'INTE' for standard (4 byte) integers
> 'REAL' for single precision (4 byte) floating point reals
> 'LOGI' for standard (4 byte) logicals
> 'DOUB' for double precision (8 byte) floating point reals
> 'CHAR' for characters
>
> Note that if the data type is 'CHAR', then NEL is the total number of CHARACTER*8 elements.
>
> The contents of the block follows the descriptor (on a new record). This may be formatted or unformatted. If formatted, the form of output used may be read back by FORTRAN list input. If unformatted, each data block must be read back in the same block structure as it was written out.
>
> The header/record structure enables files to be converted from formatted to unformatted form, or scanned for a specific item. For example, a restart file will contain a record header with an identifier of 'PRESSURE' , type 'REAL' and with the number of elements set to the number of active cells in the study.
>
> In the formatted case the actual formats used for the data blocks are :
>
> | | |
> |---|---|
> | Integers | - 6(1X, I11) |
> | Reals | - 4(1X, E16.8) |
> | Double Precisions | - 3(1X, D22.14) |
> | Logicals | - 25(1X, L2) |

Character strings - 7(1X, A1, A8, A1)

The header line is written using a (1X, A1, A8, A1, 1X, I11, 1X, A1, A4, A1). The A1 fields are used to output quotes so that the data may be read using list input.

The A1 fields are used to output quotes so that the data may be read using list input.

A generic example of such a format is as follows:

```
'LOGIHEAD'         20 'LOGI'
 F  F  F  T  F  T  F  F  T  F  F  F  F  F  F  F  F  F  F
'ZGRP    '          1 'CHAR'
'G       '
'IWEL    '         72 'INTE'
        1         1         3         3         1         1
        3         2         2         0         1         0
        1         2         0         2         0      -100
        0         0         0         0         1         1
        0         0         0         0         0         0
        0         7         0         0         0         0
        5         1         1         1         1         1
        1         4         4         0         1         0
        1         2         0         1         0      -100
        0         0         0         0         1         1
       -1         0         0         0         0         0
        0         7         0         0         0         0
'PRESSURE'         15 'REAL'
  .39684106+004  .39684106+004  .39684106+004  .39684106+004
  .39684106+004  .39872349+004  .39872349+004  .39872349+004
  .39872349+004  .39872349+004  .40060627+004  .40060627+004
  .40060627+004  .40060627+004  .40060627+004
```

A partial listing of the TRIAL.FUNSMRY file is shown in Appendix 2. Also, several routines, in the supplied libraries, can handle this type of file. Refer to Appendix 1 for Routine Description and Libraries Source Code Listings.

6. PRACTICAL RESULTS

The data set considered in this section was a modified version of TDATA3, supplied by ECL (ECLIPSE® USER'S MANUAL, 1993). This file is listed in Appendix 2.1. The main modifications were:

- The number of wells considered in the present work;
- Production constraints;
- Production started with one well. The other ones were put in the drilling queue;
- Multi-level hierarchy was removed from the original data;

Costs concerning injection systems for water and gas were not considered.

6.1 Cases Considered

Considering the data shown in Table 3-1 we ran the optimizing program for several values of the tolerance parameter ε from Equation (6) (see Table 6-1).

Tolerances
10^{-6}
10^{-5}
10^{-3}
10^{-2}
10^{-1}

Table 6-1

Table 6-2 shows the initial polytope location. As this is a two parameter problem (*starting time* and *development time*) the polytope is a triangle (3 vertex polygon).

Parameter Description	Initial Value (months)
Starting Time for Vertex 1	1
Development Time for Vertex 1	1
Starting Time for Vertex 2	5
Development Time for Vertex 2	2
Starting Time for Vertex 3	2
Development Time for Vertex 3	5

Table 6-2

Figure 6-1 shows the oil production profile for Field 1. We must keep in mind that the maximum oil production rate for a single well is 1,500 BPD and so the number of wells can easily be derived from the curve.

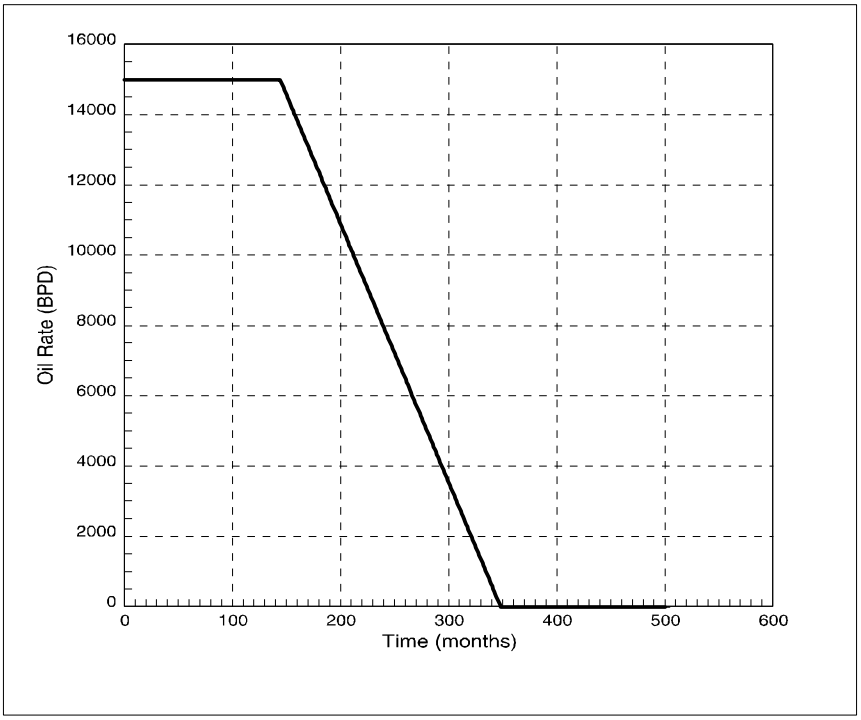

Figure 6-1

Table 6-3 shows the results obtained for the different values of tolerance. By simple sorting, the optimal location obtained from the surface is at (240.0, 55.0) with a function value equal to US$ 643.45 million. This point is referred to here as "apparent" optimum. The error and the distance to the *apparent* optimum were evaluated considering this location and the centroid of the polytope for the last iteration.

Tolerance ε	**Function Evaluations**	**Number of Movements**	**Function Value (US$/$10^6$)**	**Error (%)**	***Suggested* Optimum (months)**	**Distance to *Apparent* Optimum (months)**
10^{-6}	95+4*	46	643.38	0.01	225.3 54.2	14.7
10^{-5}	83+4*	40	643.38	0.01	225.3 54.2	14.7
10^{-3}	23+4*	13	642.06	0.22	221.8 96.7	45.5
10^{-2}	21+4*	12	641.70	0.27	227.9 98.7	45.3
10^{-1}	0+4*	0	410.87	36.15	2.67 2.67	243.0

(*) three evaluations to initialize the polytope and one at the final centroid for report purposes.

Table 6-3

Figure 6-3 shows the polytope movements towards the optimal location. The optimum found by the polytope method is referred to here as "suggested" optimum. The *apparent* optimum is shown in each plot as a small black square. The *suggested* optimum is the last polytope movement. We can observe in Figure 6-3 the efficiency of the polytope method according to the specific tolerance. The average run for each function evaluation takes five minutes.

The 'distance to the *apparent* optimum' was calculated as being the length of the straight line between the *suggested* optimum and the *apparent* optimum. This value gives us an idea how far we are from the known optimal result. It should not be considered as a measure of project parameters since its calculation involves quantities with different physical meanings (see Figure 6-2).

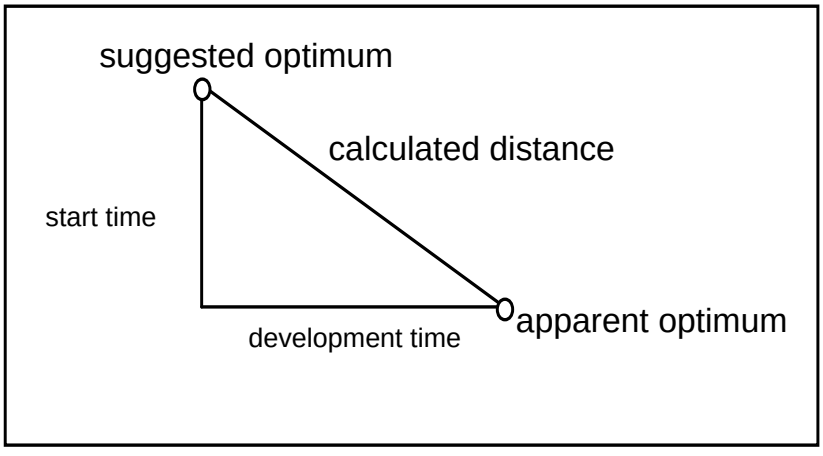

Figure 6-2

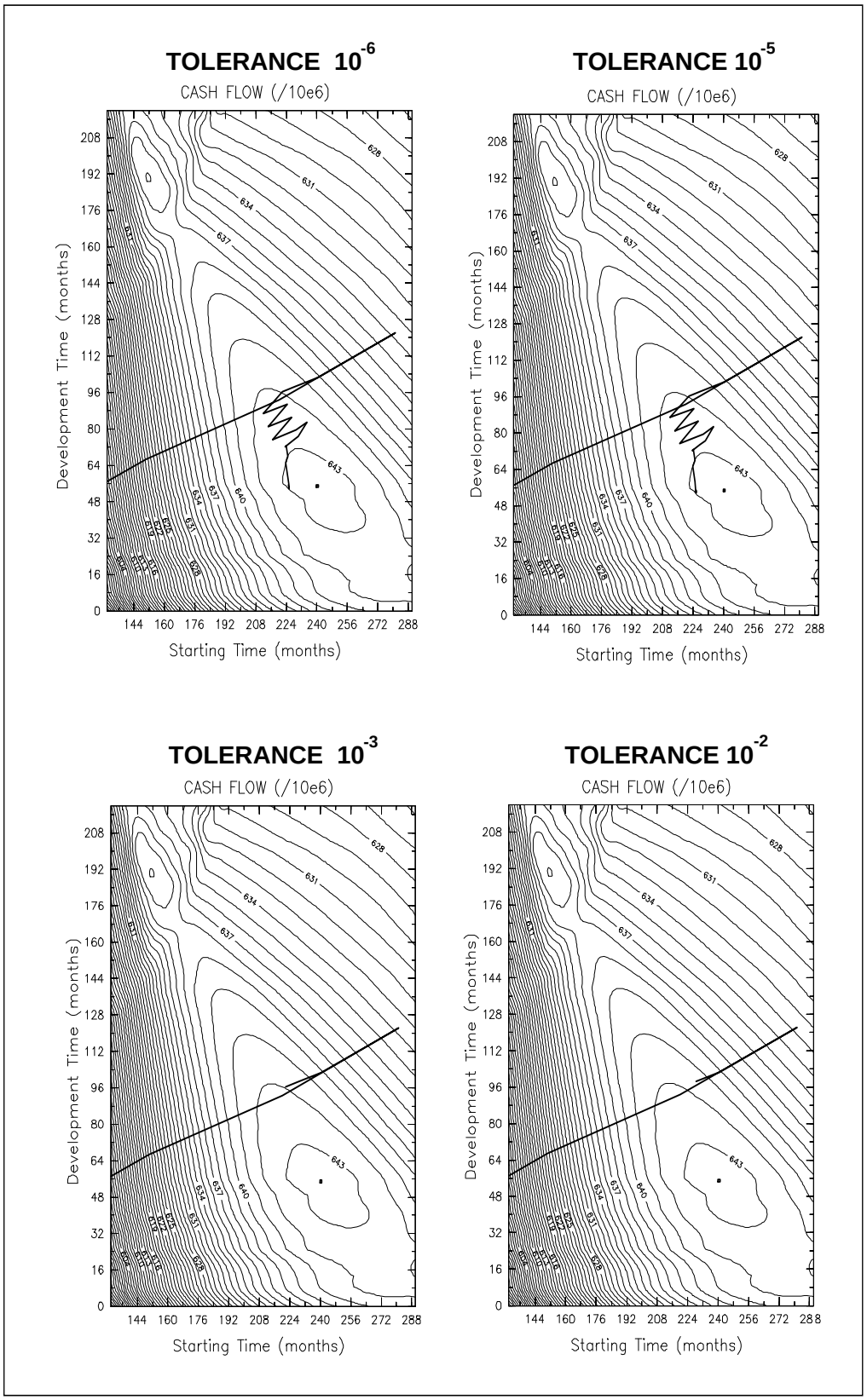
TOLERANCE 10-6
CASH FLOW (/10e6)
Development Time (months)
Starting Time (months)
TOLERANCE 10-5
CASH FLOW (/10e6)
Development Time (months)
Starting Time (months)
TOLERANCE 10-3
CASH FLOW (/10e6)
Development Time (months)
Starting Time (months)
TOLERANCE 10-2
CASH FLOW (/10e6)
Development Time (months)
Starting Time (months)

Figure 6-3

The oil production profile for both fields (Figure 6-4), rig operational schedule, number of wells of Field 1 and Field 2 (Figure 6-5) and actual monetary movement in net present value (NPV) for the project (Figure 6-6) as a function of time are provided for the *suggested* optimum where the tolerance ε was 10^{-5}.

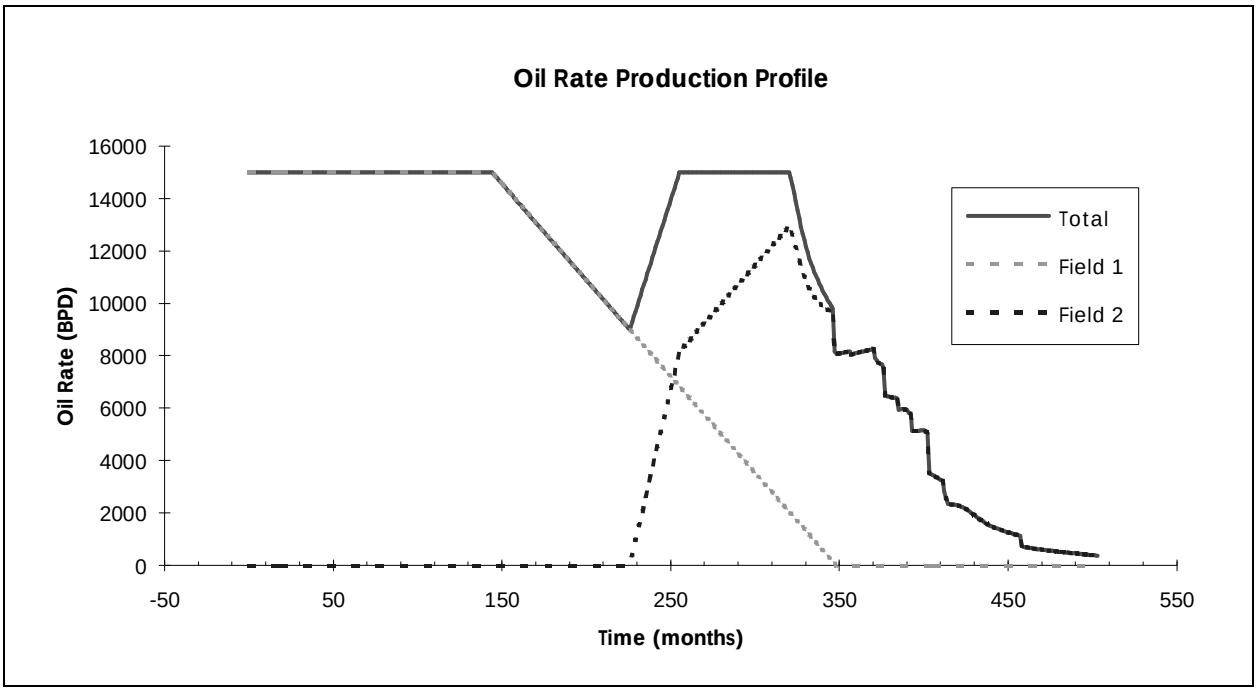

Figure 6-4

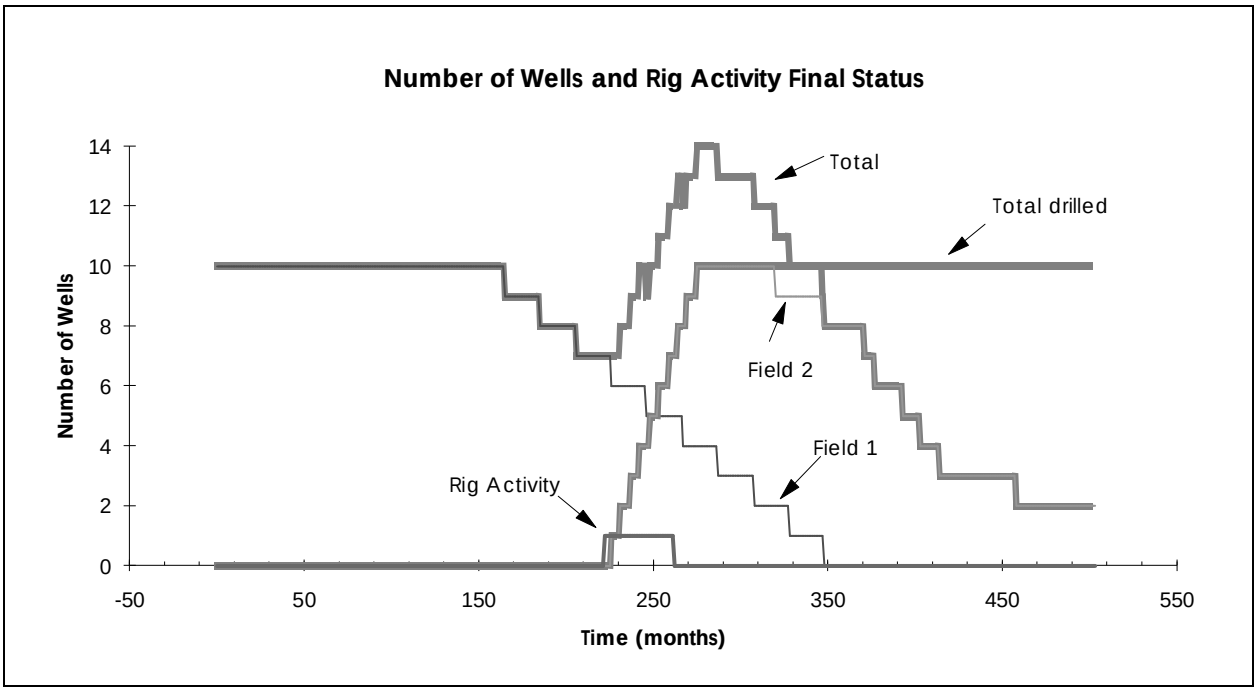

Figure 6-5

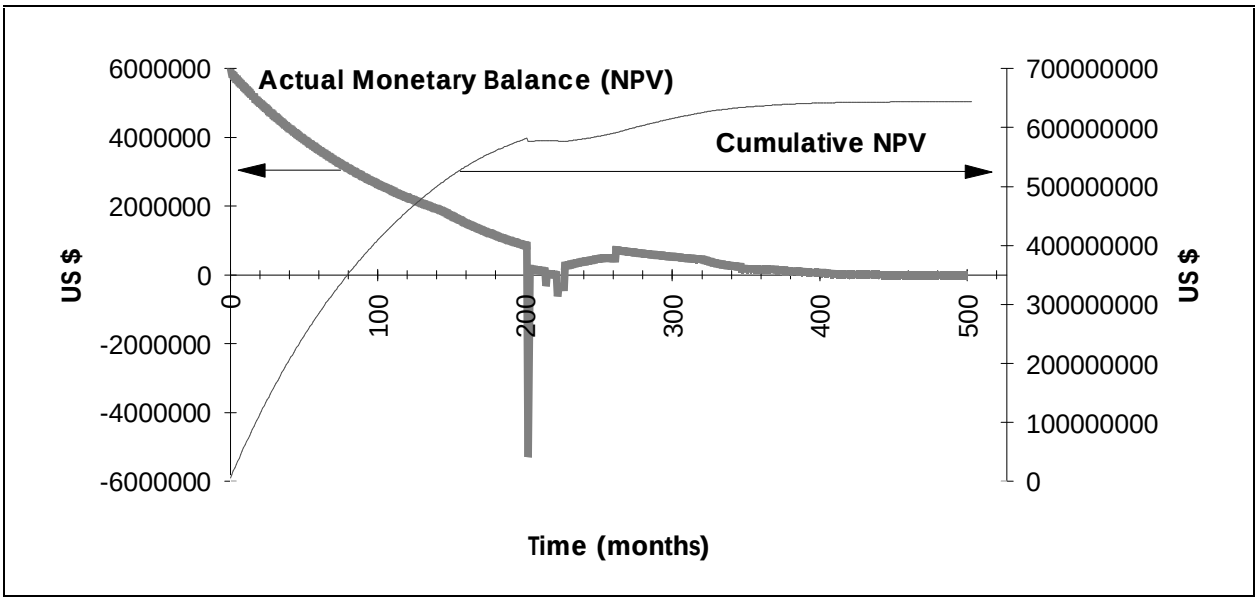

Figure 6-6

6.2 Other Polytope Sizes

As this method is strongly sensitive to polytope size and orientation, we also tried other sizes. We kept the same orientation. Table 6-4 shows the different polytope sizes, their area and the ratio height/d, where d is the distance from the *apparent* optimum to the origin, in months. This parameter give us an idea about the scale problem.

Polytope type	Vertex coordinates	Width (months)	Height (months)	Area (months^2)	Height/d
I	(1,1) (45,1) (1,45)	62.2	31.1	968.0	0.13
II	(1,1) (90,1) (1,90)	125.9	62.9	3960.5	0.26
III	(1,1) (5,1) (1,5)	5.7	2.8	8.0	0.01

Table 6-4

Other geometric parameters are constant because each polytope is an isosceles right triangle. So, we obtain for all new polytope sizes:

- Aspect ratio = width / height = 2
- Skinniness = (largest angle = 90°) / 60° = 1.5
- Regularity = (3 - Skinniness) / 2 = 0.75

Table 6-5 shows the results obtained for the different polytope sizes. Here, the tolerance ε used was 10^{-5}.

Type	Function Evaluations	Number of Movements	Function Value (US$/$10^6$)	Error (%)	*Suggested* Optimum (months)	Distance to *Apparent* Optimum (months)
I	63+4*	28	643.77	-0.05	245.4 54.0	5.52
II	68+4*	34	643.76	-0.05	238.1 54.0	2.16
III	50+4*	26	640.32	0.49	187.3 137.1	97.53

(*) 3 evaluations to initialize the polytope and 1 evaluation at the final centroid report.

Table 6-5

Figure 6-7 shows the paths followed by the new polytopes towards the optimal point. In these cases the method reached the region near to the *apparent* optimum for polytope types I and II and failed for polytope type III.

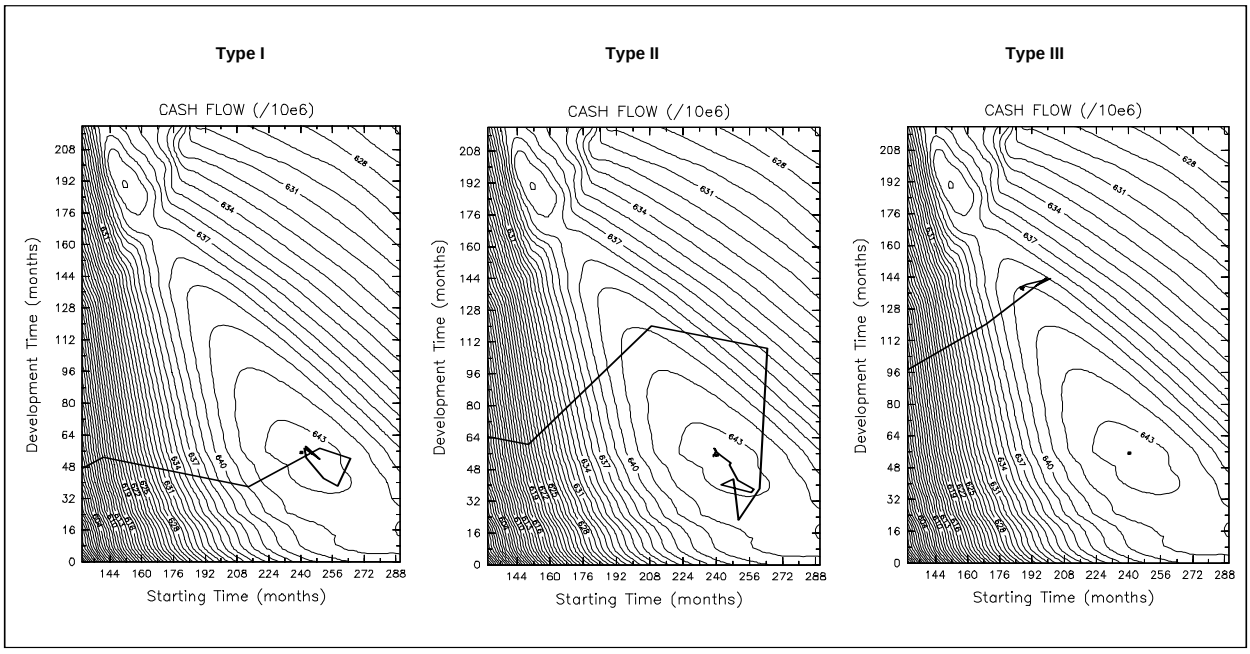

Figure 6-7

Figures 6-8 through 6-10 show the final production and development schedules for the optimal strategy suggested by the polytope type I. The corresponding data are listed in Appendix 2.6.

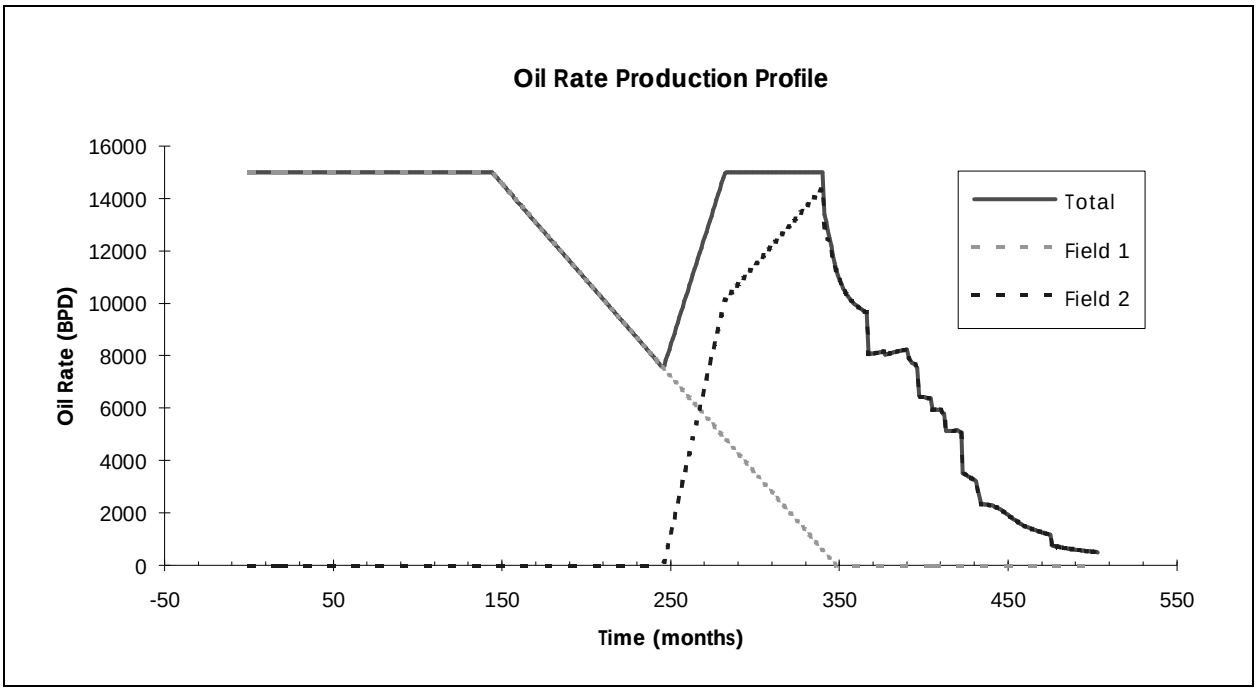

Figure 6-8

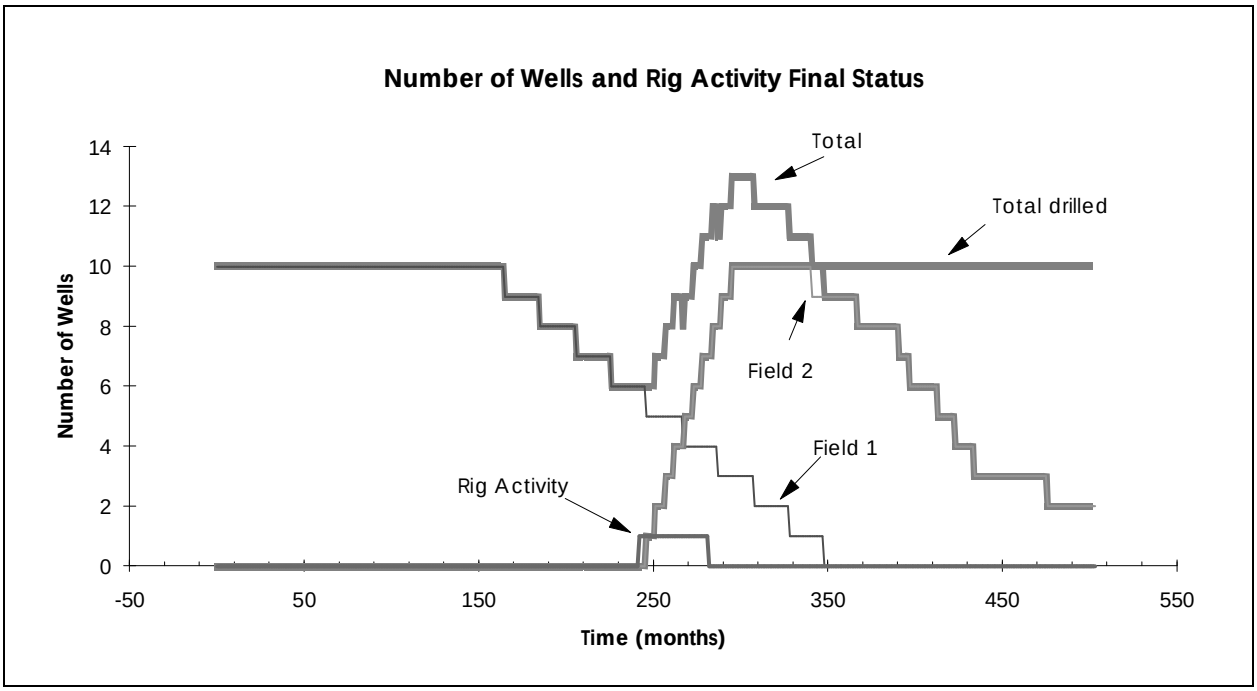

Figure 6-9

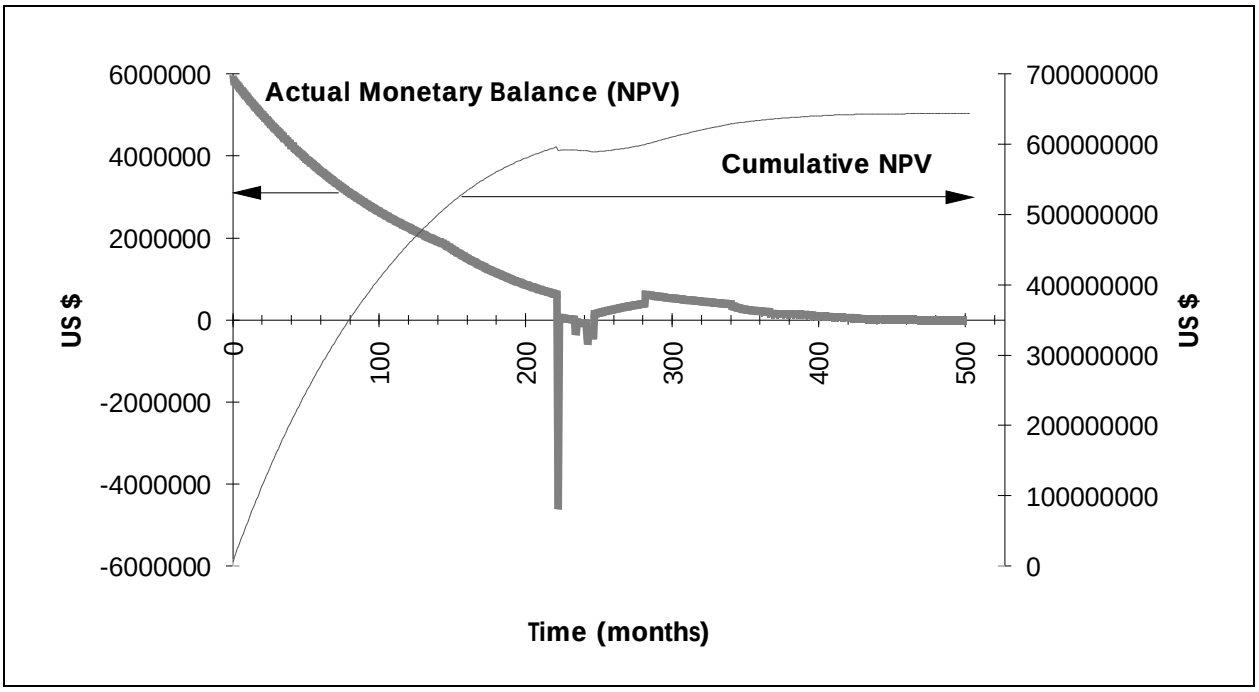

Figure 6-10

6.3 Starting at a New Location

We can observe in Figure 4-2 the occurrence of a local maximum in the vicinity of the point (144, 204). This is a special location because of the conditions of our particular problem: constant decline rate. So, we can anticipate that the optimum could be such that the *start time* is equal to the time when Field 1 production decline starts and the *development time* lasts until Field 1 production becomes zero. This location can be referred to here as the "expected" optimal point. Also, the shape of the surface presents some steps that originate from the rig allocation procedure as described in Section 3.1.

Using the tolerance ε of 10^{-5}, we started the original polytope and the polytope type II at the location (144, 204) in order to observe the behavior. The original polytope was chosen because, in spite of its small size, it previously converged on the global maximum region. The polytope type II was chosen due to its larger size. The *expected* optimal point was assumed to be the new polytope's centroid (see Table6-6).

Polytope type	**Original**	**II**

Parameter Description	Initial Value (months)	Initial Value (months)
Starting Time for Vertex 1	142.3	114.3
Development Time for Vertex 1	202.3	174.3
Starting Time for Vertex 2	146.3	203.3
Development Time for Vertex 2	203.3	174.3
Starting Time for Vertex 3	143.3	114.3
Development Time for Vertex 3	206.3	263.3

Table 6-6

Figure 6-11 shows the paths taken by both polytopes. The original polytope, a small one, became trapped in the local maximum region converging on a local optimum, and failed to reach the *apparent* optimum. After 42 function evaluations and 21 movements the *suggested* optimum was reached at (147.6, 199) with a cash flow value of US$ 640.38 million. Polytope type II, the biggest one, succeeded in reaching the global maximum region and converged on a optimal location at (246.7, 46.0), with a cash flow value of US$ 643.58 million, after 61 function evaluations and 27 movements.

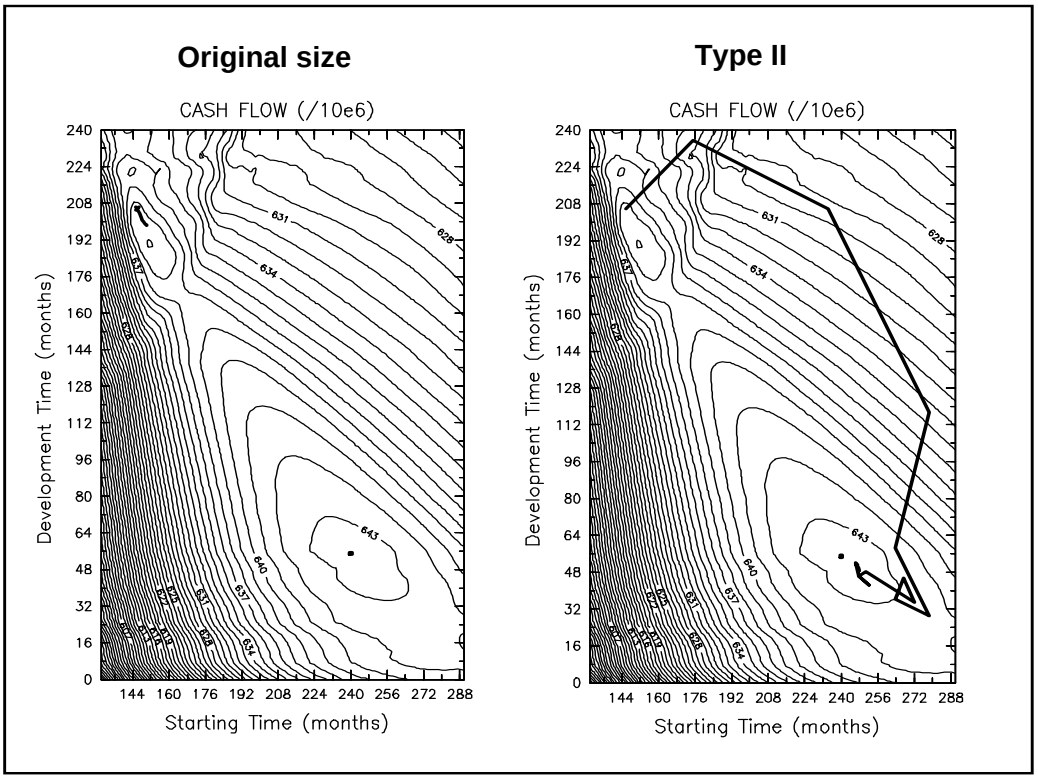

Figure 6-11

6.4 Discussion

From Table 6-3 we can note that the polytope efficiency varies strongly with the tolerance. Tolerance values which are too large can lead to no polytope movement. Tolerance values which are too small do not add any extra precision.

Although the difference between the objective function value at the *apparent* optimum and the ones shown in Table 6-3 may seem small, the effects in the observed parameters can be drastic. Neglecting the worst case and referring to the *apparent* optimum, the variation of the *starting time* parameter is in the range from 13 to 18 months while the range for the *development time* parameter, the one that most affects the results, is from 1 to 43 months.

Figure 6-3 shows that the polytope followed almost a straight line towards the optimal region. It happened due to the surface steepness and the way that the polytope moves flipping about the centroid of the opposite nodes. It is not assured that the next polytope movement will be towards the steepest direction. However, generally speaking, moving in the steepest direction does not assure convergence to a global maximum.

Based on Table 6-5 we could conclude that polytope type II obtained the best result due to its minimal distance to the *apparent* optimum. However, we must be aware that we do not know the optimal location and that the *apparent* optimum is the optimum for the surface control points only. We must focus on the objective function value that best represents the problem to be optimized. On this basis, the polytope type I obtained the best value for the objective function.

Figure 6-4 shows that Field 2 oil production could not maintain the maximum rate during Field 1 decline. The data set we used did not assure that the reservoir had sufficient potential to maintain the maximum capacity of the installed facilities. This is the reason for the better solution obtained by polytope type I in lieu of the *expected* optimum. This demonstrates the ability of this current approach to handle a realistic case in which the intuitive answer is not optimal.

Figure 6-5 shows that the most economical scheme is to contract the rig to drill the required wells in just one single campaign. We can also observe the number of wells operating at any specific time. The second rig was not required at any time.

Figure 6-6 shows the monetary balance, in actual values, affecting each specific time. We can also observe in this graph the perturbation in the 'cumulative NPV' curve caused by the realization of the second platform project.

The polytope type I reached an optimal point better than the *apparent* optimum, generating a negative error. It means that the so called *apparent* maximum was actually superseded by the *suggested* optimum since the *suggested* optimum was not limited to lie on the surface control points. The *suggested* optimum also superseded the *expected* one due to reasons already explained.

The polytope type II reached a better optimal location than the *apparent* optimum but still worse than the one reached by polytope type I.

The polytope type III shows how sensitive the method is to the polytope size. The *suggested* optimum is far from the results obtained previously. If we compare the results with the optimum suggested by the original polytope, running with the same tolerance ε, we can observe how the polytope geometry can affect the effectiveness of the optimization. This is a common case of the scale problem.

It is usual practice to restart the polytope from the suggested optimum to avoid local maxima. The restart procedure was not required for the polytopes considered so far. A restart procedure can be implemented in the optimizing program algorithm to automatically restart the polytope after it has converged on a *suggested* optimum. The number of restarts must be user defined or a new tolerance must be supplied to compare all *suggested* optima. It is also usual to start the polytope from different locations and check if the *suggested* optima are within a satisfactory range.

Figure 6-11 shows, once again, how this method is sensitive to the polytope initial size. From the same starting location one polytope became trapped and the other one converged on a optimum that superseded the *apparent* optimum.

7. CONCLUSIONS

The optimizing procedure developed so far works for an actual case and is ready to be refined in different ways.

If we compare the effort spent to generate the whole surface (60 ECLIPSE® runs) with the one made by the optimizing procedure to find the maximum we might be lead to conclude that it is cheaper (and better) to generate the surface. However this is not generally true. The particular parameter *starting time* is the only one that allows such a treatment, since it does not depend on the simulator at all. For other parameters to be considered in the future, it will not be possible to eliminate a variable due to the nonlinear relationships between them. Had this been the case here, it would have required more than 3,600 ECLIPSE® runs to generate the surface with the same level of detail.

An algorithm must also be defined to find the optimum value of the objective function. The problem here is the shape of such a function and the possibility of the occurrence of local maxima. For this reason all assumptions must be consistent with the limitations imposed by the actual field parameters.

As expected, the polytope search method is very sensitive to the initial size and orientation of the polytope. The scale problem is an important factor to be considered in any optimization problem. The restart procedure must be always performed in order to reduce the uncertainties about the *suggested* maximum. Also, different sizes and shapes for the polytope must be tried.

Even if we generate the whole surface it is not assured that the optimal location is defined, as shown by two of our polytope searches that converged on better optima that lay between points on the surface grid.

Because of the costs due to the second platform contract and facilities acquisition the maxima region lays in the fast *development time* range. We can note on Figure 7-10 the negative balance due to these costs. So, a fast development of the second field is required in order to pay for such a disbursement.

Despite not achieving production at full capacity during an interim period, the algorithm suggested an optimum different from the *expected* maximum. We might think that it should be better to keep the production at the maximum capacity of the installed facilities. However, the economical analysis showed it to be better to delay the second field development. The

perturbation observed in the 'cumulative NPV' curve is the responsible for the displacement of the optimum from the *expected* location to the *suggested* location in the vicinity of the *apparent* optimum. Two parameters can explain this behavior: oil price and interest rate. The perturbation brings down the curve and so it will reach lower values at its end. The algorithm locates the position where this effect is minimized, considering the costs and incomes due to the second field development. Such a location is close to the region where the 'cumulative NPV' curve becomes almost horizontal and the overall production can recover the amount spent to develop the second field.

8. FURTHER RESULTS

The results obtained in Section 6 show the *suggested* optimum in unexpected locations as reported in Table 6-5. This lead us to investigate in more detail the composition of the profit surface. Figure 8-1 shows the cumulative production surface with a plateau of maximum recovery. This means that any production strategy within this plateau provides much the same recovery and is the region of optimized recovery.

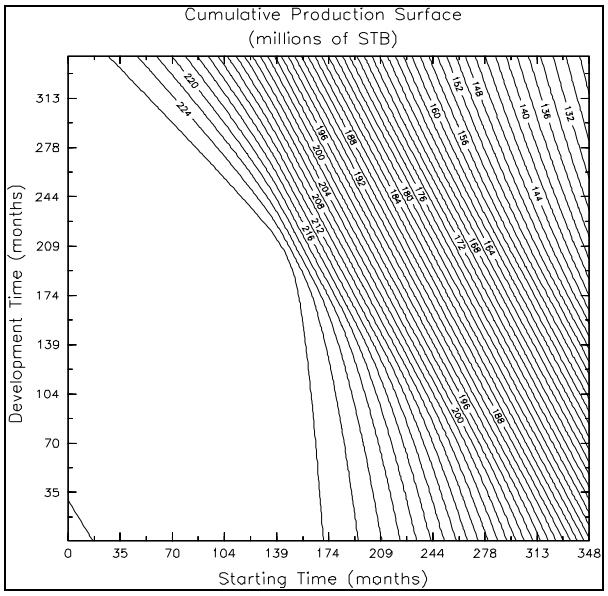

Figure 8-1

The work developed so far shows that there is a well defined optimal strategy that arises when we consider the development costs. To observe the significance of each component we split the objective function calculation over its components. The main terms are: rig mobilization costs, rig operational costs, drilling costs and platform contract costs. The production facility costs were always considered. In this analysis we assumed that the platform costs are fixed and we varied the remaining costs.

By considering each of these costs separately we observed their contribution to the original surface. Figure 8-2 shows that the optimum remains close to the *expected* optimum location at (144, 204) except for the case of platform costs which suggests that a faster development can pay quickly for the high disbursement.

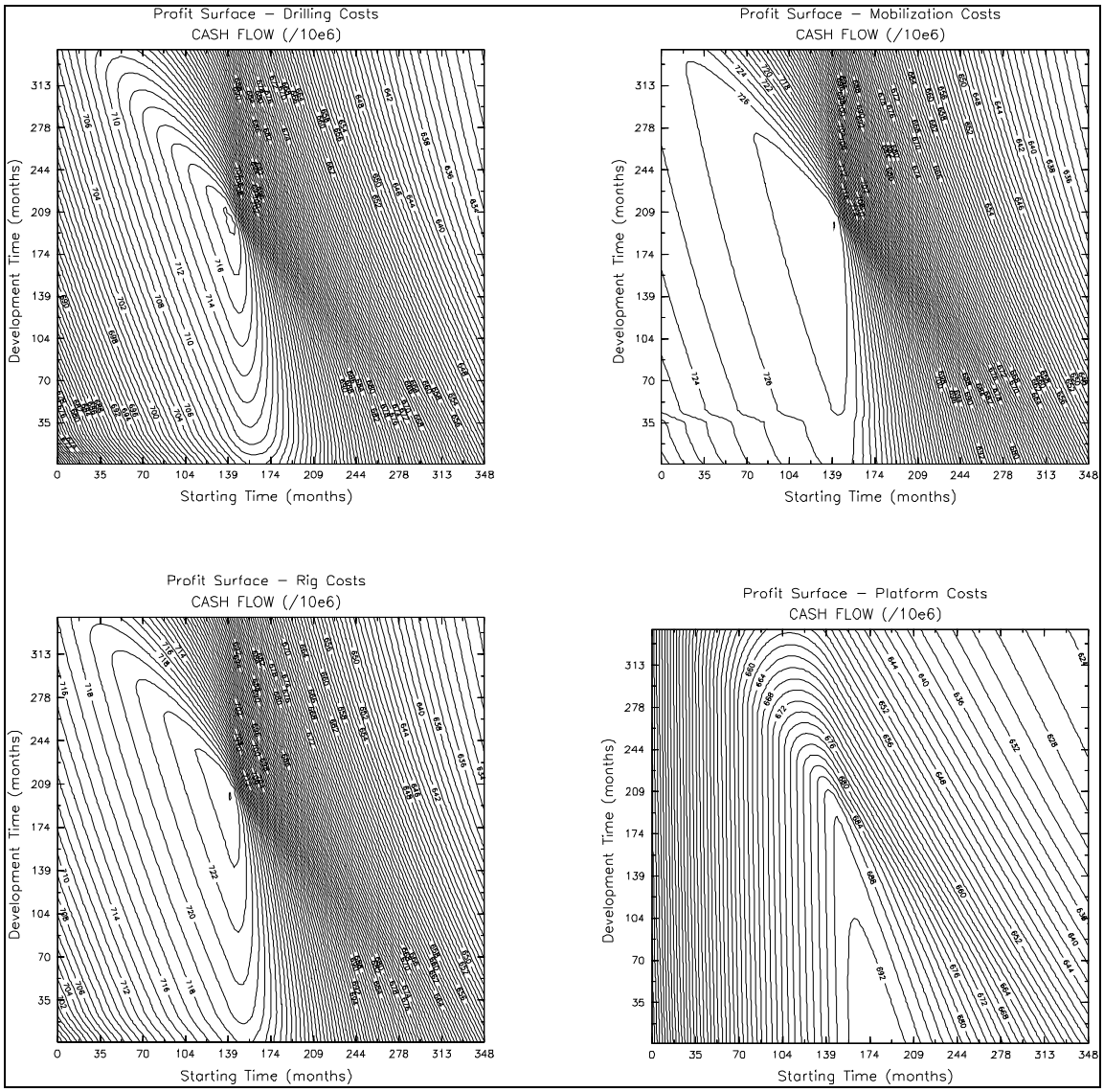

Figure 8-2

When we considered two components at the same time we observed that the optimum location is extremely sensitive to the mobilization costs. The composition of mobilization and drilling costs provides the most drastic optimum location change. Figure 8-3 shows also that when rig, drilling and rig mobilization costs only are considered we obtain a surface very similar to the one obtained when all costs are taken into account (Figure 4-2).

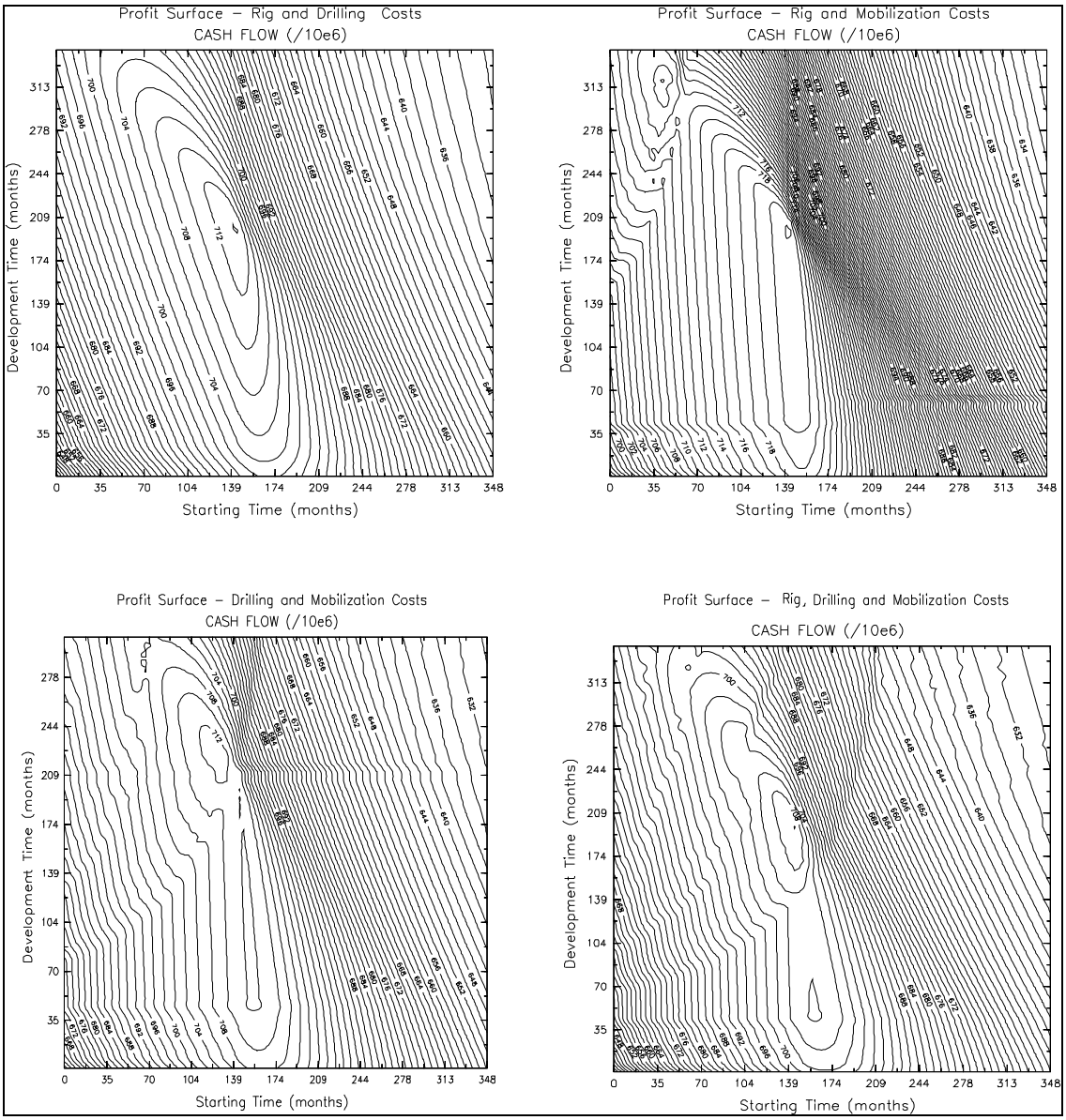

Figure 8-3

The goal was then to find out how the mobilization costs cause the change in the *expected* optimum location. We first analyzed the combination of rig mobilization and drilling costs. With this sensitivity analysis in mind we could later consider all costs. Figure 8-4 shows the profit surface for selected values of rig mobilization costs. The sequence in Figure 8-4 shows that when the rig mobilization costs become significant the optimum location moves towards the one suggested by the case where only the platform costs were considered (Figure8-2).

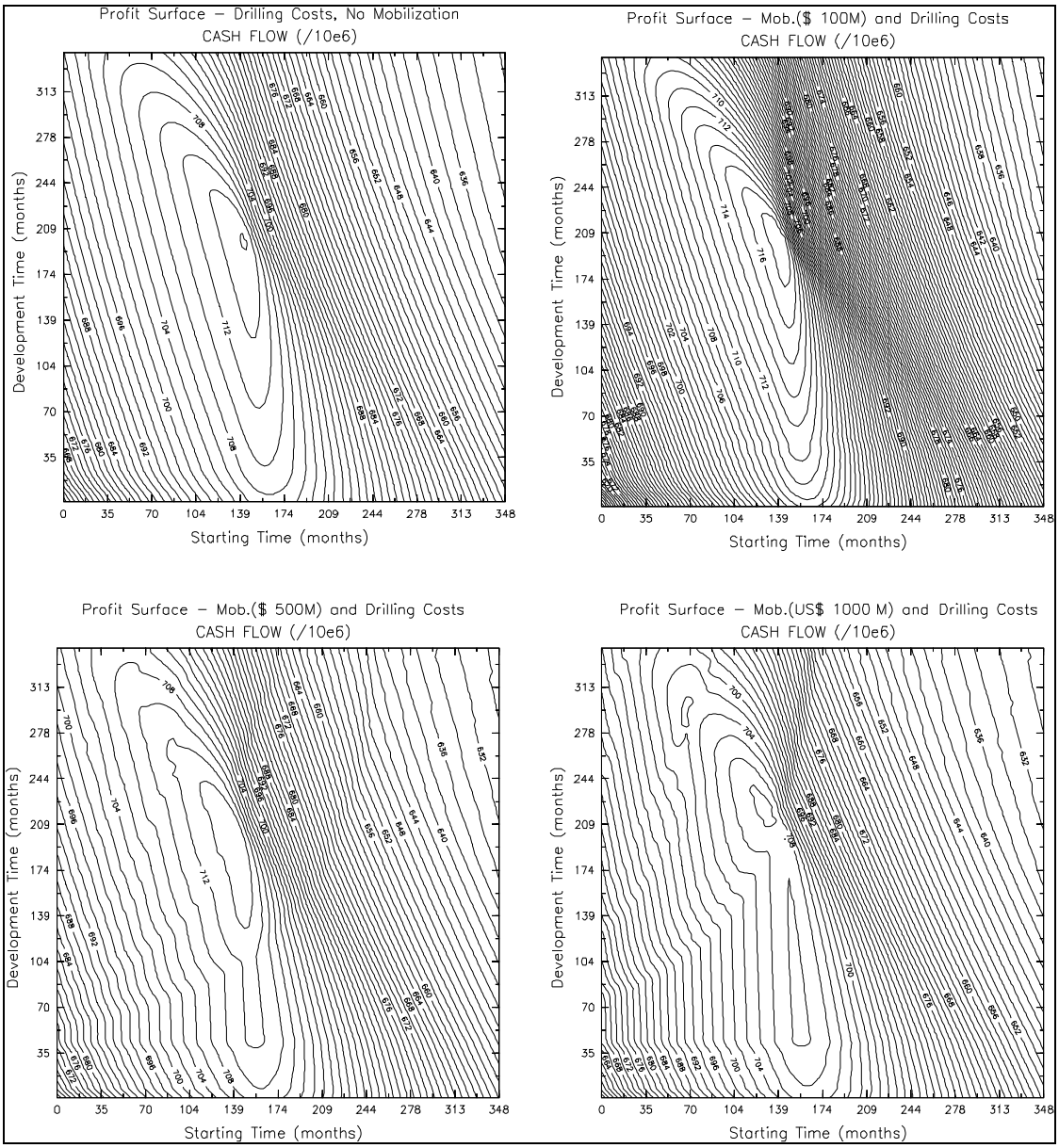

Figure 8-4

On varying the mobilization costs and considering all the other costs as in the base case we obtained similar surfaces. When there is no rig mobilization costs a pear-shaped plateau is obtained, with the rig and drilling costs accounting for one end of the plateau and the platform costs for the other one (Figure 8-5).

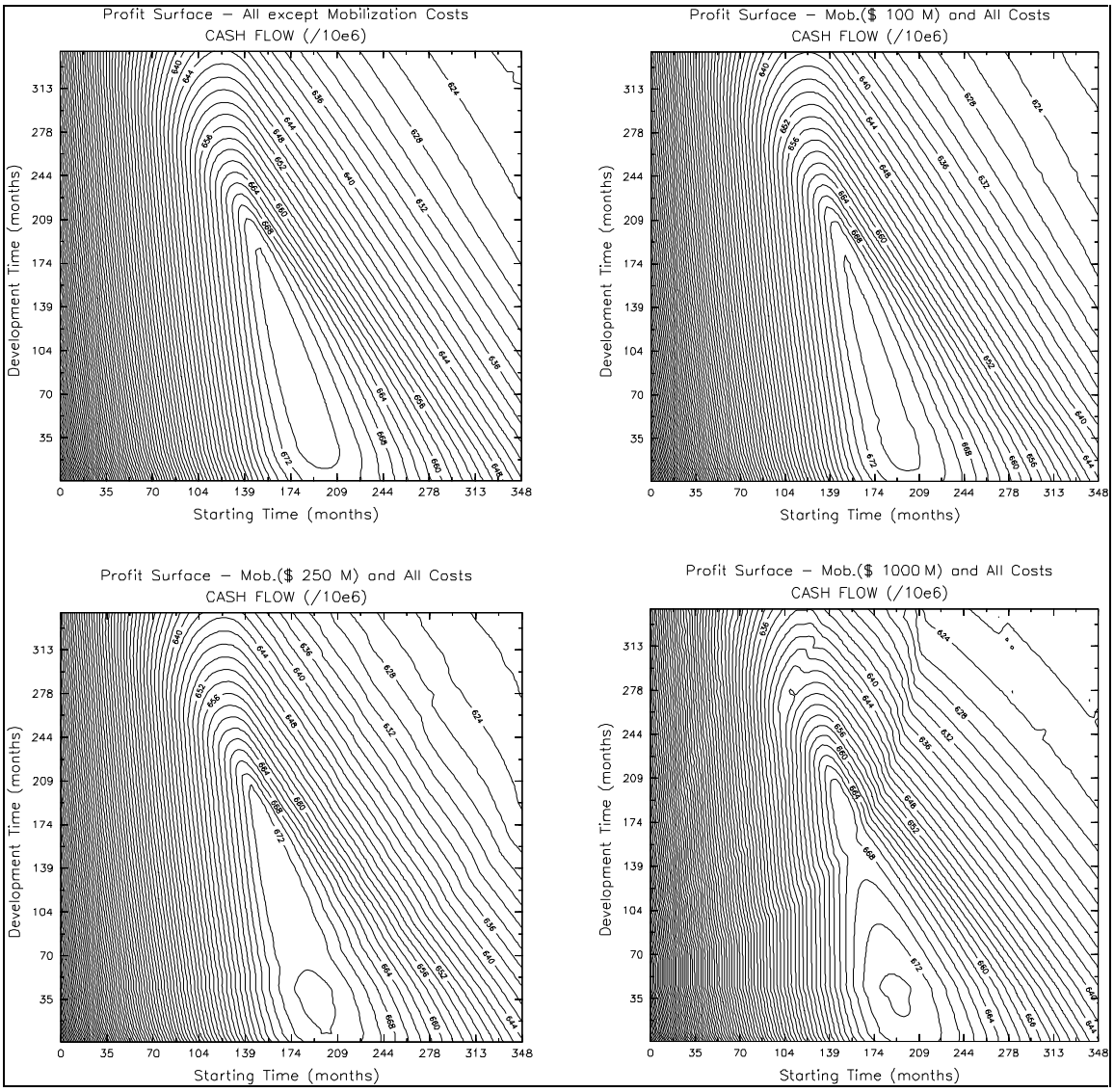

Figure 8-5

The analysis tried to investigate the contribution of each term regardless of whether it depends on others, e. g., the rig mobilization costs should not occur without rig operational costs and/or rig drilling costs. The main goal was a sensitivity analysis.

SUMMARY:

The sensitivity analysis could also be performed for the platform cost component. We expect this component to be a dominant factor for the final *suggested* optimum location. The rig mobilization cost contributes to a better definition of the *suggested* location.

The results obtained in this section support the following conclusions:

- Depending on the cost values involved the optimum location may not coincide with the *expected* one;

- If we ignore this possibility, substantial profit may be foregone if the development strategy is based on reserve optimization only;

- One can obtain more profit without necessarily having more production.

NOMENCLATURE

Discounted Net Cash Flow: Balance of a specific time expressed in present values.

Apparent optimal location: Maximum found by simply sorting the surface control points.

Development time (dt): Time period imposed on Field 2 to attain production at full capacity.

Expected optimal location: Maximum expected by considering an ideal case for the problem.

Pay-out time: Time when the balance is no longernegative.

Starting time (t_{i2}): Time when Field 2 production starts.

Suggested optimal location: Maximum (local or global) found by the polytope search method.

REFERENCES

1. ECLIPSE® manual, Exploration Consultants Ltd., England, Copyright 1993

2. Gill P. E., Murray W., Wright M. H.: *Practical Optimization*, California, USA: Academic Press, 1992.

3. Mannarino, R.: *Introdução à Análise Econômica*, Rio de Janeiro, Brazil: Editora Campus, 1991

4. Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P.: *Numerical Recipes in C*, New York, N.Y.: Cambridge University Press, 1992.

APPENDICES

1. SOURCE CODES

1.1 Main Program Source Listing

```
#include <stdio.h>
#include <math.h>
#define  NRANSI
#include "/ekofisk/suprid/bit/RECIPES/c/recipes_c-ansi/include/nr.h"
#include "/ekofisk/suprid/bit/RECIPES/c/recipes_c-ansi/include/nrutil.h"

#ifndef _custo_h
#include "custo.h"
#endif

#define NP      2
#define MP      NP+1

/* space of pointers to double functions that take 1 double argument */
typedef double (*fn) (double *);

/* header for myamoeba routine */
void myamoeba(float **p, float y[], int ndim, float ftol, \
            float (*funk)(float []), int *nfunk);

/* header for the routine to be minimized. It just call the  */
/* objective function in a proper way                        */
float funk(float *);
```

```
int main()
{
 int i,nfunc,j,ndim=NP;
 float *x, *y, **p;

 sched = 0;
 x = vector(1,NP);
 y = vector(1,MP);
 p = matrix(1,MP,1,NP);
 remove("optimum.log");

 ReadVariables();

 p[1][1] = sttime1;   /* guess for vertex 1, parameter 1 start time   */
 p[1][2] = dvtime1;   /* guess for vertex 1, parameter 2 develop time */
 p[2][1] = sttime2;   /* guess for vertex 2, parameter 1 */
 p[2][2] = dvtime2;   /* guess for vertex 2, parameter 2 */
 p[3][1] = sttime3;   /* guess for vertex 3, parameter 1 */
 p[3][2] = dvtime3;   /* guess for vertex 3, parameter 2 */

 /* print initial center of the polytope */
 x[1] = (p[1][1]+p[2][1]+p[3][1])/3.;
 x[2] = (p[1][2]+p[2][2]+p[3][2])/3.;
 printf(" %f  %f \n",x[1],x[2]);

 for(i=1; i<=MP; i++)
  {printf("setting leg %d \n",i);
    for(j=1; j<=NP; j++) x[j] = p[i][j];
    y[i] = funk(x);
   }

 myamoeba(p,y,ndim,ftol,funk,&nfunc);

 printf(" Polytope vertices \n");
 printf("  i       x[i]          y[i]       function \n");
 printf(" ---  -----------    ----------  ----------------- \n");
 for(i=1; i<=MP; i++)
       {printf("%3d ",i);
         for(j=1;j<=NP;j++)  printf(" %12.6f ",p[i][j]);
         printf(" %12.6f \n",-y[i]);
         }

 printf(" \n Polytope stopped after %d function evaluations \n",nfunc);
 x[1] = (p[1][1]+p[2][1]+p[3][1])/3.;
 printf(" mean value for parameter 1 = %f \n",x[1]);
```

```
 x[2] = (p[1][2]+p[2][2]+p[3][2])/3.;
 printf(" mean value for parameter 2 = %f \n",x[2]);
 sched = 1;
 printf(" value for function @ means = %f \n",-funk(x));
 printf(" tolerance                  = %f \n",ftol);
 printf(" Allocated bytes            = %f \n",A_BYTES);
 printf(" Deallocated bytes          = %f \n\n",F_BYTES);

 free_vector(x,1,NP);
 free_vector(y,1,MP);
 free_matrix(p,1,MP,1,NP);
}
/* ------------------------------------------------------- */
float funk(float p[])
{
 timeADT  vtime[MAX_PERIODS];
 float    cf;
 registro first;
 string   infile;
 FILE     *fout;
 int      i;

 infile  = CopyString((string) "../ecl/ROOT.DATA");

 AllocateTimeVector(vtime);
 Initialize(vtime);
 ReadFile(&first, infile);  /* price for parameters !! */

 if(remove("TRIAL.FUNSMRY") != 0)                    \
       PrintLog("TRIAL.FUNSMRY not removed. First run probably\n");
 if(remove("TRIAL.FSMSPEC") != 0)                    \
       PrintLog("TRIAL.FSMSPEC not removed. First run probably\n");

 RunSimulator(first,periods,(double)p[1],(double)p[2],vtime);

 cf =(float) -CashFlowAt(first,periods,(double)p[1],(double)p[2],vtime);

 if(sched > 0)
   if((fout = fopen("optSched.out","w")) == NULL) { PrintLog("Cannot open schedule file\n");}
     else
      {FPRINTF(fout,"schedule: tstart = %f  devl_time = %f \n",p[1],p[2]);
       FPRINTF(fout,"  Time  qTotal     q1      q2  Wells1  Wells2  tWells  dWells2 Rig1 Rig2
Secnd       Np   Local_Money     NPV      Cumulative\n");
       for(i=0; i<MAX_PERIODS; i++)
         if(vtime[i]->time != (-2.0 * ZERO_TIME_INDEX))
```

```
        FPRINTF(fout,"%6.1f %6.0f %7.1f %7.1f  %4d    %4d    %4d    %4d   %3d  %3d   %3d
%10.0f %10.0f  %10.0f  %10.0f\n",
        vtime[i]->time,vtime[i]->qtotal,vtime[i]->q1,vtime[i]->q2,
        vtime[i]->wells1,vtime[i]->wells2,vtime[i]->tot_wells,
        vtime[i]->drill_wells2,
        vtime[i]->rig[0],vtime[i]->rig[1],vtime[i]->secondary,
        vtime[i]->np,vtime[i]->local_money,vtime[i]->corrected_money,
        vtime[i]->payOut);
     }

 FreeTimeVector(vtime);
 FreeFile(first);
 FreeBlock(infile,strlen(infile)+1);
 for(i = 0; i < NUMBER_WELLS; i++)
    FreeBlock(wellNames[i], strlen(wellNames)+1);

 return(cf);
}
/* ---------------------------------------------------- */
#undef  NRANSI
```

1.2 Routine Description

The optimizing program uses the following definitions and data structures:

MAX_PERIODS	maximum dimension for time related arrays
MCTAB	maximum number of columns for production data table
MAX_RIGS	maximum number of rigs in campaign
ZERO_TIME_INDEX	how many entries are left before the one for time zero
NUMBER_WELLS	maximum number of well names handled

TimeADT data structure stores all information for a specific time:

```
typedef struct {
          double time;              current time
          int wells1;               number of  producing wells for Field 1
          int wells2;               number of  producing wells for Field 2
          int drill_wells2;         number of total drilled wells for Field 2
          int tot_wells;                  number of total producing wells
          int rig[MAX_RIGS];              array of flags to set operating rig
          int secondary;            number of contracted secondary permanent structure
          double q1, q2, qtotal;     oil rates for Field 1, Field 2 and total, respectively
```

```
    double np;                  total cumulative production
    double local_money;             historical value of the monetary balance
    double corrected_money;     present value of the monetary balance (NPV)
    double payOut;              cummulative NPV
    } *timeADT;
```

*Registro data structure stores a string and links to another data structure of same type:

```
typedef struct reg {
        string line;                text string
        struct reg *link;           pointer to another structure of same type
        } *registro;
```

The description for the routines is as follows:

void BuildSurface(int sched,double periodo, double start_time1,double start_time2,
double delta_start_time, double devl_rate1, double devl_rate2,
double delta_devl_rate, registro first);

Routine used to build the whole surface for the Objective Function. Initial, final and increment values for the two parameters are supplied by the user. The output is a ternary (x,y,z) for the whole domain.

Input parameters:

sched	flag to activate the schedule report. 0= no 1= yes
periodo	maximum investigated time interval, in months
start_time1	lower limit for the *start time* parameter, in months
start_time2	upper limit for the *start time* parameter, in months
delta_start_time	increment for the *start time* parameter, in months
devl_rate1	lower limit for the *development time* parameter, in months
devl_rate2	upper limit for the *development time* parameter, in months
delta_devl_rate	increment for the *development time* parameter, in months
first	pointer to structure with the first line of the ROOT file

double CashFlowAt(registro first, double periodo, double tstart, double deltat,
timeADT vtime[]);

Master routine for the whole package. After the simulation run this routine calls all necessary routines to evaluate the cash flow.

Input parameters:

first	pointer to structure with the first line of the ROOT file

periodo	maximum investigated time interval, in months
tstart	value for the *start time* parameter, in months
deltat	value for the *development time* parameter, in months

Output parameters:

vtime	vector of pointers to timeADT structure completly loaded

Returns cash flow value for the case (tstart, deltat)

void AllocateTimeVector(timeADT vtime[]);

Routine to allocate dynamic memory to the structure that stores all information for each timestep.

Input/output parameter:

vtime	vector of pointers to timeADT structure

void FreeTimeVector(timeADT vtime[]);

Routine to free the dynamicaly allocated memory.

Input/output parameter:

vtime	vector of pointers to timeADT structure

void Initialize(timeADT vtime[]);

Routine to initialize the structure with the default values.

Input/output parameter:

vtime	vector of pointers to timeADT structure

void MakeQ2Feasible(int t, timeADT vtime[]);

Routine to place the rig for each new well required to achieve the current oil production. It first checks if there is room available in rig schedule and then places the rig.

Input parameter:

t	current time, in months

Output parameter

vtime	vector of pointers to timeADT structure

int FindRoom(int *troom, int *nrig, int t, double *f, timeADT vtime[]);

Routine to schedule a rig for drilling a well. It first finds a room in time scale for all MAX_RIGS (number of available rigs) rigs. Then analizes if it is economically better to drill the well just after another one or to pay the mobilization cost for a new rig.

Input parameter:

t	current time, in months

Output parameters:

*troom	time when the rig can operate
*nrig	number of the selected rig
*f	*f=0 for no mobilization cost, *f=1 otherwise
vtime	vector of pointers to timeADT structure

Returns TRUE if room was found, FALSE otherwise

void PlaceRig(int nrig, int t, double f, timeADT vtime[]);

Routine to lock the time scale during rig activity and to apply drilling operational costs.

Input parameters:

nrig	number of the selected rig
t	current time, in months
f	f=0 for no mobilization cost, f=1 otherwise

Output parameter:

vtime	vector of pointers to timeADT structure

void PlaceSecondary(int t, double f, timeADT vtime[]);

Routine to lock the time scale during secondary permanent drilling plataform activity and to apply drilling costs.

Input parameters:

t	current time, in months
f	f=0 for no mobilization cost, f=1 otherwise

Output parameter:

vtime	vector of pointers to timeADT structure

double RigCost(int time, double factor);

Routine to evaluate total rig operational cost corrected to present value.

Input parameters:

time	current time, in months
factor	factor=0 for no mobilization cost, factor=1 otherwise

Returns evaluated rig cost

double SecondaryCost(int time, double factor);

Routine to evaluate total secondary structure operational cost corrected to present value.

Input parameters:

time	current time, in months
factor	factor=0 for no mobilization cost, factor=1 otherwise

Returns evaluated secondary structure cost

double EvaluateQ(double t, double tabl[][MAX_PERIODS], int tabl_size);

Routine to look up production profile table and return the oil rate for a specific time.

Input parameters:

t	current time, in months
tabl	table with field production data
tabl_size	size of the table tabl

:

Returns oil rate for time t

double Production_Cost(double q_period, double time_interval, double nwells);

Routine to evaluate the production cost for a certain oil rate, time interval and number of wells.

Input parameters:

q_period	oil rate in BPD
time_interval	time_interval valid for q_period, in months
nwells	number of wells required to produce q_period

Returns the cost to produce q_period during time_interval

void RunSimulator(registro first, double periodo, double tstart,double deltat,
timeADT vtime[]);

Routine to generate the ECLIPSE® input data file and to dispatch the simulator run.

Input parameters:

first	pointer to structure with the first line of the ROOT file
periodo	maximum investigated time interval, in months
tstart	value for the *start time* parameter, in months
deltat	value for the *development time* parameter, in months
vtime	vector of pointers to timeADT structure

void LoadTables(int *tabl1_size, double tabl1[][MAX_PERIODS], double tstart,
int *tabl2_size, double tabl2[][MAX_PERIODS], timeADT vtime[]);

Routine to load into tables the Field1 oil production profile and the simulator results.

Output parameters:

tabl1_size	size of the table Field 1 production data
tabl1	table with Field 1 production data
tstart	current time, in months
tabl2_size	size of the table Field 2 production data
tabl2	table with Field 2 production data
vtime	vector of pointers to timeADT structure

void ReadFile(registro *first, string infile);

Routine to read the ROOT.DATA file into a dynamically allocated memory.

Input parameter:

infile	file name

Output parameters:

*first	pointer to structure with the first line of the ROOT file

void WriteFile(FILE *outf, registro first);

Routine to dump the file ROOT.DATA from the dynamically allocated memory.

Input parameter:

*outf	file handler
first	pointer to structure with the first line of the ROOT file

void FreeFile(registro first);

Routine to free the memory used to save the file ROOT.DATA.

Input parameter:

first	pointer to structure with the first line of the ROOT file

void PrintLog(string text);

Routine to print control messages and error status into a log file.

Input parameter:

text	string to be printed into log file

void ReadVariables(void);

Routine to read the input data file optVar.in.

Returns values for all input variables

1.3 Main Header Source Listing

```
#ifndef _custo_h
#define _custo_h

#include <stdio.h>
#include <math.h>

#ifndef _lib193U_h
#include "/pangea/home/pete/bit/clib/193Ulib.h"
#endif

#define FPRINTF (void)fprintf
#define FSCANF  (void)fscanf
#define FCLOSE  (void)fclose
```

```
#define LINESIZE 256

#define MAX_PERIODS             1000
#define MCTAB                      4
#define MAX_RIGS                   2
#define ZERO_TIME_INDEX          100
#define BUSY                       1
#define FREE                       0
#define CONTRACTED                 1
#define AVAILABLE                  0

#define NUMBER_WELLS              50

typedef struct {
            double time;
            int wells1;
            int wells2;
            int drill_wells2;
            int tot_wells;
            int rig[MAX_RIGS];
            int secondary;     /* # of contracted secondary struc */
            double q1, q2, qtotal;
            double np;
            double local_money;
            double corrected_money;
            double payOut;
            } *timeADT;

typedef struct reg {
                string line;
                struct reg *link;
                } *registro;

float     sttime1,dvtime1,sttime2,dvtime2,sttime3,dvtime3,ftol;
double interest_year, QMAX1_per_well, t_decl1, t_final1,periods, oil_price, platform_cost,
         rig_cost, mob_cost,offshore_well_drill_cost, facilities_initial_cost, overhead_factor,
         insurance_cost, f, operational_cost_per_well, handling_cost, MAX_CAPACITY,
        QMAX1, interest_month;
int       sched,rig_capcty, lastRoom[MAX_RIGS];
string  wellNames[NUMBER_WELLS];

void BuildSurface(int sched,double periodo, double start_time1,double start_time2,
                    double delta_start_time, double devl_rate1, double devl_rate2,
                    double delta_devl_rate, registro first);
```

```
double CashFlowAt(registro first, double periodo, double tstart, double deltat,
                  timeADT vtime[]);
void AllocateTimeVector(timeADT vtime[]);
void FreeTimeVector(timeADT vtime[]);
void Initialize(timeADT vtime[]);
void MakeQ2Feasible(int t, timeADT vtime[]);
int FindRoom(int *troom, int *nrig, int t, double *f, timeADT vtime[]);
void PlaceRig(int nrig, int t, double f, timeADT vtime[]);
void PlaceSecondary(int t, double f, timeADT vtime[]);
double RigCost(int time, double factor);
double SecondaryCost(int time, double factor);
double EvaluateQ(double t, double tabl[][MAX_PERIODS], int tabl_size);
double Production_Cost(double q_period, double time_interval, double nwells);
void RunSimulator(registro first, double periodo, double tstart,double deltat, timeADT vtime[]);
void LoadTables(int *tabl1_size, double tabl1[][MAX_PERIODS], double tstart,
                int *tabl2_size, double tabl2[][MAX_PERIODS], timeADT vtime[]);
void ReadFile(registro *, string );
void WriteFile(FILE *, registro);
void FreeFile(registro);
void PrintLog(string text);
void ReadVariables(void);

#endif
```

1.4 Main Library Source Listing

```
#include <sys/types.h>
#include <sys/wait.h>
#ifndef _custo_h
#include "custo.h"
#endif

/*---------------------------------------------------------------------*/
void BuildSurface(int sched,double periodo,double start_time1, double start_time2,
                  double delta_start_time, double devl_time1, double devl_time2,
                  double delta_devl_time,registro first)
{
 double t, t_start2, devl_time, taux;
 double q1, q2, np, cash_flow, q_period, cost_period;
 timeADT vtime[MAX_PERIODS];
 FILE *fout;
 int i;
```

```
 /* periodo = tfinal1 + devl_time2 + 60.;    attention */

 AllocateTimeVector(vtime);

  /* We now assume that the upper limit for start_time (start_time2) must
    not be greater t_final1 once that it is not interesting for us.
    Also, the upper limit for devl_time (devl_time2)
*/

 if(start_time2 > t_final1) start_time2 = t_final1;
 if(devl_time2  > t_final1)  devl_time2 = t_final1;

/* here we could consider start_time2+devl_time2  > t_final1. But we
   allow the solution be natural once that rates lower than MAX_CAPACITY
   will generated lower profits
*/

 for(devl_time=devl_time1; devl_time <= devl_time2;
     devl_time += delta_devl_time)
 {
  switch(sched)
   {
    case 1: if((fout = fopen("optSched.out","w")) == NULL)
              PrintLog("Cannot open sched\n");
            break;
    case 2: if((fout = fopen("optSched.out","a")) == NULL)
              PrintLog("Cannot open sched\n");
            break;
   }

  if(remove("TRIAL.RSM") != 0)
        PrintLog("Cannot remove TRIAL.RSM file. First run probably\n");
  RunSimulator(first,periodo,0.,devl_time,vtime);

  for(t_start2=start_time1; t_start2 <= start_time2;
     t_start2 += delta_start_time)
   {
    Initialize(vtime);

    cash_flow = CashFlowAt(first,periodo, t_start2,devl_time,vtime);

    printf(" %7.3f  %5.1f  %15.2f \n",(float) t_start2,
          (float) devl_time, cash_flow/1000000.);
     /*  (float) MAX_CAPACITY/devl_time/100., cash_flow/1000000.);  */
```

```
   if(sched > 0)
     {
     FPRINTF(fout,"schedule: tstart = %f  devl_time = %f \n",
           t_start2,devl_time);
     FPRINTF(fout,"production declining in %f months \n",
           t_final1-t_decl1);
     FPRINTF(fout,"Time   qTotal     q1      q2   Wells1  Wells2 ",
              " tWells  dWells2",
              " Rig1 Rig2 Secnd     Np    Local_Money     NPV\n");
     for(i=0; i<MAX_PERIODS; i++)
      if(vtime[i]->time != (-2.0 * ZERO_TIME_INDEX))
        FPRINTF(fout,"%6.1f %6.0f %7.1f %7.1f  %4d  %4d  %4d  %4d  %3d  %3d  %3d
%10.0f  %10.0f  %10.0f \n",   \
        vtime[i]->time,vtime[i]->qtotal,vtime[i]->q1,vtime[i]->q2,
        vtime[i]->wells1,vtime[i]->wells2,vtime[i]->tot_wells,
        vtime[i]->drill_wells2,
        vtime[i]->rig[0],vtime[i]->rig[1],vtime[i]->secondary,
        vtime[i]->np,vtime[i]->local_money,vtime[i]->corrected_money);
     }

  }

 if(sched > 0) fclose(fout);

 }
 FreeTimeVector(vtime);
 return;
}
/*---------------------------------------------------------------------*/
double CashFlowAt(registro first, double periodo, double tstart,
              double deltat,timeADT vtime[])
{
 double q1, q2, np, cash_flow, nrig, q_period, cost_period, t, taux;
 double time_interval, nwells, A;
 double tabl1[MCTAB][MAX_PERIODS], tabl2[MCTAB][MAX_PERIODS];
 int tabl1_size, tabl2_size;
 int    i,j,k,actual;
 bool   done;
 double trash;
 FILE *fhist, *fsim_out;
 string temp;

 q1 = q2 = np = 0.0;
 taux = -1.0;
```

```
interest_month = pow((interest_year + 1.0), 1.0/12.) - 1.0;

temp = GetBlock(90);
sprintf(temp,"c: period %7.2f   tstart %7.2f   dev_time %7.2f \n",periodo,tstart,deltat);
PrintLog(temp);
FreeBlock(temp, 90);

LoadTables(&tabl1_size, tabl1, tstart, &tabl2_size, tabl2,vtime);

for(t=0; t < periodo; t++)
  {
   actual       = (int) t + ZERO_TIME_INDEX;
   time_interval = (t - taux) * 30.0;

   q1  = EvaluateQ(t,tabl1,tabl1_size);
   q2  = EvaluateQ(t,tabl2,tabl2_size);
   q_period = (q1+q2);

   if(q2 > 0.0) MakeQ2Feasible(actual,vtime);
   if(q_period > MAX_CAPACITY) { q2 = MAX_CAPACITY - q1;
                                 q_period = (q1+q2);
                               }

   vtime[actual]->time   = t;
   vtime[actual]->q1     = q1;
   vtime[actual]->q2     = q2;
   vtime[actual]->qtotal = q_period;

   vtime[actual]->wells1    = ceil(q1/QMAX1_per_well);
   vtime[actual]->tot_wells = vtime[actual]->wells1 +
                              vtime[actual]->wells2;

   nwells = vtime[actual]->tot_wells; /* number of producer wells */

   if(q_period == 0.0)
     { cost_period = operational_cost_per_well * 2.; }/* min cost */
    else { cost_period = 0.;  }

   cost_period += Production_Cost(q_period,time_interval,nwells);

   vtime[actual]->local_money +=  (q_period * time_interval *
                                   oil_price - cost_period);

   vtime[actual]->np = vtime[actual-1]->np + q_period * time_interval;
   taux = t;
```

```
  }

 /* locate first time to contract platform 2 years before.
    Payment is 30% in the first month and the remaining 70%
    will be paid monthly for 2 years.
    Note that vtime[i]->time is initialized with a flag number
    and when the timestep is touched its value is corrected    */

 i    = -1;
 done = FALSE;
 while(!done)  if(vtime[++i]->drill_wells2 != 0)
               { done = TRUE;
                 time0 = i;
                 j     = time0 - 24;
                 if(vtime[j]->time == (-2.0 * ZERO_TIME_INDEX))
                    vtime[j]->time = vtime[time0]->time - 24;
                vtime[j]->local_money -= 0.30 * platform_cost;
                A = 0.70 *  platform_cost *
                      pow((1.0+interest_month),24.) * interest_month /
                      (pow((1.0+interest_month),24.) - 1.);
                for(k=j+1; k<= time0; k++)
                   { if(vtime[k]->time == (-ZERO_TIME_INDEX * 2.0))
                        vtime[k]->time = vtime[k-1]->time + 1;
                     vtime[k]->local_money -= A;
                   }
               }

 /* buy facilities one year before the first drilled well and pay in the same way
    as done for the platform                                          */

 j = time0 - 12;
 A  = facilities_initial_cost * MAX_CAPACITY;
 vtime[j]->local_money -= 0.30 * A;
 A = 0.70 * A * pow((1.0+interest_month),12.) * interest_month /
       (pow((1.0+interest_month),12.) - 1.);
 for(k=j+1; k<= time0; k++) vtime[k]->local_money -= A;

/* evaluate NPV for each timestep and then perform cashflow */

cash_flow = 0;
for(i=0; i<MAX_PERIODS; i++)
   if(vtime[i]->time != (-2.0 * ZERO_TIME_INDEX))
     {
      vtime[i]->corrected_money = vtime[i]->local_money *
                        pow((1.0+interest_month),-vtime[i]->time);
```

```
     cash_flow += vtime[i]->corrected_money;
     vtime[i]->payOut = cash_flow;
    }
 return(cash_flow);
}
/*-------------------------------------------------------------------*/
void MakeQ2Feasible(int t, timeADT vtime[])
{
 double f;
 int i, troom, nrig, new_wells;
 int isec = 0;

 new_wells = vtime[t]->drill_wells2 - vtime[t-1]->drill_wells2;

 /* solve rigs allocation for each well */
 for(i=0; i<new_wells; i++)
    if(FindRoom(&troom,&nrig,t,&f,vtime)) PlaceRig(nrig,troom,f,vtime);
 return;
}
/*-------------------------------------------------------------------*/
bool FindRoom(int *troom, int *nrig, int t, double *f, timeADT vtime[])
{
 int i,k;
 bool found;

 *troom = t;
 *nrig  = 0;
 found  = FALSE;
 while(!found)
     {
      if(*troom == 0) break;
      (*troom)--;
      for(k=0; k<MAX_RIGS; k++)
        if(vtime[*troom]->rig[k] == FREE)
          {
           found = TRUE;                    /* room found */
           for(i=*troom; i>(*troom)-4; i--)      /* check 4 months */
             if(vtime[i]->rig[k] == BUSY) found = FALSE;
           if(found)                  /* there is really a room */
             {
              *nrig = k;
              *f = 0.;    /* it means that rig has no mob_cost */
              if( ( (*troom-4) > lastRoom[k] ) &&
                        ( RigCost(*troom,1.) > RigCost(lastRoom[k]+4,0.) ))
                   *troom = lastRoom[k]+4;
```

```
                else *f = 1.;

              lastRoom[k] = *troom;
              break;
             }
           }
      }
 return found;
}
/*-------------------------------------------------------------------*/
void PlaceRig(int nrig, int time, double f, timeADT vtime[])
{
 int i;

 for(i=time; i>time-4; i--)
    {
     vtime[i]->time         = i - ZERO_TIME_INDEX;
     vtime[i]->rig[nrig]    = BUSY;
     vtime[i]->local_money -= (rig_cost + offshore_well_drill_cost);
    }

 vtime[time-3]->local_money -= f * mob_cost;
 return;
}
/*-------------------------------------------------------------------*/
void PlaceSecondary(int time, double f, timeADT vtime[])
{
 int i;

 for(i=time; i>time-4; i--)
    {
     vtime[i]->time         = i - ZERO_TIME_INDEX;
     vtime[i]->secondary++;
     vtime[i]->local_money -= (rig_cost + offshore_well_drill_cost);
    }

 vtime[time-3]->local_money -= f * mob_cost;
 return;
}
/*-------------------------------------------------------------------*/
double RigCost(int time, double factor)
{
 double s,t,i;

 t = (double) time - ZERO_TIME_INDEX;
```

```
 s = factor * mob_cost;
 for(i=t-3; i<=t; i++) s += (rig_cost + offshore_well_drill_cost)*
                          pow((1.0+interest_month),-i);

 return s;
}
/*---------------------------------------------------------------------*/
double SecondaryCost(int time, double factor)
{
 double s,t,i;

 t = (double) time - ZERO_TIME_INDEX;
 s = factor * mob_cost;
 for(i=time; i>time-4; i--) s += (rig_cost + offshore_well_drill_cost)*
                          pow((1.0+interest_month),-i);

 return s;
}
/*---------------------------------------------------------------------*/
void AllocateTimeVector(timeADT vtime[])
{
 int i;

 for(i=0; i<MAX_PERIODS; i++)
   vtime[i] = GetBlock(sizeof(*vtime[i]));
 return;
}
/*---------------------------------------------------------------------*/
void Initialize(timeADT vtime[])
{
 int i,j;

 for(i=0; i<MAX_PERIODS; i++)
   {
    vtime[i]->time  = -ZERO_TIME_INDEX * 2.0;
    vtime[i]->wells1 = 0;
    vtime[i]->wells2 = 0;
    vtime[i]->drill_wells2 = 0;
    vtime[i]->tot_wells = 0;
    for(j=0; j<MAX_RIGS; j++)
      { vtime[i]->rig[j] = FREE;
        lastRoom[j]      = i;
      }
    vtime[i]->secondary   = AVAILABLE;
    vtime[i]->q1          = 0.0;
```

```
    vtime[i]->q2          = 0.0;
    vtime[i]->qtotal      = 0.0;
    vtime[i]->np          = 0.0;
    vtime[i]->local_money = 0.0;
    vtime[i]->corrected_money = 0.0;
   }
 return;
}
/*---------------------------------------------------------------------*/
void FreeTimeVector(timeADT vtime[])
{
 int i;

 for(i=0; i<MAX_PERIODS; i++) FreeBlock(vtime[i], sizeof(*vtime[i]));
 return;
}
/*---------------------------------------------------------------------*/
double EvaluateQ(double t, double tabl[][MAX_PERIODS], int tabl_size)
{
 int i,j;

 i = (int) t;
 return (tabl[1][i]);
}
/*---------------------------------------------------------------------*/
double Production_Cost(double q_period, double time_interval,
                       double nwells)
{
 double cop, ch, isl;

 cop  = operational_cost_per_well * nwells * time_interval/30.0; /*US$*/
 ch   = handling_cost * q_period * time_interval;       /* US$        */
 isl  = 0.05 * oil_price * (1-f) * time_interval;       /* /bbl/day   */
 return ((1+overhead_factor)*(cop+ch)+insurance_cost+(isl * q_period));
}
/*---------------------------------------------------------------------*/
 void RunSimulator(registro first, double periodo, double tstart,
                   double deltat,timeADT vtime[])
  {
   FILE *fsim_in;
   pid_t pid;
   int i;
   double c,f,t;
   string temp;
```

```
if((fsim_in = fopen("TRIAL.DATA","w")) == NULL)
  {
   printf("Cannot open simulator input file\n");
   abort();
  }

FPRINTF(fsim_in,"-- periodo %f tstart %f dev_time %f\n",periodo,tstart,deltat);
WriteFile(fsim_in,first);

f = modf(deltat,&t);    /* DEVELOPMENT TIME period */
i = (int) t;
for(c=0.; c < t; c++)     /* for(t=1; t <= deltat; t++) */
 {
  FPRINTF(fsim_in,"GCONPROD\n 'FIELD' 'ORAT' ");
  FPRINTF(fsim_in,"   %f / \n/ \nTSTEP \n30 \n/ \n",
              (c+0.5) * MAX_CAPACITY / deltat );
 }
if(f > 0.)
 {
  FPRINTF(fsim_in,"GCONPROD\n 'FIELD' 'ORAT' ");
  FPRINTF(fsim_in,"   %f / \n/ \nTSTEP \n%8.3f \n/ \n",
              (t+f/2.) * MAX_CAPACITY / deltat, f*30. );
 }
FPRINTF(fsim_in,"GCONPROD\n 'FIELD' 'ORAT' %f / \n/ \n",MAX_CAPACITY);

f = modf((double)(periodo-tstart-deltat+1.),&t);    /* QMAX period */
i = (int) t;
while(i > 50)
 {
  FPRINTF(fsim_in,"TSTEP \n50*30 \n / \n");
  i -= 50;
 }
if(i > 0) FPRINTF(fsim_in,"TSTEP \n%d*30\n/ \n", i);
if(f > 0.) FPRINTF(fsim_in,"TSTEP \n%8.3f \n/ \n", f*30.);
FPRINTF(fsim_in,"END \n / \n");
FCLOSE(fsim_in);

switch(pid=fork())
 {
  case -1: printf("ERROR on FORK\n");
           abort();

  case  0: FreeTimeVector(vtime);
           FreeFile(first);
           execl("runECLIPSE","runECLIPSE",(string) NULL);
```

```
            printf("Cannot run SIMULATOR\n");
            abort();
     }
   temp = GetBlock(80);
   sprintf(temp,"Aracne dispatching process %d \n",pid);
   PrintLog(temp);
   waitpid(pid,&i,0);
   sprintf(temp,"Aracne is back... Process %d return code %d \n",pid,i);
   PrintLog(temp);
   FreeBlock(temp, 80);
   return;
  }
/*--------------------------------------------------------------------*/
void LoadTables(int *tabl1_size, double tabl1[MCTAB][MAX_PERIODS],
      double tstart, int *tabl2_size, double tabl2[MCTAB][MAX_PERIODS],
      timeADT vtime[])
{
 FILE *fhist, *fsim_out;
 int i,j,k,n,nv,nalf,ierro, twells, limalf[MAXV];
 string nothing, s;
 char *alf[MAXV];
 float vnum[MAXV], cumw[MAXV];
 double trash;

 /*  load history table */

 for(i=0; i < MAX_PERIODS; i++)
    for(j=0; j < MCTAB ; j++) tabl1[j][i] = tabl2[j][i] = 0.;

 if((fhist = fopen("optHist.in","r")) == NULL)
           {
            PrintLog("Cannot open HISTORY file\n");
            abort();
           }
 i = 0;
 while(!feof(fhist))
    {
     FSCANF(fhist," %lf %lf ",&tabl1[0][i],&tabl1[1][i]);
     if(++i >  MAX_PERIODS)
       { printf(" index > MAX_PERIODS when loading HISTORY.\n");
            break;
           }
    }
 FCLOSE(fhist);
 *tabl1_size = --i;
```

```
/*  load well names  */

nothing = CopyString(":+:+:+:+:+");

for(i=0; i<MAXV; i++)
  { alf[i] = GetBlock(MAXV);
    memset(alf[i],'\0',MAXV);
  }

if((fsim_out = fopen("TRIAL.FSMSPEC","r")) == NULL)
          { printf("Cannot open TRIAL.FSMSPEC file\n");
            abort();
          }

GetRecordAlpha(fsim_out,"WGNAMES",alf,&nalf);
twells = 0;
for(j=0; j<nalf; j++)
  if((strcmp(alf[j],nothing) !=0) && (strcmp(alf[j],"FIELD") !=0))
    { wellNames[twells++] = CopyString( s = trim(alf[j]));
      FreeBlock(s, strlen(s)+1);
    }
FCLOSE(fsim_out);

/*  load simulation table */

if((fsim_out = fopen("TRIAL.FUNSMRY","r")) == NULL)
          { printf("Cannot open TRIAL.FUNSMRY file\n");
            abort();
          }

i = (int) tstart; /* this time is not related to ZERO_TIME_INDEX */
for(j=0; j<twells; j++) cumw[j+3] = 0.;

/* for every timestep */
while((s=LocateHDR(fsim_out,"MINISTEP")) != NULL)
 { GetRecordNum(fsim_out,"PARAMS",vnum,&nv);
   tabl2[0][i] = vnum[0]/30.;    /* time from days --> months */
   tabl2[1][i] = vnum[2];        /* total field production  */

   /* figure out the number of total drilled and producing wells */
   for(j=0; j<twells; j++)
     { if((cumw[j+3] += vnum[j+3]) != 0.)
         vtime[i + ZERO_TIME_INDEX]->drill_wells2 += 1;
       if((vnum[j+3]) != 0.) vtime[i + ZERO_TIME_INDEX]->wells2 += 1;
```

```
      }
    FreeBlock(s, strlen(s)+1);
    if((++i + ZERO_TIME_INDEX)> (MAX_PERIODS-1)) break;
  }

 *tabl2_size = --i;
 FCLOSE(fsim_out);
 for(i=0; i<MAXV; i++) FreeBlock(alf[i], MAXV);
 FreeBlock(nothing, strlen(nothing)+1);
 return;
}
/*-------------------------------------------------------------------*/
void PrintLog(string text)
{
 FILE *log;

 log = fopen("optLog.out","a");
 (void) fprintf(log,text);
 fclose(log);
 return;
}
/*-------------------------------------------------------------------*/
void ReadFile(registro *first, string infile)
{
 registro previous,actual;
 FILE *inf;
 string line;

 inf = fopen(infile,"r");

 *first = GetBlock(sizeof(**first));
 line   = GetBlock(LINESIZE);
 (void) fgets(line,LINESIZE,inf);
 (*first)->line = CopyString(line);
 (*first)->link = NULL;
 previous       = *first;

 while(fgets(line,LINESIZE,inf) != NULL)
  { actual   = GetBlock(sizeof(*actual));
    actual->line   = CopyString(line);
    actual->link   = NULL;
    previous->link = actual;
    previous = actual;
  }
```

```
 fclose(inf);
 FreeBlock(line, LINESIZE);
 return;
}
/*---------------------------------------------------------------------*/
void WriteFile(FILE *outf, registro first)
{
 registro actual;

 for(actual = first; actual->link != NULL; actual = actual->link)
  fprintf(outf,"%s",actual->line);

 fprintf(outf,"%s",actual->line);  /* prints last record */

 return;
}
/*---------------------------------------------------------------------*/
void FreeFile(registro first)
{
 registro actual, previous;

 previous = first->link;
 do { actual = previous;
      FreeBlock(actual->line, strlen(actual->line)+1);
      previous = actual->link;
      FreeBlock(actual, sizeof(*actual));
     } while(previous != NULL);

 return;
}
/*---------------------------------------------------------------------*/
void ReadVariables()
 {
  char buff[80];
  FILE *in;

  in = fopen("optVar.in","r");

  FSCANF(in,"%s %f  ",buff,&sttime1);
  FSCANF(in,"%s %f  ",buff,&dvtime1);
  FSCANF(in,"%s %f  ",buff,&sttime2);
  FSCANF(in,"%s %f  ",buff,&dvtime2);
  FSCANF(in,"%s %f  ",buff,&sttime3);
  FSCANF(in,"%s %f  ",buff,&dvtime3);
  FSCANF(in,"%s %f  ",buff,&ftol);
```

```
  FSCANF(in,"%s %lf ",buff,&periods);
  FSCANF(in,"%s %lf ",buff,&interest_year);
  FSCANF(in,"%s %lf ",buff,&MAX_CAPACITY);
  FSCANF(in,"%s %lf ",buff,&QMAX1_per_well);
  FSCANF(in,"%s %lf ",buff,&oil_price);
  FSCANF(in,"%s %lf ",buff,&platform_cost);
  FSCANF(in,"%s %lf ",buff,&rig_cost);
  FSCANF(in,"%s %lf ",buff,&mob_cost);
  FSCANF(in,"%s %i  ",buff,&rig_capcty);
  FSCANF(in,"%s %lf ",buff,&offshore_well_drill_cost);
  FSCANF(in,"%s %lf ",buff,&facilities_initial_cost);
  FSCANF(in,"%s %lf ",buff,&overhead_factor);
  FSCANF(in,"%s %lf ",buff,&insurance_cost);
  FSCANF(in,"%s %lf ",buff,&f);
  FSCANF(in,"%s %lf ",buff,&operational_cost_per_well);
  FSCANF(in,"%s %lf ",buff,&handling_cost);

  fclose(in);
  return;
 }
/*---------------------------------------------------------------------*/
```

1.5 Auxiliary Header Source Listing

```
/*
        File: 193Ulib.h
        ** Last modified Tue Apr  6 12:26:41 1993 by seligman **

        This file contains general definitions for those parts of
        Eric Roberts's genlib library that are used in CS193U,
        Spring '93. Much of the implementation was changed from
        Eric's genlib by James Finn.

        The purpose of this library is to provide a
        basic set of tools and conventions that increase the
        readability of C programs, particularly as they are used
        in a teaching environment.


 */
```

```
#ifndef _lib193U_h
#define _lib193U_h

/* Section 1: Imports */

#include <stdlib.h>

/* Section 2: New defined types */

/*
  Type: bool

  This type has two values, FALSE and TRUE, which are equal
  to 0 and 1 respectively.  Most of the advantage of defining
  this type comes from readability, because it allows the
  programmer to provide documentation that a value will
  take on only one of these two values.
 */

#ifdef THINK_C
  typedef Boolean bool;
#else
#define FALSE 0
#define TRUE 1
  typedef int bool;
  /*
  typedef enum { FALSE, TRUE } bool;
  */
#endif

/*
  Type: string

  The type string is identical to the type char *, which is
  traditionally used in C programs.  The point of defining
  a new type is principally to improve program readability.
  At the abstraction levels at which the type string is used,
  it is usually not important to take the string apart into
  its component characters.  Declaring it as a string
  emphasizes this atomicity.
 */

typedef char *string;

/*
```

```
   The type (void *) is the most general pointer type.
   Besides NULL, the only other constant of type (void *) is
   UNDEFINED, which may be used as a special sentinel value
   to indicate an undefined pointer value in those contexts in
   which NULL would be a legitimate value.
 */

#define UNDEFINED ((void *) undefined_object)

extern char undefined_object[];
extern double A_BYTES, F_BYTES;

/*
   Type: opaque

   Every time the symbol opaque is used in a type definition,
   it creates a new unique type.  The purpose of this
   mechanism is to allow the interface to hide the details
   of a type implementation from the client.  An interface
   declares a type as

       typedef opaque *mytype;

   and then uses mytype as the name of the type exported
   by the interface.  In the implementation, pointers to
   the opaque type must be converted back and forth to
   the underlying type using type casts.
 */

#define opaque struct { int dummy; }

/* Section 3: Functions */

/*
   Function: GetBlock
   Usage: ptr = GetBlock(nbytes);

   GetBlock allocates a block of memory of the given size.
   If there no memory available, GetBlock generates an error.
 */

void *GetBlock(size_t nbytes);

/*
   Function: FreeBlock
```

```
   Usage: FreeBlock(ptr, nbytes);

   FreeBlock deallocates a block of memory of pointed by ptr.
 */

void FreeBlock(void *ptr, size_t nbytes);

/*
   Function: CopyString
   Usage: s = CopyString(t);

   Copies the string t into dynamically-allocated storage.
 */

string CopyString(string s);

/*
   Function: ConcatString
   Usage: s = ConcatString(p,t);

   Copies the string p+t into dynamically-allocated storage.
 */

string ConcatString(string s, string t);

/*
   Function: GetLine
   Usage: length = GetLine(s, n);

   s[] is an array of size n.
   GetLine reads the next line from standard input.
   The line is stored in s without the terminating '\n',
   and the result is terminated with a '\0'.  At most
   n-1 characters are read, to allow room for the '\0'.

   Returns: length of the string, or -1 on end of file.
 */

int GetLine(char s[], int n);

/*
   Function: Error
   Usage: Error(msg, ...)

   Prints an error message, expanding % constructions in
```

```
   the error message string in the manner of printf.
   The program exits then exits with status code as given by
   the constant ErrorExitStatus.
 */

#define ErrorExitStatus 1
void Error(string msg, ...);

/* new routines implemented by Bittencourt */

#define MAXV 127
#define MAXC 255
void foliC(char *,int,int,float [],int *,char *[],int *,int[],int *);
char *LocateHDR(FILE *, char *);
void GetRecordAlpha(FILE *, char *,char *[],int *);
void GetRecordNum(FILE *, char *,float [], int *);

string dtoa(double, string);
string itoa(int, string);
int odd(int);
int numeric(char *);
char *trim(char *);
char *Lower(char *);
char *Upper(char *);

#endif /* _lib193U_h */
```

1.6 Auxiliary Library Source Listing

```
/*
 * 193Ulib.c
 ** Last modified Sun Jun 13 10:15:31 1993 by seligman **
 *
 * Implementation of 193Ulib.h interface for CS193U, Spring '93.
 * Modified and butchered by James Finn from Eric Roberts's genlib.c.
 * Major changes from Eric's library:
 *       1)      Threw away everything not used in CS193U.
 *       2)      Recoded many functions to eliminate unnecessary library calls.
 *       3)      Moved CopyString from strlib.c/h to 193Ulib.c/h
 *       4)      Added the GetLine function.
 *
 * 193Ulib uses the exception handling facilities of exception.h
```

```
 * in the implementation of the Error function if _EXCEPTION is defined.
 */

/* Uncomment the following line to enable exception handling. */
/* #define _EXCEPTION */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>

#ifndef _lib193U_h
#include "193Ulib.h"
#endif

#ifdef _EXCEPTION
#include "exception.h"
#endif

char undefined_object[] = "UNDEFINED";
double A_BYTES = 0.,
      F_BYTES = 0.;

/*-------------------------------------------*/
void *GetBlock(size_t nbytes)
{
   void *result = malloc(nbytes);
   A_BYTES += (double) nbytes;

   if (result == NULL)
      Error("Free storage exhausted");
   return (result);
}
/*-------------------------------------------*/
void FreeBlock(void *ptr, size_t nbytes)
{
   F_BYTES += (double) nbytes;
   free(ptr);

   return;
}
/*-------------------------------------------*/
string CopyString(string s)
{
          string t = GetBlock(strlen(s) + 1);
```

```
        strcpy(t, s);
        return(t);
}
/*-----------------------------------------*/
string ConcatString(string s, string t)
{
        string p = GetBlock(strlen(s) + strlen(t) + 1);
        strcpy(p, s);
      strcat(p,t);
        return(p);
}
/*-----------------------------------------*/

int GetLine(char s[], int n)
{
        int i, c;

        for (i = 0; i < n-1 && (c = getchar()) != EOF && c != '\n'; i++)
                s[i] = c;
        s[i] = '\0';
        if (c == EOF && i == 0) return -1;
        else return i;
}
/*-----------------------------------------*/
void Error(string msg, ...)
{
   va_list args;

   va_start(args, msg);
   fprintf(stderr, "Error: ");
   vfprintf(stderr, msg, args);
   va_end(args);
   fprintf(stderr, "\n");
#ifdef _EXCEPTION
   if (HandlerExists(&ErrorException)) raise(ErrorException);
#endif
   exit(ErrorExitStatus);
}
/*----------------------------------------      added by Bit */
string dtoa(double x,string fmt)
{
 string t;

 string s = GetBlock(100);
 sprintf(s,fmt,x);
```

```
 t = CopyString(s);
 FreeBlock(s, 100);
 return t;
}
/*---------------------------------------------------------*/
string itoa(int x,string fmt)
{
 string t;

 string s = GetBlock(100);
 sprintf(s,fmt,x);
 t = CopyString(s);
 FreeBlock(s, 100);
 return t;
}
/*---------------------------------------------------------*/
int odd(int x)
{ return(x % 2); }
/*---------------------------------------------------------*/
void foliC(char *s,int inic,int ifim, float vnum[],int *nv,char *alf[],
          int *nalf, int limalf[], int *ierro)
{
 int i,j,k,load,alreadyAdded;
 double x;
 char *temp[MAXV];

 if(s == (char *) NULL) return;

 for(i=0; i<MAXV; i++) { temp[i] = GetBlock(MAXV);
                         vnum[i] = 0.;
                         memset(alf[i],'\0',MAXV);
                         memset(temp[i],'\0',MAXV);
                        }
 j=k=load=*ierro=0;
 alreadyAdded=1;
 i = -1;

 do
  {
   i++;
   switch (s[i])
    {
     case '\0': ;
     case '\t': ;
     case '\n': break;
```

```
    case '\'': ;
    case '"' : load = (load+1) % 2;
               if(load) i++;
                 else alreadyAdded = 0;
    case ' ' : if(!alreadyAdded)
                { j++;
                  k = 0;
                  alreadyAdded++;
                }
               if(load) temp[j][k++] = s[i];
               break;
    default  : temp[j][k++]  = s[i];
               if(!load) alreadyAdded = 0;
   }
  } while((s[i] != '\n') && (s[i] != '\0'));

 i = -1;
 j = -1;
 *nalf = 0;
 *nv   = 0;
 while(strlen(temp[++i]) != 0)
   if(numeric(temp[i])) vnum[(*nv)++] = (float) atof(temp[i]);
     else { strcpy(alf[(*nalf)++],temp[i]);
            limalf[++j] = i;
          }

 for(i=0; i<MAXV; i++) FreeBlock(temp[i], MAXV);

 return;
}
/*---------------------------------------------------------------------*/
/*   printf("%s %f %f\n",t,atof(t),strtod(t,(char **)NULL));          */
/*---------------------------------------------------------------------*/
int numeric(char *s)
{
 int bad,n;

 bad = 0;
 for(n=0; *s; s++) if(isdigit(*s)) n = 1;
                   else switch (*s)
                        {
                         case 'e': ;
                         case 'E': if(n==0) bad = 1;
                         case '-': ;
                         case '+': ;
```

```
                        case '.': break;
                        default : n = 0;
                       }
 if (bad) return 0;
 return n;
}
/*---------------------------------------------------------------------*/
char *trim(char *s)
{
  /* local variables */
    int i,j;
    int iblank;       /* count how many blanks have been printed */
    bool print_blank; /* flag on if a blank can be printed */
    bool separator;   /* flag on if is to handle blank in a
                         special way   */
    char c,*t,*r;

 if(s == (char *) NULL) return ((char *) NULL);

 /* set initial values */
 t           = GetBlock(strlen(s)+1);
 memset(t,'\0',strlen(s)+1);
 iblank      = 0;
 separator   = FALSE;
 print_blank = FALSE;
 i = j       = -1;


 while((c=s[++i]) != '\0')
  { switch (c)
    { case '\t': ;     /* consider as "blank" avoiding "break" */
      case ' ' : iblank++;              /* count blanks */
                 if(j!=-1) separator = TRUE;
                 break;
      default:   if(separator) t[++j] = ' ';
                 t[++j] = c;
                 separator   = FALSE;
                 print_blank = TRUE;
                 iblank      = 0;
                 break;
    }
  }
 r = CopyString(t);
 FreeBlock(t,strlen(s)+1);
 return r;
```

```
}
/*-------------------------------------------------------------------*/
char *Lower(char *s)
{
 int i = 0;
 char *t;

 if(s == (char *) NULL) return ((char *) NULL);

 t = GetBlock(strlen(s) + 1);
 memset(t,'\0',strlen(s)+1);
 while(s[i] != '\0')
    { t[i] = tolower(s[i]);
      i++;
    }

 return t;
}
/*-------------------------------------------------------------------*/
char *Upper(char *s)
{
 int i = 0;
 char *t;

 if(s == (char *) NULL) return ((char *) NULL);

 t = GetBlock(strlen(s) + 1);
 memset(t,'\0',strlen(s)+1);
 while(s[i] != '\0')
    { t[i] = toupper(s[i]);
      i++;
    }

 return t;
}
/*-------------------------------------------------------------------*/
char *LocateHDR(FILE *fin, char *sl)
{
 char *t,*s,*r, *alf;
 int i,j,k,load,alreadyAdded;

 if(sl == (char *) NULL) return ((char *) NULL);

 alf = GetBlock(MAXV);
/*
```

```
 FreeBlock(s, strlen(s)+1);
         here we free before because it is going to be returned.
            I don't know, it must be checked
*/

 s = GetBlock(MAXC);
 t = CopyString("0");
 r = CopyString("0");

 do
  {
   memset(s,'\0',MAXC);
   if(fgets(s,MAXC,fin) == NULL) { FreeBlock(s, MAXC);
                                   FreeBlock(alf, MAXV);
                                   FreeBlock(r, strlen(r)+1);
                                   FreeBlock(t, strlen(t)+1);
                                   return ((char *)NULL);
                                  }

   memset(alf,'\0',MAXV);
   FreeBlock(t, strlen(t)+1);
   FreeBlock(r, strlen(r)+1);
   j=k=load=0;
   alreadyAdded=1;
   i = -1;

   do
    {
     i++;
     switch (s[i])
      {
       case '\0': ;
       case '\t': ;
       case '\n': break;
       case '\'': ;
       case '"' : load = (load+1) % 2;
                  if(load) i++;
                    else alreadyAdded = 0;
       case ' ' : if(!alreadyAdded)
                    { j++;
                      k = 0;
                      alreadyAdded++;
                    }
                  if(load) alf[k++] = s[i];
                  break;
```

```
      default  : alf[k++]  = s[i];
                 if(!load) alreadyAdded = 0;
      }
     } while((s[i] != '\n') && (s[i] != '\0') && (j == 0));

  } while (strcmp((r=trim(sl)),(t=trim(alf))) != 0);

 FreeBlock(t, strlen(t)+1);
 FreeBlock(r, strlen(r)+1);
 t = CopyString(s);
 FreeBlock(s, MAXC);
 FreeBlock(alf, MAXV);
 return (t);
}
/*-------------------------------------------------------*/
void GetRecordNum(FILE *fin, char *sl, float values[], int *n)
{
 char *s, *alf[MAXV];
 int  i,j,k,nv,nalf,limalf[MAXV],ierro;
 float vnum[MAXV];

 if(sl == (char *) NULL) return;
 *n = 0;

 if((s = LocateHDR(fin,sl)) != NULL)
   { for(j=0; j<MAXV; j++) alf[j] = GetBlock(MAXV);
     foliC(s,0,MAXC,vnum,&nv,alf,&nalf,limalf,&ierro);
     *n = (int) vnum[0];
     nv = (int) *n / 4;
     i  = 0;
     for(k=0; k < nv; k++)
       { for(j=0; j < 4; j++) fscanf(fin,"%f",&values[j+i]);
         i += 4;
       }
     nv = *n % 4;
     for(j=0; j < nv; j++) fscanf(fin,"%f",&values[j+i]);

     FreeBlock(s, strlen(s)+1);
     for(j=0; j<MAXV; j++) FreeBlock(alf[j], MAXV);
   }

 return;
}
/*-------------------------------------------------------*/
void FOLIGetRecordNum(FILE *fin, char *sl, float values[], int *n)
```

```
{
 char *s, *alf[MAXV];
 int  i,j,nv,nalf,limalf[MAXV],ierro;
 float vnum[MAXV];

 if(sl == (char *) NULL) return;
 *n = 0;

 if((s = LocateHDR(fin,sl)) != NULL)
   { for(j=0; j<MAXV; j++) alf[j] = GetBlock(MAXV);
     foliC(s,0,MAXC,vnum,&nv,alf,&nalf,limalf,&ierro);
     *n = (int) vnum[0];
     i  = 0;
     FreeBlock(s, strlen(s)+1);
     s = GetBlock(MAXC+1);
     do { (void) fgets(s,MAXC,fin);
          foliC(s,0,MAXC,vnum,&nv,alf,&nalf,limalf,&ierro);
          for(j=0; j<nv; j++) values[j+i] = vnum[j];
          i += nv;
        } while (i < *n);

     FreeBlock(s, MAXC+1);
     for(j=0; j<MAXV; j++) FreeBlock(alf[j], MAXV);
   }

 return;
}
/*-------------------------------------------------------------*/
void GetRecordAlpha(FILE *fin, char *sl, char *alpha[], int *n)
{
 char *s, *alf[MAXV];
 int i,j,nalf,nv,limalf[MAXV],ierro;
 float vnum[MAXV];

 if(sl == (char *) NULL) return;
 *n = 0;

 if((s = LocateHDR(fin,sl)) != NULL)
   { for(j=0; j<MAXV; j++) alf[j] = GetBlock(MAXV);

     foliC(s,0,MAXC,vnum,&nv,alf,&nalf,limalf,&ierro);
     *n = (int) vnum[0];
     i  = 0;
     FreeBlock(s, strlen(s)+1);
     s = GetBlock(MAXC+1);
```

```
    do { (void) fgets(s,MAXC,fin);
         foliC(s,0,MAXC,vnum,&nv,alf,&nalf,limalf,&ierro);
         for(j=0; j<nalf; j++) alpha[i+j] = trim(alf[j]);
         i += nalf;
       } while (i < *n);

    FreeBlock(s, MAXC+1);
    for(j=0; j<MAXV; j++) FreeBlock(alf[j], MAXV);
   }

 return;
}
/*----------------------------------------------------------*/
```

2. DATA SET LISTINGS

2.1 Modified TDATA3.DATA File

```
-- ================================================================
-- ORIGINAL DATA SET TDATA3 PROBLEM STATEMENT
-- THIS PROBLEM ILLUSTRATES THE USE OF GROUP AND FIELD PRODUCTION
-- AND INJECTION CONTROLS IN A MULTI-LEVEL HIERARCHY.  A PLATFORM
-- PRODUCES AT A TARGET LIQUID RATE OF 20,000 STB/DAY, WHILE A SECOND
-- PLATFORM MAKES UP THE FIELD'S OIL PRODUCTION RATE TO A TARGET OF
-- 40,000 STB/DAY.  CERTAIN GROUPS RE-INJECT HALF THEIR PRODUCED GAS,
-- AND WATER IS INJECTED TO MAKE UP THE TOTAL VOIDAGE REPLACEMENT
-- FRACTION TO 0.8 .  WELLS ARE PLACED IN A DRILLING QUEUE AND
-- OPENED AUTOMATICALLY TO MAINTAIN TARGET RATES.
-- ================================================================NRUNSPEC

DIVIX   NDIVIY   NDIVIZ   QRDIAL
   9       9       3       F /
- OIL   WATER   GAS   DISSOLVED GAS   VAPOURISED OIL
  T     T     T        T            F /
- UNIT CONVENTION -  'METRIC', 'FIELD', OR 'LAB'
   'FIELD'  /
- NRPVT    NPPVT    NTPVT
   15     15       1 /
- NSSFUN   NTSFUN   QDIRKR    QREVKR
   16       1   /
- NDRXVD    NTEQUL    NDPRVD
    /
- NTFIP
    3   /
- NWMAXZ   NCWMAX   NGMAXZ   NWGMAX
   30      3       10      10 /
- QEXGOP   NWFRIC   NUPCOL
    /
- MXMFLO   MXMTHP   MXMWCT   MXMGFR   MXMALQ   NMMVFT
    /
- MXSFLO   MXSTHP   NMSVFT
   /
- NANAQU   NCAMAX   NIFTBL   NRIFTB
   /
```

```
- IDATE    ZMNTH    IYEAR
   1      'JAN'     1983  /
- QSOLVE   NSTACK QFMTOU  QFMTIN  QUNOUT  ...
   F      1*       F     1*       F    /
SAVE
/

GRID    ============================================================
--      THIS SECTION SPECIFIES THE GEOMETRY OF THE 9 X 9 X 3 GRID, AND
--      SETS THE ROCK POROSITIES AND PERMEABILITIES.
--      ------------------------------------------------------------


-- THE CELL DIMENSIONS ( DX, DY, DZ ), PERMEABILITIES AND POROSITIES
-- ARE THE SAME FOR EACH CELL.  THE CELL TOP DEPTHS ARE SET FOR THE TOP
-- LAYER ONLY. THE FIELD SLOPES DOWN IN THE Y-DIRECTION.  USING MULTX
-- WE SET UP TWO VERTICAL IMPERMEABLE BARRIERS, DIVIDING THE FIELD INTO
-- THREE ISOLATED SEGMENTS.

--   ARRAY   VALUE    ------ BOX ------
EQUALS
    'DX'     1000    /
    'DY'     1000    /
    'DZ'     20      /
    'PERMX'  300     /
    'PERMY'  300     /
    'PERMZ'  30      /
    'PORO'   0.3     /
    'TOPS'   7000     1  9  1  1  1  1   /
    'TOPS'   7020     1  9  2  2  1  1   /
    'TOPS'   7040     1  9  3  3  1  1   /
    'TOPS'   7060     1  9  4  4  1  1   /
    'TOPS'   7080     1  9  5  5  1  1   /
    'TOPS'   7100     1  9  6  6  1  1   /
    'TOPS'   7120     1  9  7  7  1  1   /
    'TOPS'   7140     1  9  8  8  1  1   /
    'TOPS'   7160     1  9  9  9  1  1   /
    'MULTX'  0        3  3  1  9  1  3   /
    'MULTX'  0        6  6  1  9  1  3   /
/

RPTGRID
  0 /

PROPS   ============================================================
--      THE PVT PROPERTIES AND ROCK-FLUID DATA ARE THE SAME AS IN THE
--      CHAPPELEAR PROBLEM.  PLEASE REFER TO SAMPLE DATA SET 2 FOR
--      COMMENTS.
--      ------------------------------------------------------------

SWFN

  0.22  0     7
```

```
 0.3   0.07  4
 0.4   0.15  3
 0.5   0.24  2.5
 0.6   0.33  2
 0.8   0.65  1
 0.9   0.83  0.5
 1     1     0     /

SGFN

 0     0       0
 0.04  0       0.2
 0.1   0.022   0.5
 0.2   0.1     1
 0.3   0.24    1.5
 0.4   0.34    2
 0.5   0.42    2.5
 0.6   0.5     3
 0.7   0.8125  3.5
 0.78  1       3.9    /

SOF3

0        0         0
0.2      0         0
0.38     0.00432   0
0.4      0.0048    0.004
0.48     0.05288   0.02
0.5      0.0649    0.036
0.58     0.11298   0.1
0.6      0.125     0.146
0.68     0.345     0.33
0.7      0.4       0.42
0.74     0.7       0.6
0.78     1         1         /

PVTW
3000  1.00341  3.0D-6  0.96  0  /

ROCK
3600    4.0D-6   /

DENSITY
45   63.02   0.0702  /

PVDG
 400  5.9   0.013
 800  2.95  0.0135
1200  1.96  0.014
1600  1.47  0.0145
2000  1.18  0.015
2400  0.98  0.0155
2800  0.84  0.016
```

```
3200  0.74  0.0165
3600  0.65  0.017
4000  0.59  0.0175
4400  0.54  0.018
4800  0.49  0.0185
5200  0.45  0.019
5600  0.42  0.0195 /


PVTO
  0.165     400   1.012    1.17  /
  0.335     800   1.0255   1.14  /
  0.500    1200   1.038    1.11  /
  0.665    1600   1.051    1.08  /
  0.828    2000   1.063    1.06  /
  0.985    2400   1.075    1.03  /
  1.130    2800   1.087    1.00  /
  1.270    3200   1.0985   0.98  /
  1.390    3600   1.11     0.95  /
  1.500    4000   1.12     0.94  /
  1.600    4400   1.13     0.92  /
  1.676    4800   1.14     0.91  /
  1.750    5200   1.148    0.9   /
  1.810    5600   1.155    0.89
           6000   1.1504   0.89
           6400   1.1458   0.89
           6800   1.1412   0.89
           7200   1.1367   0.89  /
/

RPTPROPS
  8*0  /

REGIONS    ====================================================
--         THERE ARE THREE FLUIDS-IN-PLACE REGIONS, SEPARATED BY THE
--         TWO VERTICAL IMPERMEABLE BARRIERS.
--         ------------------------------------------------------------

--    ARRAY    VALUE     ------ BOX ------

EQUALS
    'FIPNUM'   1     1  3  1  9  1  3   /
    'FIPNUM'   2     4  6  1  9  1  3   /
    'FIPNUM'   3     7  9  1  9  1  3   /
/

-- PRINT FIPNUM ARRAY

RPTREGS
  0   0   0   0   0   /

SOLUTION    ============================================================
--          THIS SECTION DEFINES THE INITIAL CONDITIONS IN THE FIELD.
```

```
--        ------------------------------------------------------------

-- SET THE EQUILIBRATION DATA
--
--  DATUM  DATUM   OWC    OWC    GOC    GOC   RSVD   RVVD   SOLN
--  DEPTH  PRESS  DEPTH   PCOW  DEPTH   PCOG  TABLE  TABLE  METH

EQUIL
    7010   4000   9000    0.0   7010    0.0    0      0       5  /

-- SWITCH ON OUTPUT OF INITIAL CONDITIONS

RPTSOL
  6*0 4 4*0 /


SCHEDULE  ============================================================
--       THIS SECTION DEFINES THE OPERATIONS TO BE SIMULATED.
--       ------------------------------------------------------------


-- SET CONTROLS ON OUTPUT AT EACH REPORT TIME

RPTSCHED
 0   0   0   0   0   0   4  8*0 /

TSTEP
0.001
/
END
seq
```

2.2 ROOT.DATA File

```
LOAD
  'TDATA3'
/
RESTART
  'TDATA3' 0
/

SUMMARY   ============================================================
--       THIS SECTION SPECIFIES DATA TO BE WRITTEN TO THE SUMMARY
--       FILES, WHICH MAY LATER BE USED WITH THE GRAPHICS PACKAGE.
--
RUNSUM
RPTONLY

-- OIL PRODUCTION RATE FROM FIELD AND EACH PLATFORM AND SUB-PLATFORM
```

```
FOPR
WOPR
/

-- SWITCH ON REPORT OF SUMMARY FILE CONTENTS
RPTSMRY
 1
/

SCHEDULE ============================================================
--       THIS SECTION DEFINES THE OPERATIONS TO BE SIMULATED.


-- SET CONTROLS ON OUTPUT AT EACH REPORT TIME

RPTSCHED
15*0
/

-- SET SPECIFICATION DATA FOR THE WELLS. (Note commented wells)
--
--  WELL    GROUP    LOCATION    BHP   PREF   DRAIN
--  NAME    NAME       I  J     DEPTH  PHAS    RAD

WELSPECS
--  'PA1'   'GR-A1'     3  2     1*    'OIL'  -1.0  /
--  'PA2'   'GR-A1'     1  3     1*    'OIL'  -1.0  /
    'PA3'   'GR-A1'     2  4     1*    'OIL'  -1.0  /
    'PA4'   'GR-A2'     2  7     1*    'OIL'  -1.0  /
    'PA5'   'GR-A2'     1  6     1*    'OIL'  -1.0  /
--  'PA6'   'GR-A2'     3  5     1*    'OIL'  -1.0  /
    'PB1'   'GR-B1'     6  2     1*    'OIL'  -1.0  /
    'PB2'   'GR-B1'     4  3     1*    'OIL'  -1.0  /
--  'PB3'   'GR-B1'     5  4     1*    'OIL'  -1.0  /
--  'PB4'   'GR-B2'     5  7     1*    'OIL'  -1.0  /
    'PB5'   'GR-B2'     4  6     1*    'OIL'  -1.0  /
    'PB6'   'GR-B2'     6  5     1*    'OIL'  -1.0  /
    'PC1'   'GR-C1'     9  2     1*    'OIL'  -1.0  /
--  'PC2'   'GR-C1'     7  3     1*    'OIL'  -1.0  /
    'PC3'   'GR-C1'     8  4     1*    'OIL'  -1.0  /
    'PC4'   'GR-C2'     8  7     1*    'OIL'  -1.0  /
--  'PC5'   'GR-C2'     7  6     1*    'OIL'  -1.0  /
--  'PC6'   'GR-C2'     9  5     1*    'OIL'  -1.0  /
    'IGA'   'GR-A1'     1  1     1*    'GAS'  -1.0  /
    'IGC'   'GR-C1'     7  1     1*    'GAS'  -1.0  /
    'IWA1'  'GR-A2'     1  9     1*    'WAT'  -1.0  /
    'IWA2'  'GR-A2'     2  9     1*    'WAT'  -1.0  /
    'IWA3'  'GR-A2'     3  9     1*    'WAT'  -1.0  /
    'IWB1'  'GR-B2'     4  9     1*    'WAT'  -1.0  /
    'IWB2'  'GR-B2'     5  9     1*    'WAT'  -1.0  /
    'IWB3'  'GR-B2'     6  9     1*    'WAT'  -1.0  /
    'IWC1'  'GR-C2'     7  9     1*    'WAT'  -1.0  /
```

```
   'IWC2'  'GR-C2'     8  9     1*    'WAT'  -1.0  /
   'IWC3'  'GR-C2'     9  9     1*    'WAT'  -1.0  /
/

-- SET UP THE GROUP TREE STRUCTURE.  PLAT-A AND PLAT-B ARE TWO
-- PLATFORMS.  SP-B AND SP-C ARE SUB-PLATFORMS SUBORDINATE TO
-- PLAT-B.  WELL-GROUPS GR-A1 AND GR-A2 ARE SUBORDINATE TO PLAT-A,
-- GR-B1 AND GR-B2 TO SP-B, AND GR-C1 AND GR-C2 TO SP-C.

GRUPTREE
 'GR-A1'  'PLAT-A'  /
 'GR-A2'  'PLAT-A'  /
 'SP-B'   'PLAT-A'  /
 'SP-C'   'PLAT-A'  /
 'GR-B1'  'PLAT-A'   /
 'GR-B2'  'PLAT-A'   /
 'GR-C1'  'PLAT-A'   /
 'GR-C2'  'PLAT-A'   /
 /


-- SET THE WELL COMPLETION DATA, TO SPECIFY ALL THE WELL CONNECTIONS.
-- THE PRODUCERS ARE COMPLETED IN LAYERS 1 - 3 , THE GAS INJECTORS IN
-- LAYER 1 ONLY, AND THE WATER INJECTORS IN LAYERS 2 AND 3.  EACH WELL
-- IS COMPLETED IN ONE VERTICAL COLUMN OF GRID BLOCKS, SO THE
-- CONNECTIONS HAVE THE I- AND J- LOCATIONS THAT WERE SET IN WELSPECS.
-- THE CONNECTION TRANSMISSIBILITY FACTORS ARE CALCULATED INTERNALLY.
-- WELL NAME ROOTS ARE USED TO REDUCE THE VOLUME OF DATA.
--
--  WELL    -LOCATION-  OPEN/  SAT  CONN  BORE
--  NAME    I  J K1-K2  SHUT   TAB  FACT  DIAM

COMPDAT
  'P*'     0  0  1  3  'OPEN'  0    0.0  0.333  /
  'IG*'    0  0  1  1  'OPEN'  0    0.0  0.333  /
  'IW*'    0  0  2  3  'OPEN'  0    0.0  0.333  /
/

-- SET THE CONTROL DATA FOR ALL THE PRODUCERS.  THE WELLS ARE PLACED
-- UNDER GROUP CONTROL, AS EACH GROUP WILL INITIALLY BE UNDER GROUP
-- OR FIELD CONTROL.  EACH WELL IS GIVEN A MAXIMUM GAS RATE OF 30,000
-- MSCF/DAY, AND A MINIMUM BHP OF 2,000 PSIA.  A WELL NAME ROOT IS USED
-- TO DECLARE ALL WELLS  SHUT, THEN SELECTED WELLS ARE OPENED.
--
--  WELL    OPEN/  CNTL   OIL   WATER  GAS   LIQU  VOID  BHP
--  NAME    SHUT   MODE   RATE  RATE   RATE  RATE  RATE

WCONPROD
  'P*'    'SHUT' 'GRUP'  1500.   1*    1*     2*       2000 /
/

WELOPEN
 'PA4'  /
```

```
/

-- SET THE CONTROL DATA FOR INJECTION WELLS.  THEY ARE ALL PLACED
-- UNDER GROUP CONTROL AND SUBJECT TO A MAXIMUM BHP OF 5500 PSIA.
-- THREE WATER INJECTORS ARE INITIALLY CLOSED.
--
-- WELL   INJ   OPEN/  CNTL   FLOW   RESV  BHP
-- NAME   TYPE  SHUT   MODE   RATE   RATE

WCONINJE
  'IG*'     'GAS'   'OPEN' 'GRUP'   2*       5500  /
  'IW*'     'WAT' 'OPEN' 'GRUP'   2*       5500  /
  'IWA2'  'WAT' 'SHUT' 'GRUP'   2*       5500  /
  'IWB2'  'WAT' 'SHUT' 'GRUP'   2*       5500  /
  'IWC2'  'WAT' 'SHUT' 'GRUP'   2*       5500  /
/

-- PLACE THE WELLS THAT ARE INITIALLY CLOSED IN THE DRILLING QUEUE.
-- THESE WILL BE OPENED WHEN NECESSARY TO MAINTAIN GROUP AND FIELD
-- PRODUCTION AND INJECTION TARGETS.

QDRILL
'PB1'   'PC1'   'PA3'  'PB2'  'PC3' 'PA5' 'PB5'   'PC4' 'PB6' 'IWA2'  'IWB2'  'IWC2'
/

-- SET AN ECONOMIC LIMIT OF   5 STB/DAY OF OIL FOR THE PRODUCERS.  THEY
-- WILL BE SHUT IF THEIR PRODUCTION RATE FALLS BELOW THIS LIMIT.
-- THE WELLS WILL BE WORKED OVER IF THEIR WATER CUT EXCEEDS 0.7 OR IF
-- THEIR GOR EXCEEDS 10.0 .
--
--  WELL   MIN   MIN   MAX   MAX   MAX   WORK   END
--  NAME   OIL   GAS   WCT   GOR   WGR   OVER   RUN

WECON
    'P*'    5     1*   0.7   10.0   1*   'CON'  'NO'  /
/

-- SET THE GROUP INJECTION CONTROL DATA.  PLAT-A AND SP-C EACH
-- RE-INJECT HALF THEIR GAS PRODUCTION.  PLAT-A AND PLAT-B ALSO
-- INJECT WATER TO MAKE UP THEIR TOTAL VOIDAGE REPLACEMENT FRACTION
-- TO 0.8 .  THE WATER INJECTED BY PLAT-B IS SHARED BETWEEN SP-B
-- AND SP-C IN PROPORTION TO THEIR NET VOIDAGE (PRODUCTION VOIDAGE
-- MINUS RESERVOIR VOLUME OF INJECTED GAS).
--
-- GROUP    PHASE    CNTL   SRAT VRAT REIN VREP AVAIL GUID  GRAT
-- NAME     INJ     MODE                  FLD  RATE  DEFN

GCONINJE
  'PLAT-A'  'GAS'   'REIN'   1*   1*   0.5 /
  'SP-C'    'GAS'   'REIN'   1*   1*   0.5 /
  'PLAT-*'  'WAT'   'VREP'   1*   1*   1*  0.8  /
  'SP-*'    'WAT'   'FLD'    1*   1*   1*   1*   1*   1*   'NETV'  /
/
```

```
-- STOP THE RUN IF ALL THE PRODUCERS IN PLAT-A, SP-B OR SP-C ARE SHUT
--
-- GROUP   MIN   MIN   MAX   MAX   MAX   WORK   END
-- NAME    ORAT  GRAT  WCT   GOR   WGR   OVER   RUN

GECON
  'PLAT-A'   6*                      'YES'  /
  'SP-*'     6*                    'YES'  /
/

-- IF AN ECONOMIC LIMIT OR A GROUP RATE LIMIT IS BROKEN BY MORE THAN
-- TWENTY PERCENT, THE TIME STEP WILL BE RE-CALCULATED.

WLIMTOL
 0.2  /

-- THE GROUP/FIELD PRODUCTION AND INJECTION TARGETS SHOULD BE MET
-- WITHIN A TOLERANCE OF TWO PERCENT.

GCONTOL
 0.02 /

-- SET THE GROUP PRODUCTION CONTROL DATA.  THE FIELD HAS A TARGET OIL
-- PRODUCTION RATE OF 40,000 STB/DAY.  PLAT-A HAS ITS OWN TARGET LIQUID
-- PRODUCTION RATE OF 20,000 STB/DAY, INDEPENDENT OF THE FIELD TARGET.
-- PLAT-A, SP-B AND SP-C EACH HAVE A MAXIMUM GAS RATE OF 70,000 MSCF/DAY,
-- WITH THEIR WORST-OFFENDING WELLS BEING WORKED OVER IF THIS IS EXCEEDED.
--
-- GROUP     CNTL   OIL   WATER  GAS   LIQU   LIMIT    AVAIL
-- NAME      MODE   RATE  RATE   RATE  RATE   ACTION   FLD
```

2.3 TRIAL.DATA Sample File

```
-- periodo 504.000000 tstart 137.325516 dev_time 225.239578
LOAD
  'TDATA3'
/
RESTART
  'TDATA3' 0
/

SUMMARY  ============================================================
--       THIS SECTION SPECIFIES DATA TO BE WRITTEN TO THE SUMMARY
--       FILES, WHICH MAY LATER BE USED WITH THE GRAPHICS PACKAGE.
--       ------------------------------------------------------------
RUNSUM
RPTONLY
```

```
-- OIL PRODUCTION RATE FROM FIELD AND EACH PLATFORM AND SUB-PLATFORM

FOPR
WOPR
/

-- SWITCH ON REPORT OF SUMMARY FILE CONTENTS
RPTSMRY
 1
/

SCHEDULE ============================================================
--       THIS SECTION DEFINES THE OPERATIONS TO BE SIMULATED.
--       ---------------------------------------------------------


-- SET CONTROLS ON OUTPUT AT EACH REPORT TIME

RPTSCHED
15*0
/

-- SET SPECIFICATION DATA FOR THE WELLS.
--
--  WELL    GROUP    LOCATION    BHP   PREF   DRAIN
--  NAME    NAME       I  J     DEPTH  PHAS    RAD

WELSPECS
--  'PA1'   'GR-A1'     3  2     1*    'OIL'  -1.0  /
--  'PA2'   'GR-A1'     1  3     1*    'OIL'  -1.0  /
    'PA3'   'GR-A1'     2  4     1*    'OIL'  -1.0  /
    'PA4'   'GR-A2'     2  7     1*    'OIL'  -1.0  /
    'PA5'   'GR-A2'     1  6     1*    'OIL'  -1.0  /
--  'PA6'   'GR-A2'     3  5     1*    'OIL'  -1.0  /
    'PB1'   'GR-B1'     6  2     1*    'OIL'  -1.0  /
    'PB2'   'GR-B1'     4  3     1*    'OIL'  -1.0  /
--  'PB3'   'GR-B1'     5  4     1*    'OIL'  -1.0  /
--  'PB4'   'GR-B2'     5  7     1*    'OIL'  -1.0  /
    'PB5'   'GR-B2'     4  6     1*    'OIL'  -1.0  /
    'PB6'   'GR-B2'     6  5     1*    'OIL'  -1.0  /
    'PC1'   'GR-C1'     9  2     1*    'OIL'  -1.0  /
--  'PC2'   'GR-C1'     7  3     1*    'OIL'  -1.0  /
    'PC3'   'GR-C1'     8  4     1*    'OIL'  -1.0  /
    'PC4'   'GR-C2'     8  7     1*    'OIL'  -1.0  /
--  'PC5'   'GR-C2'     7  6     1*    'OIL'  -1.0  /
--  'PC6'   'GR-C2'     9  5     1*    'OIL'  -1.0  /
    'IGA'   'GR-A1'     1  1     1*    'GAS'  -1.0  /
    'IGC'   'GR-C1'     7  1     1*    'GAS'  -1.0  /
    'IWA1'  'GR-A2'     1  9     1*    'WAT'  -1.0  /
    'IWA2'  'GR-A2'     2  9     1*    'WAT'  -1.0  /
    'IWA3'  'GR-A2'     3  9     1*    'WAT'  -1.0  /
    'IWB1'  'GR-B2'     4  9     1*    'WAT'  -1.0  /
```

```
   'IWB2'  'GR-B2'     5  9     1*   'WAT'  -1.0  /
   'IWB3'  'GR-B2'     6  9     1*   'WAT'  -1.0  /
   'IWC1'  'GR-C2'     7  9     1*   'WAT'  -1.0  /
   'IWC2'  'GR-C2'     8  9     1*   'WAT'  -1.0  /
   'IWC3'  'GR-C2'     9  9     1*   'WAT'  -1.0  /
/

-- SET UP THE GROUP TREE STRUCTURE.  PLAT-A AND PLAT-B ARE TWO
-- PLATFORMS.  SP-B AND SP-C ARE SUB-PLATFORMS SUBORDINATE TO
-- PLAT-B.  WELL-GROUPS GR-A1 AND GR-A2 ARE SUBORDINATE TO PLAT-A,
-- GR-B1 AND GR-B2 TO SP-B, AND GR-C1 AND GR-C2 TO SP-C.

GRUPTREE
 'GR-A1'  'PLAT-A'  /
 'GR-A2'  'PLAT-A'  /
 'SP-B'   'PLAT-A'  /
 'SP-C'   'PLAT-A'  /
 'GR-B1'  'PLAT-A'   /
 'GR-B2'  'PLAT-A'   /
 'GR-C1'  'PLAT-A'   /
 'GR-C2'  'PLAT-A'   /
 /


-- SET THE WELL COMPLETION DATA, TO SPECIFY ALL THE WELL CONNECTIONS.
-- THE PRODUCERS ARE COMPLETED IN LAYERS 1 - 3 , THE GAS INJECTORS IN
-- LAYER 1 ONLY, AND THE WATER INJECTORS IN LAYERS 2 AND 3.  EACH WELL
-- IS COMPLETED IN ONE VERTICAL COLUMN OF GRID BLOCKS, SO THE
-- CONNECTIONS HAVE THE I- AND J- LOCATIONS THAT WERE SET IN WELSPECS.
-- THE CONNECTION TRANSMISSIBILITY FACTORS ARE CALCULATED INTERNALLY.
-- WELL NAME ROOTS ARE USED TO REDUCE THE VOLUME OF DATA.
--
--  WELL    -LOCATION-  OPEN/  SAT  CONN  BORE
--  NAME    I  J K1-K2  SHUT   TAB  FACT  DIAM

COMPDAT
  'P*'     0  0  1  3  'OPEN'  0    0.0  0.333  /
  'IG*'    0  0  1  1  'OPEN'  0    0.0  0.333  /
  'IW*'    0  0  2  3  'OPEN'  0    0.0  0.333  /
/

-- SET THE CONTROL DATA FOR ALL THE PRODUCERS.  THE WELLS ARE PLACED
-- UNDER GROUP CONTROL, AS EACH GROUP WILL INITIALLY BE UNDER GROUP
-- OR FIELD CONTROL.  EACH WELL IS GIVEN A MAXIMUM GAS RATE OF 30,000
-- MSCF/DAY, AND A MINIMUM BHP OF 2,000 PSIA.  A WELL NAME ROOT IS USED
-- TO DECLARE ALL WELLS  SHUT, THEN SELECTED WELLS ARE OPENED.
--
--  WELL    OPEN/ CNTL   OIL   WATER  GAS   LIQU  VOID  BHP
--  NAME    SHUT  MODE   RATE  RATE   RATE  RATE  RATE

WCONPROD
  'P*'    'SHUT' 'GRUP'  1500.   1*    1*     2*        2000 /
/
```

```
WELOPEN
 'PA4'  /
/

-- SET THE CONTROL DATA FOR INJECTION WELLS.  THEY ARE ALL PLACED
-- UNDER GROUP CONTROL AND SUBJECT TO A MAXIMUM BHP OF 5500 PSIA.
-- THREE WATER INJECTORS ARE INITIALLY CLOSED.
--
-- WELL   INJ   OPEN/  CNTL   FLOW   RESV  BHP
-- NAME   TYPE  SHUT   MODE   RATE   RATE

WCONINJE
  'IG*'   'GAS' 'OPEN' 'GRUP'   2*        5500  /
  'IW*'   'WAT' 'OPEN' 'GRUP'   2*        5500  /
  'IWA2'  'WAT' 'SHUT' 'GRUP'   2*        5500  /
  'IWB2'  'WAT' 'SHUT' 'GRUP'   2*        5500  /
  'IWC2'  'WAT' 'SHUT' 'GRUP'   2*        5500  /
/

-- PLACE THE WELLS THAT ARE INITIALLY CLOSED IN THE DRILLING QUEUE.
-- THESE WILL BE OPENED WHEN NECESSARY TO MAINTAIN GROUP AND FIELD
-- PRODUCTION AND INJECTION TARGETS.

QDRILL
'PB1'   'PC1'   'PA3'  'PB2'  'PC3' 'PA5' 'PB5'   'PC4' 'PB6'
'IWA2'  'IWB2'  'IWC2'
/

-- SET AN ECONOMIC LIMIT OF   5 STB/DAY OF OIL FOR THE PRODUCERS.  THEY
-- WILL BE SHUT IF THEIR PRODUCTION RATE FALLS BELOW THIS LIMIT.
-- THE WELLS WILL BE WORKED OVER IF THEIR WATER CUT EXCEEDS 0.7 OR IF
-- THEIR GOR EXCEEDS 10.0 .
--
--  WELL   MIN   MIN   MAX   MAX   MAX   WORK   END
--  NAME   OIL   GAS   WCT   GOR   WGR   OVER   RUN

WECON
    'P*'    5     1*   0.7   10.0   1*   'CON'  'NO'  /
/

-- SET THE GROUP INJECTION CONTROL DATA.  PLAT-A AND SP-C EACH
-- RE-INJECT HALF THEIR GAS PRODUCTION.  PLAT-A AND PLAT-B ALSO
-- INJECT WATER TO MAKE UP THEIR TOTAL VOIDAGE REPLACEMENT FRACTION
-- TO 0.8 .  THE WATER INJECTED BY PLAT-B IS SHARED BETWEEN SP-B
-- AND SP-C IN PROPORTION TO THEIR NET VOIDAGE (PRODUCTION VOIDAGE
-- MINUS RESERVOIR VOLUME OF INJECTED GAS).
--
-- GROUP    PHASE    CNTL   SRAT VRAT REIN VREP AVAIL GUID  GRAT
-- NAME     INJ      MODE                  FLD  RATE  DEFN

GCONINJE
  'PLAT-A'  'GAS'   'REIN'   1*   1*   0.5 /
```

```
  'SP-C'   'GAS'  'REIN'  1*  1*  0.5 /
  'PLAT-*' 'WAT'  'VREP'  1*  1*  1*  0.8  /
  'SP-*'   'WAT'  'FLD'   1*  1*  1*  1*  1*  1*  'NETV'  /
/

-- STOP THE RUN IF ALL THE PRODUCERS IN PLAT-A, SP-B OR SP-C ARE SHUT
--
-- GROUP   MIN   MIN   MAX   MAX   MAX   WORK   END
-- NAME    ORAT  GRAT  WCT   GOR   WGR   OVER   RUN

GECON
  'PLAT-A'  6*                    'YES'  /
  'SP-*'    6*                    'YES'  /
/

-- IF AN ECONOMIC LIMIT OR A GROUP RATE LIMIT IS BROKEN BY MORE THAN
-- TWENTY PERCENT, THE TIME STEP WILL BE RE-CALCULATED.

WLIMTOL
 0.2  /

-- THE GROUP/FIELD PRODUCTION AND INJECTION TARGETS SHOULD BE MET
-- WITHIN A TOLERANCE OF TWO PERCENT.

GCONTOL
 0.02 /

-- SET THE GROUP PRODUCTION CONTROL DATA.  THE FIELD HAS A TARGET OIL
-- PRODUCTION RATE OF 40,000 STB/DAY.  PLAT-A HAS ITS OWN TARGET LIQUID
-- PRODUCTION RATE OF 20,000 STB/DAY, INDEPENDENT OF THE FIELD TARGET.
-- PLAT-A, SP-B AND SP-C EACH HAVE A MAXIMUM GAS RATE OF 70,000 MSCF/DAY,
-- WITH THEIR WORST-OFFENDING WELLS BEING WORKED OVER IF THIS IS EXCEEDED.
--
-- GROUP     CNTL   OIL   WATER  GAS   LIQU   LIMIT    AVAIL
-- NAME      MODE   RATE  RATE   RATE  RATE   ACTION   FLD

GCONPROD
 'FIELD' 'ORAT'    33.297878 /
/
TSTEP
30
/
GCONPROD
 'FIELD' 'ORAT'    99.893634 /
/
TSTEP
30
/
GCONPROD
 'FIELD' 'ORAT'    166.489390 /
/
TSTEP
30
```

```
/
GCONPROD
 'FIELD' 'ORAT'    233.085146 /
/
TSTEP
30
/
GCONPROD
 'FIELD' 'ORAT'    299.680902 /
/
TSTEP
30
/
GCONPROD
 'FIELD' 'ORAT'    366.276658 /
/
TSTEP
30
/
GCONPROD
 'FIELD' 'ORAT'    432.872414 /
/
TSTEP
30
/

------------------------- Skipped lines.  Oil rate increases monthly up to ----------------------


GCONPROD
 'FIELD' 'ORAT'    14992.022553 /
/
TSTEP
   7.187
/
GCONPROD
 'FIELD' 'ORAT' 15000.000000 /
/
TSTEP
50*30
 /
TSTEP
50*30
 /
TSTEP
42*30
/
TSTEP
  13.047
/
END
 /
```

2.4 Optimizing Program Input File

This is the file optVar.in.

starting_time_1	1.00
development_time_1	1.00
starting_time_2	45.00
development_time_2	1.00
starting_time_3	1.00
development_time_3	45.00
tolerance	1.e-5
maximum_simulated_period	504.00
interest_year	0.10
max_capacity	15000.00
qmax1_per_well	1500.00
oil_price	15.00
platform_cost	100000000.00
rig_cost	600000.00
mob_cost	1000000.00
rig_capcty	4
offshore_well_drill_cost	1400000.00
facilities_initial_cost	400.00
overhead_factor	0.15
insurance_cost	12500.00
f	0.13
operational_cost_per_well	5000.00
handling_cost	1.00

2.5 TRIAL.FUNSMRY File (Partial)

```
'SEQHDR  '          1 'INTE'
-1749003837
'MINISTEP'          1 'INTE'
         0
'PARAMS  '         25 'REAL'
  0.00000000E+00   0.00000000E+00   0.00000000E+00   0.00000000E+00
  0.00000000E+00   0.00000000E+00   0.00000000E+00   0.00000000E+00
  0.00000000E+00   0.00000000E+00   0.00000000E+00   0.00000000E+00
  0.00000000E+00   0.00000000E+00   0.00000000E+00   0.00000000E+00
```

```
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00
'MINISTEP'         1 'INTE'
         4
'PARAMS  '        25 'REAL'
 0.30000000E+02  0.82135521E-01  0.33297878E+02  0.00000000E+00
 0.33297878E+02  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00
```

----- Skipped lines ----

```
'SEQHDR  '         1 'INTE'
-1749003803
'MINISTEP'         1 'INTE'
       164
'PARAMS  '        25 'REAL'
 0.48300000E+04  0.13223820E+02  0.10688619E+05  0.11337433E+04
 0.10256753E+04  0.13539669E+04  0.13399415E+04  0.14875211E+04
 0.13477710E+04  0.00000000E+00  0.15000000E+04  0.15000000E+04
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00
'SEQHDR  '         1 'INTE'
-1749003802
'MINISTEP'         1 'INTE'
       165
'PARAMS  '        25 'REAL'
 0.48600000E+04  0.13305955E+02  0.10755215E+05  0.11227734E+04
 0.10355911E+04  0.13816908E+04  0.13734559E+04  0.15000000E+04
 0.13417035E+04  0.00000000E+00  0.15000000E+04  0.15000000E+04
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00
'SEQHDR  '         1 'INTE'
-1749003802
'MINISTEP'         1 'INTE'
       166
'PARAMS  '        25 'REAL'
 0.48900000E+04  0.13388090E+02  0.10821811E+05  0.11346453E+04
 0.10428413E+04  0.14041510E+04  0.14008636E+04  0.15000000E+04
 0.13393092E+04  0.00000000E+00  0.15000000E+04  0.15000000E+04
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
 0.00000000E+00
```

----- Skipped lines ----

```
 'SEQHDR  '         1 'INTE'
-1749003754
 'MINISTEP'         1 'INTE'
       370
 'PARAMS  '        25 'REAL'
  0.10957187E+05  0.29999144E+02  0.12211382E+03  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.64818825E+02  0.57295002E+02
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00
 'SEQHDR  '         1 'INTE'
-1749003754
 'MINISTEP'         1 'INTE'
       371
 'PARAMS  '        25 'REAL'
  0.10987187E+05  0.30081278E+02  0.11901762E+03  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.63104309E+02  0.55913315E+02
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00
 'SEQHDR  '         1 'INTE'
-1749003753
 'MINISTEP'         1 'INTE'
       372
 'PARAMS  '        25 'REAL'
  0.11017187E+05  0.30163414E+02  0.11600484E+03  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.61434059E+02  0.54570786E+02
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00
 'SEQHDR  '         1 'INTE'
-1749003753
 'MINISTEP'         1 'INTE'
       373
 'PARAMS  '        25 'REAL'
  0.11030234E+05  0.30199135E+02  0.11468005E+03  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.60686497E+02  0.53993557E+02
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00  0.00000000E+00  0.00000000E+00  0.00000000E+00
  0.00000000E+00
```

2.6 Optimal Point Final Status

start time = 245.43 development time = 54.02

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
0	15000	15000	0	10	0	10	0	0	0	450000	5868875	5868875	5868875
1	15000	15000	0	10	0	10	0	0	0	900000	5868875	5822446	11691321
2	15000	15000	0	10	0	10	0	0	0	1350000	5868875	5776384	17467705
3	15000	15000	0	10	0	10	0	0	0	1800000	5868875	5730687	23198392
4	15000	15000	0	10	0	10	0	0	0	2250000	5868875	5685351	28883743
5	15000	15000	0	10	0	10	0	0	0	2700000	5868875	5640374	34524117
6	15000	15000	0	10	0	10	0	0	0	3150000	5868875	5595753	40119870
7	15000	15000	0	10	0	10	0	0	0	3600000	5868875	5551484	45671354
8	15000	15000	0	10	0	10	0	0	0	4050000	5868875	5507566	51178920
9	15000	15000	0	10	0	10	0	0	0	4500000	5868875	5463996	56642916
10	15000	15000	0	10	0	10	0	0	0	4950000	5868875	5420770	62063686
11	15000	15000	0	10	0	10	0	0	0	5400000	5868875	5377886	67441572
12	15000	15000	0	10	0	10	0	0	0	5850000	5868875	5335341	72776913
13	15000	15000	0	10	0	10	0	0	0	6300000	5868875	5293133	78070046
14	15000	15000	0	10	0	10	0	0	0	6750000	5868875	5251258	83321304
15	15000	15000	0	10	0	10	0	0	0	7200000	5868875	5209715	88531019
16	15000	15000	0	10	0	10	0	0	0	7650000	5868875	5168501	93699520
17	15000	15000	0	10	0	10	0	0	0	8100000	5868875	5127613	98827133
18	15000	15000	0	10	0	10	0	0	0	8550000	5868875	5087048	103914181
19	15000	15000	0	10	0	10	0	0	0	9000000	5868875	5046804	108960985
20	15000	15000	0	10	0	10	0	0	0	9450000	5868875	5006878	113967863
21	15000	15000	0	10	0	10	0	0	0	9900000	5868875	4967269	118935132
22	15000	15000	0	10	0	10	0	0	0	10350000	5868875	4927972	123863104
23	15000	15000	0	10	0	10	0	0	0	10800000	5868875	4888987	128752091
24	15000	15000	0	10	0	10	0	0	0	11250000	5868875	4850310	133602401
25	15000	15000	0	10	0	10	0	0	0	11700000	5868875	4811939	138414340
26	15000	15000	0	10	0	10	0	0	0	12150000	5868875	4773871	143188211
27	15000	15000	0	10	0	10	0	0	0	12600000	5868875	4736105	147924316
28	15000	15000	0	10	0	10	0	0	0	13050000	5868875	4698637	152622953
29	15000	15000	0	10	0	10	0	0	0	13500000	5868875	4661466	157284419
30	15000	15000	0	10	0	10	0	0	0	13950000	5868875	4624589	161909008
31	15000	15000	0	10	0	10	0	0	0	14400000	5868875	4588004	166497012
32	15000	15000	0	10	0	10	0	0	0	14850000	5868875	4551708	171048720
33	15000	15000	0	10	0	10	0	0	0	15300000	5868875	4515699	175564419
34	15000	15000	0	10	0	10	0	0	0	15750000	5868875	4479975	180044394
35	15000	15000	0	10	0	10	0	0	0	16200000	5868875	4444534	184488928
36	15000	15000	0	10	0	10	0	0	0	16650000	5868875	4409373	188898301
37	15000	15000	0	10	0	10	0	0	0	17100000	5868875	4374490	193272791
38	15000	15000	0	10	0	10	0	0	0	17550000	5868875	4339883	197612674

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
39	15000	15000	0	10	0	10	0	0	0	18000000	5868875	4305550	201918224
40	15000	15000	0	10	0	10	0	0	0	18450000	5868875	4271489	206189713
41	15000	15000	0	10	0	10	0	0	0	18900000	5868875	4237697	210427410
42	15000	15000	0	10	0	10	0	0	0	19350000	5868875	4204172	214631582
43	15000	15000	0	10	0	10	0	0	0	19800000	5868875	4170912	218802494
44	15000	15000	0	10	0	10	0	0	0	20250000	5868875	4137916	222940410
45	15000	15000	0	10	0	10	0	0	0	20700000	5868875	4105181	227045591
46	15000	15000	0	10	0	10	0	0	0	21150000	5868875	4072704	231118295
47	15000	15000	0	10	0	10	0	0	0	21600000	5868875	4040485	235158780
48	15000	15000	0	10	0	10	0	0	0	22050000	5868875	4008521	239167301
49	15000	15000	0	10	0	10	0	0	0	22500000	5868875	3976809	243144110
50	15000	15000	0	10	0	10	0	0	0	22950000	5868875	3945348	247089458
51	15000	15000	0	10	0	10	0	0	0	23400000	5868875	3914136	251003594
52	15000	15000	0	10	0	10	0	0	0	23850000	5868875	3883171	254886765
53	15000	15000	0	10	0	10	0	0	0	24300000	5868875	3852451	258739216
54	15000	15000	0	10	0	10	0	0	0	24750000	5868875	3821974	262561190
55	15000	15000	0	10	0	10	0	0	0	25200000	5868875	3791739	266352929
56	15000	15000	0	10	0	10	0	0	0	25650000	5868875	3761742	270114671
57	15000	15000	0	10	0	10	0	0	0	26100000	5868875	3731983	273846654
58	15000	15000	0	10	0	10	0	0	0	26550000	5868875	3702459	277549113
59	15000	15000	0	10	0	10	0	0	0	27000000	5868875	3673168	281222281
60	15000	15000	0	10	0	10	0	0	0	27450000	5868875	3644110	284866391
61	15000	15000	0	10	0	10	0	0	0	27900000	5868875	3615281	288481672
62	15000	15000	0	10	0	10	0	0	0	28350000	5868875	3586680	292068352
63	15000	15000	0	10	0	10	0	0	0	28800000	5868875	3558306	295626658
64	15000	15000	0	10	0	10	0	0	0	29250000	5868875	3530156	299156814
65	15000	15000	0	10	0	10	0	0	0	29700000	5868875	3502229	302659043
66	15000	15000	0	10	0	10	0	0	0	30150000	5868875	3474522	306133565
67	15000	15000	0	10	0	10	0	0	0	30600000	5868875	3447035	309580600
68	15000	15000	0	10	0	10	0	0	0	31050000	5868875	3419765	313000365
69	15000	15000	0	10	0	10	0	0	0	31500000	5868875	3392711	316393076
70	15000	15000	0	10	0	10	0	0	0	31950000	5868875	3365871	319758947
71	15000	15000	0	10	0	10	0	0	0	32400000	5868875	3339244	323098191
72	15000	15000	0	10	0	10	0	0	0	32850000	5868875	3312827	326411018
73	15000	15000	0	10	0	10	0	0	0	33300000	5868875	3286619	329697637
74	15000	15000	0	10	0	10	0	0	0	33750000	5868875	3260618	332958255
75	15000	15000	0	10	0	10	0	0	0	34200000	5868875	3234823	336193078
76	15000	15000	0	10	0	10	0	0	0	34650000	5868875	3209233	339402311
77	15000	15000	0	10	0	10	0	0	0	35100000	5868875	3183844	342586155
78	15000	15000	0	10	0	10	0	0	0	35550000	5868875	3158657	345744812
79	15000	15000	0	10	0	10	0	0	0	36000000	5868875	3133668	348878480
80	15000	15000	0	10	0	10	0	0	0	36450000	5868875	3108878	351987358
81	15000	15000	0	10	0	10	0	0	0	36900000	5868875	3084283	355071641
82	15000	15000	0	10	0	10	0	0	0	37350000	5868875	3059883	358131524
83	15000	15000	0	10	0	10	0	0	0	37800000	5868875	3035676	361167200
84	15000	15000	0	10	0	10	0	0	0	38250000	5868875	3011661	364178861

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
85	15000	15000	0	10	0	10	0	0	0	38700000	5868875	2987835	367166696
86	15000	15000	0	10	0	10	0	0	0	39150000	5868875	2964198	370130894
87	15000	15000	0	10	0	10	0	0	0	39600000	5868875	2940749	373071643
88	15000	15000	0	10	0	10	0	0	0	40050000	5868875	2917484	375989127
89	15000	15000	0	10	0	10	0	0	0	40500000	5868875	2894404	378883531
90	15000	15000	0	10	0	10	0	0	0	40950000	5868875	2871506	381755037
91	15000	15000	0	10	0	10	0	0	0	41400000	5868875	2848789	384603826
92	15000	15000	0	10	0	10	0	0	0	41850000	5868875	2826252	387430078
93	15000	15000	0	10	0	10	0	0	0	42300000	5868875	2803894	390233972
94	15000	15000	0	10	0	10	0	0	0	42750000	5868875	2781712	393015684
95	15000	15000	0	10	0	10	0	0	0	43200000	5868875	2759706	395775390
96	15000	15000	0	10	0	10	0	0	0	43650000	5868875	2737874	398513264
97	15000	15000	0	10	0	10	0	0	0	44100000	5868875	2716214	401229478
98	15000	15000	0	10	0	10	0	0	0	44550000	5868875	2694726	403924204
99	15000	15000	0	10	0	10	0	0	0	45000000	5868875	2673408	406597612
100	15000	15000	0	10	0	10	0	0	0	45450000	5868875	2652258	409249870
101	15000	15000	0	10	0	10	0	0	0	45900000	5868875	2631276	411881146
102	15000	15000	0	10	0	10	0	0	0	46350000	5868875	2610460	414491606
103	15000	15000	0	10	0	10	0	0	0	46800000	5868875	2589808	417081414
104	15000	15000	0	10	0	10	0	0	0	47250000	5868875	2569320	419650734
105	15000	15000	0	10	0	10	0	0	0	47700000	5868875	2548994	422199728
106	15000	15000	0	10	0	10	0	0	0	48150000	5868875	2528829	424728557
107	15000	15000	0	10	0	10	0	0	0	48600000	5868875	2508823	427237380
108	15000	15000	0	10	0	10	0	0	0	49050000	5868875	2488976	429726356
109	15000	15000	0	10	0	10	0	0	0	49500000	5868875	2469285	432195641
110	15000	15000	0	10	0	10	0	0	0	49950000	5868875	2449751	434645392
111	15000	15000	0	10	0	10	0	0	0	50400000	5868875	2430371	437075763
112	15000	15000	0	10	0	10	0	0	0	50850000	5868875	2411144	439486907
113	15000	15000	0	10	0	10	0	0	0	51300000	5868875	2392069	441878976
114	15000	15000	0	10	0	10	0	0	0	51750000	5868875	2373145	444252121
115	15000	15000	0	10	0	10	0	0	0	52200000	5868875	2354371	446606492
116	15000	15000	0	10	0	10	0	0	0	52650000	5868875	2335746	448942238
117	15000	15000	0	10	0	10	0	0	0	53100000	5868875	2317268	451259506
118	15000	15000	0	10	0	10	0	0	0	53550000	5868875	2298936	453558442
119	15000	15000	0	10	0	10	0	0	0	54000000	5868875	2280749	455839191
120	15000	15000	0	10	0	10	0	0	0	54450000	5868875	2262705	458101896
121	15000	15000	0	10	0	10	0	0	0	54900000	5868875	2244805	460346701
122	15000	15000	0	10	0	10	0	0	0	55350000	5868875	2227046	462573747
123	15000	15000	0	10	0	10	0	0	0	55800000	5868875	2209428	464783175
124	15000	15000	0	10	0	10	0	0	0	56250000	5868875	2191949	466975124
125	15000	15000	0	10	0	10	0	0	0	56700000	5868875	2174608	469149732
126	15000	15000	0	10	0	10	0	0	0	57150000	5868875	2157405	471307137
127	15000	15000	0	10	0	10	0	0	0	57600000	5868875	2140338	473447475
128	15000	15000	0	10	0	10	0	0	0	58050000	5868875	2123405	475570880
129	15000	15000	0	10	0	10	0	0	0	58500000	5868875	2106607	477677487
130	15000	15000	0	10	0	10	0	0	0	58950000	5868875	2089941	479767428

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
131	15000	15000	0	10	0	10	0	0	0	59400000	5868875	2073408	481840836
132	15000	15000	0	10	0	10	0	0	0	59850000	5868875	2057005	483897841
133	15000	15000	0	10	0	10	0	0	0	60300000	5868875	2040732	485938573
134	15000	15000	0	10	0	10	0	0	0	60750000	5868875	2024587	487963160
135	15000	15000	0	10	0	10	0	0	0	61200000	5868875	2008571	489971731
136	15000	15000	0	10	0	10	0	0	0	61650000	5868875	1992681	491964412
137	15000	15000	0	10	0	10	0	0	0	62100000	5868875	1976917	493941329
138	15000	15000	0	10	0	10	0	0	0	62550000	5868875	1961277	495902606
139	15000	15000	0	10	0	10	0	0	0	63000000	5868875	1945761	497848367
140	15000	15000	0	10	0	10	0	0	0	63450000	5868875	1930368	499778735
141	15000	15000	0	10	0	10	0	0	0	63900000	5868875	1915097	501693832
142	15000	15000	0	10	0	10	0	0	0	64350000	5868875	1899947	503593779
143	15000	15000	0	10	0	10	0	0	0	64800000	5868875	1884916	505478695
144	15000	15000	0	10	0	10	0	0	0	65250000	5868875	1870004	507348699
145	14926	14927	0	10	0	10	0	0	0	65697794	5839763	1846008	509194707
146	14853	14853	0	10	0	10	0	0	0	66143382	5810651	1822274	511016981
147	14779	14779	0	10	0	10	0	0	0	66586765	5781539	1798801	512815782
148	14706	14706	0	10	0	10	0	0	0	67027941	5752426	1775584	514591366
149	14632	14632	0	10	0	10	0	0	0	67466912	5723314	1752623	516343989
150	14559	14559	0	10	0	10	0	0	0	67903676	5694202	1729913	518073902
151	14485	14485	0	10	0	10	0	0	0	68338235	5665090	1707453	519781355
152	14412	14412	0	10	0	10	0	0	0	68770588	5635978	1685241	521466596
153	14338	14338	0	10	0	10	0	0	0	69200735	5606866	1663273	523129869
154	14265	14265	0	10	0	10	0	0	0	69628676	5577754	1641547	524771416
155	14191	14191	0	10	0	10	0	0	0	70054412	5548642	1620060	526391476
156	14118	14118	0	10	0	10	0	0	0	70477941	5519529	1598811	527990287
157	14044	14044	0	10	0	10	0	0	0	70899265	5490417	1577797	529568084
158	13971	13971	0	10	0	10	0	0	0	71318382	5461305	1557015	531125099
159	13897	13897	0	10	0	10	0	0	0	71735294	5432193	1536463	532661562
160	13824	13824	0	10	0	10	0	0	0	72150000	5403081	1516139	534177701
161	13750	13750	0	10	0	10	0	0	0	72562500	5373969	1496040	535673741
162	13676	13677	0	10	0	10	0	0	0	72972794	5344857	1476165	537149906
163	13603	13603	0	10	0	10	0	0	0	73380882	5315744	1456510	538606416
164	13529	13529	0	10	0	10	0	0	0	73786765	5286632	1437074	540043490
165	13456	13456	0	9	0	9	0	0	0	74190441	5263270	1419405	541462895
166	13382	13382	0	9	0	9	0	0	0	74591912	5234158	1400387	542863282
167	13309	13309	0	9	0	9	0	0	0	74991176	5205046	1381581	544244863
168	13235	13235	0	9	0	9	0	0	0	75388235	5175934	1362985	545607848
169	13162	13162	0	9	0	9	0	0	0	75783088	5146822	1344597	546952445
170	13088	13088	0	9	0	9	0	0	0	76175735	5117710	1326415	548278860
171	13015	13015	0	9	0	9	0	0	0	76566176	5088597	1308436	549587296
172	12941	12941	0	9	0	9	0	0	0	76954412	5059485	1290658	550877954
173	12868	12868	0	9	0	9	0	0	0	77340441	5030373	1273080	552151034
174	12794	12794	0	9	0	9	0	0	0	77724265	5001261	1255699	553406733
175	12721	12721	0	9	0	9	0	0	0	78105882	4972149	1238514	554645247
176	12647	12647	0	9	0	9	0	0	0	78485294	4943037	1221522	555866769

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
177	12574	12574	0	9	0	9	0	0	0	78862500	4913925	1204721	557071490
178	12500	12500	0	9	0	9	0	0	0	79237500	4884813	1188109	558259599
179	12426	12427	0	9	0	9	0	0	0	79610294	4855700	1171685	559431284
180	12353	12353	0	9	0	9	0	0	0	79980882	4826588	1155447	560586731
181	12279	12279	0	9	0	9	0	0	0	80349265	4797476	1139392	561726123
182	12206	12206	0	9	0	9	0	0	0	80715441	4768364	1123519	562849642
183	12132	12132	0	9	0	9	0	0	0	81079412	4739252	1107825	563957467
184	12059	12059	0	9	0	9	0	0	0	81441176	4710140	1092310	565049777
185	11985	11985	0	8	0	8	0	0	0	81800735	4686778	1078294	566128071
186	11912	11912	0	8	0	8	0	0	0	82158088	4657665	1063118	567191189
187	11838	11838	0	8	0	8	0	0	0	82513235	4628553	1048116	568239305
188	11765	11765	0	8	0	8	0	0	0	82866176	4599441	1033284	569272589
189	11691	11691	0	8	0	8	0	0	0	83216912	4570329	1018621	570291210
190	11618	11618	0	8	0	8	0	0	0	83565441	4541217	1004126	571295336
191	11544	11544	0	8	0	8	0	0	0	83911765	4512105	989796	572285132
192	11471	11471	0	8	0	8	0	0	0	84255882	4482993	975630	573260762
193	11397	11397	0	8	0	8	0	0	0	84597794	4453881	961626	574222388
194	11324	11324	0	8	0	8	0	0	0	84937500	4424768	947783	575170171
195	11250	11250	0	8	0	8	0	0	0	85275000	4395656	934098	576104269
196	11176	11177	0	8	0	8	0	0	0	85610294	4366544	920571	577024840
197	11103	11103	0	8	0	8	0	0	0	85943382	4337432	907199	577932039
198	11029	11029	0	8	0	8	0	0	0	86274265	4308320	893982	578826021
199	10956	10956	0	8	0	8	0	0	0	86602941	4279208	880916	579706937
200	10882	10882	0	8	0	8	0	0	0	86929412	4250096	868002	580574939
201	10809	10809	0	8	0	8	0	0	0	87253676	4220983	855236	581430175
202	10735	10735	0	8	0	8	0	0	0	87575735	4191871	842619	582272794
203	10662	10662	0	8	0	8	0	0	0	87895588	4162759	830147	583102941
204	10588	10588	0	8	0	8	0	0	0	88213235	4133647	817820	583920761
205	10515	10515	0	8	0	8	0	0	0	88528676	4104535	805636	584726397
206	10441	10441	0	7	0	7	0	0	0	88841912	4081173	794713	585521110
207	10368	10368	0	7	0	7	0	0	0	89152941	4052061	782802	586303912
208	10294	10294	0	7	0	7	0	0	0	89461765	4022949	771030	587074942
209	10221	10221	0	7	0	7	0	0	0	89768382	3993836	759395	587834337
210	10147	10147	0	7	0	7	0	0	0	90072794	3964724	747896	588582233
211	10074	10074	0	7	0	7	0	0	0	90375000	3935612	736531	589318764
212	10000	10000	0	7	0	7	0	0	0	90675000	3906500	725299	590044063
213	9926	9927	0	7	0	7	0	0	0	90972794	3877388	714199	590758262
214	9853	9853	0	7	0	7	0	0	0	91268382	3848276	703229	591461491
215	9779	9779	0	7	0	7	0	0	0	91561765	3819164	692388	592153879
216	9706	9706	0	7	0	7	0	0	0	91852941	3790051	681674	592835553
217	9632	9632	0	7	0	7	0	0	0	92141912	3760939	671087	593506640
218	9559	9559	0	7	0	7	0	0	0	92428676	3731827	660624	594167264
219	9485	9485	0	7	0	7	0	0	0	92713235	3702715	650285	594817549
220	9412	9412	0	7	0	7	0	0	0	92995588	3673603	640068	595457617
221	9338	9338	0	7	0	7	0	0	0	93275735	3644491	629972	596087589
222	9265	9265	0	7	0	7	0	0	0	93553676	-26384621	-4524662	591562927

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
223	9191	9191	0	7	0	7	0	0	0	93829412	370030	62954	591625881
224	9118	9118	0	7	0	7	0	0	0	94102941	340918	57542	591683423
225	9044	9044	0	7	0	7	0	0	0	94374265	311806	52212	591735635
226	8971	8971	0	6	0	6	0	0	0	94643382	288444	47918	591783553
227	8897	8897	0	6	0	6	0	0	0	94910294	259331	42741	591826294
228	8824	8824	0	6	0	6	0	0	0	95175000	230219	37643	591863937
229	8750	8750	0	6	0	6	0	0	0	95437500	201107	32622	591896559
230	8676	8677	0	6	0	6	0	0	0	95697794	171995	27679	591924238
231	8603	8603	0	6	0	6	0	0	0	95955882	142883	22812	591947050
232	8529	8529	0	6	0	6	0	0	0	96211765	113771	18021	591965071
233	8456	8456	0	6	0	6	0	0	0	96465441	84659	13303	591978374
234	8382	8382	0	6	0	6	0	0	0	96716912	-1744454	-271958	591706416
235	8309	8309	0	6	0	6	0	0	0	96966176	-341971	-52891	591653525
236	8235	8235	0	6	0	6	0	0	0	97213235	-371083	-56940	591596585
237	8162	8162	0	6	0	6	0	0	0	97458088	-400195	-60921	591535664
238	8088	8088	0	6	0	6	0	0	0	97700735	-429307	-64836	591470828
239	8015	8015	0	6	0	6	0	0	0	97941176	-458420	-68685	591402143
240	7941	7941	0	6	0	6	0	0	0	98179412	-487532	-72468	591329675
241	7868	7868	0	6	0	6	0	0	0	98415441	-516644	-76188	591253487
242	7794	7794	0	6	0	6	0	1	0	98649265	-3545756	-518748	590734739
243	7721	7721	0	6	0	6	0	1	0	98880882	-2574868	-373726	590361013
244	7647	7647	0	6	0	6	0	1	0	99110294	-2603980	-374961	589986052
245	7574	7574	0	6	0	6	0	1	0	99337500	-2633092	-376154	589609898
246	7639	7500	139	5	1	6	1	1	0	99566665	-2607231	-369513	589240385
247	7843	7427	417	5	1	6	1	1	0	99801956	1058247	148795	589389180
248	8047	7353	694	5	1	6	1	1	0	100043372	1139082	158894	589548074
249	8251	7279	972	5	1	6	1	1	0	100290912	1219918	168823	589716897
250	8456	7206	1250	5	1	6	1	1	0	100544578	1300753	178586	589895483
251	8660	7132	1527	5	2	7	2	1	0	100804369	1375839	187400	590082883
252	8864	7059	1805	5	2	7	2	1	0	101070285	1456674	196841	590279724
253	9068	6985	2083	5	2	7	2	1	0	101342325	1537510	206121	590485845
254	9272	6912	2360	5	2	7	2	1	0	101620491	1618345	215241	590701086
255	9476	6838	2638	5	2	7	2	1	0	101904782	1699181	224205	590925291
256	9681	6765	2916	5	2	7	2	1	0	102195199	1780016	233013	591158304
257	9885	6691	3194	5	3	8	3	1	0	102491740	1855102	240921	591399225
258	10089	6618	3471	5	3	8	3	1	0	102794406	1935937	249430	591648655
259	10293	6544	3749	5	3	8	3	1	0	103103197	2016773	257789	591906444
260	10497	6471	4027	5	3	8	3	1	0	103418113	2097608	266001	592172445
261	10701	6397	4304	5	3	8	3	1	0	103739155	2178444	274066	592446511
262	10906	6324	4582	5	4	9	4	1	0	104066321	2253529	281270	592727781
263	11110	6250	4860	5	4	9	4	1	0	104399613	2334365	289054	593016835
264	11314	6177	5137	5	4	9	4	1	0	104739029	2415200	296698	593313533
265	11518	6103	5415	5	4	9	4	1	0	105084571	2496036	304202	593617735
266	11722	6029	5693	5	4	9	4	1	0	105436238	2576871	311569	593929304
267	11926	5956	5971	4	4	8	4	1	0	105794029	2663457	319491	594248795
268	12131	5882	6248	4	5	9	5	1	0	106157946	2738542	325899	594574694

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
269	12335	5809	6526	4	5	9	5	1	0	106527988	2819378	332864	594907558
270	12539	5735	6804	4	5	9	5	1	0	106904155	2900213	339699	595247257
271	12743	5662	7081	4	5	9	5	1	0	107286447	2981049	346405	595593662
272	12947	5588	7359	4	5	9	5	1	0	107674864	3061884	352984	595946646
273	13151	5515	7637	4	6	10	6	1	0	108069406	3136970	358779	596305425
274	13356	5441	7914	4	6	10	6	1	0	108470073	3217805	365113	596670538
275	13560	5368	8192	4	6	10	6	1	0	108876866	3298641	371324	597041862
276	13764	5294	8470	4	6	10	6	1	0	109289783	3379476	377414	597419276
277	13968	5221	8748	4	6	10	6	1	0	109708825	3460311	383384	597802660
278	14172	5147	9025	4	7	11	7	1	0	110133993	3535397	388604	598191264
279	14376	5074	9303	4	7	11	7	1	0	110565285	3616233	394345	598585609
280	14581	5000	9581	4	7	11	7	1	0	111002703	3697068	399971	598985580
281	14785	4927	9858	4	7	11	7	1	0	111446245	3777904	405483	599391063
282	14989	4853	10136	4	7	11	7	0	0	111895913	5858739	623844	600014907
283	15000	4779	10221	4	7	11	7	0	0	112345913	5863125	619372	600634279
284	15000	4706	10294	4	8	12	8	0	0	112795913	5857375	613870	601248149
285	15000	4632	10368	4	8	12	8	0	0	113245913	5857375	609013	601857162
286	15000	4559	10441	4	8	12	8	0	0	113695913	5857375	604195	602461357
287	15000	4485	10515	3	8	11	8	0	0	114145913	5863125	600004	603061361
288	15000	4412	10588	3	8	11	8	0	0	114595913	5863125	595257	603656618
289	15000	4338	10662	3	9	12	9	0	0	115045913	5857375	589969	604246587
290	15000	4265	10735	3	9	12	9	0	0	115495913	5857375	585302	604831889
291	15000	4191	10809	3	9	12	9	0	0	115945913	5857375	580671	605412560
292	15000	4118	10882	3	9	12	9	0	0	116395913	5857375	576078	605988638
293	15000	4044	10956	3	9	12	9	0	0	116845913	5857375	571520	606560158
294	15000	3971	11029	3	9	12	9	0	0	117295913	5857375	566999	607127157
295	15000	3897	11103	3	10	13	10	0	0	117745913	5851625	561961	607689118
296	15000	3824	11177	3	10	13	10	0	0	118195913	5851625	557515	608246633
297	15000	3750	11250	3	10	13	10	0	0	118645913	5851625	553105	608799738
298	15000	3677	11324	3	10	13	10	0	0	119095913	5851625	548729	609348467
299	15000	3603	11397	3	10	13	10	0	0	119545913	5851625	544388	609892855
300	15000	3529	11471	3	10	13	10	0	0	119995913	5851625	540082	610432937
301	15000	3456	11544	3	10	13	10	0	0	120445913	5851625	535809	610968746
302	15000	3382	11618	3	10	13	10	0	0	120895913	5851625	531570	611500316
303	15000	3309	11691	3	10	13	10	0	0	121345913	5851625	527365	612027681
304	15000	3235	11765	3	10	13	10	0	0	121795913	5851625	523193	612550874
305	15000	3162	11838	3	10	13	10	0	0	122245913	5851625	519054	613069928
306	15000	3088	11912	3	10	13	10	0	0	122695913	5851625	514948	613584876
307	15000	3015	11985	3	10	13	10	0	0	123145913	5851625	510874	614095750
308	15000	2941	12059	2	10	12	10	0	0	123595913	5857375	507330	614603080
309	15000	2868	12132	2	10	12	10	0	0	124045913	5857375	503317	615106397
310	15000	2794	12206	2	10	12	10	0	0	124495913	5857375	499335	615605732
311	15000	2721	12279	2	10	12	10	0	0	124945913	5857375	495385	616101117
312	15000	2647	12353	2	10	12	10	0	0	125395913	5857375	491466	616592583
313	15000	2574	12427	2	10	12	10	0	0	125845913	5857375	487578	617080161
314	15000	2500	12500	2	10	12	10	0	0	126295913	5857375	483720	617563881

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
315	15000	2427	12574	2	10	12	10	0	0	126745913	5857375	479894	618043775
316	15000	2353	12647	2	10	12	10	0	0	127195913	5857375	476097	618519872
317	15000	2279	12721	2	10	12	10	0	0	127645913	5857375	472331	618992203
318	15000	2206	12794	2	10	12	10	0	0	128095913	5857375	468594	619460797
319	15000	2132	12868	2	10	12	10	0	0	128545913	5857375	464887	619925684
320	15000	2059	12941	2	10	12	10	0	0	128995913	5857375	461209	620386893
321	15000	1985	13015	2	10	12	10	0	0	129445913	5857375	457561	620844454
322	15000	1912	13088	2	10	12	10	0	0	129895913	5857375	453941	621298395
323	15000	1838	13162	2	10	12	10	0	0	130345913	5857375	450350	621748745
324	15000	1765	13235	2	10	12	10	0	0	130795913	5857375	446787	622195532
325	15000	1691	13309	2	10	12	10	0	0	131245913	5857375	443252	622638784
326	15000	1618	13382	2	10	12	10	0	0	131695913	5857375	439746	623078530
327	15000	1544	13456	2	10	12	10	0	0	132145913	5857375	436267	623514797
328	15000	1471	13529	1	10	11	10	0	0	132595913	5863125	433241	623948038
329	15000	1397	13603	1	10	11	10	0	0	133045913	5863125	429813	624377851
330	15000	1324	13677	1	10	11	10	0	0	133495913	5863125	426413	624804264
331	15000	1250	13750	1	10	11	10	0	0	133945913	5863125	423040	625227304
332	15000	1177	13824	1	10	11	10	0	0	134395913	5863125	419693	625646997
333	15000	1103	13897	1	10	11	10	0	0	134845913	5863125	416373	626063370
334	15000	1029	13971	1	10	11	10	0	0	135295913	5863125	413079	626476449
335	15000	956	14044	1	10	11	10	0	0	135745913	5863125	409811	626886260
336	15000	882	14118	1	10	11	10	0	0	136195913	5863125	406569	627292829
337	15000	809	14191	1	10	11	10	0	0	136645913	5863125	403352	627696181
338	15000	735	14265	1	10	11	10	0	0	137095913	5863125	400161	628096342
339	15000	662	14338	1	10	11	10	0	0	137545913	5863125	396996	628493338
340	15000	588	14412	1	10	11	10	0	0	137995913	5863125	393855	628887193
341	13405	515	12891	1	9	10	10	0	0	138398074	5237520	349047	629236240
342	13141	441	12700	1	9	10	10	0	0	138792309	5132918	339369	629575609
343	12806	368	12438	1	9	10	10	0	0	139176481	5000104	327973	629903582
344	12502	294	12208	1	9	10	10	0	0	139551551	4879986	317562	630221144
345	12192	221	11972	1	9	10	10	0	0	139917323	4757278	307128	630528272
346	11848	147	11701	1	9	10	10	0	0	140272755	4620810	295957	630824229
347	11539	74	11466	1	9	10	10	0	0	140618925	4498579	285849	631110078
348	11256	0	11256	0	9	9	10	0	0	140956601	4392235	276884	631386962
349	11065	0	11065	0	9	9	10	0	0	141288537	4316480	269956	631656918
350	10889	0	10889	0	9	9	10	0	0	141615200	4246887	263502	631920420
351	10730	0	10730	0	9	9	10	0	0	141937106	4184092	257552	632177972
352	10596	0	10596	0	9	9	10	0	0	142254998	4131141	252281	632430253
353	10475	0	10476	0	9	9	10	0	0	142569263	4083251	247384	632677637
354	10367	0	10367	0	9	9	10	0	0	142880269	4040256	242842	632920479
355	10270	0	10270	0	9	9	10	0	0	143188356	4001735	238624	633159103
356	10184	0	10184	0	9	9	10	0	0	143493864	3967683	234722	633393825
357	10106	0	10106	0	9	9	10	0	0	143797052	3937073	231069	633624894
358	10037	0	10037	0	9	9	10	0	0	144098175	3909827	227654	633852548
359	9977	0	9977	0	9	9	10	0	0	144397483	3885862	224469	634077017
360	9922	0	9922	0	9	9	10	0	0	144695140	3864088	221445	634298462

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
361	9871	0	9871	0	9	9	10	0	0	144991272	3843942	218548	634517010
362	9822	0	9823	0	9	9	10	0	0	145285946	3824710	215734	634732744
363	9776	0	9776	0	9	9	10	0	0	145579215	3806177	212991	634945735
364	9731	0	9731	0	9	9	10	0	0	145871145	3788492	210324	635156059
365	9692	0	9692	0	9	9	10	0	0	146161897	3772949	207804	635363863
366	9667	0	9667	0	9	9	10	0	0	146451919	3763314	205633	635569496
367	8086	0	8086	0	8	8	10	0	0	146694497	3142929	170376	635739872
368	8088	0	8088	0	8	8	10	0	0	146937125	3143576	169063	635908935
369	8089	0	8089	0	8	8	10	0	0	147179803	3144248	167761	636076696
370	8097	0	8097	0	8	8	10	0	0	147422701	3147142	166587	636243283
371	8107	0	8107	0	8	8	10	0	0	147665909	3151238	165484	636408767
372	8118	0	8118	0	8	8	10	0	0	147909450	3155630	164404	636573171
373	8127	0	8127	0	8	8	10	0	0	148153273	3159353	163296	636736467
374	8138	0	8138	0	8	8	10	0	0	148397408	3163477	162216	636898683
375	8150	0	8150	0	8	8	10	0	0	148641921	3168459	161186	637059869
376	8162	0	8162	0	8	8	10	0	0	148886793	3173197	160150	637220019
377	8040	0	8040	0	8	8	10	0	0	149127979	3124547	156447	637376466
378	8066	0	8066	0	8	8	10	0	0	149369945	3134855	155721	637532187
379	8087	0	8087	0	8	8	10	0	0	149612567	3143499	154915	637687102
380	8107	0	8107	0	8	8	10	0	0	149855765	3151103	154062	637841164
381	8124	0	8124	0	8	8	10	0	0	150099490	3158070	153181	637994345
382	8141	0	8141	0	8	8	10	0	0	150343727	3164813	152293	638146638
383	8158	0	8158	0	8	8	10	0	0	150588467	3171450	151405	638298043
384	8174	0	8174	0	8	8	10	0	0	150833680	3177705	150504	638448547
385	8188	0	8188	0	8	8	10	0	0	151079327	3183426	149582	638598129
386	8203	0	8203	0	8	8	10	0	0	151325416	3189260	148671	638746800
387	8217	0	8217	0	8	8	10	0	0	151571925	3194798	147751	638894551
388	8230	0	8230	0	8	8	10	0	0	151818824	3199953	146818	639041369
389	8242	0	8242	0	8	8	10	0	0	152066094	3204848	145880	639187249
390	8255	0	8255	0	8	8	10	0	0	152313729	3209665	144943	639332192
391	7915	0	7915	0	7	7	10	0	0	152551175	3080946	138030	639470222
392	7812	0	7812	0	7	7	10	0	0	152785524	3040068	135121	639605343
393	7743	0	7743	0	7	7	10	0	0	153017805	3012782	132849	639738192
394	7699	0	7699	0	7	7	10	0	0	153248780	2995536	131043	639869235
395	7671	0	7671	0	7	7	10	0	0	153478915	2984452	129526	639998761
396	7548	0	7548	0	7	7	10	0	0	153705360	2935758	126404	640125165
397	6490	0	6490	0	6	6	10	0	0	153900070	2522685	107759	640232924
398	6445	0	6445	0	6	6	10	0	0	154093420	2504740	106146	640339070
399	6441	0	6441	0	6	6	10	0	0	154286644	2503081	105237	640444307
400	6429	0	6429	0	6	6	10	0	0	154479526	2498557	104216	640548523
401	6416	0	6416	0	6	6	10	0	0	154672006	2493249	103172	640651695
402	6403	0	6404	0	6	6	10	0	0	154864110	2488294	102152	640753847
403	6395	0	6395	0	6	6	10	0	0	155055945	2484743	101199	640855046
404	6386	0	6386	0	6	6	10	0	0	155247532	2481471	100266	640955312
405	5946	0	5946	0	6	6	10	0	0	155425903	2307055	92482	641047794
406	5960	0	5961	0	6	6	10	0	0	155604718	2312904	91982	641139776

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
407	5961	0	5961	0	6	6	10	0	0	155783555	2313208	91267	641231043
408	5958	0	5958	0	6	6	10	0	0	155962300	2311979	90497	641321540
409	5951	0	5951	0	6	6	10	0	0	156140843	2309318	89677	641411217
410	5939	0	5939	0	6	6	10	0	0	156319022	2304521	88783	641500000
411	5811	0	5811	0	6	6	10	0	0	156493350	2253696	86138	641586138
412	5807	0	5807	0	6	6	10	0	0	156667548	2251976	85392	641671530
413	5146	0	5146	0	5	5	10	0	0	156821930	1996207	75094	641746624
414	5149	0	5149	0	5	5	10	0	0	156976404	1997420	74546	641821170
415	5151	0	5151	0	5	5	10	0	0	157130928	1998075	73980	641895150
416	5151	0	5151	0	5	5	10	0	0	157285463	1998232	73401	641968551
417	5151	0	5151	0	5	5	10	0	0	157439988	1998097	72815	642041366
418	5157	0	5157	0	5	5	10	0	0	157594698	2000534	72327	642113693
419	5163	0	5163	0	5	5	10	0	0	157749575	2002735	71834	642185527
420	5165	0	5165	0	5	5	10	0	0	157904526	2003724	71301	642256828
421	5124	0	5124	0	5	5	10	0	0	158058249	1987503	70164	642326992
422	5073	0	5073	0	5	5	10	0	0	158210443	1967327	68902	642395894
423	3535	0	3535	0	4	4	10	0	0	158316489	1364048	47396	642443290
424	3496	0	3496	0	4	4	10	0	0	158421356	1348474	46484	642489774
425	3458	0	3458	0	4	4	10	0	0	158525081	1333419	45601	642535375
426	3418	0	3418	0	4	4	10	0	0	158627618	1317727	44708	642580083
427	3380	0	3380	0	4	4	10	0	0	158729010	1302619	43846	642623929
428	3343	0	3343	0	4	4	10	0	0	158829303	1288126	43015	642666944
429	3306	0	3306	0	4	4	10	0	0	158928471	1273269	42182	642709126
430	3265	0	3265	0	4	4	10	0	0	159026410	1257048	41316	642750442
431	3225	0	3225	0	4	4	10	0	0	159123163	1241399	40478	642790920
432	2850	0	2850	0	4	4	10	0	0	159208661	1092849	35353	642826273
433	2566	0	2566	0	4	4	10	0	0	159285645	980505	31468	642857741
434	2350	0	2350	0	3	3	10	0	0	159356138	900581	28674	642886415
435	2347	0	2347	0	3	3	10	0	0	159426538	899347	28408	642914823
436	2343	0	2343	0	3	3	10	0	0	159496829	897924	28139	642942962
437	2337	0	2337	0	3	3	10	0	0	159566936	895483	27840	642970802
438	2330	0	2330	0	3	3	10	0	0	159636843	892851	27539	642998341
439	2319	0	2319	0	3	3	10	0	0	159706401	888244	27180	643025521
440	2301	0	2301	0	3	3	10	0	0	159775417	881081	26748	643052269
441	2277	0	2277	0	3	3	10	0	0	159843714	871603	26251	643078520
442	2248	0	2248	0	3	3	10	0	0	159911165	860441	25709	643104229
443	2227	0	2227	0	3	3	10	0	0	159977967	851858	25251	643129480
444	2201	0	2201	0	3	3	10	0	0	160043997	841683	24753	643154233
445	2167	0	2167	0	3	3	10	0	0	160109020	828394	24169	643178402
446	2127	0	2127	0	3	3	10	0	0	160172818	812229	23510	643201912
447	2080	0	2080	0	3	3	10	0	0	160235212	793691	22792	643224704
448	2031	0	2031	0	3	3	10	0	0	160296143	774386	22061	643246765
449	1983	0	1983	0	3	3	10	0	0	160355628	755304	21347	643268112
450	1933	0	1933	0	3	3	10	0	0	160413611	735485	20623	643288735
451	1882	0	1882	0	3	3	10	0	0	160470081	715502	19904	643308639
452	1832	0	1832	0	3	3	10	0	0	160525047	695666	19199	643327838

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
453	1785	0	1785	0	3	3	10	0	0	160578595	676958	18535	643346373
454	1739	0	1739	0	3	3	10	0	0	160630775	658890	17897	643364270
455	1696	0	1696	0	3	3	10	0	0	160681653	641717	17293	643381563
456	1654	0	1654	0	3	3	10	0	0	160731285	625265	16716	643398279
457	1615	0	1615	0	3	3	10	0	0	160779738	609710	16172	643414451
458	1578	0	1578	0	3	3	10	0	0	160827089	595160	15661	643430112
459	1544	0	1544	0	3	3	10	0	0	160873396	581394	15178	643445290
460	1511	0	1511	0	3	3	10	0	0	160918720	568407	14721	643460011
461	1480	0	1480	0	3	3	10	0	0	160963109	556074	14288	643474299
462	1451	0	1451	0	3	3	10	0	0	161006626	544570	13881	643488180
463	1424	0	1424	0	3	3	10	0	0	161049352	534124	13507	643501687
464	1400	0	1400	0	3	3	10	0	0	161091339	524373	13156	643514843
465	1376	0	1376	0	3	3	10	0	0	161132617	515021	12819	643527662
466	1353	0	1353	0	3	3	10	0	0	161173210	505973	12494	643540156
467	1331	0	1331	0	3	3	10	0	0	161213137	497182	12180	643552336
468	1309	0	1309	0	3	3	10	0	0	161252415	488629	11876	643564212
469	1289	0	1289	0	3	3	10	0	0	161291088	480638	11589	643575801
470	1270	0	1270	0	3	3	10	0	0	161329180	472969	11314	643587115
471	1251	0	1251	0	3	3	10	0	0	161366705	465483	11047	643598162
472	1232	0	1232	0	3	3	10	0	0	161403677	458181	10788	643608950
473	1214	0	1214	0	3	3	10	0	0	161440109	451067	10536	643619486
474	1197	0	1197	0	3	3	10	0	0	161476017	444140	10292	643629778
475	1180	0	1180	0	3	3	10	0	0	161511414	437401	10056	643639834
476	765	0	765	0	2	2	10	0	0	161534369	278950	6362	643646196
477	752	0	752	0	2	2	10	0	0	161556923	273658	6192	643652388
478	739	0	739	0	2	2	10	0	0	161579090	268546	6029	643658417
479	726	0	726	0	2	2	10	0	0	161600883	263612	5871	643664288
480	714	0	714	0	2	2	10	0	0	161622311	258807	5718	643670006
481	702	0	702	0	2	2	10	0	0	161643377	254010	5568	643675574
482	691	0	691	0	2	2	10	0	0	161664093	249399	5424	643680998
483	679	0	679	0	2	2	10	0	0	161684474	244986	5285	643686283
484	669	0	669	0	2	2	10	0	0	161704536	240766	5153	643691436
485	659	0	659	0	2	2	10	0	0	161724292	236723	5027	643696463
486	649	0	649	0	2	2	10	0	0	161743753	232839	4905	643701368
487	639	0	639	0	2	2	10	0	0	161762930	229091	4788	643706156
488	630	0	630	0	2	2	10	0	0	161781832	225461	4675	643710831
489	621	0	621	0	2	2	10	0	0	161800467	221930	4565	643715396
490	612	0	612	0	2	2	10	0	0	161818840	218484	4459	643719855
491	604	0	604	0	2	2	10	0	0	161836958	215113	4355	643724210
492	596	0	596	0	2	2	10	0	0	161854826	211809	4254	643728464
493	587	0	587	0	2	2	10	0	0	161872447	208562	4156	643732620
494	579	0	579	0	2	2	10	0	0	161889827	205364	4060	643736680
495	571	0	571	0	2	2	10	0	0	161906967	202210	3966	643740646
496	563	0	564	0	2	2	10	0	0	161923872	199095	3874	643744520
497	556	0	556	0	2	2	10	0	0	161940543	196016	3784	643748304
498	548	0	548	0	2	2	10	0	0	161956983	192974	3696	643752000

Time	qTotal	q1	q2	Wells 1	Wells 2	total Wells	drilled Wells 2	Rig 1	Rig 2	Np	Local Money	NPV	Cumulative NPV
499	540	0	540	0	2	2	10	0	0	161973196	189968	3609	643755609
500	533	0	533	0	2	2	10	0	0	161989184	187008	3525	643759134
501	526	0	526	0	2	2	10	0	0	162004951	184085	3443	643762577
502	518	0	518	0	2	2	10	0	0	162020499	181194	3362	643765939
503	511	0	511	0	2	2	10	0	0	162035830	178329	3282	643769221