2000

Restimulation candidate selection using virtual intelligence

Khalid Y. Mohamad
West Virginia University

Restimulation Candidate Selection using Virtual Intelligence

By

Khalid Mohamad

A thesis

Submitted to the faculty of the

College of Engineering and Mineral Resources

West Virginia University

In partial fulfillment of the Requirements for the Degree of

Master of Science in Petroleum and Natural Gas Engineering

Sam Ameri, Chair

Turgay Ertekin, Ph.D.

Dan Wood

Shahab Mohaghegh, Ph.D.

Department of Petroleum and Natural Gas Engineering

Morgantown, West Virginia

2000

ABSTRACT

Restimulation Candidate Selection using Virtual Intelligence

Khalid Y. Mohamad

Due to the importance of well deliverability maintenance, a committee of specialists from Dominion East Ohio and other service companies meets every year to select the wells to be included in the deliverability maintenance plan. The application tool not only help in selecting the wells for deliverability maintenance plan but goes beyond that by designing the most optimum frac recipe.

The purpose of this study is to develop an engineering tool that will help petroleum engineers making a better decision for selecting well candidate and design well restimulation. The project focuses on a gas storage field and use data such as well location, stimulation time and recipe and deliverability statistics.

This tool reduces the time engineers spend designing optimum treatment schedules by proposing a solution based on virtual intelligence. Neural networks, genetic algorithms and a fuzzy support system are integrated into a software application to achieve the required goals.

The software application is a user-friendly application compiled in a Visual Basic programming language linked to and access database.

Acknowledgment

I would like to take this opportunity to express my deepest gratitude for my parents for their patience and support during the duration of my study abroad, I am grateful to all what they did for me.

Also I would like to express my gratitude to my brother Yacoub for his support, encouragements and patience throughout my study and school days.

I would like to express my appreciation to my Advisor DR. Shahab Mohaghegh, for his support, advice and his guidance through my long trip for my MS degree in the field of Petroleum and Natural Gas Engineer. Special thanks to all my professors in the department of Petroleum and Natural Gas Engineering for their help and support.

Special thanks also goes to my graduate committee Dr. Sam Ameri, Dr. Turgay Ertekin, Dan Wood and Dr. Shahab Mohaghegh.

I would like to thank my Friend Andrei Popa for his help and advice. I also would like to thank all my fiends who stood beside me through the good and bad times.

TABLE OF CONTENETS

List of Tables

List of Figures

1. Introduction

1.1 Introduction

"Gas storage in depleted reservoirs is a well-known complex technique that takes advantage of the naturally occurring storage media, and of the knowledge accumulated over their exploitation period[1]." This takes place when the market demand falls below the supply. Storage and production operation both are affected by the reservoir deliverability.

Over the years the problem of decreasing deliverability in gas storage wells has seen on one hand the cost of tremendous efforts by the engineers, and on the other hand, the cost of economic resources. The decrease of deliverability is caused by many factors; the most significant ones being bacteria and well damage. To maintain and improve the well deliverability many appropriate methods, such as fracturing, chemical treatments, and coil tubing, have been developed.

The main objective of this study is to develop a Virtual Intelligence System that provides support for engineers in their decision making when selecting candidates for stimulation treatment.

Data availability and structure were the main motivations on deciding to use Virtual Intelligence Systems. The field data from a gas storage field have been provided by Dominion East Ohio. It consists of well completion information, well deliverability history, and treatment recipes. Numerical simulators usually require comprehensive data; for example, the accurate determination of reservoir parameters such as permeability and porosity is mandatory in using a numerical simulator. The method developed in this study is a valuable tool and can be used when a lack of engineering data is encountered.

1.2 Background

Today, the idea of using Virtual Intelligence Systems in solving engineering problems is not something new. Abroad spectrum of Petroleum Engineering Problems is currently solved by neural network based applications, such as reservoir characterization, well testing, and permeability predicting from well log data. There are also numerous applications that are widely used in the industry that utilize virtual intelligence. There have been numerous published papers and research work examining the application of the Virtual Intelligence Systems in the petroleum industry. This study will explore the role of Virtual Intelligence Systems in assisting the engineer in the process of selecting a candidate well for stimulation.

Well stimulation or fracturing treatment is one of the methods that has been used widely by Dominion East Ohio to improve the deliverability of the Clinton Sand. Every year a committee of specialists selects the wells to be included in the deliverability maintenance plan. The collection of all associated data for the fracturing program had started in 1967. The data for this project were collected from the actual files in the DEO location at North Canton, Ohio. The data were carefully reviewed, revised, and affirmed before their storage in a relational database. This database provides a straightforward and quick access to wells data and treatment jobs.

1.3 Well Testing

Each well in the field is tested annually with a two-point test during the storage injection season[1]. The company uses a two point pseudo-modified isochronal testing procedure to obtain a pseudo back –pressure performance curve.

The test is a pseudo-modified isochronal test for the following reasons:

1. The test is performed during injection rather than withdrawal seasons.
2. The time that the well is shut-in is not equal to the injection time as in a true modified isochronal test.

The equation used for backpressure analysis is:

$$Q = C*(Pf^2 - Ps^2)^n$$

Where:

Q=production rate, MSCF/D

Pf = static bottom hole pressure, psia

Ps = flowing bottom hole pressure, psia

n = slope of the back pressure curve

C= performance coefficient

The data needed to solve this equation is obtained by performing the pseudo-modified isochronal test.

The field procedure to perform the test is as follows:

1. Shut the well in one day in advance of the test.
2. Record the shut in well pressure and well temperature before starting the test.
3. Open the well for 30 minutes, record the flowing pressure, differential pressure, and flow temperature at the end of the test.

4. Shut the well in for 60 minutes; Record the static pressure and temperature.
5. Open the well for 30 minutes: record flow pressure, differential pressure, and flow temperature at the end of the test.

Plotting the log of the pressure squared versus the log of the flow rate for the test points will yield several points that can be plotted for the best line curve fit. The slope and intercept will be found using the linear least square method. Q100, which is the equivalent flow rate at a common delta P^2 of 100,000, is then calculated using the corresponding C and n values. This procedure does not provide information such as permeability, skin, drainage radius, or fracture length. However, it has historical significance and it gives some relative indication of well performance.

1.4 Project Approach

As mentioned earlier, one of the system's purposes is to assist engineers in selecting candidate wells for treatment. Access to data is critical at this point, namely, the engineer should be able to consider numerous different aspects prior to making a decision. This will be accomplished with much more ease if the data is organized into a database. Wisely querying the database facilitates the process of reaching a decision for the engineer. Since this project relies heavily on the organization of the data, the first stage of this research was to reconstruct the database.

Once the decision is made as to which wells should be treated, post-treatment deliverability can be estimated for these wells. The next step was to design and train artificial neural networks. This implies that for each separate neural network, there is the process of selecting appropriate inputs, finding the right network architecture, training algorithm, and choosing training parameters. In order to accomplish this task several neural networks were used. Input for these neural networks is obtained by querying the database, thus this step heavily relies on both the database and the Virtual Intelligence.

The next step is the application of the system. Genetic algorithms combine "pieces" of treatment recipes always selecting the best-fitted "individuals." Specialized neural networks are used as a fitness function.

Finally, a decision support system based on the theory of Fuzzy Logic is applied to generate the best-ranked well list for stimulation.

2. Artificial Intelligence

2. 1 Artificial Neural Networks

Artificial neural networks are mathematical systems processed through processing units. These processing units are mathematical formulas known as transfer functions. The transfer functions are linked through a large number of adjustable weights that come from other connected neurons.

"The function of artificial neural network is to mimic the learning process of the human brain[4]". In other words, it processes information in a similar way the human brain does. It does this by imitating the parallel architecture of the human brain by providing a large number of highly interconnected units to imitate the human neuron.

The neural network gathers its knowledge by detecting the patterns and relationships that exist in the data. Using this powerful methodology, it is possible to introduce a solution to a particular problem if the set of inputs and outputs of data exist, regardless if the relationship between them is unknown. Thus, the employment of neural network in forecasting and predicting can save substantial time and human resources. The advantages of using a neural net can be summarized as follows:

- It can deal with non-linearity in the data,
- Only a set of input and output data are required to start analyzing the system,
- It can handle noise in data,
- It can handle a large number of parameters.

The main building units of the neural net are the neurons. These neurons are located in three main blocks as shown in figure 2.1. The first block consists of one layer containing the input neurons, or input data. The middle block might consist of more than

one layer called hidden layers as shown in the figure. Each hidden layer might consist of a certain number of neurons. Hidden layers are connected to the input layer and the output layer with adjustable weights. Finally the third block is the output layer, which consists of a certain number of neurons representing the output parameters.

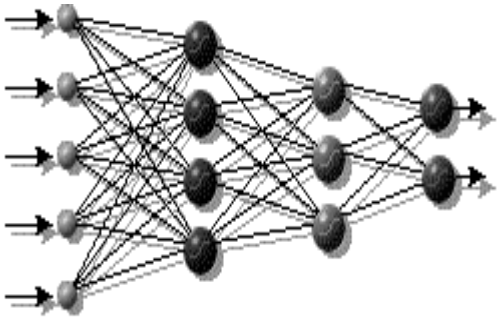

Figure 2.1: Typical back propagation network.

Training of neural network involves three stages: the feed-forward of the input training pattern, the calculation of the associated error, and the adjustment of the weights.

In the first stage after dividing the data into input and output data, the input layer that consists of a certain number of neurons equal to the number of input data, passes one block of data to the following layer. As mentioned before the two layers are connected to each other through a number of adjustable weights that help to control the flow of data. This step is called feed-forward dynamics. The net input at each neuron is determined by the sum of the weight multiplied by the received input as shown in Equation, 1

$$Y_i = \sum w_{ji} * Input_i \qquad (1)$$

The processing unit, transfer function, is then applied according to Equation, 2

$$F(Y) = \frac{1}{1 + e^{(Y_i + bias)}} \qquad (2)$$

The output of this sigmoid function is a continuous real value between 0 and 1. Figure 2.2 shows graphical representation of an activation function[3].

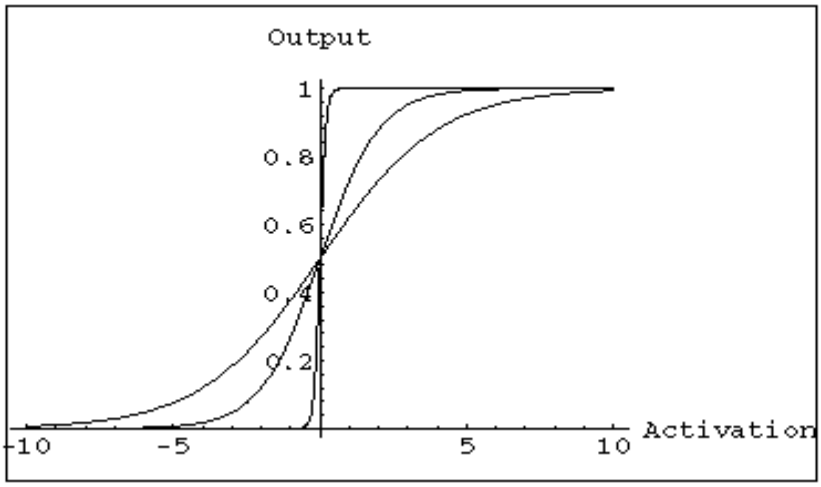

Figure 2.2: Transfer functions for back propagation network.

In the second stage, each computed output is compared with the real output and the error is then calculated.

Finally, after the error is calculated the weights are adjusted base on the error calculated and the activation of the hidden layer.

2. 2 Genetic Algorithms

Genetic Algorithms are powerful search algorithms based on the mechanism of natural evolution. By mimicking this process, genetic algorithms are able to "evolve" solutions to real world problems, if they have been suitably encoded. Genetic algorithms are global search and optimization routines that mimic the Darwin Evolution theory. Genetic algorithms achieve their objective by operating on a population of strings that are evolved over time. These genetic strings are decoded to form a particular parameter set that represents a solution to the problem. They combine primary genetic operations of reproduction such as selection, cross over, and mutation. These operators are very simple, involving nothing more complex than random number generation, string copying, and partial string exchanging; "yet the resulting search performance is wide-ranging and impressive.[6]"

Genetic Algorithm can be used to find a solution in considerably less time than their conventional counter parts. The idea of Genetic Algorithms is to simulate the way nature uses evolution. It uses the concept of survival of the fittest. The good solutions reproduce to form new and hopefully superior solutions in the population, while the bad solutions are removed.

The Genetic Algorithm lets us obtain solutions to problems that do not have a precisely defined solving method; or if they do, following the exact solving method would take substantially more time.

Genetic Algorithms begin with a population of binary individuals. "The individual, commonly referred to as a chromosome or string, is a computer compatible representation of a function or problem[9]." Each individual in the population is evaluated and its fitness is measured. Only the high fitness individuals are to be considered for the next generation. The previously mentioned operators produce the next generation of the population. The primary operators involved in the genetic algorithm process are:

Selection

Selection is the process of choosing individuals from the genetic population, which have the high fitness values. Typical selection methods include roulette and tournament.

Cross over

Recombination forms new individuals by generating copies of high-fitness individuals. A recombination technique, commonly known as crossover, creates new individuals by taking sub strings from two individuals to form an offspring.

Mutation

Mutation is the flipping of the binary bit value from 0 to 1, and doing so in an opposite manner.

Pseudo Code of The Genetic Algorithm process[5]

```
GA Algorithm
// Start with initiate time
t = 0;
// Initialize a usually random population of individuals
Initpopulation P (t);
// Evaluate fitness of all initial individuals of population
Evaluate P (t);
//test for termination criterion (time, fitness, etc)
```

```
While not done do
        //increase the time counter
        t = t + 1
        //select a sub-population for offspring production
        P' = select-parents P (t);
        // Select the "genes" of selected parents
        Recombine P (t);
        //perturb the mated population stochastically
        Mutate P (t);
        //evaluate its new fitness
        Evaluate P (t);
```

2. 3 Fuzzy Logic and Fuzzy Expert System

"Fuzzy logic is a powerful problem solving methodology[10]" based on fuzzy set theory. It provides a simple way of representing reality and defines vague information by resembling human decision-making. Unlike the traditional logic set, where we have the classification of true or false, fuzzy logic is based on conventional logic that has been extended to handle the concept of partial false and partial truth between "completely true" and "completely false." A fuzzy value is a continuous value ranging from zero to one. What this means is that a statement in Fuzzy Logic is either true by various degrees or false by various degrees starting from zero to one. Fuzzy set theory gives mathematical backgrounds and foundations for the description and handling of human knowledge.

The basic definition of a classical set is a collection of elements. Elements can either belong to the set, meaning completely inside the set, or not belong to the set, meaning completely outside the set. Fuzzy Sets, which are elements that may only partially belong to the set, or belong to the set in different degrees, are called grades of membership. To make an approximate comparison, these memberships are described in a Fuzzy number. A Fuzzy number consists of a number and word that provides a linguistic description, which later helps in setting the rules and explaining the decision making process.

Fuzzy Expert Systems are the most commonly used tools of Fuzzy Logic. Just like Fuzzy Logic, Fuzzy Expert Systems are also based on fuzzy set theory. Fuzzy Expert Systems are systems designed to review a collection of fuzzy membership functions and rules to reach a specific decision or conclusion. This conclusion should be close if not the same as the conclusion reached by a team of experts.

Fuzzy Expert Systems are considered to be unique intelligent tools because they have the ability to explain the reasoning behind the decision made. They use knowledge and linguistic rules defined by experts in the problem field. They are used in several wide-ranging fields like Linear and Nonlinear Control, Pattern Recognition, and Financial Systems.

Fuzzy Expert Systems consist of three main processes: Fuzzification, Inference Engine, and Defuzzification. The Fuzzication process "is a mapping from the observed input to the fuzzy sets defined in the corresponding universe of discourse[11]." In this process the input is translated from its crisp or a numerical value to a fuzzy output with linguistic term and a corresponding grade of membership.

The Inference Process is " the decision making logic, which determines fuzzy outputs corresponding to a fuzzified inputs, with respect to fuzzy rules. The designer must specify which rules, implication, and aggregation operators are used." Whenever the inputs are not clear and reflect belief rather than proof, Inference takes place employing inference steps similar to those a human brain might take. The Fuzzy inference engine is built using rules based on linguistic statements.

As a result of the Fuzzification and inference processes, several rules will be fired to generate a symbolic result or an output fuzzy subset. The process of converting this output fuzzy subset to a crisp number is known as defuzzification Process. In other words, defuzzification is the calculation of a crisp numerical value as linguistic output based on the symbolic results.

3.0 Relational Database

Train accurate artificial neural network requires lots of data and requires these data to be organized and updateable. Therefore, for a fast and an efficient retrieval for data, a fine structured relational database was needed. This database took in consideration the continuous flow of data, especially frac recipe data and deliverability maintenance data, and the use of these data in the application. Other useful feature that database can provide is the retrieving of specific information.

3.1 Development of Database

The data inside the database were divided into several tables connecting to each other by structure query language, SQL, provided by the jet engine comes with Access database. Each table contains related information different than other tables. For instance information regarding a well completion and location are stored in one table, while frac recipe is stored in other table. Some table in the database cannot be browse by the user they conations other useful information used by the application. For example, default Rules and user Rules are stored in the database.

4. Data Analysis

4.1 Clinton Reservoir Characteristics

This study is based on real data provided by Dominion East Ohio, for an underground storage reservoir located in northeast Ohio. The reservoir under investigation is a tight gas formation called Clinton Sandstone. It is tight gas sand with low permeability. 'Sand occurs in lenses and is discontinuous from one well to another making generalizing of well properties from core data difficult[2]."

Table 4.1: Clinton Reservoir Characteristics

Reservoir Type	Tight Gas Sand
Reservoir description	Heterogeneous-lenses
Average Depth	4000 ft
Average Pressure	1500 psi
Primary Porosity	Naturally Fractured
Average Porosity	6%
Average Permeability	0.1 mD
Spacing	20 – 40 acres

4.2 Data Statistics

The data includes information for 804 wells. There are a total of 1589-frac job records with frac job recipes. Some wells were fraced more than once and sometimes up to four times as shown in the figure below. Two-point flow test are also included in the database. The two-point flow test was performed in a regular basis in this reservoir.

4.2.1 Records that can be used in Analysis

Many records were lost due to the lack of isochronal test values. Most of the lost records occurre before the year of 1967. The main reason is that the recording of the isochronal test data started at 1967. On several occasions records of the Prior to Frac Deliverability readings were missing, and only Post Frac Deliverability readings were available, on the other hand Post Frac Deliverability was not recorded for some of the recent jobs.

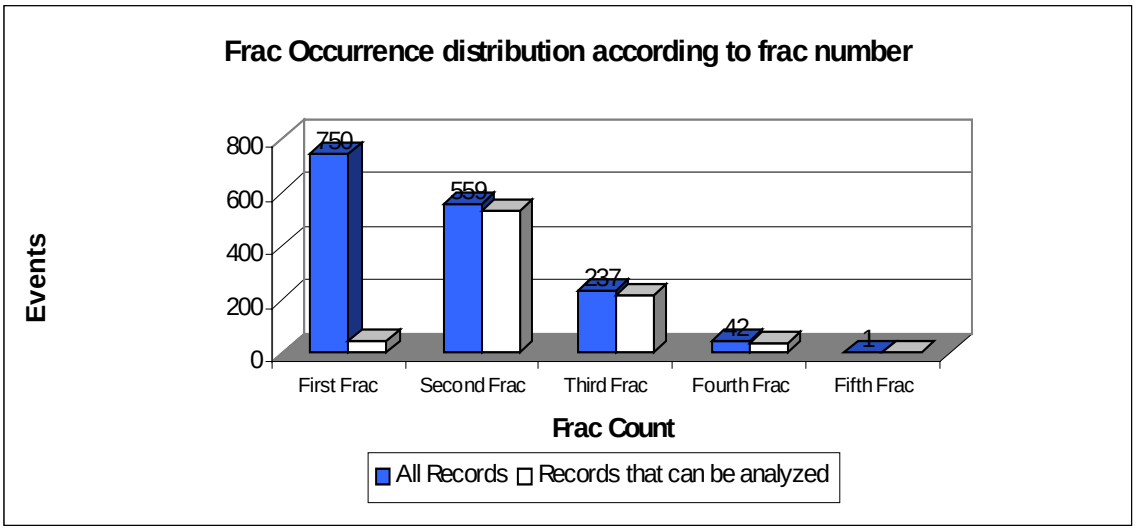

Figure 4.1: Frac Occurrence distribution according to frac number.

As shown in the figure the percentage lost for the first frac was almost 95%, while it was only 5.4% and 8.9 % for the second and the third frac respectively.

4.2.2 Statistical Analysis

After excluding all the records that were not useful for analysis, a detailed statistical analysis was performed on the data. The following is some of the most important statistics.

4.2.2.1 Statistics on Service Companies:

The Service Company is considered one of the important parameters. It reflects the role of the quality control on the frac job performed.

The figure below shows that Halliburton, HA, performed most of the frac jobs, followed by Dowel Schlumberger, DS.

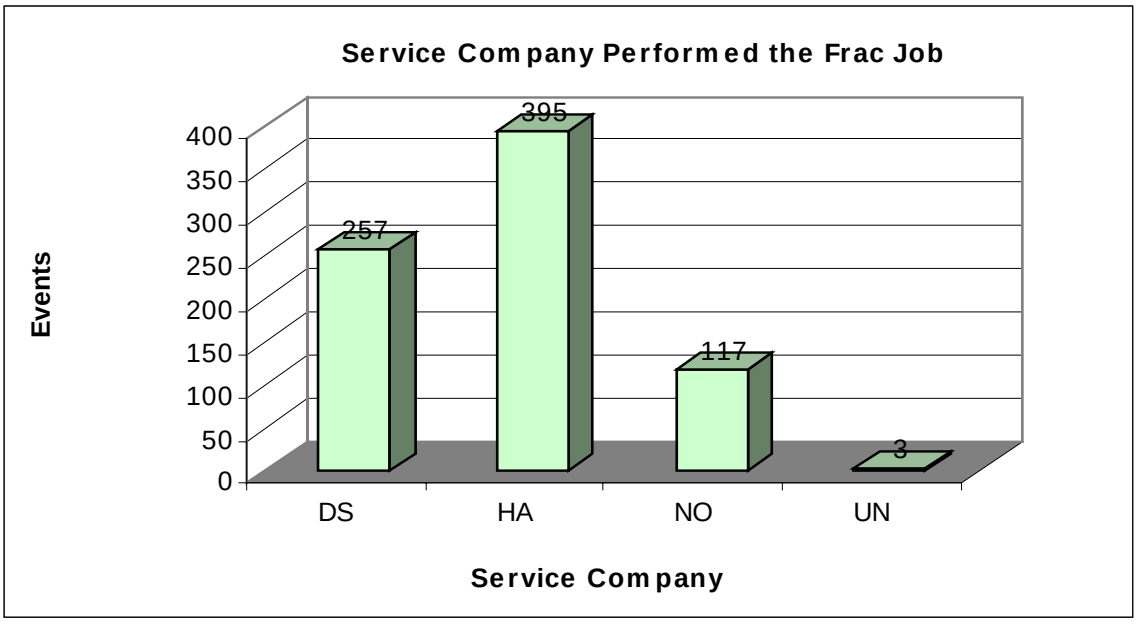

Figure 4.2: Service Company Performed the Frac Job

Some companies were among the data excluded from the database as was mentioned in the previous section.

4.2.2.2 Statistics on Frac Fluid:

For frac jobs that have been performed before 1967, water was the frac fluid that has been used most commonly, after excluding those records; figure 4.3 shows that most of the fracs were preformed using gel as the frac fluid.

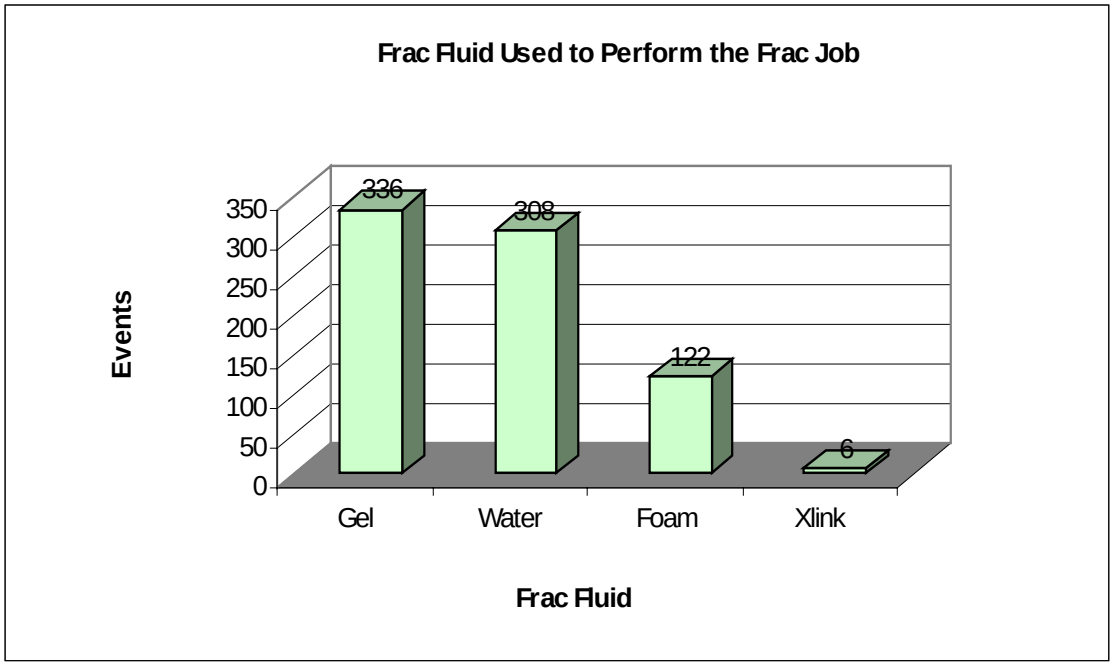

Figure 4.3: Frac fluid used to perform the frac job.

Next we explored the distribution of the use of different frac fluids with time. This is to identify if a pattern exists.

A. Distribution of Frac Fluid Over the Years:

In late sixties and early seventies, water used to be considered the primary frac fluid. Figure 4.4 shows the decrease in the use of water as frac fluid and the increase in the use of gel as frac fluid. The use of foam does not significantly change even though there was a period in the late 80's when its use was dropped remarkably. It

starts to increase again to less than its normal use in the 90's. Recent years show the use of cross-link as a new type of frac fluid. It is not included in the figure below because the number of fracs using cross-link as a frac fluid is relatively small comparing to other fracs using other frac fluids.

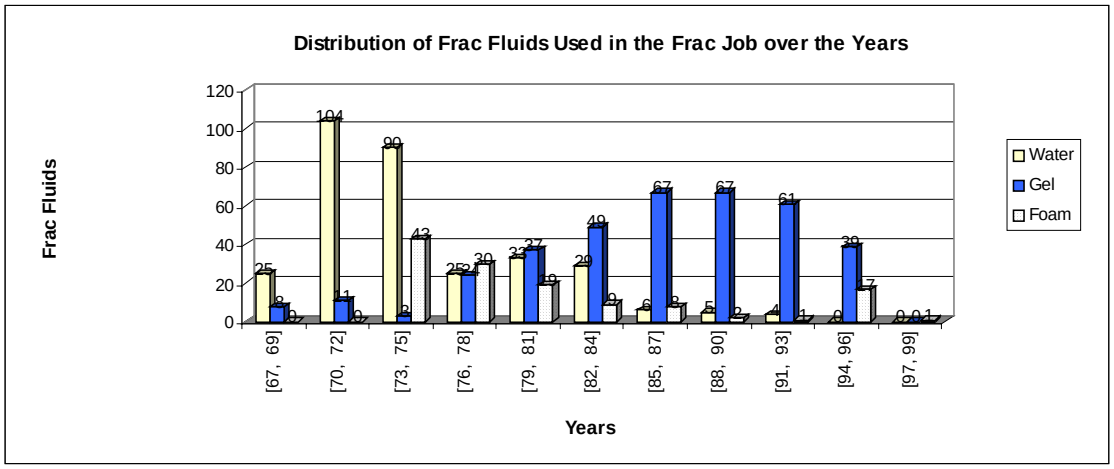

Figure 4.4: Distribution of frac fluids used in the frac job over the years.

B. Service Company and Frac Fluid:

When water has been used as a frac fluid, figure 4.5 shows that Dowel Schlumberger was the company that used water most frequently. They performed 61% of their frac jobs using water as frac fluid. While Halliburton performed only 35% of their frac job using water. On the other hand, Halliburton used Gel as frac fluid more frequently than any other company. Using the information provided in the previous section, it can be seen that Halliburton performed 81 frac jobs in the years after 1990, while Dowel Schlumberger did not perform any. This gives a good indication why the use of gel increased in the recent years while the use of water dropped.

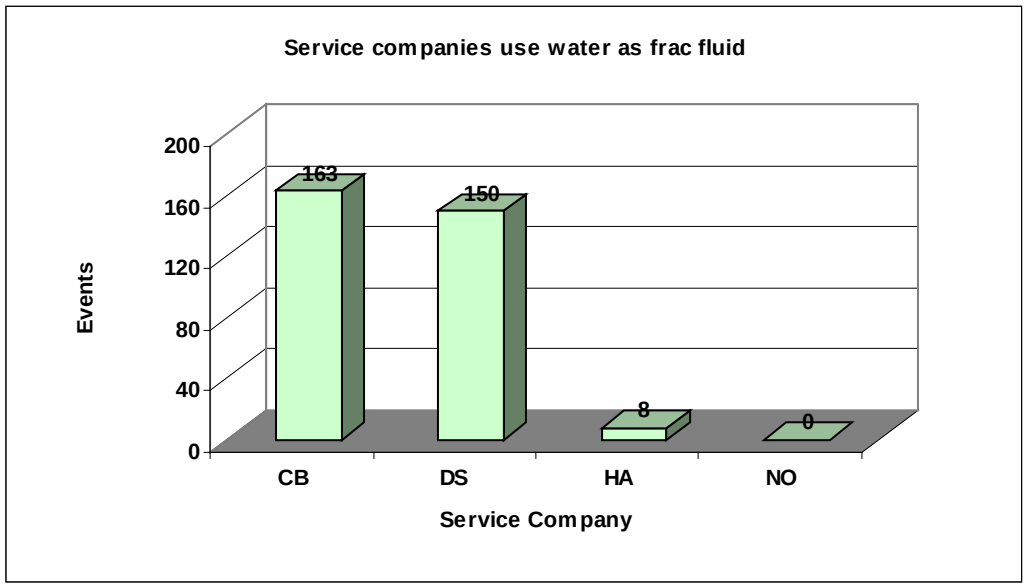

Figure 4.5: Service companies use water as frac fluid.

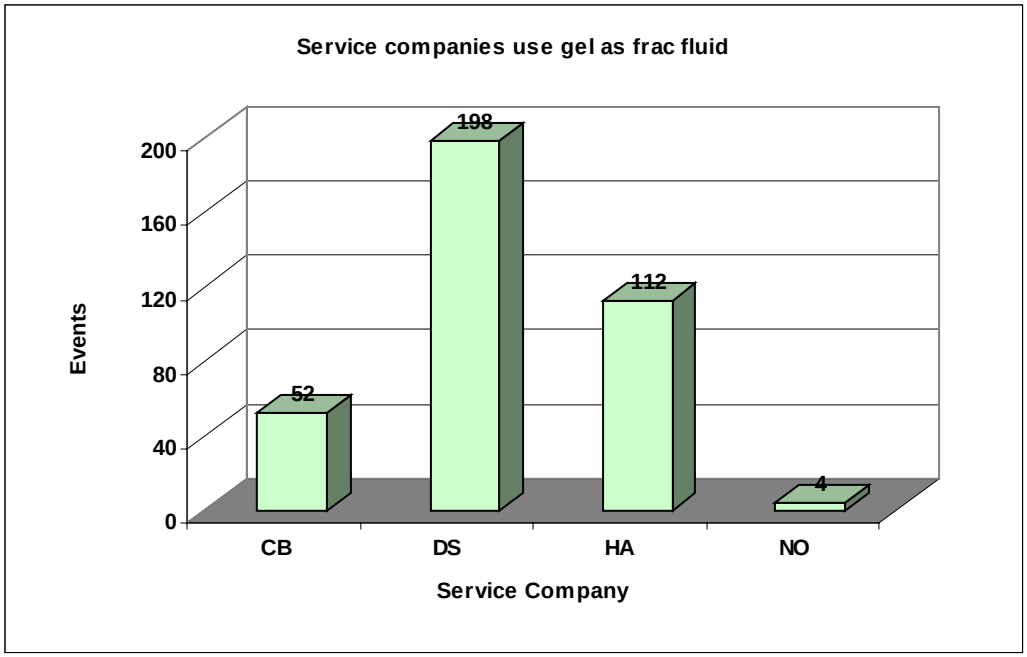

Figure 4.6: Service companies use gel as frac fluid.

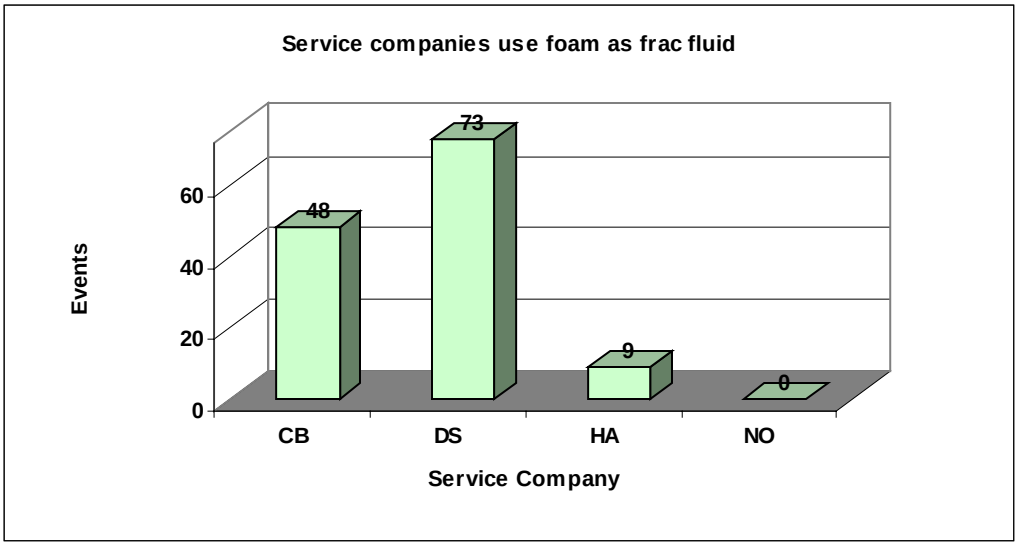

Figure 4.7: Service Companies use foam as frac fluid.

4.2.2.3 Distribution of Frac Fluid Volume and Fluid Pumping Rate:

Analyzing water, gel and foam as the frac fluids yield the following:

A. Water:

The minimum fluid volume used for water was 5 barrels, while the maximum volume was 1917 barrels. The average volume used was 693 barrels. The following figure shows the distribution of the used volume over equally divided intervals.

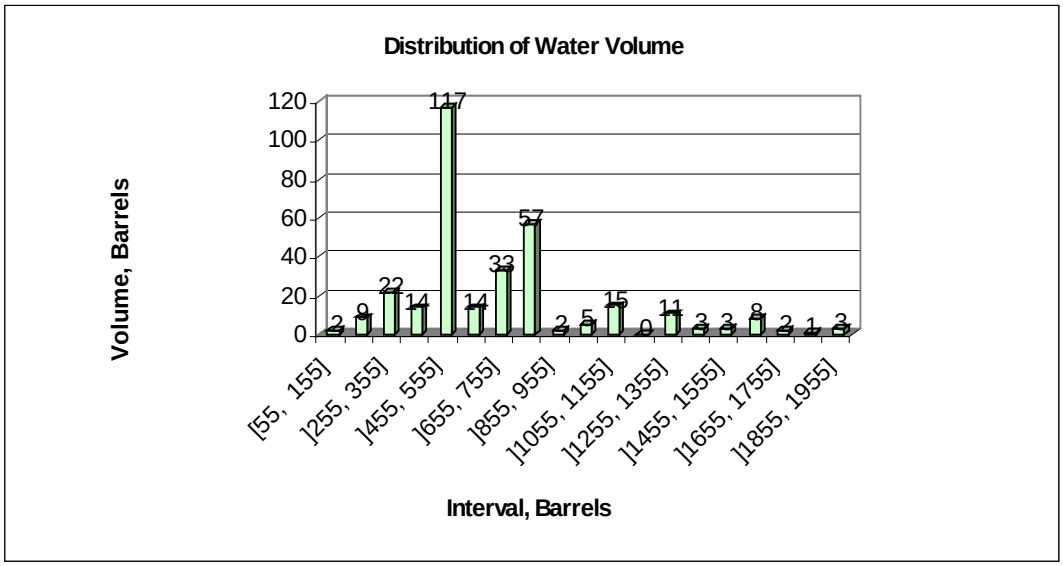

Figure 4.8: Distribution of water volume.

The minimum-pumping rate used in a water based frac job was 3.75 BPM, the maximum 70 BPM and the average 30.5 BPM.

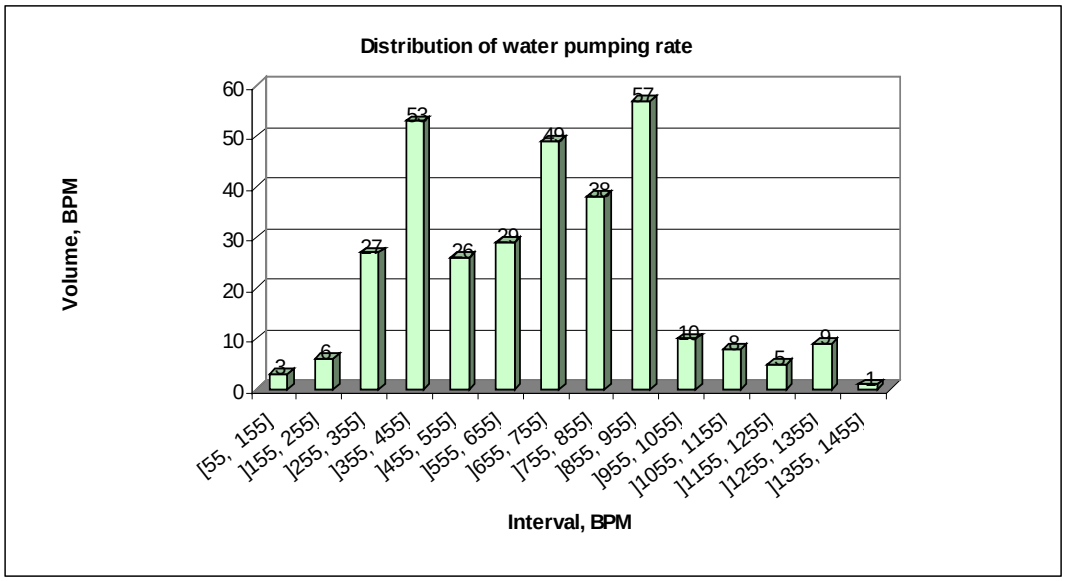

Figure 4.9: Distribution of water pumping rate.

B. Gel:

The minimum fluid volume used was 218 barrels, the maximum was 622 barrels, and the average was 544 barrels. Between the range of 515 to 565 , the gel reaches its highest volume.

For the pumping rate, the minimum value was 3.5 BMP and the maximum Value was 71 BMP. These two values are almost equal to the water minimum and maximum values, but the average was 15.8 BMP, which is half of the water Pumping rate average.

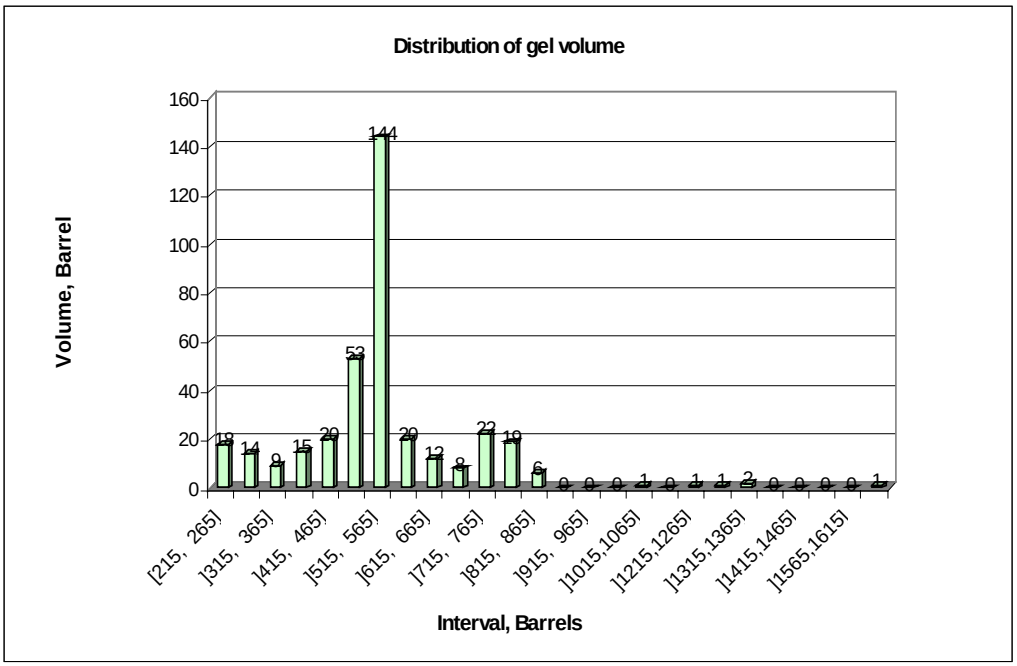

Figure 4.10: Distribution of gel volume.

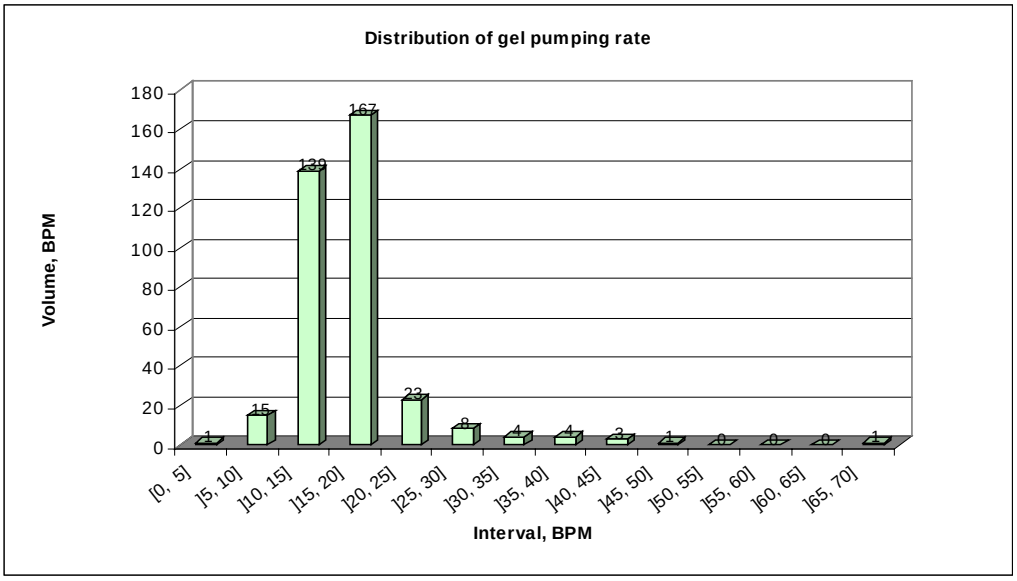

Figure 4.11: Distribution of gel pumping rate.

C. Foam:

The minimum fluid volume used for foam was 25 barrels, the maximum was 624 barrels and the average was 262 barrels. Figure 4.12 shows the distribution of the used volume.

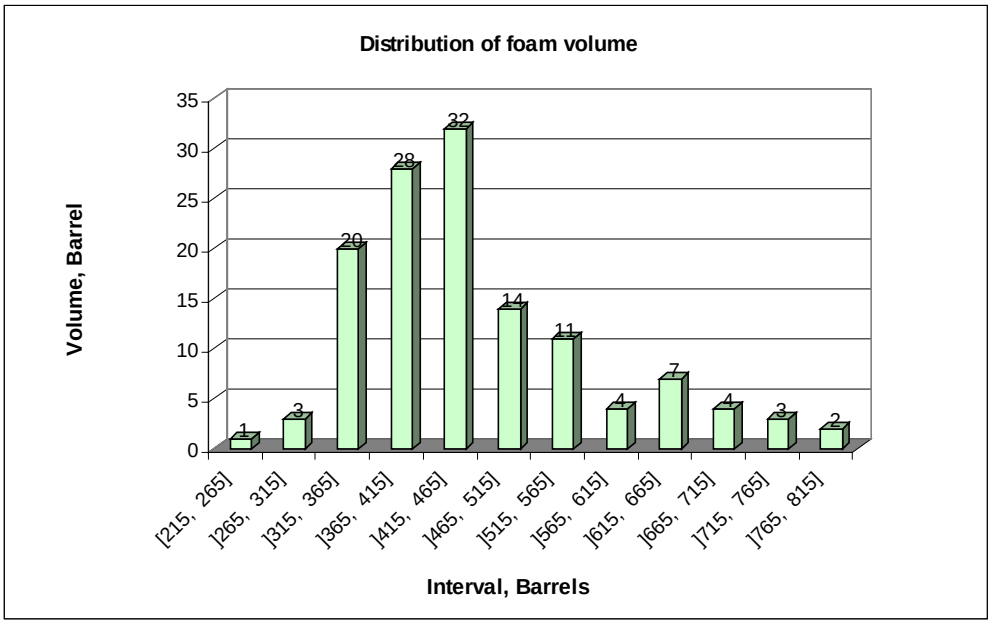

Figure 4.12: Distribution of foam volume

For the pumping rate, the minimum value was 2.4 BMP and the maximum value was 31 BMP. These two values are almost equal to the water minimum and maximum values, but the average was 14.56 BMP, which is half of the water pumping rate average.

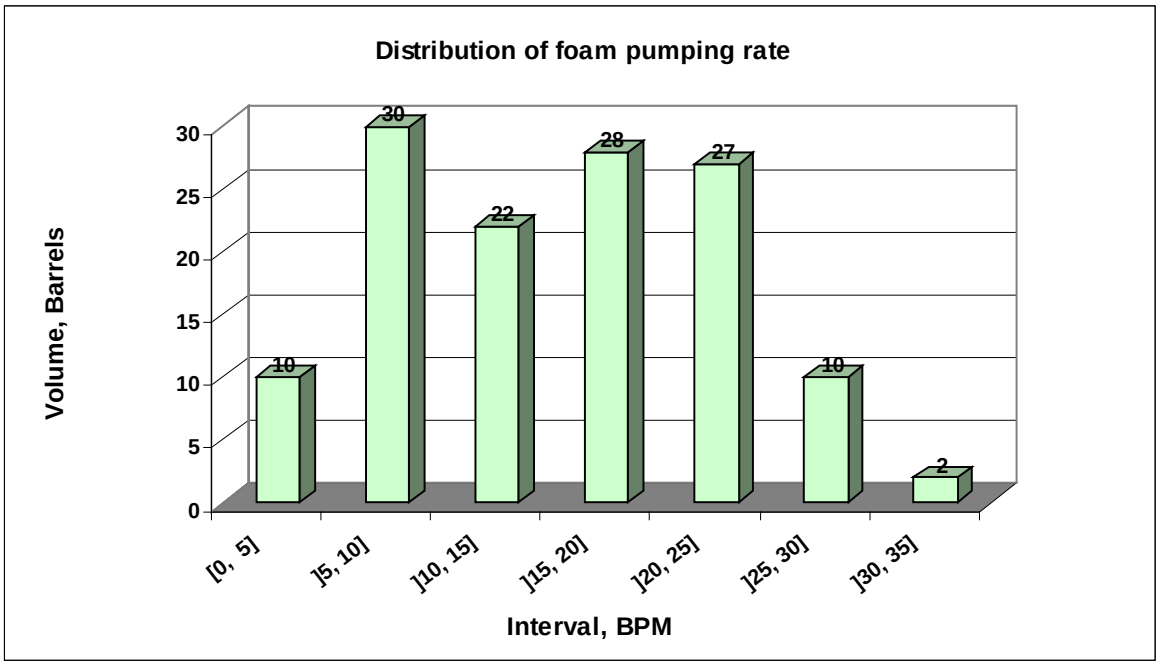

Figure 4.13: Distribution of foam pumping rate.

4.2.2.4 Distribution of Sand Volume:

The minimum sand volume used was 13 sacks, while the maximum volume was 670 sacks. The average volume used was 168 sacks. The following figure shows the distribution of the used sand volume over equally divided intervals.

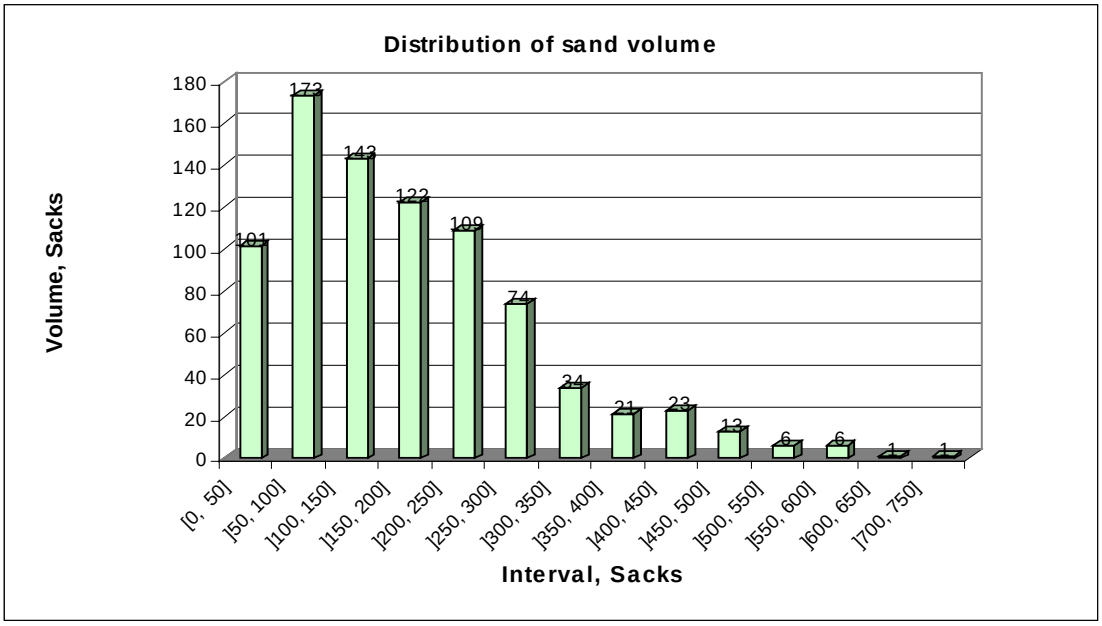

Figure 4.14: Distribution of sand volume.

5. Artificial Neural Network

In order to apply conventional modeling technique for optimization of fracture design great amount of data is required. For instance data like rock deformation, vertical in-situ stress distribution, permeability and porosity distribution, hydrocarbon saturation and pore pressure distribution, and many other data. Obtaining these data is very expensive and cost lots of time. Therefore applying a traditional method is extremely difficult due to the lack of data. For that reason using Artificial Neural Network is the best choice for such problems. Artificial Neural Network does not only mimic human brain in thinking it also investigates the hidden relation between parameters that engineers would argue their importance in conventional modeling and analysis process.

5.1 Preparing Data for Training

As previously mentioned the data for this application was divided into four categories according to the performed frac number. The first category contains all the wells that have been fraced at least once, while the second category contains all the wells that have been fraced at least twice and the third category contains all the wells that have been fraced at least three times.

Table 5.1, Shows the data categories based on frac number.

FRAC CATEGORY	NUMBER OF RECORDS
All Data	779
1st Frac	37
2nd Frac	526
3rd Frac	216

The trained Artificial Neural Networks based on these categorized sets of data are called specialized ANN. Therefore, a well that has been fraced three times will show up in first, second and third frac categories.

On the other hand all data are organized together, in one database file, to form the last category that is called the All Data category. The trained ANN base on All Data category is called the general ANN. The frac number is used as an input parameter in the training process, only for the All Data category.

As shown in following table, Table 6.1, there are a total of 779 records involved in the training procedure.

5.1.1 Selecting Input and Output Parameters:

The first step in the training process is the selection of inputs and the desired outputs parameters. The selection criteria were based on the well information, frac date and recipe information and isochronal test data. The table below shows the suggested input parameters chosen for this training process. "Post Frac Deliverability" was considered as the only output in the training process. "Post Frac Deliverability" is the peak value after the well clean up. It was carefully selected from the provided deliverability data.

Table 5.2, Input and output parameters for both specialized and general neural network

CATEGORY	PARAMETER NAME	TYPE	DESCRIPTION
Location	Well No	Input	Well Number
	X Coordinate	Input	Well x axis coordinate
	Y Coordinate	Input	Well y axis coordinate
	Elevation	Input	Well Head Elevation

Frac Time	Frac Date	Input	Frac Date (Numerical input)
	Years Before This Frac	Input	Years since last fraced performed
	Number of months before frac	Input	Number of months Deliverability was taken before this frac
Frac Recipe	Frac Fluid	Input	Frac fluid used
	Viscosity	Input	Viscosity of the frac fluid
	Fluid Volume	Input	Volume used in the frac job
	N2 Rate	Input	Nitrogen rate
	N2 Volume SCF	Input	Nitrogen Volume SCF
	Sand Volume	Input	Sand Volume (Sacks)
	Sand Concentration	Input	Sand Concentration
	Sand Mesh	Input	Sand mesh (Numerical input)
	Acid Volume	Input	Acid Volume
	Average Rate	Input	Frac fluid injection rate
	Service Company	Input	Company performed the job
Production Statistics	Q min	Input	Minimum deliverability before this frac
	Q max	Input	Maximum deliverability before this frac
	Q Average	Input	Average deliverability before this frac
	Q100 Before This Frac	Input	Deliverability before this frac

Input parameters can be divided into four categories, location category, frac time category, frac recipe category and production statistics category as shown in Table 6.2. The first category includes the unique well information such as the well number, well

location coordinates, and well elevation. This category gives each well a unique identity, and distinguishes it from other wells. Because there was no change in the frac number, frac number was not used as an input parameter in the 1st Frac, 2nd Frac and the 3rd Frac categories.

The third category includes the well frac job date, which in the Visual Basic Application will be the recent date, the number of years since last frac, and the number of months since last isochronal test where the last deliverability was recorded. For the first frac category the number of years before the frac was excluded since it is the first frac performed, but when it was recombined with other categories in the All Data category it was recorded as zero.

Since the objective of the training process is to determine the well performance after stimulation, input should include, besides the well information, the frac job recipe. The frac job recipe includes all parameters used to frac the pay zone such as fluid type, fluid volume and viscosity, sand volume and concentration, sand mesh, acid volume and average rate.

The last category contains the deliverability history. There were four input parameters in this category. Minimum deliverability, maximum deliverability, average deliverability, and finally the last deliverability recorded before the frac is performed. These statistics could change for the same well after each frac. For instance, as shown in figure 1, a well has a maximum deliverability of 1147 (MCF/D), and a minimum deliverability of 315 (MCF/D), after the first frac is performed as in Figure 1. These two values were used in the analysis of the second frac. After the second frac, the well deliverability reaches a new minimum value of 146 (MCF/D), although it maintains its

maximum. Therefore in the analysis of the third frac the maximum value stays as 1147 (MCF/D), while the minimum value is 146 (MCF/D). Average and last deliverability recorded changes in the analysis for each new frac.

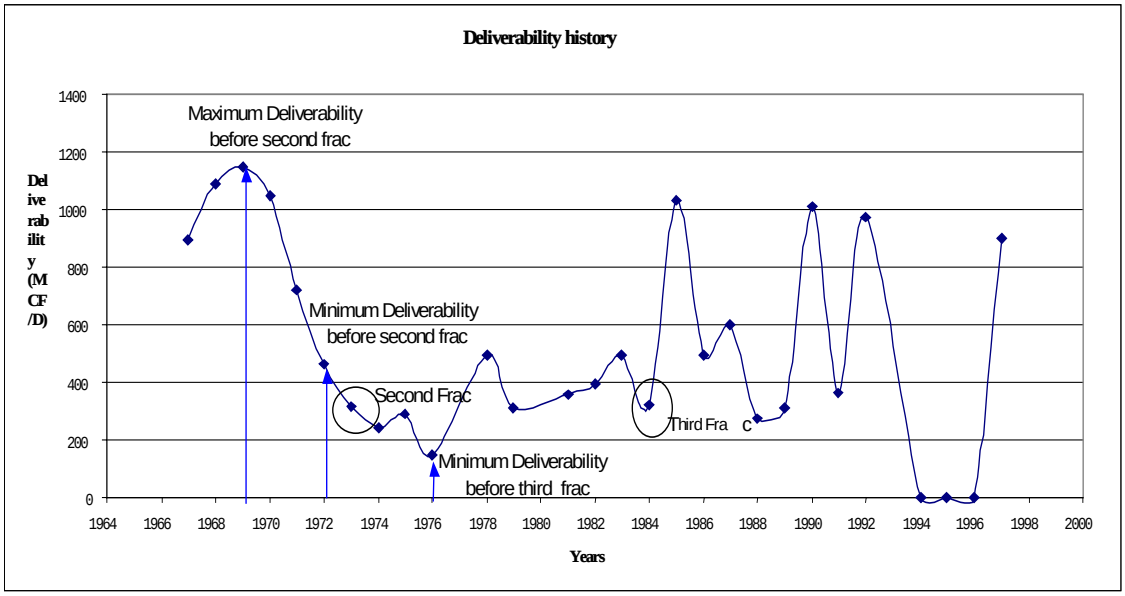

Figure 5.1: Shows the deliverability history and the change in the minimum and maximum value

5.2 Neural Networks:

"Since Artificial Neural Network, ANN, can learn, theoretically they can be trained to predict well response to certain stimulation[3]." Thus, a representative data that covers all the possibilities must be provided to the system in order to produce a well-trained ANN.

5.2.1 Selecting the Best Neural Net Structure:

Several neural net designs were tested. The most effective design was a network consisting of five layers. One input layer, one output layer, and three hidden layers. The input layer is fully connected to each hidden layer. The number of neurons in each hidden layer, the learning rate, and the momentum are subject to design and modification during the training process. Table 5.3 shows these modifications for each trained net of the specialized and general ANN.

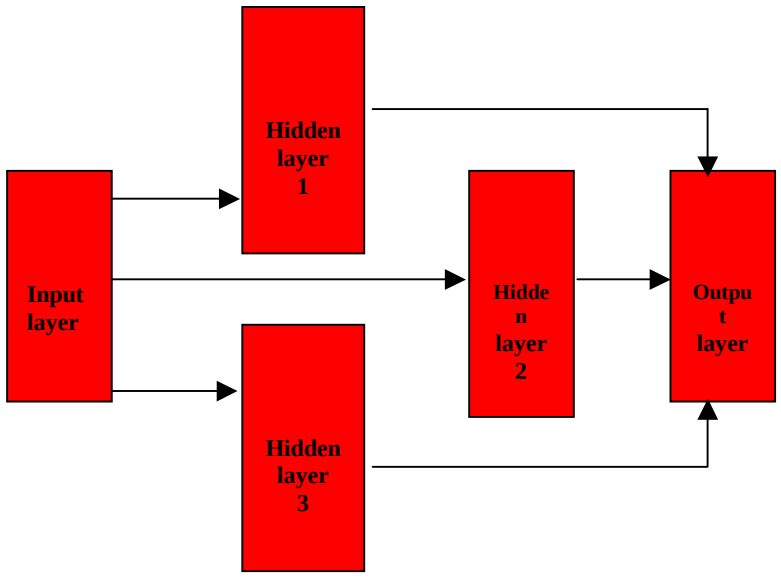

Figure 5.2: Artificial Neural Network structure used in the training process.

Table 5.3: Neural network training parameters

NET TYPE	NEURAL NETWORK TRAINING INFORMATION					
	Number of Inputs	Number of Outputs	Hidden layers Number	Number of Neurons in Hidden layers	Learning Rate	Momentum
1st Frac	21	1	3	13	0.1	0.2
2nd Frac	20	1	3	13	0.1	0.62
3rd Frac	16	1	3	13	0.1	0.74
All Frac	15	1	3	13	0.04	0.1

5.2.2 Training Process:

All input parameters are used in the training process, but can be manipulated accordingly to achieve desired results. For example, after all the input parameters are used in the training process, the training process is then repeated, this time systematically removing and replacing various parameters individually. This way, the effect of different parameters upon the training process can be observed. The correlation coefficient for training and testing sets decreased when removing some inputs parameter, and decreased when moving other parameters.

After reaching the highest correlation coefficient for both training and testing patterns, some other modification steps were applied. First, all the training is carried out using the whole set of data. After the out-layers are identified, the training process is then repeated, this time removing the out-layers' records for each related parameter from the training set.

During the training process a detailed and meticulous process was followed to identify and remove noise from the data set. This process also included the identification of unique data records that had a training value. This identification is done by plotting the actual deliverability versus the network deliverability and investigates the points with the highest error. It was assured that these records were used in the training patterns. Therefore all the data patterns were reexamined after random selection of training and testing sets to ensure that no unique data pattern had been removed from the training data set. This process helps in training the network for all different possibilities. The following figure presents some of these data records.

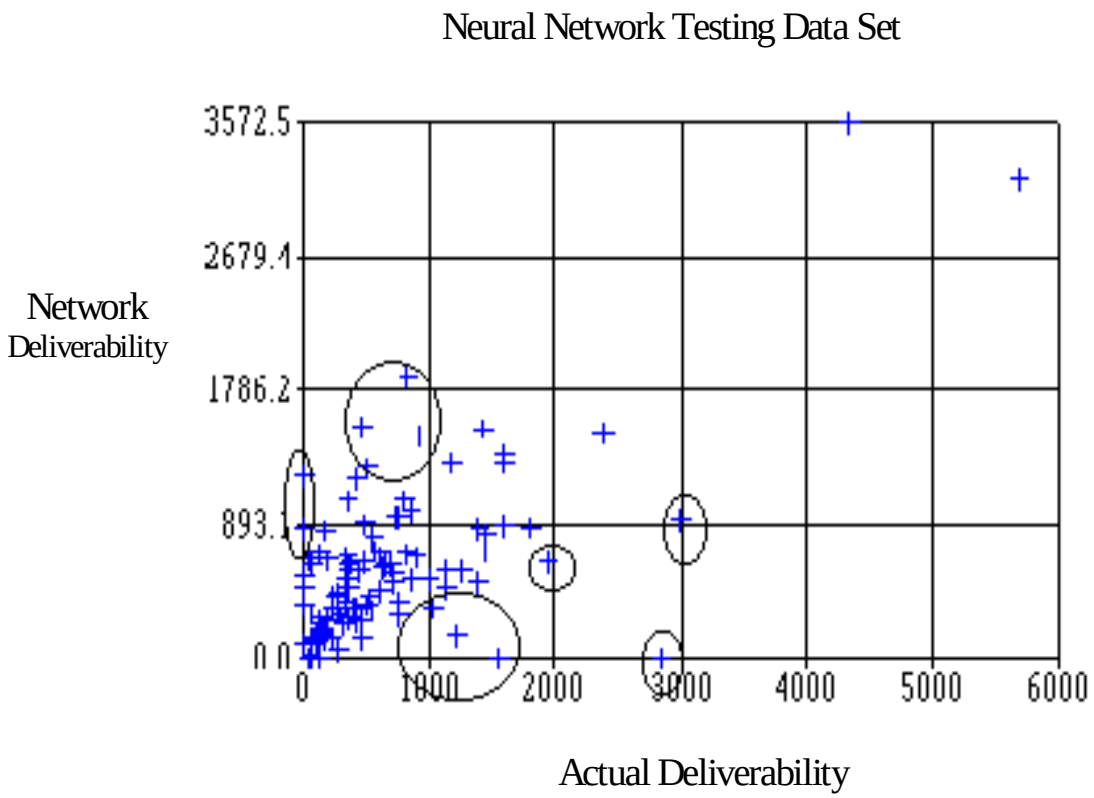

Figure 5.3: Neural network analysis for testing pattern identifies unique data records that had a training value.

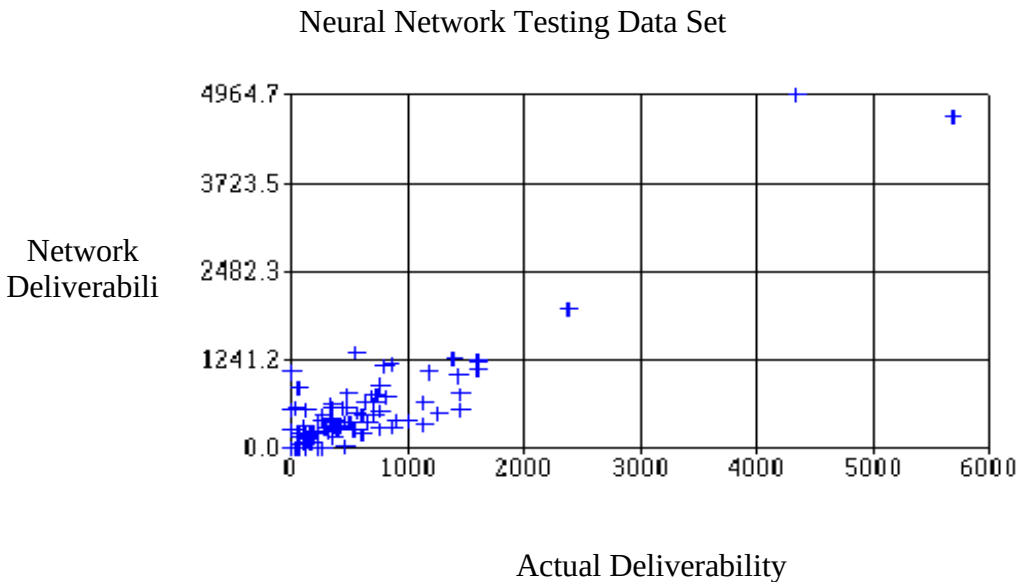

Figure 5.4: Neural network analysis for testing pattern after all unique data records with training value removed to the training pattern

The pervious mentioned records were not moved into the training pattern at once. Instead they were removed from the testing pattern to the training pattern one by one. The following figure shows the increase in the correlation coefficient of the second frac-testing pattern every time a record with training value is removed into the training pattern.

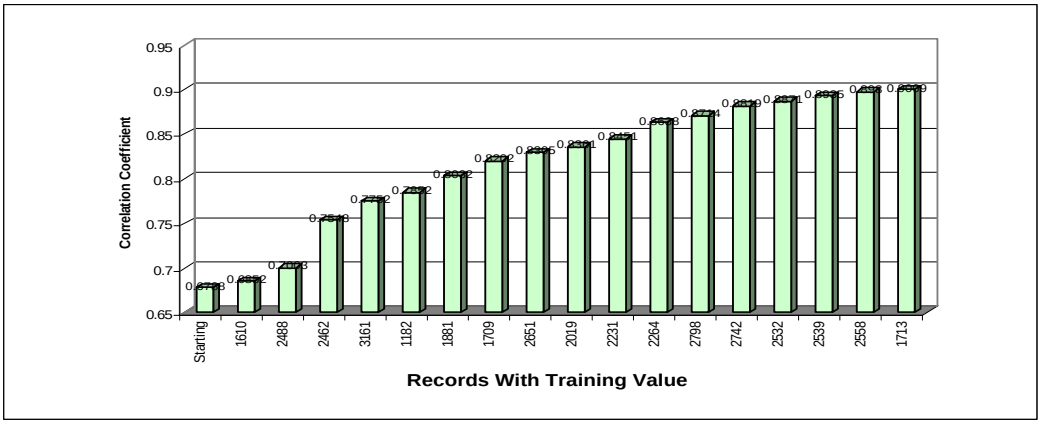

Figure 5.5: Correlation coefficient for second frac testing pattern.

Not all the removed records were replaced in the training pattern. Some of these records do not have any training value. These records were considered as noise and were excluded from the training process.

The table below presents the final results accomplished after the training process is finished

Table 5.4: Final results of the four data categories.

NETWORK PROCESSING	FIRST CATEGORY		SECOND CATEGORY		THIRD CATEGORY		FOURTH CATEGORY	
	Training	Testing	Training	Testing	Training	Testing	Training	Testing
Patterns processed:	26	10	414	94	171	40	611	144
R squared:	0.916	0.83	0.867	0.728	0.831	0.808	0.8151	0.789
r squared:	0.9179	0.833	0.882	0.734	0.8319	0.840	0.8199	0.7952
Correlation coefficient r:	0.959	0.913	0.939	0.856	0.912	0.916	0.9055	0.891

The following figures show the final training process results.

First Frac:

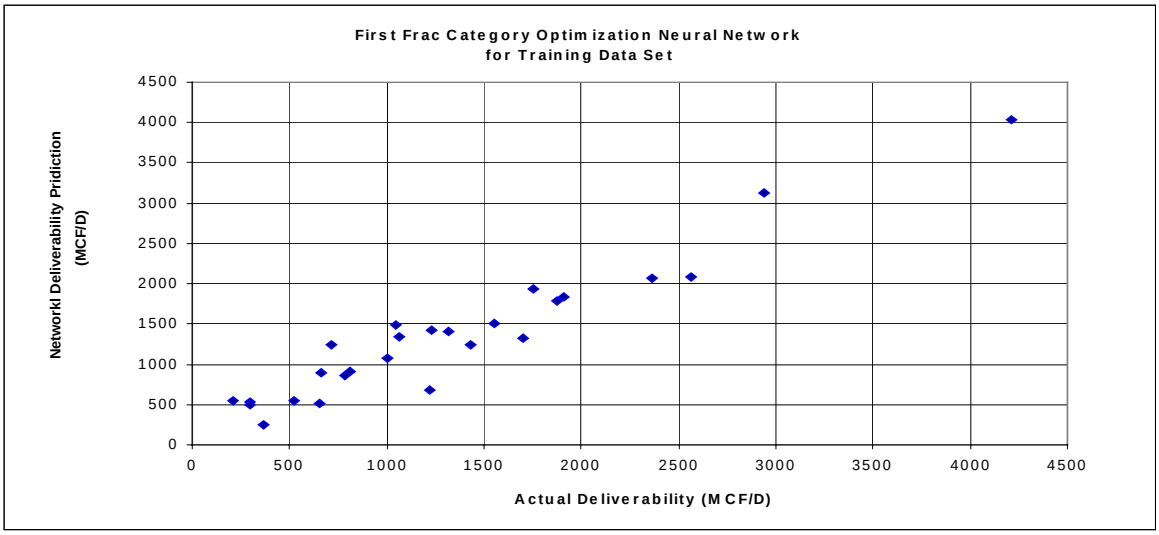

Figure 5.6: Deliverability training network for first frac with accuracy of 95.5%.

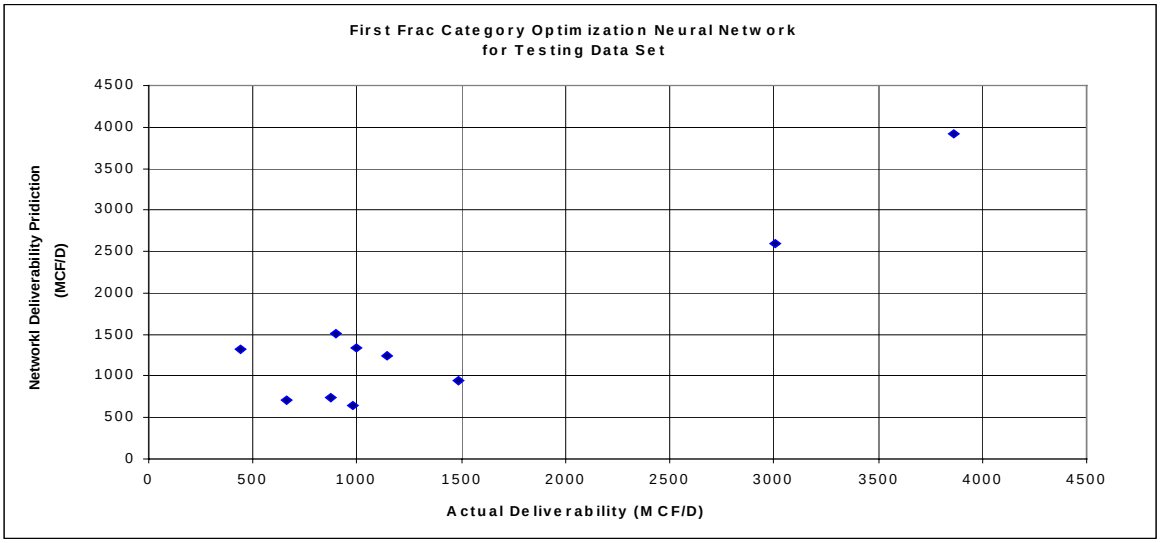

Figure 5.7: Deliverability testing network for first frac with accuracy of 91.3%.

Second Frac

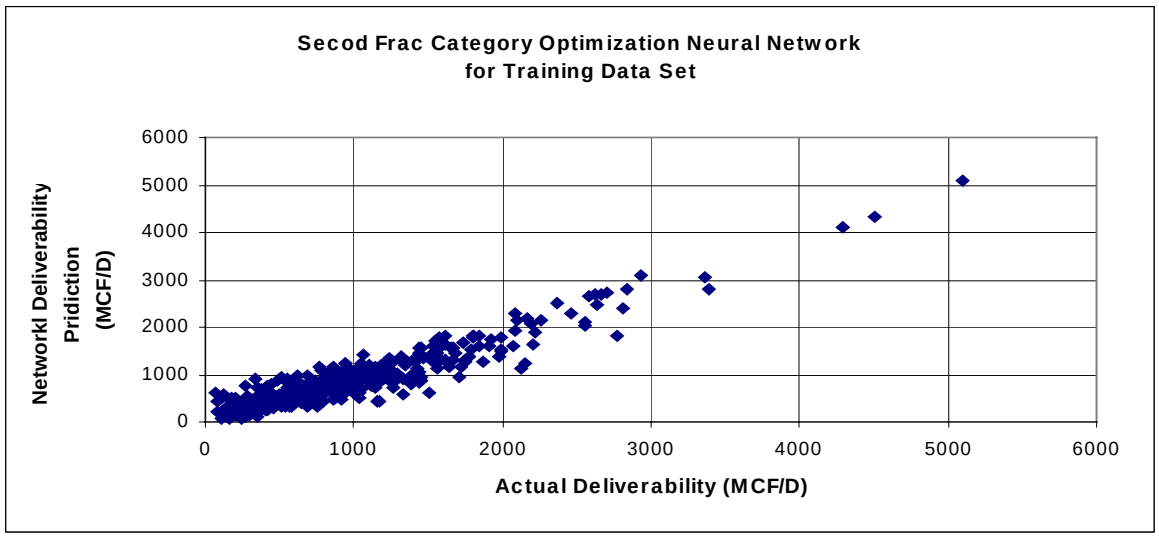

Figure 5.8: Deliverability training network for second frac with accuracy of 90.8%.

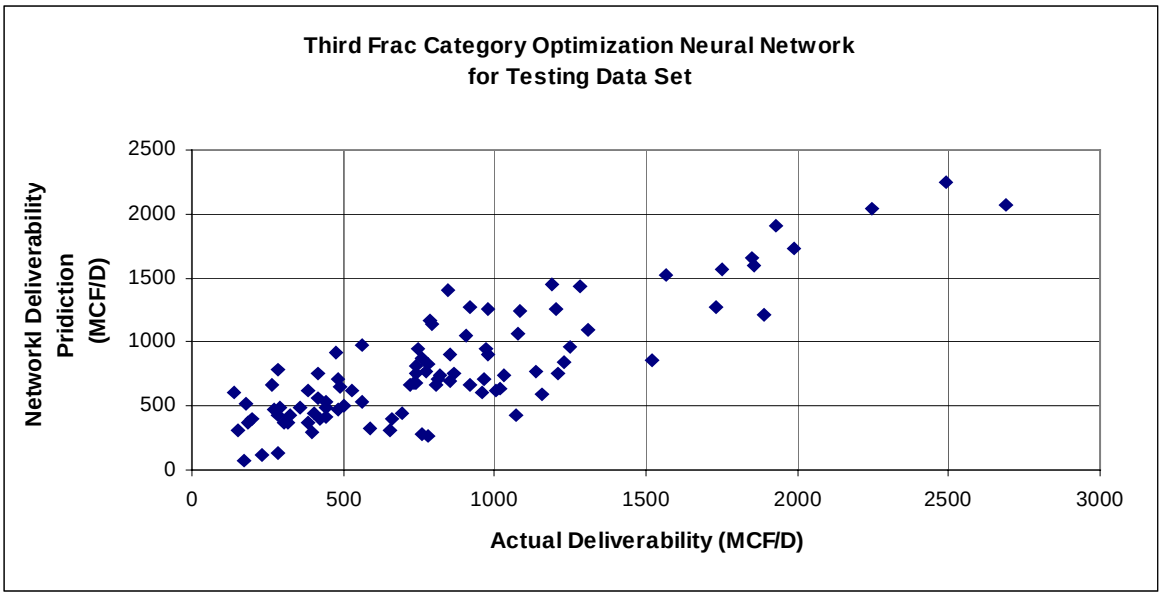

Figure 5.9: Deliverability testing network for second frac with accuracy of 90.3%.

Third Frac

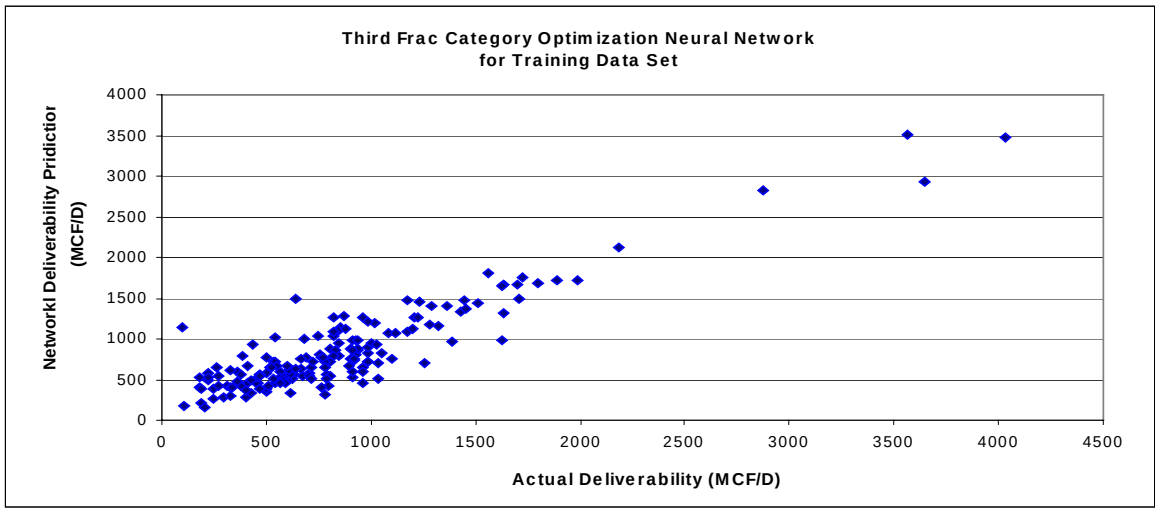

Figure 5.10: Deliverability training network for third frac with accuracy of 91.2%.

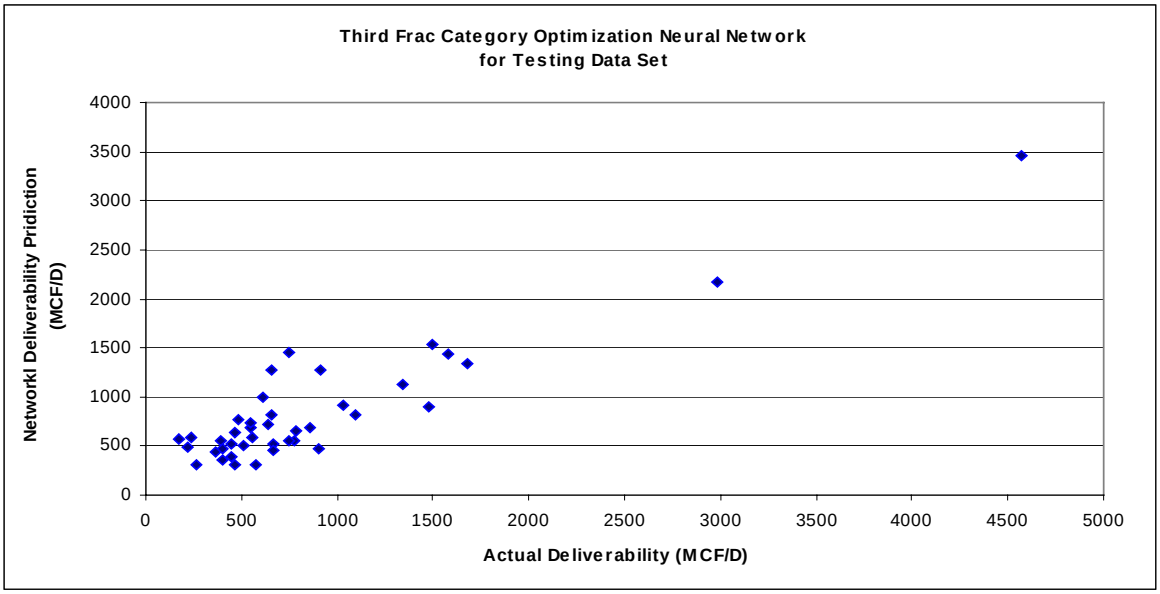

Figure 5.11: Deliverability testing network for third frac with accuracy of 91.6%.

All Frac

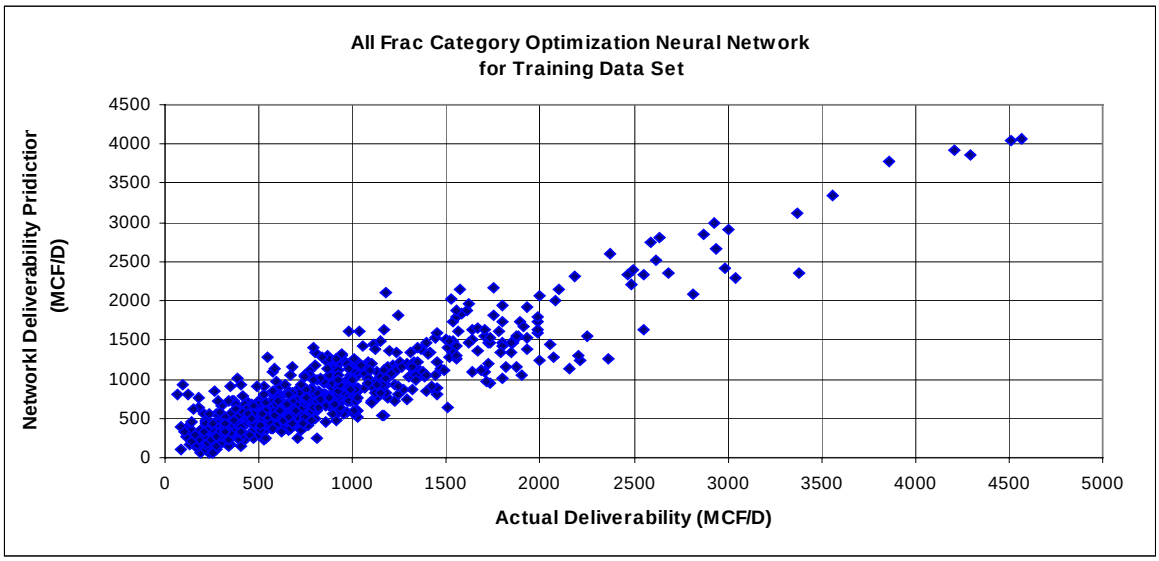

Figure 5.12: Deliverability training network for all frac with accuracy of 90.5%.

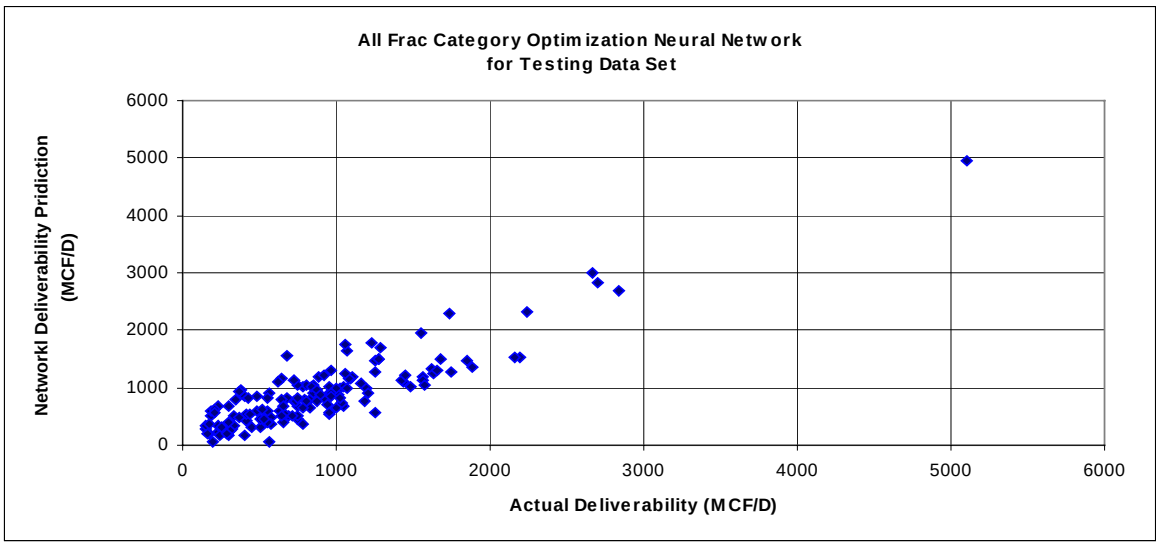

Figure 5.13: Deliverability testing network for all frac with accuracy of 89.1%.

6. Genetic Algorithms

Using conventional method for solving problem with discontinuity in the existing data and have a large number of input parameters involve in the solution, is very time consuming if not impossible. Therefore genetic algorithms are practical tool in solving such complex problems. They can cover large search spaces in a short amount of time without requiring an exact mathematical model of the problem**.** As mentioned previously the idea of Genetic Algorithms is to simulate the way nature uses evolution. The surviving chromosomes are used to generate the next generation of chromosomes. The good chromosomes are used to form new and hopefully better solutions in the next population, while the bad solutions are discarded. This process is repeated until no improvement in the Fitness value is observed.

Genetic Algorithm starts by generating many random solutions to the problem. These randomly created solutions are known as initial population. This population can be either a digital population or an analog population. The digitized population consists of a binary string, a series of zeroes and ones, while the analog population consists of a series of real numbers. Each of these numbers is located inside of a bit. The figure below shows examples of both kinds of strings.

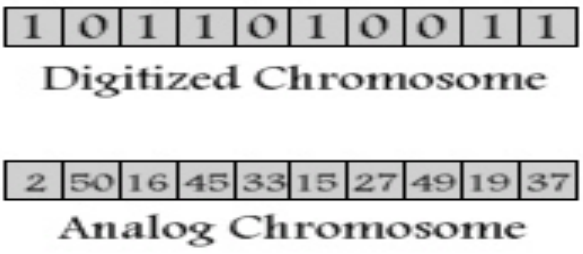

Figure 6.1: A digitized and analog chromosome

6.1 Developing the Genetic Algorithms

6.2 Analogs and Digital Initialization

The first step in developing a Genetic Algorithm is to calculate the chromosome length. In this application the length of the chromosome was calculated according to the chromosome type selected.

In the digitized chromosome type, the chromosome length depends on the total lengths of all the genes. The genes' length depends on the minimum and maximum values of the optimized parameter. The gene length can be measured according to the following equation:

$$L_{gene} = \text{round}\ \{\text{Log}_2\ [(\text{Max} - \text{Min}) * 10^m + 1]\}$$

Where:

L = Gene Length (Number of bits)

Min = Maximum value of the parameter

Max = Minimum value of the parameter

m = number of decimal digits

Table 6.1: Frac recipe optimization encoding.

INPUT PARAMETER	NUMBER OF BITS			
	First Frac	Second Frac	Third Frac	All Frac
Frac Fluid	2	2	2	2
Fluid Volume	11	11	10	11
N2 Rate	9	12	11	12
N2 Volume SCF	18	19	19	19
Sand Volume	9	9	9	9
Sand Concentration	4	4	4	4
Sand Mesh	2	3	2	3
Acid Volume	10	12	12	12
Average Rate	6	6	5	6
Service Company	2	2	2	2
Total Length	71	79	78	80

The total chromosome length is the sum of all the genes' lengths calculated.

In the analog chromosome type the length of the chromosome is equal to the number of the parameters used in the optimization. The reason is that each gene in the analog chromosome consists of only one bit. This bit value depends on the minimum and maximum values of the parameter to be optimized. Each bit value is randomly initialized and its value located between the minimum and maximum. Figure 6.1 shows an analog chromosome.

6.3 Fitness

After the chromosome is evaluated, a value is achieved. This value is known as the fitness value. The fitness value is a numerical value, which is proportional to the ability of the individual represented by that chromosome.

6.4 Chromosome Selection

Chromosome selection is very important as it is used to select both parents for the new chromosomes and to choose the chromosomes that survive and help to develop the next generation. During this phase of the genetic search, individuals are selected from the population to produce offspring chromosomes, which will make up the next generation. These parents are selected randomly from the population, using their fitness value, which favors the most fit individuals. Good individuals will probably be selected several times in a generation while weak ones may not be selected at all. In This project two types of selections are used:

6.4.1 Classical Selection Method, (Roulette-wheel selection).

The Classical Selection Method[5] of the new population is based on a probability distribution with respect to fitness evaluation. For this process a so-called "roulette wheel" is used. The process employs the following steps:

1. Calculate the fitness value eval (v_j) for each chromosome v_j (j=1 …pop_size)
2. Find the total population fitness

$$F = \sum_{j=1}^{Pop\_size} eval(v_j)$$

3. Calculate the probability of a selection p_j for each chromosome v_j (j=1…pop_size)

$$P_j = \frac{eval(v_j)}{F}$$

4. Calculate the cumulative probability q_j for each chromosome v_j (j=1…pop_size)

$$F = \sum_{k=1}^{j} (P_k)$$

The selection process is based on spinning the roulette wheel pop_size times; each time a single chromosome is selected for the new population. Specifically, this is how it works:

- Generate a random number r from the range [0 ...1]
- If r < q_1 then select the first chromosome (v1) otherwise select the j-th chromosome v_j (2<= j <= pop_size) such that $q_{j-1} < r < q_j$.

Obviously, some of the chromosomes will be selected more than once. This is due to the probability distribution, the best chromosomes get more copies, the average stay even, and the worst die off.

6.4.2 CVOR Selection Method

Considering the case of a population of 100 chromosomes, each chromosome has the same chance to be selected to generate the new population. Some of them will obviously be selected more than once. The CVOR selection methodology assigns a better probability to breed and generate to the offspring of the best chromosomes in the population. This can be done by selecting only the best x chromosomes in the population and using them to create the new population.

Basically each case is represented by a linear function with a characteristic slope. The smaller the number of the chromosomes selected the higher the values of the slope. Likewise, if more chromosomes are selected to generate a new population, the probability to breed for the strongest ones is lower.

Lets take the case of 25 chromosomes. Each of the four strongest chromosomes has the probability to breed with another four following. In this way eight new chromosomes are generated. The next six strongest chromosomes have a probability to breed of three. The process continues until the last chromosome in the population is allowed to breed.

6.5 Crossover

The crossover operator is very important in a genetic algorithm. It allows portions in chromosomes to be combined with portions of other chromosomes. The selected chromosomes are the parent chromosomes while the new generated chromosomes are the offspring. The mechanics of the crossover is to choose a corresponding point or points along the parent chromosomes at random. The parents used are swapped at each crossover point to produce two new full-length chromosomes. These two offspring inherit genes from each parent. Crossover is not necessarily applied to all pairs of individuals selected for mating when a choice is made. It depends on a probability specified by the user, this is typically between 0.5 and 1.0. If crossover is not applied, the offspring are simply duplications of the parents. There are three types of crossovers used in this project:

6.5.1 Classic Crossover

Classical crossover consists of generating a random number between one and the total number of bits in the chromosome that will determine where two the chromosomes will exchange parts.

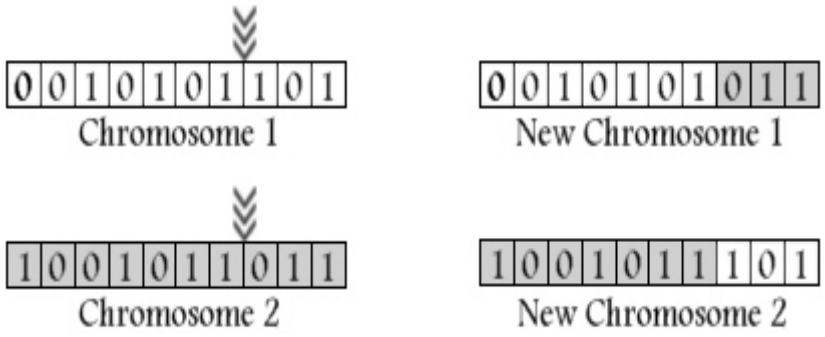

Figure 6.2: Digitize classical crossover mechanism.

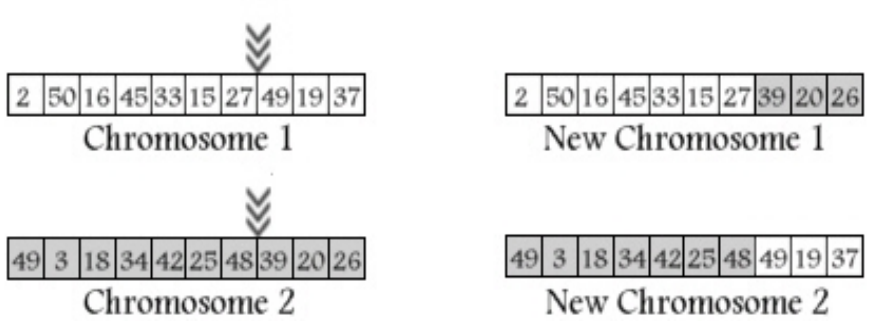

Figure 6.3: Analog classical crossover mechanism

6.5.2 Double Crossover

Double crossover consists of generating two random numbers, between one and total number of bits in the chromosome that will determine where the chromosomes are broken in to three parts and the middle part is exchanged.

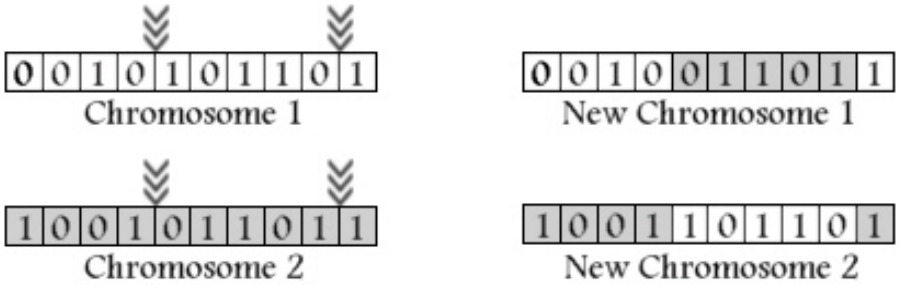

Figure 6.4. Digitize double crossover mechanism.

6.5.3 Uniform Crossover

Uniform crossover consists of generating an indefinite number of random numbers, between one and total number of bits in the chromosome that will represent the bits exchanged between the two chromosomes.

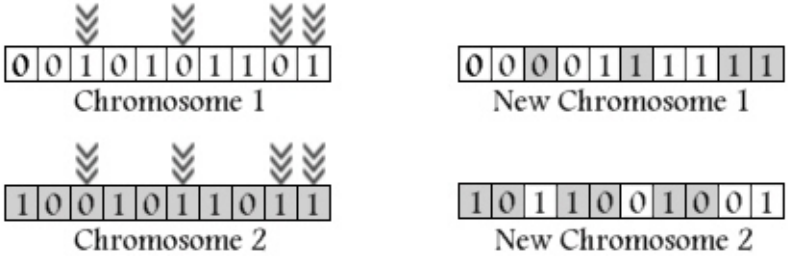

Figure 6.5: Digitize uniform crossover mechanism.

The analog part is not different but instead of swapping zeros and one we swap a total parameter.

6.7 Mutation Process

Mutation is another major operator that is used in genetic algorithms. It is implemented by randomly selecting a bit in the chromosome and altering its value of that bit from one to zero or zero to one in digitize chromosome, and populate a new value between parameter rang in the analog chromosome. This process will randomly change the encoded bit information for the newly created individual population. “Typically, mutation is performed with a small frequency of occurrence.”

Mutation probability is represented by a number between zero and one. The default probability of mutation used in the project was 0.25.

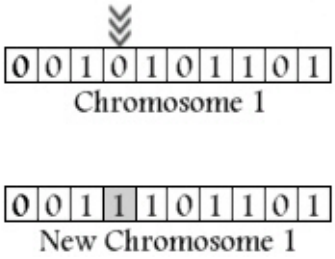

Figure 6.6: Digitize mutation mechanism.

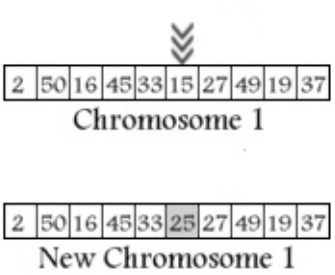

Figure 6.7: Analog mutation mechanism.

7. Fuzzy Expert System

Fuzzy Expert System is a system that applies the technical knowledge of an expert to solve a particular problem. "The Fuzzy Expert System uses a collection of fuzzy membership functions and rules, instead of Boolean Logic, to reason about data"[11]. The rules in a Fuzzy Expert System are in a linguistic form. They consist of a condition part and a conclusion part. The condition describes to what degree the rule applies, while the conclusion assigns a membership function to each of one or more output variables. The set of rules in a Fuzzy Expert System is known as the rule base or knowledge base. The Figure below shows how the Fuzzy Expert System works.

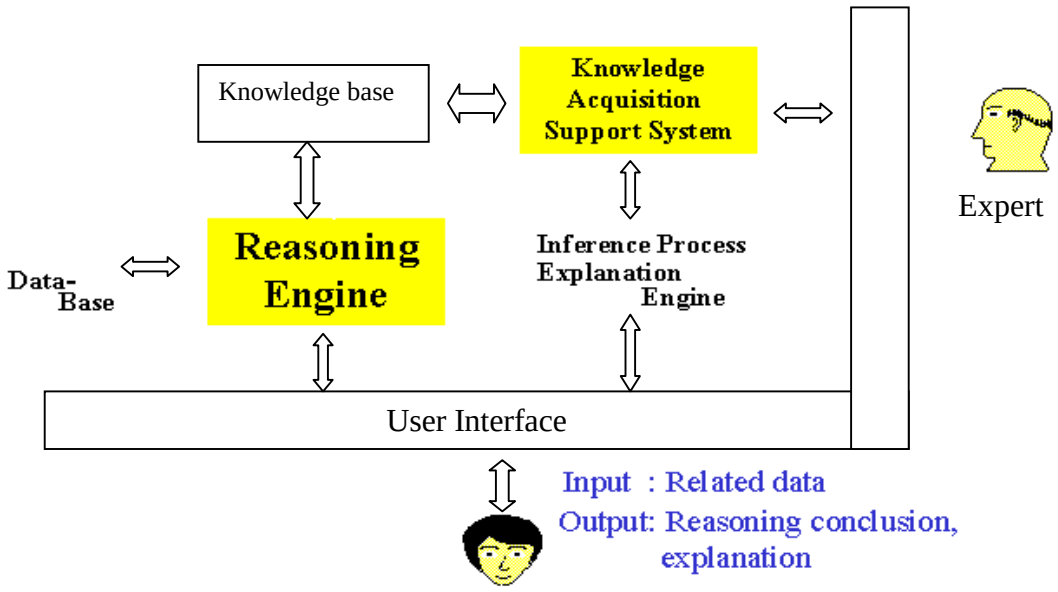

Figure 7.1: Shows the Fuzzy Expert System Steps

Before explaining the main process of the Fuzzy Expert System it is useful to know some of the common terms used in this process.

7.1 Universe of Discourse

Universe of Discourse is the total problem space. It can be divided into as many parts as desired. These parts, known as fuzzy sets, are shown in the figure below. In designing the Universe of Discourse, the consideration of all input possibility is taken. Therefore the Universe of Discourse usually starts from the minimum value and ends at the maximum value of the data set.

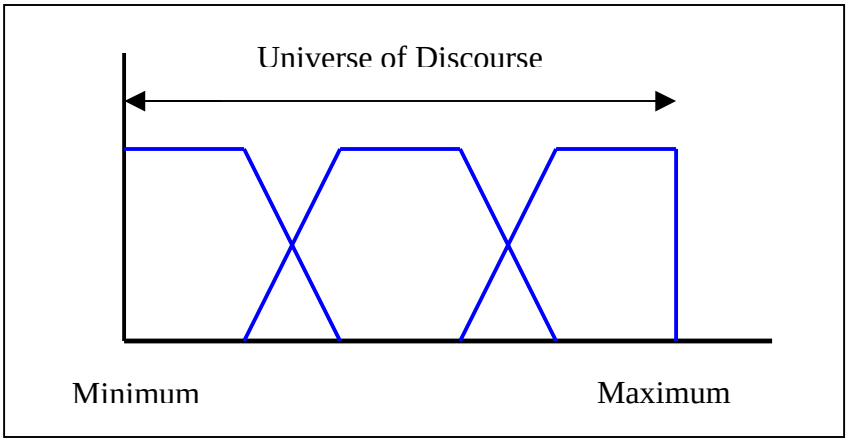

Figure 7.2: Universe of Discourse divided into three Fuzzy Sets

7.2 Fuzzy Set

A classical Set is basically defined as a collection of elements. In Fuzzy Sets, elements might partially belong to the set, or belong to the same set but in different degrees that are called grades of membership. To make an approximate comparison, these memberships are described in fuzzy numbers. A fuzzy number consists of a number and a linguistic description which later helps in setting the rules and explaining the decision making process.

7.2.1 Fuzzy Sets Initialization

Fuzzy sets can be initialized in two main ways:

a) **Intuition initialization:** Applying human experience in choosing the domain of the fuzzy set and its characteristics involving the linguistic variable.

b) **Inductive reasoning initialization:** is an automatic generation of membership functions.

7.2.2 Fuzzy Sets Presentation

Fuzzy Sets can be presented in many different ways. The most common representations are:

a) **Linear Representation:** Where the sets' true value grows until the sets' reach the point where the true and false values are equal. Then a decline begins where the degree of truth decreases and the degree of false increases.

b) **S-Curve (Sigmoid, Logistic)**: The same idea is still applied as a linear representation but the curve shape is different.

c) **Plateau (Trapezoid):** In the Plateau, the range of where the true value equals

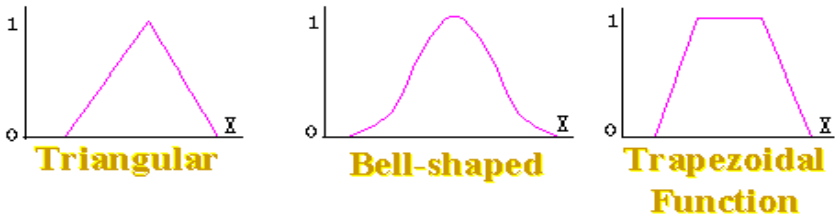

Figure 7.3: Shows the most common Sets shape

7.3 Fuzzy Sets Operator

An element is a member of a fuzzy set if it falls within the domain of the set. On the other hand due to the way the sets' overlap an element can be a member of two fuzzy sets. For that, operators like, **and,** and **or** are needed to explain the element grades of membership. The operators can be explained as following:

7.3.1 Conjunctive system of rules operator

The logical operator **and** operates the intersection in Fuzzy sets. The **and** operator is supported by taking the minimum of the truth membership.

The degree of membership of an element in the intersection of two fuzzy sets is the minimum, or the smaller value of its degree of membership individually in the two sets forming the intersection. For example if a well has 0.8 degree of membership in the set of candidates to be fractured, and 0.5 degree of membership in the set of likely to be a candidate. Then the well belongs to the set of likely to be a candidate.

Min {0.5, 0.8} = 0.5

7.3.2 Disjunctive system of rules operator

The logical operator **or** operates the union in Fuzzy sets. The **or** operator is supported by taking the maximum of the truth membership.

Max {.05, 0.8} = 0.8

7.4 Rules

The rules in a Fuzzy Expert System are usually in a linguistic form similar to the following:

IF ***the Average Stimulation Deliverability is LOW*** *&* ***the Year / Frac Ratio is HIGH*** *&* ***the Post Stimulation Deliverability is AVERAGE then the well is*** *likely a candidate.*

Where "Average Stimulation Deliverability", "Year Frac / Ratio" and "Post Stimulation Deliverability" are input variables, and likely a candidate is an output. *LOW, AVERAGE* and *HIGH* are the linguistic parts of the membership function defined on the inputs variable. The condition describes to what degree the rule applies, while the conclusion assigns a membership function to each of one or more output variables. The set of rules in a Fuzzy Expert System is known as the rule base or knowledge base.

7.5 Fuzzification Process

Fuzzification process is a mapping from the observed input to the fuzzy sets defined in the corresponding universe of discourse. In this process the input is translated from its crisp or a numerical value to a fuzzy output with linguistic term and a corresponding grade of membership.

Fuzzification is labeling the crisp value of a numerical input with a linguistic term and determining the corresponding grade of membership. Under Fuzzification, the membership functions defined on the input variables are applied to their actual values, to determine the degree of truth for each rule premise.

7.6 Inference Process

Inference Process is the decision-making logic, which determines fuzzy outputs corresponding to fuzzified inputs, with respect to the fuzzy rules. Fuzzy inference engine is built using rules based on linguistic statements. The expert must specify which rules

and which Fuzzy Sets Operators are used. When ever the input is not crystal clear and reflects belief rather than proof Inference take place employing inference steps similar to that of a human brain way of thinking. In the inference process, the truth-value for the premise of each rule is computed, and applied to the conclusion part of each rule. This results in one fuzzy subset to be assigned to each output variable for each rule.

7.7 Defuzzification Process

As a result of fuzzification and inference process several rules will be fired to generate a symbolic results or output fuzzy subset. The process of converting this output fuzzy subset to a crisp number is known as Defuzzification process. On other word Defuzzification is the calculation of a crisp numerical value as the controller output based on the symbolic results.

There are many ways to convert non-fuzzy output to crisp. The most used methods are Center of Area, centroid, Mean of Maximum." Centroid method is the most used methods. In centroid method, the crisp value of the output variable is computed by finding the value of the center of gravity of the membership function for the fuzzy value. In the Mean of Maximum method, one of the variable values at which the fuzzy subset has its maximum truth-value is chosen as the crisp value for the output variable.

$$c_i = \frac{\sum_k^m \mu_{ik} * (x_i) * y_k}{\sum_k^m \mu_{ik} * (x_i)}$$

Where:

μ_{ik} = Fuzzy membership function

x_{ik} = Input variable

7.8 Implementation

After using the genetic algorithm coupled with the neural net to generate a list of candidates, an expert opinion is important to consider which of the selected candidates can be more voluble that the others. Therefore the use of Fuzzy Expert Decision becomes important in this stage of the solution.

7.8.1 Input Selection

In order to implement the Fuzzy Expert System in the current project, defining the input parameters was the first step. Consequently, for the present problem three inputs were selected. They are:

1. Average Stimulation Deliverability (MSCF/D). In order to compare between the deliverability of two wells, one must compare between their average deliverability histories.
2. Year - Frac Number Ratio. Well deliverability is affected by two input parameters, which are “Years since last frac” and “ Frac Number”. In the case of the first parameter one must notice that: if number of “Years since last frac” increases the well will be much better candidate for fracturing. The second input parameter represents the number of times the well has been fraced until current analysis.
3. Post Stimulation Deliverability (MSCF/D). This represents the deliverability of the well generated by the genetic algorithm optimization.

To set the Universe of Discourse for each input parameter, range of the parameter for the entire set of data is scanned and only the lowest and the highest values are selected. This range is the total problem space also known as the Universe of Discourse.

Table 7.1: Shows the range of each input parameter for Fuzzy Expert System according to the frac number.

FRAC NO	INPUT PARAMETER	MINIMUM	MAXIMUM
First Frac	Average Deliverability	36	2796
	Year Frac Ratio	0	33
	Post Frac Deliverability	205	4209
Second Frac	Average Deliverability	41	4256
	Year Frac Ratio	1	19
	Post Frac Deliverability	70	5105
Third Frac	Average Deliverability	94	2197
	Year Frac Ratio	0.33	10
	Post Frac Deliverability	97	4570
All Frac	Average Deliverability	36	4256
	Year Frac Ratio	0	33
	Post Frac Deliverability	70	5105

7.8.2 Fuzzy Set Initialization

The Universe of Discourse in the application is divided into three sets. These sets are defined by trapezoidal shape with height of 1.0. These sets can be initialized automatically or as user input or combination of both.

In the automatic initialization, the division of the range into different number of sets was performs by dividing the entire Universe of Discourse by the desired number of sets. The range of each set can be calculated using the following mathematical model:

$$\text{Set Range} = \frac{\text{Universe of Discourse}}{N}$$

Where:

N = desired number of sets

Then each set is divided into three equal intervals using the following relation:

$$Interval = \frac{Max - Min}{(2 * N) - 1}$$

Where the Max and the Min values are the highest and the lowest values of each individual parameter.

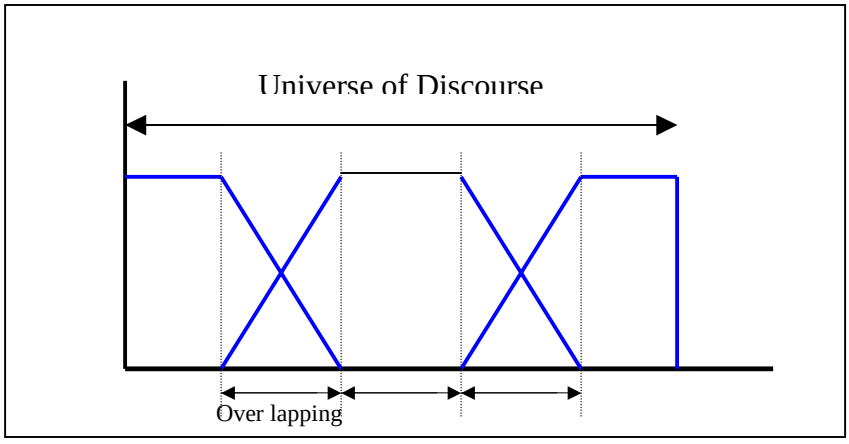

Figure 7.4: Shows how each set is divided into three equal intervals and the sets over overlapping

Each set starts from the end of the second interval of the previous set. The overlapping between all the sets is equal to 30%. Notice that the first and the last sets are divided into only two intervals. The reason for that was the first set should always start with truth-value of one while the last set always end with false value of one.

Each of these subsets was labeled differently from the other subsets in the Universe of Discourse. This labeling system was used later in constructing the Rule System. In the default case the subsets were named as follows:

- Low Subset - first set starting from the minimum value until 1/3 of the entire range.
- Average Subset – is the middle set, and
- High Subset – last set starting from 2/3 the range until the maximum value.

7.8.3 Defining the Rules and Aggregation Process

Rules are the heart of the Expert Decision System. They play an important role in the decision-making logic. They also provide a descriptive presentation of the problem output. Rules are based on linguistic statements that can be constructed as a function of human experts. An example of the rules construction is given below:.

- *IF* ***the Average Stimulation Deliverability is LOW with a value of 0.6*** *&* ***the Year Frac Ratio is HIGH with a value of 0.4*** *&* ***the Post Stimulation Deliverability is Average with a value of 0.8*** *then* ***the well is*** *likely a candidate by value of 0.4*

In the same way, the **Or** operator takes the "control' of the process when more than one rule provide the same linguistic output but with different fuzzified value.

Lets consider the next example: assuming that three rules are fired, and they all provide the same linguistic output " likely candidate" but with different fuzzified value {0.4, 0.6, 0.2} then the rule is likely candidate with fuzzified value of 0.6 degree.

The outputs of this step consist of two parts, symbolic part and numerical part. Also they form fuzzy subset output.

7.8.4 Degree of Confidence

The process of converting the fuzzy output to a crisp value is known as Defuzzification process. The crisp resulting from the de-fuzzification process is called the degree of confidant.

The fuzzy output is converted to a crisp value using the average weight method. For example for three fuzzy outputs of 0.25 candidate, 0.7 likely candidate and 0.157 not candidate the degree of confidence is 53%.

Fuzzy system, provide a rich and meaningful addition to standard logic. The application generated using Fuzzy Expert System helps in mimic the human reasoning. Since we are using Fuzzy Expert System the accuracy of a solution we reach depends on our confidence on the system of rules that was established.

8. Conclusion

The study shows that virtual intelligence can be used successfully and efficiently to select candidate wells for refracturing in a Gas Storage Field. An application tool based on virtual intelligence technique was successfully developed. The Tool developed can now help engineers not only to identify well need to be fraced, but also in designing frac design in less time and resources. The integrated virtual intelligence technique includes four components, Databases component, artificial neural network component, genetic algorithm, fuzzy decision support component.

The Databases provides a detail look at the available data. This database will assist engineers to structure the available data in a way that is easy to be update. The database provides the needed data and work as storage facility for the other three components of the application.

An artificial neural network component that provides a neural model of the simulation process and the formation respond in the Clinton sand.

A genetic algorithm that use the neural model, the second component, as a fitness function to identify the best frac design that result in highest possible post-frac deliverability.

Finally, a fuzzy decision support component that combine the data driven portion of the analysis, the second and the third component, and integrated with the engineering expertise and knowledge related to the Clinton sand to assist engineer in making the final decision in identifying refracturing candidates for each year.

The design of the application is flexible and updateable. New modification on the application front end, interface, code, or backend, can be easily modified. All neural networks can be retrained and replaced in the application.

References

1. Platon, V.: “Virtual Intelligence Application in Gas Storage Well Stimulation Design and Optimization” MS Thesis, West Virginia University, Morgantown WV, 1998.
2. Balan, B.: “A hybrid Neuro-Genetic Approach Optimization of Frac Treatment Design of Gas Storage Wells” MS Thesis, West Virginia University, Morgantown WV, 1996.
3. Popa, A.: “A hybrid Neuro-Genetic Approach Optimization of Frac Treatment Design of Gas Storage Wells” MS Thesis, West Virginia University, Morgantown WV, 1999.
4. Fausett, L.: “ Fundamentals of neural Networks Architecture, Algorithms, and Applications” Prentice Hall, Englewood Cliffs, NJ, 1994
5. Michalewics, Z.: “Genetic Algorithm+ Data Structure =Evolution Programs”, Springer-Verlag, Berlin, 1992.
6. Haupt, R. & Haupt, S.: “Practical Genetic Algorithms” John Wiley & Sons, INC, New York, 1998
7. Ross, T.: “Fuzzy logic with Engineering Applications” McGraw-Hill, Inc, New York 1995.
8. "Genetic "Genetic Algorithms in Search, Optimization and Machine Learning" D.E. Goldberg. http://www2.psy.uq.edu.au/~brainwav/manual/BackProp.html
9. "An Overview of Genetic Algorithms" - D Beasley, D Bull, R Martin. http://www.us.vergenet.net/~horms/papers/honours1/html/node56.html
10. Nelson M. and Illigworth W.I.:”A Practical Guide to Neural Nets”, Addison-Wesley Publishing Company Inc., 1992.

11. Hykin S. :”Neural Networks. A Comprehensive Foundation”, Macmillan College Publishing Inc., New York, 1994.

APPENDIX –Visual Basic Code

Splash Form

```
Option Explicit

Private Sub Form_Load()
   Me.Left = (Screen.Width - Width) / 2
   Me.Top = (Screen.Height - Height) / 2
End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub SSPanel1_Click(Index As Integer)

   If Index = 0 Then
      FrmEog_Frac_info.Show
   ElseIf Index = 1 Then
      frmNeural_GA.Show
   ElseIf Index = 2 Then
      frmFuzzy.Show

   End If
   Me.Hide

End Sub
```

Start Module

```
Global Inputs%, Outputs%                  ' the number of inputs and outputs of the
network
   Global Netnumber%                       '  the number assigned to this net
   Global Netisopen%                      '  makes sure nets are only opened once
   Global Frac_ID()                      As String
   Public DatReturen                     As DataType
   Public DataFor                       As DataUse
   Public Getting_Datafor                  As GettingDatafor
   Public Data_Existance                   As DataExistance
   Public GA_Type                        As GAType
   Public Chromosome_SelectionType              As Chromosome_Selection
   Public Cross_OverType                   As CrossOverType
   Public SubSet_Type                     As SubSetType
   Public Diffazification_Type               As DiffazificationType
   ' Function prototypes for the NSHELL2.DLL
   Declare Function OpenNet% Lib "Ns2-32.dll" (ByVal Path$, Netnumber%, Inputs%,
Outputs%)
```

```
   Declare Function FireNet% Lib "Ns2-32.dll" (Netnumber%, Inputs_array#,
Outputs_array#)
   Declare Function CloseNet% Lib "Ns2-32.dll" (Netnumber%)


Sub Main()
   frmSplash.Show
   frmSplash.Refresh
End Sub
```

Well_&_Frac Information Form

```
Option Explicit
   'References ADO 2.5
   'ADO Objects Used
   Private sConnection       As ADODB.Connection
   Private sRecordset        As ADODB.Recordset
   Private sCommand          As ADODB.Command
   Private sParameter        As ADODB.Parameter
   Private strSQL            As String
   Dim myData()              As Variant
   Private AddNewFlag        As Boolean

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Form_Load()


   'Form location
   Me.Left = (Screen.Width - Me.Width) / 2
   Me.Top = (Screen.Height - Me.Height) / 2
   'Start Tab
   Me.SSTab_FracDesign.Tab = 0
   Me.cmd_addNEw_Deliverability.Top = 4560
   Me.cmd_Cancel.Visible = False
   Me.MSHFlexGrid_Deliverability.Height = 4000


   'Openning the database
   Call Open_Database

   With ADO_Well_Info
      'stating that the form was just open
      AddNewFlag = True
      'Sorting data from database
      .Recordset.Sort = "WELL_NO ASC"
   End With
```

```
    AddNewFlag = False

    Call DG_Well_No_RowColChange(1, 1)

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Open_Database()

    'Setting up the connection String to the MDB File
    Set sConnection = New ADODB.Connection
    'Open the connection to the DB
    With sConnection
      'Telling ADO to use Jolt
      .Provider = "Microsoft.Jet.OLEDB.4.0"
      .ConnectionString = App.Path & "\EOGDB.mdb"
      .CursorLocation = adUseClient
      .Open
    End With

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Open_RecordSet(mySQL)

    'Setting up the Recordset
    Set sRecordset = New Recordset

     With sRecordset
          .Source = mySQL
          .ActiveConnection = sConnection
          .LockType = adLockOptimistic
          .CursorType = adOpenKeyset
          .Open
    End With

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub DG_Well_No_RowColChange(LastRow As Variant, ByVal LastCol As
Integer)

    On Error GoTo Proc_Err

    Dim i As Integer
    'If the form was just open exit this procdure
    If AddNewFlag = True Then Exit Sub
```

```
    Call Deliverability

    'Getting the frac job information
    'Preparing SQL string to get frac information
    strSQL = "SELECT * FROM tbl_Fracrecipe WHERE [WELL No]='" &
txt_Well_No.Text & "'"
    strSQL = strSQL + " ORDER BY [FRAC No]"

    Call Open_RecordSet(strSQL)
    'Counting the records #
    Me.txt_No_of_Frac.Text = sRecordset.RecordCount

    'condition if there is no record available
    If sRecordset.EOF = True And sRecordset.BOF = True Then
       MsgBox ("There is no Frac Recipe Available For Well " & txt_Well_No.Text)
          'Empty all text box
          For i = Me.txt_Add_Frac.LBound To Me.txt_Add_Frac.UBound
            Me.txt_Add_Frac(i).Text = ""
          Next
       Exit Sub
    Else
       'Looping through all the text boxes
       For i = Me.txt_Add_Frac.LBound To Me.txt_Add_Frac.UBound
          'if the value is Null then keep the text box empty
          If IsNull(sRecordset.Fields(i)) Then
             Me.txt_Add_Frac(i).Text = ""
          ElseIf sRecordset.Fields(i) <> "" Then
            txt_Add_Frac(i).Text = sRecordset.Fields(i)
          End If
       Next
    End If

    Me.cmd_AddFracRecipe.Caption = "Add New Frac Recipe"
    Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Deliverability()

    'This Code gets all deliverablity data store them inside grid
    ' and then draw the graph
```

```
    'Writting the SQL for each well excluding and value = zero
    ' and ordering them according to the year
    strSQL = "SELECT YEAR, Q100 FROM tbl_Q100 WHERE [WeLL No]='" &
txt_Well_No.Text & "' And [Q100] <> 0"
    strSQL = strSQL + " ORDER BY [YEAR]"
    'Open record set
    Call Open_RecordSet(strSQL)

    'if there was no Deliverability values then alert
    If sRecordset.RecordCount = 0 Then
        MsgBox ("There is no Deliverability History For Well " & txt_Well_No.Text)
    End If

    'Preparing the grid
    With Me.MSHFlexGrid_Deliverability
        .ColWidth(0) = 520
        .ColWidth(1) = 530
        'Attaching the record set to the flex grid
        Set Me.MSHFlexGrid_Deliverability.DataSource = sRecordset.DataSource
        Me.MSChart1.RowCount = Me.MSHFlexGrid_Deliverability.Rows
        Me.MSChart1.RowLabelCount = Me.MSHFlexGrid_Deliverability.Rows
        Me.MSChart1.ColumnCount = Me.MSHFlexGrid_Deliverability.Cols
        Set Me.MSChart1.DataSource = sRecordset.DataSource

        Call Scaling_MS_Curves

    End With

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Add_Well_Click()

    On Error GoTo Proc_Err

    If Me.cmd_Add_Well.Caption = "Add New Well" Then
        Me.cmd_Add_Well.Caption = "Submit New Well"
        AddNewFlag = True
        With ADO_Well_Info
            .Recordset.AddNew
        End With
        AddNewFlag = False
        Exit Sub
    Else
        Me.cmd_Add_Well.Caption = "Add New Well"
        Exit Sub
```

```
    End If

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmdUpdate_Click()

    On Error GoTo Proc_Err

    With ADO_Well_Info
        .Recordset.Update
    End With

 Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Delete_Well_Click()

    On Error GoTo Proc_Err



    strSQL = "SELECT * FROM tbl_Fracrecipe WHERE [WELL No]= '" &
txt_Well_No.Text & "'"
    Call Open_RecordSet(strSQL)

    If sRecordset.RecordCount > 0 Then
        MsgBox (" This Well have " & sRecordset.RecordCount & " Frac Recipe, Delete
All Frac Recipes First ")
    Else
        With ADO_Well_Info
            .Recordset.Delete
        End With
    End If

Exit Sub
```

```
Proc_Err:
   MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Scaling_MS_Curves()

   'Scaling the MS chart

   With MSChart1.Plot.Axis(VtChAxisIdX)
      .AxisTitle.Text = "Years"
     .AxisScale.Type = VtChScaleTypeLinear
      .AxisGrid.MinorPen.Style = VtPenStyleDashed
      .CategoryScale.Auto = True
   End With

   With MSChart1.Plot.Axis(VtChAxisIdY)
      .AxisTitle.Text = "Deliverability (MSCF/D)"
      .AxisScale.Type = VtChScaleTypeLinear
      .AxisGrid.MinorPen.Style = VtPenStyleSolid
      .CategoryScale.Auto = True
      .ValueScale.MajorDivision = 15
   End With

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmdMove_Click(Index As Integer)

   Dim i As Integer
   Dim X As Integer
   Dim FracNo As String
   Dim Msg, Style, Title, Help, Ctxt, Response, MyString

   X = sRecordset.RecordCount
   'Checking if there is no recordset
   If sRecordset.EOF = True And sRecordset.BOF = True Then
      MsgBox ("There is no Frac Recipe Available For Well " & txt_Well_No.Text)
   Else
      With sRecordset
         'Moving to the first record
         If Index = 0 Then
               .MoveFirst
         ElseIf Index = 1 Then
```

```
            'Moving Back
            If .AbsolutePosition = adPosBOF Then
              .MoveFirst
               GoTo FastExit
            Else
              .MovePrevious
            End If
          ElseIf Index = 2 Then
            'Moving to next record
            If .AbsolutePosition = adPosEOF Then
               .MoveLast
               GoTo FastExit
             Else
               .MoveNext
             End If
          ElseIf Index = 3 Then
            'Moving to the last record
            .MoveLast
          End If
        End With


    If sRecordset.EOF = True Or sRecordset.BOF = True Then
       Msg = "No more Record !!  "
       Style = vbCritical
      Response = MsgBox(Msg, Style)
      GoTo FastExit
    End If

    For i = Me.txt_Add_Frac.LBound To Me.txt_Add_Frac.UBound
       If IsNull(sRecordset.Fields(i)) Then
          Me.txt_Add_Frac(i).Text = ""
       Else
          txt_Add_Frac(i).Text = sRecordset.Fields(i)
       End If
    Next i

FastExit:
    End If

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_addNEw_Deliverability_Click()

    Dim i                       As Integer
    Dim year                     As Integer
```

```
Dim Q100                    As Integer

If Me.cmd_addNEw_Deliverability.Caption = "Add New " Then

   Me.cmd_addNEw_Deliverability.Caption = "Submit"
   Me.MSHFlexGrid_Deliverability.Height = 3100
   Me.txt_Add_Deliverability.Visible = True
   Me.txt_Add_Deliverability.Top = 3700
   Me.txt_Add_Year.Visible = True
   Me.txt_Add_Year.Top = 3700
   Me.cmd_addNEw_Deliverability.Top = 4080
   Me.cmd_Cancel.Visible = True
   Me.cmd_Cancel.Top = 4560

ElseIf Me.cmd_addNEw_Deliverability.Caption = "Submit" Then

   If Me.txt_Add_Deliverability.Text = "" Or Me.txt_Add_Year.Text = "" Then
      MsgBox ("Data is missing in one or more fields"), vbCritical
      Exit Sub
   End If

   Me.cmd_addNEw_Deliverability.Caption = "Add New "
   Me.MSHFlexGrid_Deliverability.Height = 4000
   Me.cmd_addNEw_Deliverability.Top = 4560
   Me.txt_Add_Deliverability.Visible = False
   Me.txt_Add_Year.Visible = False
   Q100 = Me.txt_Add_Deliverability.Text
   year = Me.txt_Add_Year.Text
   Me.txt_Add_Deliverability.Text = ""
   Me.txt_Add_Year.Text = ""
   Me.cmd_Cancel.Visible = False


   Call Add_Deliverability_Record(year, Q100)
   Call DG_Well_No_RowColChange(1, 1)


ElseIf Me.cmd_addNEw_Deliverability.Caption = "Delete" Then
   If Me.txt_Add_Deliverability.Text = "" Or Me.txt_Add_Year.Text = "" Then
      MsgBox ("Data is missing in one or more fields"), vbCritical
      Exit Sub
   End If

   Me.cmd_addNEw_Deliverability.Caption = "Add New "
   Me.MSHFlexGrid_Deliverability.Height = 4000
   Call Add_Deliverability_Record(Me.txt_Add_Year.Text, 0)
```

```
        Me.cmd_addNEw_Deliverability.Top = 4560
        Me.txt_Add_Deliverability.Visible = False
        Me.txt_Add_Deliverability.Text = ""
        Me.txt_Add_Year.Visible = False
        Me.txt_Add_Year.Text = ""
        Me.cmd_Cancel.Visible = False
        Call DG_Well_No_RowColChange(1, 1)
    End If


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub MSHFlexGrid_Deliverability_DblClick()


    Me.cmd_addNEw_Deliverability.Caption = "Delete"
    Me.MSHFlexGrid_Deliverability.Height = 3100
    Me.txt_Add_Deliverability.Visible = True
    Me.txt_Add_Deliverability.Top = 3700
    Me.txt_Add_Year.Visible = True
    Me.txt_Add_Year.Top = 3700
    Me.cmd_addNEw_Deliverability.Top = 4080
    Me.cmd_Cancel.Visible = True
    Me.cmd_Cancel.Top = 4560

    With MSHFlexGrid_Deliverability
        Me.txt_Add_Year.Text = .TextMatrix(.Row, 0)
        Me.txt_Add_Deliverability.Text = .TextMatrix(.Row, 1)
    End With

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Add_Deliverability_Record(year, Q100)

    Dim myData(2)                   As Double
    Dim i                           As Integer

    'If there is a deliverability value needed to be added then
    If Q100 <> 0 Then
        'Preparing SQL String
        strSQL = "SELECT [WeLL No], YEAR, Q100 FROM tbl_Q100 WHERE [WeLL
No]='" & txt_Well_No.Text & "' And [Q100] <> 0"
        strSQL = strSQL + " ORDER BY [YEAR]"
        'Open Recordset
        Call Open_RecordSet(strSQL)
```

```
        'Move curser to the last record in the record set
        If sRecordset.EOF <> True Or sRecordset.BOF <> True Then
           sRecordset.MoveLast
        End If
           'Start adding
           sRecordset.AddNew
           sRecordset.Fields(0).Value = txt_Well_No.Text
            sRecordset.Fields(1).Value = year
             sRecordset.Fields(2).Value = Q100
              sRecordset.Update
               sRecordset.Close

    Else
       strSQL = "SELECT [WeLL No], YEAR, Q100 FROM tbl_Q100 WHERE [WeLL
No]='" & txt_Well_No.Text & "'"
        strSQL = strSQL + " And [YEAR] = " & year
         Call Open_RecordSet(strSQL)
          i = sRecordset.RecordCount
           sRecordset.Delete adAffectCurrent

    End If


End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Cancel_Click()

        Me.cmd_addNEw_Deliverability.Caption = "Add New "
        Me.MSHFlexGrid_Deliverability.Height = 4000
        Me.txt_Add_Deliverability.Visible = False
        Me.txt_Add_Deliverability.Text = ""
        Me.txt_Add_Year.Visible = False
        Me.txt_Add_Year.Text = ""
        Me.cmd_addNEw_Deliverability.Top = 4560
        Me.cmd_Cancel.Visible = False
        Me.cmd_Cancel.Top = 4560

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_AddFracRecipe_Click()
    'Describtion
       'This code allows user to add new frac Jobe to n existing well
    'Algorithm
       'if click on add then
```

```
        'disable user to change the frac#
        'move the curser to end of the recordset
        'Claculate some entry for the first 4 boxes
        'Clean up all text box
    'Else
        'msgbox
        'Checking for an empty box
        'Reading all the inputs and store them inside an MyData Array
        'Sending the array to the add new procedure to be added

    'end if
    'exit

On Error GoTo Proc_Err

Dim i                   As Integer

If Me.cmd_AddFracRecipe.Caption = "Add New Frac Recipe" Then
    Me.txt_No_of_Frac.Enabled = False
    If sRecordset.EOF = False Then
        sRecordset.MoveLast
    End If

    'Prepare Data to be entered inside the empty boxes
    Me.cmd_AddFracRecipe.Caption = "Submit New Frac Recipe"
    txt_Add_Frac(0).Text = Me.txt_Well_No.Text
    txt_Add_Frac(3).Text = CInt((CDbl(Date) - CDbl(sRecordset.Fields(1))) / 365)
    txt_Add_Frac(1).Text = Date
    txt_Add_Frac(2).Text = 1 + txt_No_of_Frac.Text

    For i = 4 To Me.txt_Add_Frac.UBound
        txt_Add_Frac(i).Text = ""
    Next i

ElseIf Me.cmd_AddFracRecipe.Caption = "Submit New Frac Recipe" Then

    Dim Msg, Style, Title, Response, MyString
    Msg = "Would You Like to Add New Frac Recipe  ? "
    Style = vbYesNo  ' Define buttons.
    Title = "Adding New Frac Recipe"
    Response = MsgBox(Msg, Style, Title)

    If Response = vbYes Then

        ReDim myData(Me.txt_Add_Frac.UBound)
```

```
            For i = Me.txt_Add_Frac.LBound To Me.txt_Add_Frac.UBound
               If Me.txt_Add_Frac(i).Text = "" Then
                  MsgBox ("Data is missing in one or more fields"), vbCritical
                  Exit Sub
               End If
               myData(i) = txt_Add_Frac(i).Text
            Next i

            Call AddFracRecipr(myData, "New")

            Me.cmd_AddFracRecipe.Caption = "Add New Frac Recipe"
            Me.txt_No_of_Frac.Enabled = True

         ElseIf Response = vbNo Then
            Exit Sub
         End If
      End If

      Exit Sub

Proc_Err:
      MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_UpdateRecipe_Click()

      On Error GoTo Proc_Err

      Dim i As Integer
      ReDim myData(Me.txt_Add_Frac.UBound)

      Dim Msg, Style, Title, Response, MyString
      Msg = "Would you like to save you changes  ? "
      Style = vbYesNo  ' Define buttons.
      Title = "Updating Current Frac Recipe"
      Response = MsgBox(Msg, Style, Title)

      If Response = vbYes Then
         For i = Me.txt_Add_Frac.LBound To Me.txt_Add_Frac.UBound
            If Me.txt_Add_Frac(i).Text = "" Then
               MsgBox ("Data is missing in one or more fields"), vbCritical
            End If
            myData(i) = txt_Add_Frac(i).Text
         Next i
         Call AddFracRecipr(myData, "Update")
```

```
   ElseIf Response = no Then
      Call DG_Well_No_RowColChange(1, 1)
   End If

 Exit Sub

Proc_Err:
   MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub AddFracRecipr(myData, Operation)

   'Discribtion
      'this code addnew frac recipe or update one
   'Algorithm
      'if new then
         'check the curser place and move to the last record
         'Add new record to the database
         'write a SQL string to update the frac order used for GA
      'else if update
         'Write SQL string to pull out the record needed to be updated
         'updtae
         'close
      'end

   Dim SaveData                  As New cOpen_Database
   Dim mySQL                     As String
   Dim i                      As Integer

   If Operation = "New" Then
      If sRecordset.EOF <> True Or sRecordset.BOF <> True Then
         If sRecordset.EOF = False Then
            sRecordset.MoveLast
         End If
      End If
      sRecordset.AddNew
      For i = LBound(myData) To UBound(myData)
         sRecordset.Fields(i).Value = myData(i)
      Next i
      sRecordset.Update

      strSQL = "SELECT * FROM tbl_Wells_Frac_Count WHERE [WELL No]= " &
txt_Well_No.Text
      i = Trim(txt_Add_Frac(2).Text)
      Call Open_RecordSet(strSQL)
```

```
            sRecordset.Update
            sRecordset.Fields(4).Value = i
            sRecordset.MoveLast
            sRecordset.Close

    ElseIf Operation = "Update" Then

        strSQL = "SELECT * FROM tbl_Fracrecipe WHERE [WELL No]= '" &
txt_Well_No.Text & "'"
        strSQL = strSQL + " And [FRAC No] = " & Str(Trim(txt_Add_Frac(2).Text))

        Call Open_RecordSet(strSQL)
        For i = LBound(myData) To UBound(myData)
            sRecordset.Update sRecordset.Fields(i).Name, myData(i)
        Next i

    End If

    Call DG_Well_No_RowColChange(1, 1)

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_DeletsFracRecipe_Click()

    'Discribtion
        'Deleting a frac job
    'algorithm

    On Error GoTo Proc_Err

    Dim i As Integer
    Dim FracNo As String

    If txt_Add_Frac(2).Text = "" Then MsgBox "Specify frac recipe you want to delete":
Exit Sub
    FracNo = Str(Trim(txt_Add_Frac(2).Text))

    Dim Msg, Style, Title, Response, MyString
    Msg = "Would you like to Delete this frac recipe  ? "
    Style = vbYesNo  ' Define buttons.
    Title = "Delete Current Frac Recipe"
    Response = MsgBox(Msg, Style, Title)

    If Response = vbYes Then
```

```
        strSQL = "SELECT * FROM tbl_Fracrecipe WHERE [WELL No]= '" &
txt_Well_No.Text & "'"
        strSQL = strSQL + " And [FRAC No] = " & FracNo

        Call Open_RecordSet(strSQL)

        i = sRecordset.RecordCount
        If i = 0 Then MsgBox "No record was found": Exit Sub
        sRecordset.Delete adAffectCurrent

        strSQL = "SELECT * FROM tbl_Wells_Frac_Count WHERE [WELL No]= " &
txt_Well_No.Text
        i = FracNo - 1
        Call Open_RecordSet(strSQL)
           sRecordset.Update
           sRecordset.Fields(4).Value = i
           sRecordset.MoveLast
           sRecordset.Close

        Call DG_Well_No_RowColChange(1, 1)

    ElseIf Response = vbNo Then
        Exit Sub
    End If

    Exit Sub
Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Fuzzy_Click()
    frmFuzzy.Show
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_View_Click()
    frmNeural_GA.Show
End Sub
Private Sub cmdClear_Click()
    Me.txt_Memo.Text = ""
End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmdExit_Click()

    sRecordset.Close
```

```
    sConnection.Close
    FrmEog_Frac_info.Hide
    Unload FrmEog_Frac_info

    End

End Sub

Private Sub cmd_Exit_Application_Click()
    sRecordset.Close
    sConnection.Close
    Unload FrmEog_Frac_info
    End
End Sub
```

Neural and Genetic Algorithm Form

```
Option Explicit

    Dim NeuralNet_Info              As New cOpen_Database   'information
about All NNW
    Dim WellData                    As New cInputs          'all Well Data
    Dim InputsFieldsTitle                                   'Names of the Inputs
    Dim InputsFields()              As Variant
    Dim Title()                     As String
    Dim MyFileds
    Dim myWells()                   As Integer
    Dim ChangeableInputs()
    Dim FireNet                     As New CNet_Fire
```

```
    Dim FinalInputData()                    As Double
    Dim BatchOutput                         As Double
    """""""""""""""""
    Dim Minimum()                           As Double
    Dim Maximum()                           As Double
    Dim MyChromosome                        As New GA_Class
    Dim Generation_SinceMax                 As Integer
    Dim NetOutput()                         As Double
    Dim StopFlag                            As Boolean
    Dim X, X1, Y, Y1



'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Form_Load()

    'This Code Saves the content of the table tbl_Neural_Net_Info
    'tbl_Neural_Net_Info contents all information about the neural networks
    'inside NeuralNet_Info
    On Error GoTo Proc_Err

    Dim i As Integer
    Dim j As Integer

    'Form location when loaded
    Me.Left = (Screen.Width - Width) / 2
    Me.Top = (Screen.Height - Height) / 2
    Me.SSTab_Optimization.Tab = 0

    Call optNet_Processing_Click(0)
    Call opt_OptimizationType_Click(0)

    'open the database
    NeuralNet_Info.OpenDatabase
    'Sending string to the class to open Neural Network table
    NeuralNet_Info.mySQL = "SELECT * FROM tbl_Neural_Net_Info"
    NeuralNet_Info.MyFlag = True
    'Fields Array of Neural network table
    MyFileds = Array("Frac #", "Record Exist", "Fraced", "FracID", "def_Files", "Input",
_
                "Output", "changeable Input")
    NeuralNet_Info.FieldArray = MyFileds
    'Open Record Set and store the contents of neural network info table inside DataArray
array
```

```
    DatReturen = dtDataArray
    'Get the record set
    NeuralNet_Info.OpenRecordSet
    NeuralNet_Info.CLoseDatabse

    'Filling the combo boxes with frac IDs
    For j = cmbo_FracID.LBound To cmbo_FracID.UBound
       With cmbo_FracID(j)
          .Clear
          'This combo for well list
          For i = LBound(NeuralNet_Info.DataArray) To
UBound(NeuralNet_Info.DataArray)
             'Checking if the Neural network is exist for this Frac
             If NeuralNet_Info.DataArray(i, 1) = True Then
                If NeuralNet_Info.DataArray(i, 2) <> "All Frac" Then
                   .AddItem NeuralNet_Info.DataArray(i, 2)
                End If
             End If
          Next i
          .ListIndex = 0
       End With
    Next j

    'This Veriable store all the Inputs Parameters Titels
    'Check table tbl_InputsType from the data base
    InputsFields = Input_Fields()
    'Spicifing the items that going to appear in the combo box
    Me.txt_InputParameters(0).Visible = False
    Me.lbl_InputParameters(0).Visible = False
    Me.Cmbo_Inputs.Visible = False
    opt_Net_Selection(2).Value = True
    Me.txt_StoppingCriteria.Enabled = False
    Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Input_Fields()

    'This Function open the data base and get all the inputs field titles
    'for all the network and store them inside a veriable name InputsFields

    Dim InputFields                 As New cOpen_Database
```

```
    InputFields.OpenDatabase
    InputFields.mySQL = "SELECT * FROM tbl_InputsType"
    MyFileds = Array("Field_Name", "Type", "First Frac", "Second Frac", "Third Frac", _
                "Fourth Frac", "Fifth Frac", "Sixth Frac", "All Frac")
    InputFields.FieldArray = MyFileds
    DatReturen = dtDataArray
    'Getting all details from the table
    InputFields.OpenRecordSet
    'Store the content inside Input_Fields variable
    Input_Fields = InputFields.DataArray
    'store the Fields name of the Input type table
    InputsFieldsTitle = InputFields.FieldArray

    InputFields.CLoseDatabse
    Set InputFields = Nothing

End Function

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub opt_Net_Selection_Click(Index As Integer)

    'This code fill the txt_FracID with the Frac ID from the data base
    If opt_Net_Selection(0).Value = True Or opt_Net_Selection(2).Value = True Then
        'Checking if the Neural network is exist for this Frac
        If NeuralNet_Info.DataArray(cmbo_FracID(0).ListIndex + 1, 1) = True Then
            'Excluding Never and All Frac
            If NeuralNet_Info.DataArray(cmbo_FracID(0).ListIndex + 1, 3) <> "All Frac"
Then
                Me.txt_FracID.Text = NeuralNet_Info.DataArray(cmbo_FracID(0).ListIndex +
1, 3)
            End If
        Else
            Me.txt_FracID.Text = "All Frac"
        End If

    ElseIf Me.opt_Net_Selection(1).Value = True Then
        Me.txt_FracID.Text = "All Frac"

    End If


    Me.txt_GAFracID.Text = Me.txt_FracID.Text

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub opt_Batch_Net_Selection_Click(Index As Integer)
```

```
    'This code fill the txt_Batch_FracID with the Frac ID from the data base
    If opt_Batch_Net_Selection(0).Value = True Or opt_Batch_Net_Selection(2).Value =
True Then
        'Checking if the Neural network is exist for this Frac
        If NeuralNet_Info.DataArray(cmbo_FracID(1).ListIndex + 1, 1) = True Then
            'Excluding Never and All Frac
            If NeuralNet_Info.DataArray(cmbo_FracID(1).ListIndex + 1, 3) <> "All Frac"
Then
                Me.txt_Batch_FracID.Text =
NeuralNet_Info.DataArray(cmbo_FracID(1).ListIndex + 1, 3)
            End If
        Else
            Me.txt_Batch_FracID.Text = "All Frac"
        End If
        Me.opt_Net_Selection(0).Value = True
    ElseIf opt_Batch_Net_Selection(1).Value = True Then
        Me.txt_Batch_FracID.Text = "All Frac"
        Me.opt_Net_Selection(1).Value = True
    End If

End Sub

Private Sub opt_StoppingCriteria_Click(Index As Integer)

    If opt_StoppingCriteria(0).Value = True Then
        Me.txt_StoppingCriteria.Enabled = False
        Me.txt_StoppingCriteria.Text = 15
    ElseIf opt_StoppingCriteria(1).Value = True Then
        Me.txt_StoppingCriteria.Enabled = True
        Me.txt_StoppingCriteria.Text = ""
    End If

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub optNet_Processing_Click(Index As Integer)

    'This code hides and show some controles
    'According to net type selection

    Dim i As Integer

    'Form Controls Condition
    If Me.optNet_Processing(0).Value = True Then
        'if single firing then hide the grid and bring the text boxes
```

```
        Me.framMsflxgrid.Visible = False
        Me.Frame_InputsParameters.Visible = True
        Me.Lst_Wells(0).Enabled = True
        Me.cmd_Rank.Visible = False
        Getting_Datafor = SingleOptimization

        If (txt_InputParameters.UBound) <> 0 Then
           'Clear all inputs from the text boxes
           For i = 0 To txt_InputParameters.UBound
              txt_InputParameters(i).Text = ""
           Next i
        End If

    ElseIf Me.optNet_Processing(1).Value = True Then
        'if Batch process was selected then show the flex grid
        Me.framMsflxgrid.Visible = True
        Me.Frame_InputsParameters.Visible = False
        Me.Lst_Wells(0).Enabled = False
        Me.cmd_Rank.Visible = True
        Getting_Datafor = BatchOptimization

        With MSFlex_FiredQ
           .Clear
           .TextMatrix(0, 0) = "Well No"
           .TextMatrix(0, 1) = "Pre_Stim"
                .ColWidth(2) = 1000
           .TextMatrix(0, 2) = "Deliverability"
        End With
    End If

    Call FirstTab_TxtBoxes


End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmbo_FracID_Click(Index As Integer)

    'This code fill the list box with all wells that have been fraced
    'with same number that was selected from the combo box

    On Error GoTo Proc_Err

    Dim Well                    As Variant
    Dim i                       As Integer
```

```
    If Index = 0 Then
        opt_Net_Selection_Click (0)
        Call FirstTab_TxtBoxes
    ElseIf Index = 1 Then
        opt_Batch_Net_Selection_Click (0)
        txt_GAFracID.Text = txt_Batch_FracID.Text
        Me.lst_Selected_BatchWell.Clear
    End If

    'Filling the List with Wells
    With Lst_Wells(Index)
        .Clear
        Well = Wells(Index)
        ReDim myWells(UBound(Well)) As Integer

        If UBound(Well) = 0 Then
            MsgBox ("There is no wells existed for this frac")
            Exit Sub
        Else
            For i = LBound(Well) To UBound(Well)
                .AddItem Well(i, 0)
                'This array stores the wells for later use in Batch processing
                myWells(i) = Well(i, 0)
            Next i
            .ListIndex = 0
            lbl_Counter.Caption = .ListCount & "  Wells"
        End If

    End With
    'Write some information
    Frame_InputsParameters.Caption = (LTrim(Me.txt_FracID.Text)) & " Input
Parameters"
    lbl_Count(0).Caption = Lst_Wells(1).ListCount & "  Wells"
    lbl_Count(1).Caption = lst_Selected_BatchWell.ListCount & "  Wells"

    Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Wells(Index)

    Dim WellList                    As New cOpen_Database
```

```
   'Store SQL and Field Array in the property of the open Data Class
   'then open the database and return only data array contains wells
   WellList.mySQL = "SELECT * FROM tbl_Wells_Frac_Count WHERE [Frac No]= "
& cmbo_FracID(Index).ListIndex
   WellList.FieldArray = Array("Well No")
   DatReturen = dtDataArray
   WellList.OpenDatabase
   WellList.OpenRecordSet
   WellList.CLoseDatabse
   Wells = WellList.DataArray
   Set WellList = Nothing

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Lst_Wells_DblClick(Index As Integer)

   'The following Sub select the well needed to be tested for frac
   'it brings the data from last frac job as the default input parameters
   'for the next frac
   On Error GoTo Proc_Err

   Dim i                        As Integer

   Data_Existance = Yes
   DataFor = ViewData
   Select Case Index
      Case 0
            Call FirstTab_TxtBoxes
            'Getting all the data for the specific well
            Call Well_Data(CInt(Me.Lst_Wells(0).Text), False)
               If Data_Existance = no Then Exit Sub

            Call InputFrameDesign
      Case 1
            Call SSPan_Move_Click(0)
   End Select

   Exit Sub

Proc_Err:
   MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Well_Data(Well As Integer, optimization As Boolean)
```

```
    Dim FracOrder                       As Integer

    'Getting all well fixed information store them inside cInput Class
    WellData.Well_No = Well

    If Me.opt_OptimizationType(0).Value = True Then
       cmbo_FracID(1).ListIndex = cmbo_FracID(0).ListIndex
    End If
    FracOrder = LTrim(CInt(Me.cmbo_FracID(1).ListIndex + 1))
    WellData.FracNo = FracOrder

    'Call the class Function
    WellData.GetWellData Well, FracOrder, optimization

       If Data_Existance = no Then
          Set WellData = Nothing
          Exit Function
       End If

    If optimization = False Then
          'Write into
          txt_WellNo(0).Text = Lst_Wells(0).Text
          txt_WellNo(1).Text = Lst_Wells(0).Text

          Me.txt_Pre_StimulationDeliverability.Text = WellData.Q100_Before_This_Frac
          Me.lbl_No_Months_Since_ThisQ.Caption =
Format(WellData.No_of_Months_Before_Frac, "##,##0") & "  Months"
          Me.lbl_YearsBefore.Caption = WellData.YearsBefore_This_Frac & "  Years"
    End If

    Me.txt_GA_Pre_Stimulation.Text = WellData.Q100_Before_This_Frac

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub FirstTab_TxtBoxes()

    'Clearing all text boxes
    txt_WellNo(0).Text = ""
    Me.txt_Pre_StimulationDeliverability.Text = ""
    Me.lbl_No_Months_Since_ThisQ.Caption = ""
    Me.lbl_YearsBefore.Caption = ""
    Me.txt_Post_StimulationDeliverability.Text = ""
    Me.txt_Deliverability_Increase.Text = ""


End Sub
```

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Get_FieldsName(InputsNumber)

    'This Sub gets all the input parameters Titles names and store them inside Title array
    'variable and also stores the paramteres inside ChangeableInputs array
    'then is loads them in the lable and text controles in the form
    'Remember that all the input paramteres fields name are stored in "tbl_InputsType" in
the DB

    Dim i                           As Integer
    Dim k                           As Integer
    Dim iRow                        As Integer
    Dim myData
    ReDim Title(InputsNumber)
    ReDim ChangeableInputs(InputsNumber)

    myData = WellData.Well_Input_Data

    k = 0
    "navigate throught the table column titels
    For i = 2 To UBound(InputsFieldsTitle)
        'Cheking which frac we are looking for
        'remember that InputsFieldsTitle() are data from the database
        If Me.txt_FracID.Text = InputsFieldsTitle(i) Then
            'Navigate through the inputs type to see which is fixed and
            'which is a variable
            For iRow = LBound(InputsFields) To UBound(InputsFields)
                'if the input is a varibale we check if its used in this frac
                If InputsFields(iRow, 1) = "Variable" Then
                    If InputsFields(iRow, i) = True Then

                        Title(k) = InputsFields(iRow, 0)
                        ChangeableInputs(k) = myData(iRow)
                        k = k + 1

                    End If
                End If

            Next iRow
        End If
    Next i

End Function

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub InputFrameDesign()
```

```
    Dim i                               As Integer
    Dim InputsNumber                    As Integer
    Dim myTop                           As Integer

    'These few lines are needed to determine the number of text boxes and   '
    'labels needed in designning the input form                             '
    For i = 1 To UBound(NeuralNet_Info.DataArray)
       If Me.txt_FracID.Text = NeuralNet_Info.DataArray(i, 3) Then
          InputsNumber = NeuralNet_Info.DataArray(i, 7)
       End If
    Next i

    Call Get_FieldsName(InputsNumber)

    For i = txt_InputParameters.LBound To txt_InputParameters.UBound
       txt_InputParameters(i).Text = "":  lbl_InputParameters(i).Caption = ""
    Next i

    If (txt_InputParameters.UBound) <> 0 Then
       For i = 1 To txt_InputParameters.UBound
          Unload txt_InputParameters(i)
          Unload lbl_InputParameters(i)
       Next i
    End If

    Me.Frame_InputsParameters.Height = ((txt_InputParameters(0).Top) * InputsNumber)
+ (55 * InputsNumber * 1.5) + 60
    If (txt_InputParameters.UBound) = 0 Then
       txt_InputParameters(0).Visible = True
       txt_InputParameters(0).Text = ChangeableInputs(0)
       lbl_InputParameters(0).Visible = True
       lbl_InputParameters(0).Caption = Title(0)

       For i = 1 To InputsNumber
          Load txt_InputParameters(i)
             myTop = txt_InputParameters(i - 1).Top + txt_InputParameters(i).Height + 60
             txt_InputParameters(i).Top = myTop
             txt_InputParameters(i).Visible = True
             txt_InputParameters(i).Text = ChangeableInputs(i)
          Load lbl_InputParameters(i)
             lbl_InputParameters(i).Top = myTop
             lbl_InputParameters(i).Visible = True
             lbl_InputParameters(i).Caption = Title(i)
       Next i
    End If
```

```
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_NetFire_Click()

    On Error GoTo Proc_Err

    Dim i As Integer

    SSPanel2.Visible = True

    For i = Me.txt_InputParameters.LBound To Me.txt_InputParameters.UBound
        If Me.txt_InputParameters(i).Text = "" Then
            MsgBox (" Missing Parameters  !"), vbCritical
            Exit Sub
        End If
    Next i


    'Network Processing, Single
    If Me.optNet_Processing(0).Value = True Then
        If txt_WellNo(0).Text = "" Then
            MsgBox (" Select a well  !"), vbCritical
            Exit Sub
        End If
        'Go to single Firing function
        Call SingleFiring_NetSelection

    ElseIf Me.optNet_Processing(1).Value = True Then
        'Network Processing Batch
        Call BatchFiring
    End If

    SSPanel2.Visible = False

    Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

    SSPanel2.Visible = False

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub SingleFiring_NetSelection()
```

```
'This Code detarmine the wich net type to be fired
'accordingly it goes and select the network for the
'selection then get the data and fire the network

Dim i                          As Integer
Dim Output                     As Double
Dim Output1                    As Double
Dim Pre_Stimulation            As Double
Dim Average_Output             As Double


Pre_Stimulation = Me.txt_Pre_StimulationDeliverability

'Net type Selection if its speicalized or General
If opt_Net_Selection(0).Value = True Or opt_Net_Selection(1).Value = True Then
   'go to SingleFiring sending the frac ID to the function
   Call SingleFiring(LTrim(Me.txt_FracID.Text))
   Output = (FireNet.Net_OutPut - Pre_Stimulation)
            If Output < 0 Then Output = 0

ElseIf opt_Net_Selection(2).Value = True Then

   For i = 1 To 2
      If i = 1 Then
         Call SingleFiring(LTrim(Me.txt_FracID.Text))
         Output = (FireNet.Net_OutPut - Pre_Stimulation)
            If Output < 0 Then Output = 0
      Else

         Call SingleFiring("All Frac")
         Output1 = (FireNet.Net_OutPut - Pre_Stimulation)
            If Output1 < 0 Then Output1 = 0

      End If
   Next i

   Average_Output = (Output + Output1) / 2
   Output = Average_Output

End If


If Me.optNet_Processing(0).Value = True Then
      txt_Post_StimulationDeliverability.Text = Format(Output, "####0.00")
```

```
        txt_Deliverability_Increase.Text =
Format(txt_Post_StimulationDeliverability.Text - Me.txt_Pre_StimulationDeliverability,
"####0.00")
        If txt_Deliverability_Increase.Text < 0 Then txt_Deliverability_Increase.Text =
0#

   ElseIf Me.optNet_Processing(1).Value = True Then
      BatchOutput = Output
   End If


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub BatchFiring()

      Dim j                  As Integer
      Dim i                  As Integer
      Dim MyWell                  As Integer
      Dim BatchWells_Data              As New cOpen_Database

      Me.MSFlex_FiredQ.Visible = True
      Me.Frame_InputsParameters.Visible = False
      Getting_Datafor = BatchOptimization

      With MSFlex_FiredQ
         .Clear
         .Rows = Me.Lst_Wells(0).ListCount + 3
         .ColWidth(0) = 820
         .TextMatrix(0, 0) = "Well No"
            .ColWidth(1) = 820
         .TextMatrix(0, 1) = "Pre_Stim"
              .ColWidth(2) = 1000
         .TextMatrix(0, 2) = "PFD"

         For i = 0 To Me.Lst_Wells(0).ListCount - 1
            DoEvents
            'Scrolling throught the list box to select Wells

            Lst_Wells(0).ListIndex = i
            Lst_Wells_DblClick (0)
            .TextMatrix(i + 1, 0) = Lst_Wells(0).Text
            .TextMatrix(i + 1, 1) = WellData.Q100_Before_This_Frac

            If Data_Existance = no Then
               MSFlex_FiredQ.TextMatrix(i + 1, 2) = "Missing Data"
            Else
```

```
            'selecting and Firing the net
            SingleFiring_NetSelection

            If (BatchOutput - WellData.Q100_Before_This_Frac) > 0 Then
               .TextMatrix(i + 1, 2) = Format(BatchOutput, "####0.0")
            Else
               .TextMatrix(i + 1, 2) = 0
            End If

         End If

         SSPanel2.FloodPercent = (i / Lst_Wells(0).ListCount) * 100
      Next i
   End With

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub SingleFiring(FracID)

   'This code get the input data from the input class
   '

   Dim i                                As Integer
   Dim InputsArray

   'Getting input parameters
   InputsArray = AllInputs(FracID)
   'Selecting the net
   NetSelection (FracID)

   ReDim FinalInputData(FireNet.InputNo) As Double

   For i = LBound(FinalInputData) To UBound(FinalInputData) - 1
      'storing all the data inside final input after
      'switching them to double format
      FinalInputData(i) = CDbl(InputsArray(i))
   Next

   FireNet.InputArry = FinalInputData()
   FireNet.Fire_The_Net

   Erase InputsArray
   Erase FinalInputData

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
Public Function AllInputs(FracID)

    'This code gets the original data from the input class
    'then switch the data with user inputs data
    Dim i                                   As Integer
    Dim iRow                                As Integer
    Dim iCol                                As Integer
    Dim nInput                              As Integer
    Dim k                                   As Integer
    Dim myData
    'Getting the well original data
    myData = WellData.Well_Input_Data
    k = 0: nInput = 0
    'Checking input from input type table from the data base
    For i = 2 To UBound(InputsFieldsTitle)
        'Checking the frac ID
        If FracID = InputsFieldsTitle(i) Then
            iCol = i
            'Checking which input is used for this frac
            For iRow = LBound(InputsFields) To UBound(InputsFields)
                'if the input is a varibale we check if its used in this frac
                If InputsFields(iRow, i) = True Then
                    nInput = nInput + 1
                    If InputsFields(iRow, 1) = "Variable" Then
                        myData(iRow) = Me.txt_InputParameters(k).Text
                        k = k + 1
                    End If
                End If
            Next iRow
        End If
    Next i
    'Getting data after switching all text to numerical
    DataFor = FireData
    AllInputs = Inputs_array(WellData.Changed_Well_Data(myData), iCol, nInput)

End Function

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Inputs_array(myData, iCol, nInput)

    Dim iRow As Integer
    ReDim iinputs(nInput)
    Dim k As Integer
    k = 0
        For iRow = LBound(InputsFields) To UBound(InputsFields)
            If InputsFields(iRow, iCol) = True Then
```

```
            iinputs(k) = CDbl(myData(iRow))
            k = k + 1
        End If
    Next iRow
  Inputs_array = iinputs
  Erase iinputs

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_GA_Click()

  Dim Msg, Style, Title, Response, MyString

  Msg = "Would you like to Delete Previous jobs ?"
  Style = vbYesNo  ' Define buttons.
  Title = "Delete Previous jobs "

  If Me.txt_Output(0).Text <> "" Then Response = MsgBox(Msg, Style, Title)

  If Response = vbYes Then
    cdm_DeletAll_Click
  End If

  'Preparing All controls
  Me.SSPanel2.FloodPercent = 0
  Me.SSPanel2.Visible = True
  StopFlag = False
  Set WellData = Nothing
  ''''''''''''''''''''''''''''''''
  Setup_Pic_To_Draw


  'Finding all Minimum and maximum values
 Call nPrameters_Min_Max(Trim(Me.txt_GAFracID.Text))
  ''''''''''''''''''''''''''''''''''

  'Checking which optimization is processed
  If opt_OptimizationType(0).Value = True Then

    Getting_Datafor = SingleOptimization
    Call StartSingleProcessing

  ElseIf opt_OptimizationType(1).Value = True Then

    Getting_Datafor = BatchOptimization
    Call StartBatch
```

```
    End If

    Me.SSPanel2.Visible = False

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub nPrameters_Min_Max(FracID)
    'This code open a dat base and get all the min max information for
    'the input parameteres and store them inside GA class properties
    Dim j                                   As Integer
    Dim ParameteMinMax                              As New cOpen_Database
    Dim myData

    ParameteMinMax.OpenDatabase
    'Go to the Database and get the min max table this table conatainse all min max
information
    ParameteMinMax.mySQL = " Select * FROM tbl_Min_Max WHERE FracID = '" &
FracID & "'"
     Select Case FracID
        Case "First Frac"
            'Input for first frac
           ParameteMinMax.FieldArray = Array("Frac Fluid", "Fluid Volume", "N2 Rate",
"N2 Volume SCF", "Sand Volume", _
                "Sand Concentration", "Sand Mesh", "Acid Volume", "Average Rate",
"Service Company")
        Case "Second Frac"
           'Input for second frac
           ParameteMinMax.FieldArray = Array("Frac Fluid", "Fluid Volume", "N2 Rate",
"N2 Volume SCF", "Sand Volume", _
                "Sand Concentration", "Sand Mesh", "Acid Volume", "Average Rate",
"Service Company")
        Case "Third Frac"
           'Input for third frac
           ParameteMinMax.FieldArray = Array("Frac Fluid", "Fluid Volume", "N2 Rate",
"N2 Volume SCF", "Sand Volume", _
                "Sand Concentration", "Sand Mesh", "Acid Volume", "Average Rate",
"Service Company")
        Case "Fourth Frac"
        Case "Fifth Frac"


        Case "All Frac"
           ParameteMinMax.FieldArray = Array("Frac Fluid", "Fluid Volume", "N2 Rate",
"N2 Volume SCF", "Sand Volume", _
```

```
                "Sand Concentration", "Sand Mesh", "Acid Volume", "Average Rate",
"Service Company")
   End Select

   DatReturen = dtDataArray
   ParameteMinMax.OpenRecordSet
   myData = ParameteMinMax.DataArray
   ParameteMinMax.CLoseDatabse
   'Establish 2 arrayes
   ReDim Minimum(UBound(myData, 2))
   ReDim Maximum(UBound(myData, 2))

   For j = LBound(myData, 2) To UBound(myData, 2)
      Minimum(j) = myData(LBound(myData), j)
      Maximum(j) = myData(UBound(myData), j)
   Next j

   MyChromosome.MinimumInputs_Array = Minimum
   MyChromosome.MaximumInputs_Array = Maximum


End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub StartSingleProcessing()

   Dim i As Integer

   'Checking if there is a selected well
   If (Me.txt_WellNo(1).Text) = "" Then
      MsgBox (" Select a well  !"), vbCritical
      Exit Sub
   End If
   txt_maximumOutput.Text = ""
   txt_Generation.Text = ""
   txt_GenerationSinceMax.Text = ""

   framResults.Visible = True
   'Getting all well fixed information
   Call Well_Data(Me.txt_WellNo(1).Text, True)
      If Data_Existance = no Then Exit Sub

   Call SelectedOptions

   SaveFinal_Solution MyChromosome.Best_Chromosome,
LTrim(Me.txt_GAFracID.Text)
```

```
    If Me.opt_OptimizationType(0).Value = True Then Me.SSTab_Optimization.Tab = 2

    Set WellData = Nothing


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub StartBatch()

    Dim i As Integer


    With lst_Selected_BatchWell
        For i = 0 To .ListCount - 1
            SSPanel2.FloodPercent = 0
            fram_BatchProcceing.Visible = True
            .ListIndex = i
            Me.txt_WellNo(1).Text = .Text
            Call StartSingleProcessing
        Next
        Me.SSTab_Optimization.Tab = 2
    End With


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub SelectedOptions()

    'Chromosome Selection type option
    If Opt_Chromosome_Selection(0).Value = True Then
        Chromosome_SelectionType = Classical_Selection

    ElseIf Opt_Chromosome_Selection(1).Value = True Then
        Chromosome_SelectionType = CVOR_Selection
    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''


     'Cross Over Selection
    If opt_CrossOver(0).Value = True Then
        Cross_OverType = Classic_CrossOver

    ElseIf opt_CrossOver(1).Value = True Then
        Cross_OverType = Double_CrossOver

    ElseIf opt_CrossOver(2).Value = True Then
```

```
        Cross_OverType = Uniform_CrossOver

    End If
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''


    'Gene coding selection
    If opt_Gene_Coding(0).Value = True Then
        GA_Type = Digital_GA
        Call Digital_Proccess

    ElseIf opt_Gene_Coding(1).Value = True Then
        GA_Type = Analog_GA
        Call Analog_Proccess

    End If


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub GA_NetSelection()

    'Select which Net to fire

    If Me.opt_Batch_Net_Selection(0).Value = True Or
Me.opt_Batch_Net_Selection(1).Value = True Then
         NetSelection (LTrim(Me.txt_GAFracID.Text))
    ElseIf Me.opt_Batch_Net_Selection(2).Value = True Then
        NetSelection (LTrim(Me.txt_GAFracID.Text))
    End If


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Digital_Proccess()

    Dim Generation                      As Integer
    Dim TotalGeneration                   As Integer

    'Finding the cromosome length
    MyChromosome.Obtaine_ChromosomeLength
    'Initializing Population
    MyChromosome.Initialize_Population

    TotalGeneration = Trim(Val(frmAdvance.txt_Generation.Text))
    Generation_SinceMax = 0
```

```
    StopFlag = False

    'Start Generations loop
    For Generation = 0 To TotalGeneration - 1

        If StopFlag = True Then Exit Sub

        DoEvents
        'Start decoding the population by sending the population and population size and
        'Chromosome_Parameters_Borders to the Decode_Population
        MyChromosome.Decode_Population MyChromosome.MyPopulation,
MyChromosome.Chromosome_Parameters_Borders, _
                                MyChromosome.PopulationSize

        Call GA_NetFiring(MyChromosome.PopulationSize,
MyChromosome.Decoded_Population)

        txt_Generation.Text = Val(Generation + 1)
        Generation_SinceMax = Generation_SinceMax + 1
        txt_GenerationSinceMax.Text = Generation_SinceMax
        If txt_GenerationSinceMax.Text = MyChromosome.Stopping_Criteria Then Exit
Sub
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        MyChromosome.Chromosome_Output = NetOutput
        MyChromosome.Select_Best_Chromosome (MyChromosome.MyPopulation)
        MyChromosome.CrossOver (MyChromosome.Selected_Chromosome)
        MyChromosome.Mutate_Chromosome (MyChromosome.CrossOver_Chromosome)
        MyChromosome.MyPopulation = MyChromosome.Mutated_Chromosome
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        Call Draw(Generation + 1, MyChromosome.TotalFitness)
        SSPanel2.FloodPercent = ((Generation) / TotalGeneration) * 100          '
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    Next Generation

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Analog_Proccess()

    Dim Generation                              As Integer
    Dim TotalGeneration                         As Integer

    GA_Type = Analog_GA
    'Sending the minimum and maximum input borders
    MyChromosome.Analog_Initialize_Population _
                                MyChromosome.MinimumInputs_Array,
MyChromosome.MaximumInputs_Array
```

```
   TotalGeneration = LTrim(Val(frmAdvance.txt_Generation.Text))

   For Generation = 0 To TotalGeneration - 1

      If StopFlag = True Then Exit Sub

      DoEvents
      'Since the Chromosome population in this case is alreade decoded
      MyChromosome.Decoded_Population = MyChromosome.MyPopulation

      Call GA_NetFiring(MyChromosome.PopulationSize,
MyChromosome.Decoded_Population)

      txt_Generation.Text = Val(Generation + 1)
      Generation_SinceMax = Generation_SinceMax + 1
      txt_GenerationSinceMax.Text = Generation_SinceMax
      If txt_GenerationSinceMax.Text = MyChromosome.Stopping_Criteria Then Exit
Sub
      '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      MyChromosome.Chromosome_Output = NetOutput()
      MyChromosome.Select_Best_Chromosome (MyChromosome.Decoded_Population)
      MyChromosome.CrossOver (MyChromosome.Selected_Chromosome)
      MyChromosome.Mutate_Chromosome (MyChromosome.CrossOver_Chromosome)
      MyChromosome.MyPopulation = MyChromosome.Mutated_Chromosome

      '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      Call Draw(Generation + 1, MyChromosome.TotalFitness)                    '
      SSPanel2.FloodPercent = ((Generation + 1) / TotalGeneration) * 100      '
      '''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   Next Generation

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub GA_NetFiring(PopulationSize, DecodedPopulation)

   Dim i As Integer, j As Integer, sRow          As Integer
   Dim FracID                              As String
   Dim DeltaOutput                         As Double
   Dim maximumOutput                       As Double
   Dim FinalInputData
   ReDim NetOutput(UBound(DecodedPopulation)) As Double
   ReDim Best_Cromosome(0, UBound(DecodedPopulation, 2)) As Integer

   FracID = Trim(Me.txt_GAFracID.Text)
   'Select which Net to fire
   maximumOutput = 0
```

```
    'sending the Decoded parameters to the input class
    WellData.Well_GA_Input_Data = DecodedPopulation

    For sRow = 0 To PopulationSize - 1
          'Storing decoded parameters in the class
          DataFor = FireData
          WellData.GA_Input_Selection FracID, sRow

          'Select only data for the specific frac
          For i = 2 To UBound(InputsFieldsTitle)
             'Checking the frac ID
             If FracID = InputsFieldsTitle(i) Then
                   GA_NetSelection
                   FinalInputData = Inputs_array(WellData.Well_Input_Data, i,
FireNet.InputNo)
             End If
          Next
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

          ReDim InputsArray(UBound(FinalInputData)) As Double
          For i = LBound(FinalInputData) To UBound(FinalInputData) - 1
             InputsArray(i) = CDbl(FinalInputData(i))
          Next i

          FireNet.InputArry = InputsArray()
          FireNet.Fire_The_Net
          NetOutput(sRow) = FireNet.Net_OutPut
          DeltaOutput = NetOutput(sRow) - WellData.Q100_Before_This_Frac
'Me.txt_GA_Pre_Stimulation
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
          'Selecting the max in each generation
          If NetOutput(sRow) > maximumOutput Then
             Call BestChromosome(InputsArray, DeltaOutput)
          End If
       Next sRow

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function BestChromosome(myData, maximumOutput)

    'Compearing the max from each generation to the previouse generation max
    If maximumOutput > Val(Me.txt_maximumOutput.Text) Then

       Me.txt_maximumOutput = Format((maximumOutput), "####0.00")

       Generation_SinceMax = 0
```

```
        MyChromosome.Best_Chromosome = myData
        MyChromosome.Best_Net_Output = maximumOutput
    End If

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub SaveFinal_Solution(myData, FracID)

    Dim DataInput                               As New cOpen_Database

    ReDim InPutData(UBound(InputsFields) + 1) As Variant
    Dim i, k, iRow, iCol                         As Integer
    Dim mySQL                                   As String

    For i = 2 To UBound(InputsFieldsTitle)
        'Checking the frac ID
        If FracID = InputsFieldsTitle(i) Then
            k = 0

            For iRow = LBound(InPutData) To UBound(InPutData) - 1
                If InputsFields(iRow, 0) = "Frac No" Then InPutData(iRow) =
WellData.FracNo

                If InputsFields(iRow, i) = True Then
                    InPutData(iRow) = CDbl(myData(k))
                    k = k + 1
                ElseIf InputsFields(iRow, i) <> True And InputsFields(iRow, 0) <> "Frac No"
Then
                    InPutData(iRow) = 0

                End If

            Next iRow
        End If
    Next i

    WellData.FinalInputData (InPutData)
    DataFor = ViewData
    WellData.Well_Data

    InPutData = WellData.Well_Input_Data

     ReDim Preserve InPutData(UBound(InputsFields) + 1) As Variant
    InPutData(UBound(InPutData)) = MyChromosome.Best_Net_Output

    mySQL = "select * from tbl_FracRecipe_Output"
```

```
    DataInput.InputRecords = InPutData()
    DataInput.InPutData DataInput.InputRecords, mySQL, "GA"

    Cmd_Refresh_Click

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub opt_OptimizationType_Click(Index As Integer)

    If Me.opt_OptimizationType(0).Value = True Then
        fram_BatchProcceing.Visible = False
        Me.framResults.Visible = True
    ElseIf Me.opt_OptimizationType(1).Value = True Then
        fram_BatchProcceing.Visible = True
        Me.framResults.Visible = False
        Me.txt_WellNo(1).Text = ""
        Me.txt_GA_Pre_Stimulation.Text = ""

    End If

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub NetSelection(MyNet As String)

     Dim i                          As Integer
     Dim InputsNumber               As Integer
    'Looking for the Frac ID in the array same like the one selected
    'in the combo box
    For i = 1 To UBound(NeuralNet_Info.DataArray)
         If MyNet = NeuralNet_Info.DataArray(i, 3) Then
            'The associated neural network now assigned to Netname need to be fired
            FireNet.NetName = NeuralNet_Info.DataArray(i, 4)
            FireNet.InputNo = NeuralNet_Info.DataArray(i, 5)
            FireNet.OutputNo = NeuralNet_Info.DataArray(i, 6)
            InputsNumber = LTrim(NeuralNet_Info.DataArray(i, 7))
        End If
    Next i

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub lst_Selected_BatchWell_DblClick()
    Call SSPan_Move_Click(2)
End Sub
```

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub SSPan_Move_Click(Index As Integer)

    If Index = 0 Then
        'move one item from left to right
        If Lst_Wells(1).ListIndex >= 0 Then
            lst_Selected_BatchWell.AddItem Lst_Wells(1).Text
            Lst_Wells(1).RemoveItem Lst_Wells(1).ListIndex
        End If
    ElseIf Index = 1 Then
        'move All item from left to right
        Do While Lst_Wells(1).ListCount
            lst_Selected_BatchWell.AddItem Lst_Wells(1).List(0)
            Lst_Wells(1).RemoveItem 0
        Loop
    ElseIf Index = 2 Then
        'move one item from  right to left
        If lst_Selected_BatchWell.ListIndex >= 0 Then
            Lst_Wells(1).AddItem lst_Selected_BatchWell.Text
            lst_Selected_BatchWell.RemoveItem lst_Selected_BatchWell.ListIndex
        End If
    ElseIf Index = 3 Then
        'Move All item from  right to left
        Do While lst_Selected_BatchWell.ListCount
            Lst_Wells(1).AddItem lst_Selected_BatchWell.List(0)
            lst_Selected_BatchWell.RemoveItem 0
        Loop
    End If

    lbl_Count(0).Caption = Lst_Wells(1).ListCount & "  Wells"
    lbl_Count(1).Caption = lst_Selected_BatchWell.ListCount & "  Wells"

End Sub



'Private Sub txt_InputParameters_MouseMove(Index As Integer, Button As Integer, Shift
As Integer, X As Single, Y As Single)
'       'If Me.lbl_InputParameters(Index).Caption = "Frac Fluid" Then
'           'Me.txt_InputParameters(Index).Visible = False
'           'With Cmbo_Inputs
'               '.Top = txt_InputParameters(Index).Top
'               '.Visible = True
'               '.AddItem "Water"
'               '.AddItem "Gel"
```

```
'               '.AddItem "Foam"
'               '.ListIndex = 0
'            'End With
'       'ElseIf Me.lbl_InputParameters(Index).Caption = "Frac Fluid" Then
'
'       'ElseIf Me.lbl_InputParameters(Index).Caption = "Frac Fluid" Then
'
'       'End If
'
'End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
'Private Sub txt_InputParameters_MouseDown(Index As Integer, Button As Integer,
Shift As Integer, _
'                                X As Single, Y As Single)
'
'
'End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
'Private Sub txt_InputParameters_MouseUp(Index As Integer, Button As Integer, Shift
As Integer, _
'                                X As Single, Y As Single)
'
'    'If Me.lbl_InputParameters(Index).Caption = "Frac Fluid" Then
'         '   txt_InputParameters(Index).Visible = True
'          ' txt_InputParameters(Index).Text = Me.Cmbo_Inputs.Text
'   ' End If
'     '   Cmbo_Inputs.Visible = False
'
'
'End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmdChemStop_Click()
    StopFlag = True
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_ViewData_Click(Index As Integer)
    FrmEog_Frac_info.Show
End Sub
'Private Sub cmdPatchInterrupt_Click()
'    'This code stop the patch processing and by clicking againe
'    'it will restart it
'    Dim BatchFlag               As Boolean
'
'    If cmdPatchInterrupt.Caption = "Interrupt" Then
'        cmdPatchInterrupt.Caption = "Continue"
'        BatchFlag = True
```

```
'
'       While BatchFlag = True
'          DoEvents
'       Wend
'
'    ElseIf cmdPatchInterrupt.Caption = "Continue" Then
'
'       cmdPatchInterrupt.Caption = "Interrupt"
'       BatchFlag = False
'
'    End If
'
'End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub MSFlex_FiredQ_Click()
    'once clicking on the cell the well information
    'pumps to the user

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmdSort_Click()
    'MSFlex_FiredQ.Sort = 1
End Sub

Public Sub Copy_Frac_ID()

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Setup_Pic_To_Draw()

'    Me.MSFG_GA_Results.Visible = False
    Me.Pic_GA.Visible = True
    Me.Pic_GA.Cls
    Me.Pic_GA.Scale (0, 1)-(20, 0)
    Me.Pic_GA.DrawStyle = 0
    Me.Pic_GA.DrawWidth = 2

    X = 0:    Y = 0
    Call Draw(X, Y)

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Draw(X1, Y1)

    X1 = X1 / 5
    Y1 = Y1 / 350000
```

```
    Me.Pic_GA.Line (X, Y)-(X1, Y1), RGB(0, 255, 0)

   X = X1
   Y = Y1


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_WellInfo_Click()
    FrmEog_Frac_info.Show
    Me.Hide
End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Delete_Click()

    On Error GoTo Proc_Err

    With Adodc1
        .Recordset.Delete
        .Recordset.MoveFirst

    End With

    Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cdm_DeletAll_Click()

    Dim i As Integer

    'MsgBox ("Are you sure that you like to delete all Frac Recipes"), vbYesNo

    With Dgrid_WellNo

        With Adodc1.Recordset
            If .RecordCount > 0 Then
                For i = 0 To .RecordCount - 1
                    cmd_Delete_Click
                Next
```

```
            End If
        End With

    End With


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Cmd_Refresh_Click()

    On Error GoTo Proc_Err

    With Adodc1
        .Refresh
    End With

    Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Sort_Click()

    On Error GoTo Proc_Err

    With Adodc1

        If Me.cmd_Sort.Caption = "Sort by Deliverabiltiy" Then
            .Recordset.Sort = "Q100 DESC"
            Me.cmd_Sort.Caption = "Sort by Frac No"

        ElseIf Me.cmd_Sort.Caption = "Sort by Frac No" Then
            .Recordset.Sort = "[Frac No] ASC"
            Me.cmd_Sort.Caption = "Sort by Well No"

        ElseIf Me.cmd_Sort.Caption = "Sort by Well No" Then
            .Recordset.Sort = "[Well No] ASC"
            Me.cmd_Sort.Caption = "Sort by Deliverabiltiy"

        End If

    End With

    Exit Sub
```

```
Proc_Err:
   MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Rank_Click()

   With Me.MSFlex_FiredQ
      .Col = 2
      .Sort = 4
   End With

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmdAdvanced_Click()
   frmAdvance.Show
End Sub
Private Sub cmd_Exit_Click()
   frmNeural_GA.Hide
   Unload frmNeural_GA
   End
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Fuzzy_Click()
   frmFuzzy.Show
   Me.Hide
End Sub
```

Fuzzy Expert System Form

```
Option Explicit

   Dim OptionIndex          As Integer
   Dim Data                 As Variant
   Dim Percent              As Double
   Dim Domain()             As New c_Fuzzy_Logic
   Dim FuzzRules            As New c_Fuzzy_Logic
   Dim OutputValues         As New c_Fuzzy_Logic
   Dim OutputDomain         As New c_Fuzzy_Logic
   Dim UserInput()          As New c_Fuzzy_Logic
   Dim Field()              As String
   Dim Names()              As String
   Dim Output()             As Double
```

```
    Dim FireRule()                  As Boolean
    Dim User_Input()                As Double
    Dim Counter                     As Integer
    Dim strSQL                      As String
    Dim myFieldArray
    Dim myInput
    Dim FuzzyBatchFlag              As Boolean
    Dim User_InputFlag()            As Boolean
    Dim FireFlag                    As Boolean
    Dim WellList()                  As Double
    Dim BatchResults()              As Double
    Dim RulesFiredFlag              As Boolean


'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Form_Load()

    On Error GoTo Proc_Err

        Dim myData

        ReDim Domain(2) As New c_Fuzzy_Logic
        ReDim UserInput(2) As New c_Fuzzy_Logic
        ReDim User_InputFlag(2) As Boolean
        'Form Location
        Me.Left = (Screen.Width - Width) / 2
        Me.Top = (Screen.Height - Height) / 2
        Me.SSTab1.Tab = 0
        Me.SSFuzzyPanal.Visible = False
        FireFlag = False

        Call Filling_all_cmbos
        'Getting data from database
        Data = Get_Data_From_DB()

        opt_Domain_Click (0)
        opt_Fuzzy_Type_Click (0)
        Opt_Difuzz_Click (0)
        MSFlex_FuzzyBatchGrid.Visible = False
        FuzzyBatchFlag = False
        Call Output_Domain
        Call Rules
        RulesFiredFlag = False
        cmd_Submit_Click

    Exit Sub
```

```
Proc_Err:
   MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmbo_FracID_Click()

   'This code fill the list box with all wells that have been fraced
   'with same number that was selected from the combo box

   On Error GoTo Proc_Err

   Dim Well                        As Variant
   Dim FracNo                      As Integer
   Dim i                           As Integer


   Me.txt_FracID.Text = Me.cmbo_FracID.Text

   'Filling the List with Wells
   FracNo = cmbo_FracID.ListIndex + 1
   DatReturen = dtDataArray
   'Select all wells that have been fraced once
   strSQL = "SELECT [Well No] FROM tbl_FracRecipe_Output WHERE [Frac No]= "
& FracNo
   myFieldArray = Array("Well No")

   With lst_Wells
      .Clear
      Well = GetData(strSQL, myFieldArray)

      If Data_Existance = no Then
         If cmbo_FracID.ListIndex = cmbo_FracID.ListCount - 1 Then
            Exit Sub
         End If
         cmbo_FracID.ListIndex = cmbo_FracID.ListIndex + 1
         Exit Sub
      End If

      ReDim myWells(UBound(Well)) As Integer

      If UBound(Well) = 0 Then
         MsgBox ("There is no wells existed for this frac")
         Exit Sub
      Else
```

```
        For i = LBound(Well) To UBound(Well)
           .AddItem Well(i, 0)
        Next i
        .ListIndex = 0
      End If
   End With

   Exit Sub

Proc_Err:
   MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub lst_Wells_Click()

   'The following Sub select the well needed to be tested for frac
   'it brings the data from table

   On Error GoTo Proc_Err

   Dim FracNo                    As Integer
   Dim WellNo                    As Integer
   Dim i                         As Integer

   FracNo = cmbo_FracID.ListIndex + 1
   WellNo = Val(lst_Wells.Text)

   strSQL = "SELECT * FROM tbl_FracRecipe_Output WHERE [Frac No]= " & FracNo
   strSQL = strSQL + "And [Well No]= " & WellNo
   myFieldArray = Array("Well No", "QAverge", "Years Before This Frac", "Q100")

   myInput = GetData(strSQL, myFieldArray)

   If Data_Existance = no Then Exit Sub

   For i = LBound(myInput, 2) + 1 To UBound(myInput, 2)
      If i = 2 Then
         Me.txt_Input(i - 1).Text = Format(myInput(0, i) / FracNo, "#0.00")
      Else
         Me.txt_Input(i - 1).Text = myInput(0, i)
      End If
   Next i

   Exit Sub
```

```
Proc_Err:
   MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description



End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Function Get_Data_From_DB()

   'This code gets all the needed data for constructing the
   'Fuzzy support system and store them in side an array

   Dim Fieldrange                  As New cOpen_Database
   Dim myFields

   'Sending string to the class to open Neural Network table
   strSQL = "Select * from tbl_Fuzzy"
   'Fields Array of Neural network table
   myFieldArray = Array("Field Name", "Frac ID", "Minimum", "Maximum")
   Get_Data_From_DB = GetData(strSQL, myFieldArray)

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function GetData(mySQL, MyArray)

   Dim myData                      As New cOpen_Database

   'Store SQL and Field Array in the property of the open Data Class
   'then open the database and return only data array

   myData.mySQL = mySQL
   myData.FieldArray = MyArray
   DatReturen = dtDataArray
   myData.OpenDatabase
   myData.OpenRecordSet
   myData.CLoseDatabse
   'Checking for data Existance

      If myData.DataExistance = False Then
         Data_Existance = no
         MsgBox ("There is no Data for " & Me.txt_FracID.Text), vbCritical
         GetData = 0
      Else
         Data_Existance = Yes
         GetData = myData.DataArray
      End If
```

```
    Set myData = Nothing

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub InitializeAll_Domains()

    'This sub initalise all the Fuzzy sets by sending all the needed
    'information to the Fuzzy class and

    Dim i                                   As Integer
    Dim Domain_Range                            As Double
    Dim FracID                              As Integer


    For i = LBound(Domain) To UBound(Domain)
        Call Domain_Info(i, Data)
        'Specify how many set are there
        Domain(i).Domain_SubSet = Val(Me.txt_SubsetNumber.Text)
        'Storing the # of sets inside the subclass
        Domain(i).c_Fuzzy_SubSet.Subsets = Domain(i).Domain_SubSet
        'Domain Name
        Domain(i).Doamin_Name = Field(i)
        'Domain Range
        Domain_Range = Domain(i).Domain_Range
        'Subset range in each Domain
        Domain(i).c_Fuzzy_SubSet.Subset_Range = Domain_Range /
Domain(i).Domain_SubSet
        'Interval size.....this interval is the equal distances inside the Subset
        Domain(i).c_Fuzzy_SubSet.Interval = Domain_Range / ((2 *
Domain(i).Domain_SubSet) - 1)
        'Initializing Process sending the minimum and percent value to the function
        Domain(i).c_Fuzzy_SubSet.Initializing_Subset_Points
Domain(i).Domain_Minimum, Percent
        Domain(i).c_Fuzzy_SubSet.InitializingSlop Domain(i).c_Fuzzy_SubSet.X_Points
    Next i

    'cmd_Submit_Click

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Submit_Click()

    On Error GoTo Proc_Err

    Call InitializeAll_Domains
```

```
    cmd_Input_Click

     Exit Sub
Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Private Sub Opt_Difuzz_Click(Index As Integer)
    Diffazification_Type = Index
End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub opt_Domain_Click(Index As Integer)

    OptionIndex = Index

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub opt_Fuzzy_Type_Click(Index As Integer)

    'If the Fuzzy type is Normal or Core
    If Me.opt_Fuzzy_Type(0).Value = True Then
        SubSet_Type = Normal_SubSet
        Percent = 0
    ElseIf Me.opt_Fuzzy_Type(1).Value = True Then
       SubSet_Type = Core_SubSet
       Percent = 0.5
    End If


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Domain_Info(Index, myData)

    'This code gets the min and max values from a table in the database
    'then store this values inside the class properties

    Dim i As Integer
    ReDim Names(2) As String


    For i = LBound(myData) To UBound(myData)
```

```
        If myData(i, 0) = Field(Index) And myData(i, 1) = Me.cmbo_FracID.Text Then

            Domain(Index).Domain_Minimum = myData(i, 2)
            Domain(Index).Domain_Maximum = myData(i, 3)

        End If
    Next i


End Function

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Output_Domain()

    'Preparing the output Domain

    Dim Domain_Range                    As Double
    Dim i                               As Integer

        Call opt_Fuzzy_Type_Click(0)

        'Specify how many set are there
        OutputDomain.Domain_SubSet = Val(Me.txt_SubsetNumber.Text)
        'Storing the # of sets inside the subclass
        OutputDomain.c_Fuzzy_SubSet.Subsets = OutputDomain.Domain_SubSet
        OutputDomain.Doamin_Name = "OutPut Domain or think of new name"
        'Domain Range
        Domain_Range = OutputDomain.Domain_OutPutMax -
OutputDomain.Domain_OutputMin
        'Subset range in each Domain
        OutputDomain.c_Fuzzy_SubSet.Subset_Range = Domain_Range /
OutputDomain.Domain_SubSet
        'Interval size.....this interval is the equal distances inside the Subset
        OutputDomain.c_Fuzzy_SubSet.Interval = Domain_Range / ((2 *
OutputDomain.Domain_SubSet) - 1)
        'Initializing Process
        OutputDomain.c_Fuzzy_SubSet.Initializing_Subset_Points
OutputDomain.Domain_Minimum, Percent

        OutputDomain.c_Fuzzy_SubSet.InitializingSlop
OutputDomain.c_Fuzzy_SubSet.X_Points

        Call Fuzzy_Set_Draw(1, OutputDomain.c_Fuzzy_SubSet.X_Points, 0)

End Sub
```

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Cell_Arranging(arr, i)

    'We always have 5 column first one for the Fz_sets name
    'and 4 for the input of each Fz_sets
    Dim iRow                        As Integer
    Dim iCol                        As Integer

    With Me.MSF_Fuzzy_Points
        .Clear
        .Rows = UBound(arr) + 2
        .Cols = UBound(arr, 2) + 2
        .ColWidth(0) = 1200

        For iRow = LBound(arr) To UBound(arr)

            .TextMatrix(iRow + 1, 0) = " Fuzzy Set(" & iRow & ")"

            For iCol = LBound(arr, 2) To UBound(arr, 2)

                .ColWidth(iCol + 1) = 850
                .TextMatrix(0, iCol + 1) = " Q_" & iCol
                .TextMatrix(iRow + 1, iCol + 1) = Format(arr(iRow, iCol), "##.0")

            Next iCol

        Next iRow

    End With

    Call Fuzzy_Set_Draw(0, arr, i)

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Fuzzy_Set_Draw(Pic, arr, i)

    Dim FuzzySet                    As Integer
    Dim Point                       As Integer
    Dim X                           As Double
    Dim Y                           As Integer
    Dim Y1

    Pic_FuzzyDraw(Pic).Cls
```

```
   Pic_FuzzyDraw(Pic).Scale (arr(LBound(arr), LBound(arr, 2)), (1.5))-
((arr(UBound(arr), UBound(arr, 2))), 0)

   Me.Pic_FuzzyDraw(Pic).DrawWidth = 1
   'Starting point is the first point in the array
   X = arr(LBound(arr), LBound(arr, 2))
   Y1 = Domain(i).c_Fuzzy_SubSet.Y_Values_NormalSet

   For FuzzySet = LBound(arr) To UBound(arr)
      For Point = LBound(arr, 2) To UBound(arr, 2)
         ' if we start from the first point in the array
         If FuzzySet = LBound(arr) And Point = LBound(arr, 2) Then Y = Y1(Point)
         'Sending the values to drowing Sub
         Call Draw(Pic, X, Y, arr(FuzzySet, Point), Y1(Point), 1, 0, i + 1)
         'Swaping the points
         X = arr(FuzzySet, Point)
         Y = Y1(Point)
      Next Point
   Next FuzzySet

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Draw(Pic, X, Y, X1, Y1, Draw_Width, Style, Color)

   Me.Pic_FuzzyDraw(Pic).DrawWidth = Draw_Width
   Me.Pic_FuzzyDraw(Pic).DrawStyle = Style
   Me.Pic_FuzzyDraw(Pic).Line (X, Y)-(X1, Y1), QBColor(Color)
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub MSF_Fuzzy_Points_DblClick()
    MSF_Fuzzy_Points.Text = ""
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub MSF_Fuzzy_Points_KeyPress(KeyAscii As Integer)

   Dim Msg, Title, Help, Ctxt, Response, MyString
   Dim i As Integer

   With MSF_Fuzzy_Points
      .Text = .Text + Chr$(KeyAscii)
      If KeyAscii = 8 Or KeyAscii = 8 Or KeyAscii = 8 Then
         .Text = ""
         Exit Sub
      End If

      If .Text = "" Then
```

```
            Msg = " Missing Cell Value,  Please Enter A Value "
            Response = MsgBox(Msg, 0)
            Exit Sub
        End If

    End With

    For i = opt_Domain.LBound To opt_Domain.UBound
        If opt_Domain(i).Value = True Then User_InputFlag(i) = True
    Next i

    Me.opt_InputType(1).Value = True
    Me.cmd_Input.Caption = "Submit User Input"

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Input_Click()

    'Get all the Default points for the Domain subset

    Dim i                       As Integer
    Dim iRow                    As Integer
    Dim iCol                    As Integer
    Dim myData

    FireFlag = False

    If Me.opt_InputType(0).Value = True Then

        'Searching for the active domain
        For i = opt_Domain.LBound To opt_Domain.UBound
            If opt_Domain(i).Value = True Then
                'Send the Values to the grid system
                Call Cell_Arranging(Domain(i).c_Fuzzy_SubSet.X_Points, i)
                Me.lbl_DomainName.Caption = Me.opt_Domain(i).Caption
            End If
        Next i



    ElseIf Me.opt_InputType(1).Value = True Then

        If Me.cmd_Input.Caption = "Submit User Input" Then
            'Getting all the value from the table and save then inside User_input array
            With MSF_Fuzzy_Points
                ReDim User_Input(.Rows - 2, .Cols - 2)
```

```
            For iRow = LBound(User_Input) To UBound(User_Input)
               For iCol = LBound(User_Input, 2) To UBound(User_Input, 2)
                  If .TextMatrix(iRow + 1, iCol + 1) = "" Then
                     MsgBox (" Missing Cell Value,  Please Enter a Value "), vbCritical
                     Exit Sub
                  End If
                  User_Input(iRow, iCol) = Val(.TextMatrix(iRow + 1, iCol + 1))
               Next iCol
            Next iRow

         End With
      End If

      'Save inside the class and activate only the selected Domain
      For i = opt_Domain.LBound To opt_Domain.UBound

         If opt_Domain(i).Value = True Then

            If User_InputFlag(i) = True Then
               'Save in class
               UserInput(i).c_Fuzzy_SubSet.X_Points = User_Input
               'Activate
               Call Cell_Arranging(UserInput(i).c_Fuzzy_SubSet.X_Points, i)
             ElseIf User_InputFlag(i) = False Then

               MsgBox ("There is no user inputs for " & opt_Domain(i).Caption & " !"),
vbCritical
               Exit Sub

            End If
         End If

      Next i

   End If

End Sub
"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub opt_InputType_Click(Index As Integer)
   If Me.opt_InputType(0).Value = True Then
      Me.cmd_Input.Caption = "Get Default Input"
   ElseIf Me.opt_InputType(1).Value = True Then
      Me.cmd_Input.Caption = "Get User Input"
   End If
End Sub
"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
Private Sub cmdBatch_Fuzzy_Click()

   Dim i As Integer

   If Me.Opt_Rule_States(1).Value = True Then
      MsgBox ("Select Rule System") & vbCritical
      Exit Sub
   End If

   Me.SSFuzzyPanal.Visible = True
   Me.Pic_FuzzyDraw(1).Visible = False
   FireFlag = False
   FuzzyBatchFlag = True

   With MSFlex_FuzzyBatchGrid
      .Visible = True
      .Rows = Me.lst_Wells.ListCount + 1
      .Cols = 5
      .ColWidth(0) = 900: .ColWidth(1) = 1100: .ColWidth(2) = 1100: .ColWidth(3) =
1100: .ColWidth(3) = 1100
      .TextMatrix(0, 0) = "Well #"

      .TextMatrix(0, 1) = "Not Cand."
      .TextMatrix(0, 2) = "Likely Cand."
      .TextMatrix(0, 3) = "Cand."
      .TextMatrix(0, 4) = "Conf."

   End With

   With Me.lst_Wells

      For i = 0 To .ListCount - 1
         'Move Down the list
         .ListIndex = i
         Me.MSFlex_FuzzyBatchGrid.TextMatrix(i + 1, 0) = .Text
         cmd_MakeDecision_Click
         With Me.MSFlex_FuzzyBatchGrid
            .TextMatrix(i + 1, 1) = Me.txt_Output(0).Text
            .TextMatrix(i + 1, 2) = Me.txt_Output(1).Text
            .TextMatrix(i + 1, 3) = Me.txt_Output(2).Text
            .TextMatrix(i + 1, 4) = Me.txt_Diffuzzification
         End With
          Me.SSFuzzyPanal.FloodPercent = (i / (.ListCount - 1)) * 100
      Next

   End With
```

```
    FuzzyBatchFlag = False
    Me.SSFuzzyPanal.Visible = False
    RulesFiredFlag = True

End Sub
"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_MakeDecision_Click()

'serching for the location of the input

On Error GoTo Proc_Err


    Dim InputNo                 As Integer
    Dim iInput                  As Double
    Dim myData
    ReDim Output(UBound(Domain), Domain(0).Domain_SubSet - 1) As Double
    ReDim FireRule(UBound(Domain), Domain(0).Domain_SubSet - 1) As Boolean

    Me.SSFuzzyPanal.Visible = True

    If FuzzyBatchFlag = False Then
        Me.Pic_FuzzyDraw(1).Visible = True
        MSFlex_FuzzyBatchGrid.Visible = False
    End If
    FireFlag = True


    If Me.Opt_Rule_States(1).Value = True Then
        If FuzzyBatchFlag = False Then
            MsgBox ("Select Rule System") & vbCritical
            Exit Sub
        End If
    End If

    'looping through the input text boxes
    For InputNo = txt_Input.LBound To txt_Input.UBound
        'Get the the value of the doamin
        myData = Domain(InputNo).c_Fuzzy_SubSet.X_Points
        'Getting the input
        iInput = Val(txt_Input(InputNo).Text)

        'Cheking if the input inside the range or not
        If iInput < myData(LBound(myData), LBound(myData, 2)) Then
            iInput = myData(LBound(myData), LBound(myData, 2))
        ElseIf iInput > myData(UBound(myData), UBound(myData, 2)) Then
```

```
            iInput = myData(UBound(myData), UBound(myData, 2)) - 0.5
        End If

        'Find each input location
        Call Input_Location(InputNo, iInput, myData)

       Me.SSFuzzyPanal.FloodPercent = (InputNo / txt_Input.UBound) * 100
    Next InputNo


    Call Fired_Rules

    Set OutputValues = Nothing
    Me.SSFuzzyPanal.Visible = False
    RulesFiredFlag = True
    Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Input_Location(InputNo, iInput, myData)

    'Find the location of each input and in wich subset its located
    'it checkes the universe by checking each subset alone
    'InpuNo determine the universe

    Dim i As Integer
    Dim mySlop
    Dim Y_Values

    mySlop = Domain(InputNo).c_Fuzzy_SubSet.Slops
    Y_Values = Domain(InputNo).c_Fuzzy_SubSet.Y_Values_NormalSet

    'Serching each subset in each Universe
    For i = LBound(myData) To UBound(myData)

        FireRule(InputNo, i) = False
        Output(InputNo, i) = Empty

        'which Subset in Universe the input is located in
        If iInput >= myData(i, LBound(myData, 2)) And iInput <= myData(i,
UBound(myData, 2)) Then
            Call Q_Value(InputNo, iInput, myData, i, mySlop, Y_Values)
        End If
```

```
    Next i

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Q_Value(InputNo, iInput, myData, i, mySlop, Y_Values)


    'This code test in which part the Input is located by knowing
    'Then it calculate the Y value from the slop Equation

    Dim Interval                        As Integer

    'Serching each set in each Universe
    For Interval = LBound(myData, 2) To UBound(myData, 2) - 1
        'which Interval in subdet the input is located
        If iInput >= myData(i, Interval) And iInput <= myData(i, Interval + 1) Then

            If iInput = myData(i, Interval) Then iInput = myData(i, Interval) + 0.00001
            If iInput = myData(i, Interval + 1) Then iInput = myData(i, Interval) - 0.00001

            'Calculating Q_value by using Y = m (X-X1) + Y1
            Output(InputNo, i) = (mySlop(i, Interval) * (iInput - myData(i, Interval + 1))) _
                                 + Y_Values(Interval + 1)
            FireRule(InputNo, i) = True
        End If
    Next Interval

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Fired_Rules()

    'Loop through all the rules and only activate if the rule was fired

    Dim i, j                As Integer
    Dim k, Counter1         As Integer
    ReDim FiredRule(0) As Integer

    Counter = 0
    Counter1 = 0
    'loop through the Univerrse
    For i = LBound(Output, 2) To UBound(Output, 2)
        For j = LBound(Output, 2) To UBound(Output, 2)
            For k = LBound(Output, 2) To UBound(Output, 2)

                'Checking all the sets
```

```
            If FireRule(0, i) = True And FireRule(1, j) = True And FireRule(2, k) = True
Then

                ReDim Preserve FiredRule(Counter1) As Integer

                FiredRule(Counter1) = Counter
                Counter1 = Counter1 + 1
                OutputValues.RuleCase = MSFRules.TextMatrix(Counter, 8)
                'OutputValues.RuleCase = MSFRules.TextMatrix(Counter - 1, 9)
                OutputValues.AggregatedValue = Aggregation(Output(0, i), Output(1, j),
Output(2, k))
                OutputValues.Aggregate

            End If
            'Count all the rules
            Counter = Counter + 1
        Next k
    Next j
Next i

'Sending the output domain and slop
OutputValues.ComputeDiffuzifcation OutputDomain.c_Fuzzy_SubSet.X_Points _
                        , OutputDomain.c_Fuzzy_SubSet.Slops

'Store all the Fired Rules inside a class property
FuzzRules.FiredRules = RememberFiredRule(FiredRule)

Me.txt_Output(0).Text = Format(OutputValues.NotCandidate, "##,##0.00")
Me.txt_Output(1).Text = Format(OutputValues.LikelyCandidate, "##,##0.00")
Me.txt_Output(2).Text = Format(OutputValues.Candidate, "##,##0.00")
Me.txt_Diffuzzification.Text = Format(OutputValues.Diffuzzification, "##,##0.00")

DrawingOutputSets

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Aggregation(Output1, Output2, Output3)


'Check what type of aggregation is selected

If Me.Opt_Aggregation(0).Value = True Then

    If Output1 <= Output2 And Output1 <= Output3 Then
        Aggregation = Output1
    ElseIf Output2 <= Output1 And Output2 <= Output3 Then
```

```
        Aggregation = Output2
      ElseIf Output3 <= Output1 And Output3 <= Output2 Then
        Aggregation = Output3
      End If

   ElseIf Me.Opt_Aggregation(1).Value = True Then
      Aggregation = Output1 * Output2 * Output3

   End If


End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub DrawingOutputSets()

   Dim i As Integer
   Dim j As Integer
   Dim X, Y, Y1
   Dim myPoints
   Dim mySlops

   Call Output_Domain
   myPoints = OutputDomain.c_Fuzzy_SubSet.X_Points

   mySlops = 5
   For i = LBound(myPoints) To UBound(myPoints)

      For j = LBound(myPoints, 2) To UBound(myPoints, 2)

         If j = LBound(myPoints, 2) Or j = UBound(myPoints, 2) Then
            Y1 = 0
         ElseIf j = 1 And i = LBound(myPoints) Then
            Y1 = Val(txt_Output(i).Text)
         ElseIf j = 1 And i <> LBound(myPoints) Then
            myPoints(i, j) = myPoints(i, j - 1) + (Val(txt_Output(i).Text) / mySlops)
            Y1 = Val(txt_Output(i).Text)
         ElseIf j = 2 And i <> UBound(myPoints) Then
            myPoints(i, j) = myPoints(i, j + 1) + (Val(txt_Output(i).Text) / -mySlops)
            Y1 = Val(txt_Output(i).Text)
         End If

         Call Draw(1, X, Y, myPoints(i, j), Y1, 4, 2, 2)
         X = myPoints(i, j)
         Y = Y1
      Next j
   Next i
```

```
End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Retrive_Rules_Click()

    Dim i, j, k
    Dim myRules


    If Me.Opt_Rule_States(0).Value = True Then myRules = FuzzRules.Default_Rules
    If Me.Opt_Rule_States(1).Value = True Then myRules = FuzzRules.FiredRules
    If Me.Opt_Rule_States(2).Value = True Then myRules = FuzzRules.User_Rules


    With MSFRules

        For i = LBound(myRules) To UBound(myRules)
            For j = LBound(myRules, 2) To UBound(myRules, 2)
                If Me.Opt_Rule_States(1).Value = False Then
                    If j = 0 Then .TextMatrix(i + 1, j) = "Rules (" & myRules(i, j) & ")"
                    If j <> 0 Then .TextMatrix(i + 1, j) = myRules(i, j)
                Else
                    .TextMatrix(i + 1, j) = myRules(i, j)

                End If
            Next j
        Next i
    End With

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function RememberFiredRule(FiredRules) As Variant

    Dim i, j

     With MSFRules
       ReDim myRule_s(UBound(FiredRules), 9) As String

        For i = LBound(FiredRules) To UBound(FiredRules)
            For j = 0 To .Cols - 1
                'Copying the fired rule to the array
                myRule_s(i, j) = .TextMatrix(FiredRules(i) + 1, j)
            Next j
        Next i
```

```
    End With

     RememberFiredRule = myRule_s

End Function

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Fired_Rules_Click()


   If FireFlag = False Then Exit Sub
   Me.SSTab1.Tab = 1
   Me.Opt_Rule_States(1).Value = True
   FillRulesGrid_Header
   cmd_Retrive_Rules_Click

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Rules()

   'Writting the rules into the grid rules after importing them from
   ' FuzzRules.GetRules Property of the Fuzzy Class

   Dim i, j
   Dim myData

   'Getting the data from the class
   myData = FuzzRules.GetRules

   cmd_Retrive_Rules_Click

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub MSFRules_DblClick()

   With MSFRules
      If .Col = 8 Then
         .Text = Me.cmbo_Rules_Changing.Text
      ElseIf .Col = 9 Then
         .Text = Me.cmbo_True_Degree.Text
      End If
   End With

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_SaveChages_Click()
```

```
    Dim SaveRules                 As New cOpen_Database
    Dim mySQL                     As String
    Dim sConnection               As ADODB.Connection
    Dim sRecordset                As ADODB.Recordset

    Dim i                         As Integer
    Dim j                         As Integer

    Dim Msg, Style, Title, Response, MyString

    Msg = "Would You Like to Save Your Changes ? "
    Style = vbYesNo  ' Define buttons.
    Title = "Save New Rules"
    Response = MsgBox(Msg, Style, Title)

    mySQL = "select * from tbl_FuzzyRules"
    'open File and save rules use dialog box to save the rules
    If Response = vbYes Then
       With MSFRules
          ReDim NewUserRules(.Rows - 2, 0 To 1)
          For i = 0 To .Rows - 2
             For j = LBound(NewUserRules, 2) To UBound(NewUserRules, 2)
                NewUserRules(i, j) = .TextMatrix(i + 1, 8 + j)
             Next j
          Next i
       End With

       FuzzRules.User_Rules = NewUserRules
       SaveRules.InPutData FuzzRules.User_Rules, mySQL, "Fuzzy"
       Rules
    Else
       Exit Sub
    End If

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Filling_all_cmbos()

    'Filling all Cmbos and Grid

    ReDim Field(2) As String

    Field(0) = "Average Deliverability"
    Field(1) = "Years / Frac Ratio"
    Field(2) = "Post Frac Deliverability"
```

```
    cmbo_Rules_Changing.ListIndex = 0
    cmbo_True_Degree.ListIndex = 0

    With Me.cmbo_FracID
        .ListIndex = 0
        Me.txt_FracID.Text = Me.cmbo_FracID.Text
    End With

    Call FillRulesGrid_Header

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub FillRulesGrid_Header()

    With MSFRules
        .Clear
        .Rows = 28
        .Cols = 10
        MSFRules.AllowUserResizing = flexResizeBoth
        .ColWidth(0) = 900:   .TextMatrix(0, 0) = " Rule # ":
        .ColWidth(1) = 300:   .TextMatrix(0, 1) = "  "
        .ColWidth(2) = 1700:  .TextMatrix(0, 2) = "Average Deliverability"
        .ColWidth(3) = 450:   .TextMatrix(0, 3) = "  "
        .ColWidth(4) = 1350:  .TextMatrix(0, 4) = "Year / Frac Ratio"
        .ColWidth(5) = 450:   .TextMatrix(0, 5) = "  "
        .ColWidth(6) = 1450:  .TextMatrix(0, 6) = " Incremental PDF "
        .ColWidth(7) = 600:   .TextMatrix(0, 7) = "  "
        .ColWidth(8) = 1400:  .TextMatrix(0, 8) = " Status "
        .ColWidth(8) = 1600:  .TextMatrix(0, 9) = " Truth Qualification "
    End With

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_WellInformation_Click()
    FrmEog_Frac_info.Show
End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Exit_Click()
    Me.Hide
    Unload frmFuzzy
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_systemOptimization_Click()
```

```
    frmNeural_GA.Show
    Me.Hide
    Unload frmFuzzy
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_Rank_Click()

    If Me.MSFlex_FuzzyBatchGrid.Visible = True Then
        With Me.MSFlex_FuzzyBatchGrid
            .Col = 4
            .Sort = 4
        End With
    End If

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Opt_Rule_States_Click(Index As Integer)
    If Index = 1 Then
        If RulesFiredFlag = False Then
            MsgBox ("No rules were fired yet !")
            Opt_Rule_States(0).Value = True
            Exit Sub
        End If
    End If
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub cmd_PrintToFile_Click()

    Dim i               As Integer
    Dim j               As Integer
    Dim iFile             As New File

    With Me.MSFlex_FuzzyBatchGrid
        ReDim BatchResults(.Rows - 1, .Cols - 1)
        For i = 1 To .Rows - 1
            For j = 0 To .Cols - 1
                BatchResults(i - 1, j) = .TextMatrix(i, j)
            Next j
        Next i
    End With

    iFile.FileExtention = "txt"
    iFile.FileName = "Ristimulation list"
    iFile.DataToSave = BatchResults()
    iFile.FileOutput
```

```
End Sub
Private Sub cmd_ExitApplication_Click()
   Unload frmFuzzy
   End
End Sub
```

Open Database Class

```
Option Explicit
   'References ADO 2.5
   'ADO Objects Used
   Private sConnection                As ADODB.Connection
   Private sRecordset                As ADODB.Recordset
   Private strSQL                    As String
   Private mvarMySQL                 As String
   Dim arr()                         As Variant
   Private mvarFieldArray            As Variant
   Private mvarDataArray()           As Variant
   Private mvarFieldsName            As Variant
   Private mvarRecordsCount          As Integer
   Private mvarMyFlag                As Boolean
   Dim MyArray()                     As Variant
   Dim myFieldsName()                As Variant
   Private mvarSingleOutput          As Variant
   Private mvarDataExistance         As Boolean
   Private mvarInputRecords          As Variant

   Public Enum DataType
      dtUniqueValue = 1
      dtMultiValues = 2
      dtDataArray = 3
      dtFiledsName = 4
   End Enum

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let MyFlag(ByVal vData As Boolean)
   mvarMyFlag = vData
End Property
Public Property Get MyFlag() As Boolean
   MyFlag = mvarMyFlag
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let mySQL(ByVal vData As String)
   'Store the Sql string
   mvarMySQL = vData
End Property
Public Property Get mySQL() As String
```

```
    mySQL = mvarMySQL
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@
Public Property Let FieldArray(arr As Variant)
    'Store the fields from the needed database in array
    mvarFieldArray = arr
End Property
Public Property Get FieldArray() As Variant
    FieldArray = mvarFieldArray
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub OpenDatabase()

    On Error GoTo Proc_Err

    'Setting up the connection String to the MDB File
    Set sConnection = New ADODB.Connection
    'Set sRecordset = New Recordset
    'Open the connection to the DB
    With sConnection
        .Provider = "Microsoft.Jet.OLEDB.4.0"
        .ConnectionString = App.Path + "\Eogdb.mdb"
        .CursorLocation = adUseClient
        .Open
    End With

    Exit Sub

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@
Public Sub CLoseDatabse()
     With sConnection
        .Close
     End With
End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub OpenRecordSet()

    On Error GoTo Proc_Err
    'Open the record set
```

```
    Set sRecordset = New Recordset
    sRecordset.Source = mySQL
    sRecordset.ActiveConnection = sConnection
    sRecordset.LockType = adLockOptimistic
    sRecordset.CursorType = adOpenKeyset
    sRecordset.Open

    Me.RecordsCount = sRecordset.RecordCount

    If sRecordset.RecordCount = 0 Then
        Me.DataExistance = False
        Exit Sub
    Else
       Me.DataExistance = True
    End If

    If DatReturen = DataType.dtUniqueValue Then
          'Getting the unique value
          Me.SingleOutput = sRecordset.Fields(0)
    ElseIf DatReturen = DataType.dtMultiValues Then
          'Getting multie values
          Get_MyMultiValues
    ElseIf DatReturen = DataType.dtDataArray Then
          'Convert my recordset to an array
          Get_MyArray
    ElseIf DatReturen = DataType.dtFiledsName Then
          'Getting the fields name
          Get_MyFields
    End If
    'Closing the connection
    Set sRecordset = Nothing
    Exit Sub
Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Get_MyMultiValues()
    Dim i                       As Integer
    Dim j                       As Integer

    ReDim MyArray(sRecordset.Fields.Count - 1)
    For i = 0 To sRecordset.Fields.Count - 1
          If Not IsNull(sRecordset.Fields(i)) Then
                MyArray(i) = sRecordset.Fields(i)
```

```
        Else
           Me.DataExistance = False
           Exit Sub
        End If
    Next i

    Me.DataArray = MyArray()
    Erase MyArray()
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Get_MyArray()

    Dim values                  As Variant
    Dim i                       As Integer
    Dim j                       As Integer

    'Store the recordset indise a variant Array
    'remember that each record is a field when using this method
    'to store the data inside an array
    values = sRecordset.GetRows(, , FieldArray)
    ReDim MyArray(UBound(values, 2), UBound(values))
    ' Transpose the data array
    For j = LBound(values, 2) To UBound(values, 2)  'Columns
       For i = LBound(values) To UBound(values)    'Rows
          If Not IsNull(values(i, j)) Then
             MyArray(j, i) = values(i, j)
          Else
             Me.DataExistance = False
             Exit Sub
          End If
       Next i
    Next j

    Me.DataArray = MyArray()

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Get_MyFields()

    Dim i                       As Integer
    Dim j                       As Integer
    ReDim myFieldsName(sRecordset.Fields.Count - 1) As Variant
    For i = 0 To sRecordset.Fields.Count - 1
          myFieldsName(i) = sRecordset.Fields(i).Name
    Next
```

```
    Me.FieldsName = myFieldsName()

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub InPutData(Inputs As Variant, mySQL As String, Store As String)


    Dim sConnection                  As ADODB.Connection
    Dim sRecordset                   As ADODB.Recordset
    Dim i                            As Integer
    Dim j                            As Integer

        'Setting up the connection String to the MDB File
        Set sConnection = New ADODB.Connection
        'Open the connection to the DB
        With sConnection
            .Provider = "Microsoft.Jet.OLEDB.4.0"
            .ConnectionString = App.Path + "\Eogdb.mdb"
            .CursorLocation = adUseClient
            .Open
        End With

        Set sRecordset = New Recordset

        With sRecordset
            .Source = mySQL
            .ActiveConnection = sConnection
            .LockType = adLockOptimistic
            .CursorType = adOpenKeyset
            .Open
        End With

""""""""""""""""""""""""""""""""""
        If Store = "Fuzzy" Then
            sRecordset.Update
            For i = LBound(Inputs) To UBound(Inputs)
                    sRecordset.Fields(9).Value = Inputs(i, 0)
                    sRecordset.Fields(11).Value = Inputs(i, 1)
                sRecordset.MoveNext
            Next i
            sRecordset.Close
""""""""""""""""""""""""""""""""""
        ElseIf Store = "GA" Then
            If sRecordset.EOF <> True Or sRecordset.BOF <> True Then
                sRecordset.MoveLast
            End If
```

```
         sRecordset.AddNew
         For i = LBound(Inputs) To UBound(Inputs)
               sRecordset.Fields(i).Value = Inputs(i)
         Next i
         sRecordset.Update
''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      ElseIf Store = "Update_FracNo" Then
         sRecordset.Update
            sRecordset.Fields(4).Value = Inputs
            sRecordset.MoveLast
            sRecordset.Close
''''''''''''''''''''''''''''''''''''''''''''''''''''''''

      End If

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let InputRecords(ByVal vData As Variant)
   mvarInputRecords = vData
End Property
Public Property Get InputRecords() As Variant
      InputRecords = mvarInputRecords
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let SingleOutput(ByVal vData As Variant)
   mvarSingleOutput = vData
End Property
Public Property Get SingleOutput() As Variant
      SingleOutput = mvarSingleOutput
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let RecordsCount(ByVal vData As Integer)
   mvarRecordsCount = vData
End Property
Public Property Get RecordsCount() As Integer
   RecordsCount = mvarRecordsCount
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let FieldsName(ByVal vData As Variant)
   mvarFieldsName = vData
End Property
Public Property Get FieldsName() As Variant
      FieldsName = mvarFieldsName
End Property
```

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let DataArray(arr As Variant)
   mvarDataArray() = arr
End Property
Public Property Get DataArray() As Variant
      DataArray = mvarDataArray
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let DataExistance(ByVal vData As Boolean)
   mvarDataExistance = vData
End Property
Public Property Get DataExistance() As Boolean
   DataExistance = mvarDataExistance
End Property
```

Open Net Class

```
Option Explicit

    'local variable(s) to hold property value(s)
    Private mvarNet                 As String
    Private mvarInputArry           As Variant
    Private mvarNet_OutPut          As Double
    Dim Input_Array()               As Double
    Dim ii                          As Integer
    Private mvarInputNo             As Integer
    Private mvarOutputNo            As Integer
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let OutputNo(ByVal vData As Integer)
    '# output of the Neural Network
    mvarOutputNo = vData
End Property
Public Property Get OutputNo() As Integer
    '# output of the Neural Network
    OutputNo = mvarOutputNo
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let InputNo(ByVal vData As Integer)
    '# input of the Neural Network
    mvarInputNo = vData
End Property
Public Property Get InputNo() As Integer
    '# input of the Neural Network
    InputNo = mvarInputNo
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let NetName(ByVal vData As String)

    mvarNet = vData
End Property
Public Property Get NetName() As String

    NetName = mvarNet
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let InputArry(arr)

    mvarInputArry = arr
End Property
Public Property Get InputArry()
```

```
    InputArry = mvarInputArry
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Fire_The_Net()

    'First Frac inptus & outputs declaration
    Dim Frac_Input As Integer
    Dim Frac_Output As Integer
    Dim Output
    Dim NNW_Name As String

    NNW_Name = NetName

    'Openning the NNW  only once and assigning the NNW inputs
    'and outputs # to other variables
    If Not Netisopen Then
          ii = OpenNet(NNW_Name, Netnumber, Inputs, Outputs)

       If ii > 0 Then
          MsgBox "Error returned from OpenNet: " + Str$(ii), 16, "Error"
          CloseNet (Netnumber)
          Exit Sub
       End If

       Netisopen = True
       'Inputs assigned by the NNW
       Frac_Input = Inputs
       'Outputs assigned by the NNW
       Frac_Output = Outputs
    End If

    If Frac_Input <> (Me.InputNo) Or Frac_Output <> (Me.OutputNo) Then
       MsgBox "Number of inputs or outputs in selected .DEF file is incorrect. Inputs
should be 21 and outputs should be 1", 16, "Error"
       CloseNet (Netnumber)
       Netisopen = False
       Exit Sub
    End If

    Input_Array = InputArry

    'After collecting the input inside InputArray
    'start fire the net using these inputs
    ii = FireNet(Netnumber, Input_Array(0), Output)
    If ii > 0 Then
```

```
        MsgBox "Error returned from FireNet: " + Str$(ii), 16, "Error"
        Exit Sub
    End If

    Me.Net_OutPut = Output
    'Closing the NeuralNetwork
    CloseNet (Netnumber)
    Netisopen = False

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Net_OutPut(ByVal vData As Double)

    mvarNet_OutPut = vData
End Property
Public Property Get Net_OutPut() As Double

    Net_OutPut = mvarNet_OutPut
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Genetic Algorithm Input Class

```
Option Explicit

    Dim AllWell_Data                    As New cOpen_Database
    Dim strSQL                          As String
    """""""""""""""
    Private mvarWell_No                  As Integer
    Private mvarX_Coor                   As Double
    Private mvarY_Coor                   As Double
    Private mvarElevation                As Integer
    """""""""""""""
    Private mvarQ_Min                    As Integer
    Private mvarQ_Max                    As Integer
    Private mvarQ_Average                As Integer
    """""""""""""""""""""""""""
    Private mvarNo_of_Months_Before_Frac        As Double
    Private mvarQ100_Before_This_Frac           As Integer
```

```
''''''''''''''''''''''''''''''''
Private mvarFrac_Dtae                      As Variant
Private mvarYearsBefore_This_Frac              As Integer
Private mvarFrac_Fluid                     As Variant
Private mvarViscosity                      As Variant
Private mvarFluid_Volume                   As Variant
Private mvarN2_Rate                        As Double
Private mvarN2_Volume                      As Double
Private mvarSand_Volume                    As Variant
Private mvarSand_Concentration             As Variant
Private mvarSand_Mesh                      As Variant
Private mvarAcid_Volume                    As Integer
Private mvarAverage_Rate                   As Double
Private mvarService_Company                As Variant
''''''''''''''''''''''''''''''
Private mvarOriginal_Input_Data            As Variant
Private mvarWell_Input_Data()              As Variant
Private mvarLastFrac_Date                  As Variant
Private mvarFracNo                         As Variant
Private mvarFracID                         As Variant
Private mvarWell_GA_Input_Data()              As Variant
Dim m_Well_Data()                          As Variant
Dim Final_WellData()                       As Variant

Public Enum OptimizationType
   GA_Optimization = 1
   Fuzzy_Optimization = 2
End Enum

Public Enum GettingDatafor
   SingleOptimization = 1
   BatchOptimization = 2
End Enum

Public Enum DataExistance
   Yes = 1
   no = 2
End Enum

Public Enum DataUse
   ViewData = 1
   FireData = 2
End Enum
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
Public Property Let Well_No(ByVal vData As Variant)
```

```
    mvarWell_No = vData
End Property
Public Property Get Well_No() As Variant
    Well_No = mvarWell_No
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let FracNo(ByVal vData As Variant)
    mvarFracNo = vData
End Property
Public Property Get FracNo() As Variant
    FracNo = mvarFracNo
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function GetWellData(Well As Integer, FracNo As Integer, optimization As
Boolean) As Double

    'Getting all well fixed information store them inside class properties
    On Error GoTo Proc_Err

        AllWell_Data.OpenDatabase
        'Getting Well location information
        Call Well_Location(Well)
            If Data_Existance = no Then GoTo FastExit
        'Getting well Deliverability statistics
        Well_Deliverability_Statistics (Well)
            If Data_Existance = no Then GoTo FastExit
        'Getting well Deliverability History
       Call WellDelevirability_History(Well)
            If Data_Existance = no Then GoTo FastExit

              Well_Frac_Recipe (Well)

    'Store all the data inside one Properties
    Call Well_Data

FastExit:
    AllWell_Data.CLoseDatabse

    Set AllWell_Data = Nothing

    Exit Function

Proc_Err:
    MsgBox "Error Number " & Err.Number & vbNewLine & Err.Description
```

```
End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Well_Location(Well As Integer)

    'This code gets the well locations and store them inside location properties

   DatReturen = dtMultiValues
   AllWell_Data.mySQL = "SELECT [X_COOR], [Y_COOR], [ELEVATION] FROM
tbl_Wells_Info WHERE [WELL_No]= " & Well
   AllWell_Data.OpenRecordSet

   If AllWell_Data.DataExistance = False Then
      Data_Existance = no
      If Getting_Datafor = SingleOptimization Then
         If frmNeural_GA.optNet_Processing(1).Value = True Then Exit Sub
         'If not batch processig then give rise Rerror and exit
            MsgBox ("Location Information for Well " & Me.Well_No & " is Missing "),
vbCritical
            Exit Sub
      ElseIf Getting_Datafor = BatchOptimization Then


      End If
   Else
      Data_Existance = Yes
      'start storing inside properties
      Me.X_Coor = AllWell_Data.DataArray(0)
      Me.Y_Coor = AllWell_Data.DataArray(1)
      Me.Elevation = AllWell_Data.DataArray(2)

   End If

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let X_Coor(ByVal vData As Double)
   mvarX_Coor = vData
End Property
Public Property Get X_Coor() As Double
   X_Coor = mvarX_Coor
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Y_Coor(ByVal vData As Double)
   mvarY_Coor = vData
```

```
End Property
Public Property Get Y_Coor() As Double
   Y_Coor = mvarY_Coor
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Elevation(ByVal vData As Integer)
   mvarElevation = CInt(vData)
End Property
Public Property Get Elevation() As Integer
   Elevation = mvarElevation
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Well_Deliverability_Statistics(Well As Integer)

   'This code prepaer the SQL to open the database and
   'get Q100 statistics

   Dim strSQL                    As String

   DatReturen = dtMultiValues
   strSQL = " WHERE [WeLL No]='" & Well & "' And [Q100] <> 0"
   AllWell_Data.mySQL = "SELECT MIN([Q100]), MAX([Q100]), AVG([Q100])
FROM tbl_Q100" + strSQL
   AllWell_Data.OpenRecordSet

      If AllWell_Data.DataExistance = False Then
         'if it was single optimization the give MsgBox
         If Getting_Datafor = SingleOptimization Then
            Data_Existance = no
            If frmNeural_GA.optNet_Processing(1).Value = True Then Exit Sub
            MsgBox ("Deliverability Information for Well " & Me.Well_No & " is Mising
"), vbCritical
            Exit Sub
         'if it was Batch optimization then skip and go to
         'the next well
         ElseIf Getting_Datafor = BatchOptimization Then

         End If
      Else
         Data_Existance = Yes
         Me.Q_Min = AllWell_Data.DataArray(0)
         Me.Q_Max = AllWell_Data.DataArray(1)
         Me.Q_Average = CInt(AllWell_Data.DataArray(2))
         Data_Existance = Yes
      End If
```

```
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Q_Min(ByVal vData As Integer)
    mvarQ_Min = vData
End Property
Public Property Get Q_Min() As Integer
    Q_Min = mvarQ_Min
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Q_Max(ByVal vData As Integer)
    mvarQ_Max = vData
End Property
Public Property Get Q_Max() As Integer
    Q_Max = mvarQ_Max
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Q_Average(ByVal vData As Integer)
    mvarQ_Average = vData
End Property
Public Property Get Q_Average() As Integer
    Q_Average = mvarQ_Average
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub WellDelevirability_History(Well As Integer)

    'This code prepaer the SQL to open the database and
    'get Q100 history from there we get the last Q100 before
    'Frac and date with # on months

    Dim iData

    DatReturen = dtDataArray
    strSQL = "SELECT * FROM tbl_Q100 WHERE [WeLL No]='" & Well & "' And
[Q100] <> 0"
    'Order the output by years
    AllWell_Data.mySQL = strSQL + " ORDER BY [YEAR]"
    AllWell_Data.FieldArray = Array("YEAR", "Q100")
    AllWell_Data.OpenRecordSet

    'if it was single optimization the give MsgBox
    If AllWell_Data.DataExistance = False Then
        Data_Existance = no
        If Getting_Datafor = SingleOptimization Then
            If frmNeural_GA.optNet_Processing(1).Value = True Then Exit Sub
```

```
        MsgBox ("Deliverability Information for Well " & Me.Well_No & " is Mising "),
vbCritical
        Exit Sub
      'if it was Batch optimization then skip and go to
      'the next well
      ElseIf Getting_Datafor = BatchOptimization Then

      End If
   Else
      Data_Existance = Yes
      iData = AllWell_Data.DataArray
      Me.No_of_Months_Before_Frac = iData(UBound(iData), LBound(iData, 1))    'Last
row first column
      Me.Q100_Before_This_Frac = iData(UBound(iData), UBound(iData, 2))       'Last
row second column
   End If

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let No_of_Months_Before_Frac(ByVal vData As Double)
   'Number of months Since Q100 was recorded
   'First it gets the last year the test was conducted then converted to months
   'by taking the difference between now and the last year the test was conducted
   'divid it by 12

   Dim MyYear_now              As Double
   Dim Q_Year                  As Double

   Q_Year = vData

   MyYear_now = Format((year(Date) + (Month(Date) / 12)), "##,##0.00")
   mvarNo_of_Months_Before_Frac = (MyYear_now - Q_Year) * 12

End Property
Public Property Get No_of_Months_Before_Frac() As Double
   No_of_Months_Before_Frac = mvarNo_of_Months_Before_Frac
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Q100_Before_This_Frac(ByVal vData As Integer)
   mvarQ100_Before_This_Frac = vData
End Property

Public Property Get Q100_Before_This_Frac() As Integer
   Q100_Before_This_Frac = mvarQ100_Before_This_Frac
End Property
```

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Well_Frac_Recipe(Well As Integer)

   'This code Prepare the SQL to get the frac Recipe information from the
   'Database and store them inside the Class

   Dim myData
   Dim j  As Integer
   Dim X As String

   DatReturen = dtDataArray

   AllWell_Data.mySQL = " Select * FROM tbl_Min_Max WHERE FracID = '" &
frmNeural_GA.txt_FracID.Text & "'"
   AllWell_Data.FieldArray = Array("Frac Fluid", "Fluid Volume", "N2 Rate", "N2
Volume SCF", "Sand Volume", _
         "Sand Concentration", "Sand Mesh", "Acid Volume", "Average Rate", "Service
Company")

   AllWell_Data.OpenRecordSet
   myData = AllWell_Data.DataArray
   'Establish 2 arrayes
   ReDim Minimum(UBound(myData, 2))
   ReDim Maximum(UBound(myData, 2))

   For j = LBound(myData, 2) To UBound(myData, 2)
      Minimum(j) = myData(LBound(myData), j)
      Maximum(j) = myData(UBound(myData), j)
   Next j

   myData = Analog_Initialize_Frac(Minimum, Maximum)

      Me.Frac_Fluid = myData(0)
      Me.Fluid_Volume = myData(1)
      Me.N2_Rate = myData(2)
      Me.N2_Volume = myData(3)
      Me.Sand_Volume = myData(4)
      Me.Sand_Concentration = myData(5)
      Me.Sand_Mesh = myData(6)
      Me.Acid_Volume = myData(7)
      Me.Average_Rate = myData(8)
      Me.Service_Company = myData(9)

      DatReturen = dtUniqueValue
      X = Str(Me.FracNo - 1)
```

```
      If X > 0 Then
         'AllWell_Data.mySQL = "SELECT [Frac Date] FROM tbl_Fracrecipe WHERE
[WeLL No]='" & Well & "' and [FRAC NO]= " & X

         strSQL = "SELECT [Frac Date] FROM tbl_Fracrecipe WHERE [WELL No]= '"
& Well & "' And ([FRAC No]) = " & Me.FracNo - 1
         'strSQL = strSQL + " And ([FRAC No]) =" & Me.FracNo - 1

          AllWell_Data.FieldArray = Array("FRAC No", "Frac Date")
         AllWell_Data.mySQL = strSQL

         AllWell_Data.OpenRecordSet

         Me.LastFrac_Date = AllWell_Data.SingleOutput
      Else
         Me.LastFrac_Date = 0
      End If


End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Analog_Initialize_Frac(Min, Max)
   'This Function Generate a Random Population Size using the
   'Properties of  Population Size and Minimum maximum input borders

   Dim j                           As Integer
   ReDim m_population(UBound(Min))
   Randomize
      For j = 0 To UBound(Max)
          m_population(j) = Int(Min(j) + Rnd * Max(j))
      Next j

   'store the population in MyPopulation properties
   Analog_Initialize_Frac = m_population()

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let LastFrac_Date(ByVal vData As Variant)
   mvarLastFrac_Date = vData
End Property
Public Property Get LastFrac_Date() As Variant

      LastFrac_Date = mvarLastFrac_Date
End Property
```

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Frac_Fluid(ByVal vData As Variant)
   mvarFrac_Fluid = vData
End Property
Public Property Get Frac_Fluid() As Variant

    If DataFor = ViewData Then
      If mvarFrac_Fluid = 1 Then
            mvarFrac_Fluid = "Gel"
            'Me.Viscosity = 45
      ElseIf mvarFrac_Fluid = 2 Then:          mvarFrac_Fluid = "Water"
      ElseIf mvarFrac_Fluid = 3 Then:          mvarFrac_Fluid = "Foam"
      ElseIf mvarFrac_Fluid = 4 Then:          mvarFrac_Fluid = "Foam"
      End If
   ElseIf DataFor = FireData Then
      mvarFrac_Fluid = UCase(mvarFrac_Fluid)

      If mvarFrac_Fluid = "GEL" Or mvarFrac_Fluid = "Gel" Then
         mvarFrac_Fluid = 1
      ElseIf mvarFrac_Fluid = "WATER" Or mvarFrac_Fluid = "Water" Then:
mvarFrac_Fluid = 2
      ElseIf mvarFrac_Fluid = "FOAM" Or mvarFrac_Fluid = "Foam" Then:
mvarFrac_Fluid = 3
      Else
        'mvarFrac_Fluid = 2
      End If
   End If
   Frac_Fluid = mvarFrac_Fluid
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Viscosity(ByVal vData As Variant)
   mvarViscosity = vData
End Property
Public Property Get Viscosity() As Variant
   Viscosity = mvarViscosity
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Fluid_Volume(ByVal vData As Variant)
   mvarFluid_Volume = vData
End Property
Public Property Get Fluid_Volume() As Variant
   Fluid_Volume = mvarFluid_Volume
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let N2_Rate(ByVal vData As Double)
```

```
    mvarN2_Rate = vData
End Property
Public Property Get N2_Rate() As Double
    N2_Rate = mvarN2_Rate
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 Public Property Let N2_Volume(ByVal vData As Double)
    mvarN2_Volume = vData
End Property
Public Property Get N2_Volume() As Double
    N2_Volume = mvarN2_Volume
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Sand_Volume(ByVal vData As Integer)
    mvarSand_Volume = vData
End Property
Public Property Get Sand_Volume() As Integer
    Sand_Volume = mvarSand_Volume
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Sand_Concentration(ByVal vData As Variant)
    If Me.Sand_Volume = 0 Then vData = 0
        mvarSand_Concentration = vData
End Property
Public Property Get Sand_Concentration() As Variant
    Sand_Concentration = mvarSand_Concentration
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 Public Property Let Sand_Mesh(ByVal vData As Variant)
    If Me.Sand_Volume = 0 Then vData = 0
        mvarSand_Mesh = vData
End Property
Public Property Get Sand_Mesh() As Variant

    If DataFor = ViewData Then
        If mvarSand_Mesh = 0 Then
            mvarSand_Mesh = "0"
        ElseIf mvarSand_Mesh = 1 Then:   mvarSand_Mesh = "10/20"
        ElseIf mvarSand_Mesh = 2 Then:   mvarSand_Mesh = "16/30"
        ElseIf mvarSand_Mesh = 3 Then:   mvarSand_Mesh = "20/40"
        ElseIf mvarSand_Mesh = 4 Then:   mvarSand_Mesh = "80 up"
        ElseIf mvarSand_Mesh >= 5 Then:   mvarSand_Mesh = "80/100"
        End If
    ElseIf DataFor = FireData Then
        If mvarSand_Mesh = "0" Then
            mvarSand_Mesh = 0
```

```
        ElseIf mvarSand_Mesh = "10/20" Then:    mvarSand_Mesh = 1
        ElseIf mvarSand_Mesh = "16/30" Then:    mvarSand_Mesh = 2
        ElseIf mvarSand_Mesh = "20/40" Then:    mvarSand_Mesh = 3
        ElseIf mvarSand_Mesh = "80 up" Then:    mvarSand_Mesh = 4
        ElseIf mvarSand_Mesh = "80/100" Then:   mvarSand_Mesh = 5
        End If
    End If
        Sand_Mesh = mvarSand_Mesh
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Acid_Volume(ByVal vData As Integer)
    mvarAcid_Volume = vData
End Property
Public Property Get Acid_Volume() As Integer
    Acid_Volume = mvarAcid_Volume
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Average_Rate(ByVal vData As Long)

    If Me.Fluid_Volume = 0 Then vData = 0
    mvarAverage_Rate = vData
End Property
Public Property Get Average_Rate() As Long
    Average_Rate = mvarAverage_Rate
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Service_Company(ByVal vData As Variant)
    mvarService_Company = vData
End Property

Public Property Get Service_Company() As Variant

     If DataFor = ViewData Then
        If mvarService_Company = 1 Then
                                    mvarService_Company = "DS"
        ElseIf mvarService_Company = 2 Then:       mvarService_Company = "HA"
        ElseIf mvarService_Company = 3 Then:       mvarService_Company = "NO"
        ElseIf mvarService_Company = 4 Then:       mvarService_Company = "UN"
        Else
           'mvarService_Company = "DS"
        End If

    ElseIf DataFor = FireData Then
        mvarService_Company = UCase(mvarService_Company)
        If mvarService_Company = "DS" Then
           mvarService_Company = 1
```

```
        ElseIf mvarService_Company = "HA" Then:   mvarService_Company = 2
        ElseIf mvarService_Company = "NO" Then:   mvarService_Company = 3
        ElseIf mvarService_Company = "UN" Then:   mvarService_Company = 4
        Else
           'mvarService_Company = 1
        End If

    End If
    Service_Company = mvarService_Company
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Get Frac_Dtae() As Variant

    'This cade provide the recent day if its for view
    If DataFor = ViewData Then
        ' View in day format
        Frac_Dtae = Date
    ElseIf DataFor = FireData Then
        'Provide as number for NN processing
        Frac_Dtae = CDbl(Date)
    End If

End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let YearsBefore_This_Frac(ByVal vData As Integer)

    If Me.FracNo > 0 Then
        vData = CInt((CDbl(Date) - Me.LastFrac_Date) / 365)
        mvarYearsBefore_This_Frac = vData
    Else
        mvarYearsBefore_This_Frac = 0
    End If

End Property
Public Property Get YearsBefore_This_Frac() As Integer

    If Me.FracNo > 1 Then
        mvarYearsBefore_This_Frac = CInt((CDbl(Date) - CDbl(Me.LastFrac_Date)) / 365)
    Else
        mvarYearsBefore_This_Frac = 0
    End If

    YearsBefore_This_Frac = mvarYearsBefore_This_Frac

End Property
```

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Well_Data()
    'This code stores all the parameters without considering frac order inside one Array
then ...
    'The array later stored these values inside 2 of the class properties Original_Input_Data
    'store the original values Well_Input_Data .... the one takes the changes from text
boxes

    Dim m_Well_Data(22)

        m_Well_Data(0) = Me.Well_No
        m_Well_Data(1) = Me.X_Coor
        m_Well_Data(2) = Me.Y_Coor
        m_Well_Data(3) = Me.Elevation
        'm_Well_Data(4) = Me.Frac_Dtae
        m_Well_Data(5) = Me.FracNo
        m_Well_Data(6) = Me.YearsBefore_This_Frac
        m_Well_Data(7) = Me.Q_Min
        m_Well_Data(8) = Me.Q_Max
        m_Well_Data(9) = Me.Q_Average
        m_Well_Data(10) = Me.No_of_Months_Before_Frac
        m_Well_Data(11) = Me.Q100_Before_This_Frac
        ''''''''''''''''''''''''''''''''''''''''''''''''
        If DataFor = ViewData Then
            m_Well_Data(4) = Me.Frac_Dtae
                m_Well_Data(12) = Me.Frac_Fluid
                m_Well_Data(13) = Me.Viscosity
                m_Well_Data(14) = Me.Fluid_Volume
                m_Well_Data(15) = Me.N2_Rate
                m_Well_Data(16) = Me.N2_Volume
                m_Well_Data(17) = Me.Sand_Volume
                m_Well_Data(18) = Me.Sand_Concentration
                m_Well_Data(19) = Me.Sand_Mesh
                m_Well_Data(20) = Me.Acid_Volume
                m_Well_Data(21) = Me.Average_Rate
                m_Well_Data(22) = Me.Service_Company
        ElseIf DataFor = FireData Then
            m_Well_Data(4) = Me.Frac_Dtae
                m_Well_Data(12) = Me.Frac_Fluid
                m_Well_Data(13) = Me.Viscosity
                m_Well_Data(14) = Me.Fluid_Volume
                m_Well_Data(15) = Me.N2_Rate
                m_Well_Data(16) = Me.N2_Volume
                m_Well_Data(17) = Me.Sand_Volume
                m_Well_Data(18) = Me.Sand_Concentration
```

```
            m_Well_Data(19) = Me.Sand_Mesh
            m_Well_Data(20) = Me.Acid_Volume
            m_Well_Data(21) = Me.Average_Rate
            m_Well_Data(22) = Me.Service_Company
        End If
        """""""""""""""""""""""""""""""""""
        Me.Original_Input_Data = m_Well_Data
        'Store all the data inside class property Well_Input_Data
        Me.Well_Input_Data = m_Well_Data

        Erase m_Well_Data

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Well_Input_Data(ByVal vData As Variant)
    'This property contains all well data
    mvarWell_Input_Data = vData
End Property
Public Property Get Well_Input_Data() As Variant
        Well_Input_Data = mvarWell_Input_Data
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Original_Input_Data(ByVal vData As Variant)
    mvarOriginal_Input_Data = vData
End Property
Public Property Get Original_Input_Data() As Variant
        Original_Input_Data = mvarOriginal_Input_Data
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function Changed_Well_Data(vData)

    'This code store all the values that has been changed from
    'the text box
    Dim m_Well_Data
    m_Well_Data = vData

        Me.Frac_Fluid = m_Well_Data(12)
        Me.Viscosity = m_Well_Data(13)
        Me.Fluid_Volume = m_Well_Data(14)
        Me.N2_Rate = m_Well_Data(15)
        Me.N2_Volume = m_Well_Data(16)
        Me.Sand_Volume = m_Well_Data(17)
        Me.Sand_Concentration = m_Well_Data(18)
        Me.Sand_Mesh = m_Well_Data(19)
```

```
        Me.Acid_Volume = m_Well_Data(20)
        Me.Average_Rate = m_Well_Data(21)
        Me.Service_Company = m_Well_Data(22)

    Call Me.Well_Data
    Changed_Well_Data = Me.Well_Input_Data

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Well_GA_Input_Data(vData As Variant)
    mvarWell_GA_Input_Data() = vData
End Property
Public Property Get Well_GA_Input_Data() As Variant
        Well_GA_Input_Data = mvarWell_GA_Input_Data
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@
Public Sub GA_Input_Selection(FracID As String, sRow)

    'This code attach all the decoded input parameters to
    ' the class input properties

    Dim m_Well_Data()
    m_Well_Data = Me.Well_GA_Input_Data

    Select Case FracID
        Case "First Frac"
            'Restore the new values
            Me.Frac_Fluid = m_Well_Data(sRow, 0)
                Call Viscosity_Selection(m_Well_Data(sRow, 0))
            Me.Fluid_Volume = m_Well_Data(sRow, 1)
            Me.N2_Rate = m_Well_Data(sRow, 2)
            Me.N2_Volume = m_Well_Data(sRow, 3)
            Me.Sand_Volume = m_Well_Data(sRow, 4)
            Me.Sand_Concentration = m_Well_Data(sRow, 5)
            Me.Sand_Mesh = m_Well_Data(sRow, 6)
            Me.Acid_Volume = m_Well_Data(sRow, 7)
            Me.Average_Rate = m_Well_Data(sRow, 8)
            Me.Service_Company = m_Well_Data(sRow, 9)

        Case "Second Frac"
            'Restore the new values
            Me.Frac_Fluid = m_Well_Data(sRow, 0)
                Call Viscosity_Selection(m_Well_Data(sRow, 0))
            Me.Fluid_Volume = m_Well_Data(sRow, 1)
```

```
            Me.N2_Rate = m_Well_Data(sRow, 2)
            Me.N2_Volume = m_Well_Data(sRow, 3)
            Me.Sand_Volume = m_Well_Data(sRow, 4)
            Me.Sand_Concentration = m_Well_Data(sRow, 5)
            Me.Sand_Mesh = m_Well_Data(sRow, 6)
            Me.Acid_Volume = m_Well_Data(sRow, 7)
            Me.Average_Rate = m_Well_Data(sRow, 8)
            Me.Service_Company = m_Well_Data(sRow, 9)

        Case "Third Frac"
            'Restore the new values
            Me.Frac_Fluid = m_Well_Data(sRow, 0)
               Call Viscosity_Selection(m_Well_Data(sRow, 0))
            Me.Fluid_Volume = m_Well_Data(sRow, 1)
            Me.N2_Rate = m_Well_Data(sRow, 2)
            Me.N2_Volume = m_Well_Data(sRow, 3)
            Me.Sand_Volume = m_Well_Data(sRow, 4)
            Me.Sand_Concentration = m_Well_Data(sRow, 5)
            Me.Sand_Mesh = m_Well_Data(sRow, 6)
            Me.Acid_Volume = m_Well_Data(sRow, 7)
            Me.Average_Rate = m_Well_Data(sRow, 8)
            Me.Service_Company = m_Well_Data(sRow, 9)

        Case "All Frac"
            'Restore the new values
            Me.Frac_Fluid = m_Well_Data(sRow, 0)
               Call Viscosity_Selection(m_Well_Data(sRow, 0))
            Me.Fluid_Volume = m_Well_Data(sRow, 1)
            Me.N2_Rate = m_Well_Data(sRow, 2)
            Me.N2_Volume = m_Well_Data(sRow, 3)
            Me.Sand_Volume = m_Well_Data(sRow, 4)
            Me.Sand_Concentration = m_Well_Data(sRow, 5)
            Me.Sand_Mesh = m_Well_Data(sRow, 6)
            Me.Acid_Volume = m_Well_Data(sRow, 7)
            Me.Average_Rate = m_Well_Data(sRow, 8)
            Me.Service_Company = m_Well_Data(sRow, 9)

    End Select

    DataFor = FireData
    Me.Well_Data

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Viscosity_Selection(Value)
```

```
    If Value = 1 Then
        Me.Viscosity = 15
    ElseIf Value = 2 Then
        Me.Viscosity = 1
    ElseIf Value = 3 Then
        Me.Viscosity = 45
       ' Me.Viscosity = 139.7
    End If

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub FinalInputData(InPutData As Variant)

    'This code stores all the parameters without considering frac order inside one Array
then ...
    'The array later stored these values inside 2 of the class properties Original_Input_Data
    'store the original values Well_Input_Data .... the one takes the changes from text
boxes

    Dim m_Well_Data(22)

        Me.Well_No = InPutData(0)
        Me.X_Coor = InPutData(1)
        Me.Y_Coor = InPutData(2)
        Me.Elevation = InPutData(3)
       ' Me.Frac_Dtae = InPutData(4)
        Me.FracNo = InPutData(5)
        Me.YearsBefore_This_Frac = InPutData(6)
        Me.Q_Min = InPutData(7)
        Me.Q_Max = InPutData(8)
        Me.Q_Average = InPutData(9)
        'Me.No_of_Months_Before_Frac = InPutData(10)
        Me.Q100_Before_This_Frac = InPutData(11)

        """""""""""""""""""""""""""""""""
        Me.Frac_Fluid = InPutData(12)
            Call Viscosity_Selection(Me.Frac_Fluid)
            Me.Viscosity = Me.Viscosity
        Me.Fluid_Volume = InPutData(14)
        Me.N2_Rate = InPutData(15)
        Me.N2_Volume = InPutData(16)
        Me.Sand_Volume = InPutData(17)
        Me.Sand_Concentration = InPutData(18)
        Me.Sand_Mesh = InPutData(19)
        Me.Acid_Volume = InPutData(20)
        Me.Average_Rate = InPutData(21)
```

```
        Me.Service_Company = InPutData(22)
        '''''''''''''''''''''''''''''''''''''

        Me.Well_Input_Data = m_Well_Data

End Sub
```

Fuzzy Class

```
Option Explicit

   Private mvarDoamin_Name                 As String
   Private mvarDomain_Color                As String
   Private mvarDomain_Range                As Double
   Private mvarDomain_SubSet               As Integer
   Private mvarc_Fuzzy_SubSet              As c_Fuzzy_SubSet
   Private mvarDomain_Minimum              As Double
   Private mvarDomain_Maximum              As Double
   Private mvarDomain_OutputMin            As Integer
   Private mvarDomain_OutPutMax            As Integer
   Private mvarRuleCase                    As String
   Private mvarDefault_Rules               As Variant
   Private mvarUser_Rules                  As Variant
   Private mvarAggregatedValue             As Double
   Private mvarDiffuzzification            As Double
   Private mvarLikelyCandidate             As Double
   Private mvarCandidate                   As Double
   Private mvarNotCandidate                As Double
   Private mvarFiredRules                  As Variant

   Enum DiffazificationType
      AverageWeight = 0
      Centroid = 1
   End Enum

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Class_Initialize()
   'create the mc_Fuzzy_SubSet object when the c_Fuzzy_Logic class is created
   Set mvarc_Fuzzy_SubSet = New c_Fuzzy_SubSet
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Get c_Fuzzy_SubSet() As c_Fuzzy_SubSet
   Set c_Fuzzy_SubSet = mvarc_Fuzzy_SubSet
End Property

Public Property Set c_Fuzzy_SubSet(vData As c_Fuzzy_SubSet)
   Set mvarc_Fuzzy_SubSet = vData
End Property

Public Property Let Doamin_Name(ByVal vData As String)
   'This property stores domain name
   mvarDoamin_Name = vData
End Property
```

```
Public Property Get Doamin_Name() As String
   Doamin_Name = mvarDoamin_Name
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Domain_SubSet(ByVal vData As Integer)
   'Stores the number of Subset in the domain
   mvarDomain_SubSet = vData
End Property
Public Property Get Domain_SubSet() As Integer
   Domain_SubSet = mvarDomain_SubSet
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Domain_Maximum(ByVal vData As Double)
   mvarDomain_Maximum = vData
End Property
Public Property Get Domain_Maximum() As Double
    Domain_Maximum = mvarDomain_Maximum
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Domain_Minimum(ByVal vData As Double)
   mvarDomain_Minimum = vData
End Property
Public Property Get Domain_Minimum() As Double
   Domain_Minimum = mvarDomain_Minimum
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Get Domain_Range() As Double
   Domain_Range = Me.Domain_Maximum - Me.Domain_Minimum
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Get Domain_OutPutMax() As Integer
   mvarDomain_OutPutMax = 1
   Domain_OutPutMax = mvarDomain_OutPutMax
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Public Property Get Domain_OutputMin() As Integer
    mvarDomain_OutputMin = 0
   Domain_OutputMin = mvarDomain_OutputMin
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Function GetRules() As Variant

   'openning the data base and get the Rules from the table
```

```
    Dim Fuzz_Rules                  As New cOpen_Database

    Fuzz_Rules.OpenDatabase
    DatReturen = dtDataArray
    Fuzz_Rules.mySQL = "Select * From tbl_FuzzyRules"
    'Get all the table structure
    Fuzz_Rules.FieldArray = Array("RulesNo", "Condition", "InputOne", "Condition1",
"InputTwo", _
                    "Condition2", "InputThree", "Condition3")

        Fuzz_Rules.OpenRecordSet
        GetRules = Fuzz_Rules.DataArray
    'Get the Defaults rules
    'Fuzz_Rules.FieldArray = Array("Default Rules", "Default True")

    Fuzz_Rules.FieldArray = Array("RulesNo", "Condition", "InputOne", "Condition1",
"InputTwo", _
                    "Condition2", "InputThree", "Condition3", "Default Rules", "Default
True")




        Fuzz_Rules.OpenRecordSet
        Me.Default_Rules = Fuzz_Rules.DataArray
    'Get user rules
   ' Fuzz_Rules.FieldArray = Array("User Rules", "User True")

    Fuzz_Rules.FieldArray = Array("RulesNo", "Condition", "InputOne", "Condition1",
"InputTwo", _
                    "Condition2", "InputThree", "Condition3", "User Rules", "User True")


        Fuzz_Rules.OpenRecordSet
        Me.User_Rules = Fuzz_Rules.DataArray

    Fuzz_Rules.CLoseDatabse

End Function
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let User_Rules(arr As Variant)
    mvarUser_Rules = arr
End Property
Public Property Get User_Rules() As Variant
    User_Rules = mvarUser_Rules
End Property
```

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Default_Rules(arr As Variant)
   mvarDefault_Rules = arr
End Property
Public Property Get Default_Rules() As Variant
   Default_Rules = mvarDefault_Rules
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let FiredRules(arr As Variant)
   mvarFiredRules = arr
End Property
Public Property Get FiredRules() As Variant
      FiredRules = mvarFiredRules
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let AggregatedValue(ByVal vData As Double)
   mvarAggregatedValue = vData
End Property
Public Property Get AggregatedValue() As Double
   AggregatedValue = mvarAggregatedValue
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let RuleCase(ByVal vData As String)
    mvarRuleCase = vData
End Property
Public Property Get RuleCase() As String
   RuleCase = mvarRuleCase
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Public Sub Aggregate()

   If Me.RuleCase = "Not Candidate" Then
      Me.NotCandidate = Me.AggregatedValue
   ElseIf Me.RuleCase = "Likely Candidate" Then
      Me.LikelyCandidate = Me.AggregatedValue
   ElseIf Me.RuleCase = "Candidate" Then
      Me.Candidate = Me.AggregatedValue
   End If

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let NotCandidate(ByVal vData As Double)

   If Me.NotCandidate >= vData Then
      mvarNotCandidate = Me.NotCandidate
```

```
    Else
        mvarNotCandidate = vData
    End If

End Property

Public Property Get NotCandidate() As Double
    NotCandidate = mvarNotCandidate
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Public Property Let LikelyCandidate(ByVal vData As Double)

    If Me.LikelyCandidate >= vData Then
        mvarLikelyCandidate = Me.LikelyCandidate
    Else
        mvarLikelyCandidate = vData
    End If

End Property

Public Property Get LikelyCandidate() As Double
    LikelyCandidate = mvarLikelyCandidate
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Candidate(ByVal vData As Double)

    If Me.Candidate >= vData Then
        mvarCandidate = Me.Candidate
    Else
        mvarCandidate = vData
    End If

End Property

Public Property Get Candidate() As Double
    Candidate = mvarCandidate
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Public Sub ComputeDiffuzifcation(myData, mySlop)

    Dim i               As Integer
    Dim Results(2)      As Double
    Dim Mid             As Double
    Dim AreaSum         As Double
```

```
    Dim ResultsSum          As Double

    AreaSum = 0
    ResultsSum = 0
    Results(0) = Me.NotCandidate
    Results(1) = Me.LikelyCandidate
    Results(2) = Me.Candidate

    If Diffazification_Type = AverageWeight Then
        'looping through the subsets
        For i = LBound(myData) To UBound(myData)

            AreaSum = Degree_confidence(i, Results(i), myData, mySlop) + AreaSum
            ResultsSum = Results(i) + ResultsSum

        Next

        If ResultsSum = 0 Then
            Diffuzzification = 0
            Exit Sub
        End If

        Diffuzzification = AreaSum / ResultsSum

    ElseIf Diffazification_Type = Centroid Then

        For i = LBound(myData) To UBound(myData)

            AreaSum = (Degree_confidence(i, Results(i), myData, mySlop) * Results(i)) +
AreaSum
            ResultsSum = Results(i) + ResultsSum

        Next

        If ResultsSum = 0 Then
            Diffuzzification = 0
            Exit Sub
        End If

        Diffuzzification = (AreaSum / ResultsSum) '/ AreaSum

    End If

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Diffuzzification(ByVal vData As Double)
```

```
    mvarDiffuzzification = vData
End Property
Public Property Get Diffuzzification() As Double
    Diffuzzification = mvarDiffuzzification
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Sub Class_Terminate()
    Set mvarc_Fuzzy_SubSet = Nothing
End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Private Function Degree_confidence(i, My_Y, myData, mySlop)


    Dim myPoint                     As Integer
    Dim Interval                    As Integer
    Dim TotalArea                   As Double
    Dim Area(2) As Double



    If Diffazification_Type = AverageWeight Then

          Area(i) = My_Y * ((myData(i, LBound(myData, 2)) + myData(i,
UBound(myData, 2))) / 2)

        TotalArea = Area(i)

    ElseIf Diffazification_Type = Centroid Then
        For Interval = 1 To UBound(myData, 2)
            If mySlop(i, Interval - 1) = 0 Then
                Area(Interval - 1) = My_Y * (myData(i, Interval) - myData(i, Interval - 1))
            Else
                If mySlop(i, Interval - 1) > 0 Then
                    'the set left side
                    myData(i, Interval - 1) = (-(My_Y) / mySlop(i, Interval - 1)) + myData(i,
Interval)
                    Area(Interval - 1) = 0.5 * My_Y * (myData(i, Interval) - myData(i, Interval -
1))
                ElseIf mySlop(i, Interval - 1) < 0 Then
                    'the set right side
                    myData(i, Interval - 1) = (My_Y) / mySlop(i, Interval - 1) + myData(i,
Interval)
                    Area(Interval - 1) = 0.5 * My_Y * (myData(i, Interval) - myData(i, Interval -
1))
                End If
```

```
            End If
            TotalArea = TotalArea + Area(Interval - 1)
        Next Interval

    End If

  Degree_confidence = TotalArea

End Function
```

Fuzzy Set Initialization Class

```
Option Explicit

    Private mvarSubsets                      As Integer
    Private mvarSubset_Range                 As Double

    Private mvarX_Points                     As Variant
    Private mvarInterval                     As Double
    Private mvarShape                        As String
    Private mvarSlops                        As Variant
    Private mvarState                        As String
    Private mvarOutPut_Value                 As Double
    Private mvarUser_X_Points                As Double
    Private mvarY_Values_NormalSet           As Variant
    Private mvarY_Values_CoreSet             As Variant
    Private mvarY_values                     As Variant

    Dim Y(3)                                 As Integer
    Public Enum SubSetType
        Normal_SubSet = 1
        Core_SubSet = 2
        User_Input_SubSet = 3
        OutputSubSet = 4
    End Enum
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Subsets(ByVal vData As Integer)
'   Stores the number of sunset inside the Domain
    mvarSubsets = vData
End Property

Public Property Get Subsets() As Integer
    Subsets = mvarSubsets
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
 Public Property Let Subset_Range(ByVal vData As Double)
   'Stores the range of each set
    mvarSubset_Range = vData
End Property
Public Property Get Subset_Range() As Double
    Subset_Range = mvarSubset_Range
End Property
```

```
Public Property Let Interval(ByVal vData As Double)
   'Stores the inside equale interval of subset
   'Its used in overlab
   mvarInterval = vData
End Property
Public Property Get Interval() As Double
   Interval = mvarInterval
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Public Property Let Shape(ByVal vData As String)
   'The Subset Shape ... Normal / Core
   mvarShape = vData
End Property
Public Property Get Shape() As String
   Shape = mvarShape
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub Initializing_Subset_Points(Min, Percent)

   'Initializing  Subset pionts .... With uniform  steps(Interval)
   'The Array always include 4 columns refer to the 4 Q_values that construct the Subset
   'While the Rows are variable refer to the # of Subset

   Dim X As Double
   Dim FuzzySet As Integer
   Dim Point As Integer
   ReDim m_Subset_Pionts(Me.Subsets - 1, 3) As Double

   'First Point always = to the minimum point in the domain
   X = Min
   'First loob loob through the Subsets number
   For FuzzySet = LBound(m_Subset_Pionts) To UBound(m_Subset_Pionts)
      'second loob..loobs through the points construct the subset
      For Point = LBound(m_Subset_Pionts, 2) To UBound(m_Subset_Pionts, 2)
         'Initializing subset by subset each set consists of 4 point
         m_Subset_Pionts(FuzzySet, Point) = X
         'Set the first two points in the first subset equal
         If FuzzySet = LBound(m_Subset_Pionts) Then
            If Point = LBound(m_Subset_Pionts, 2) Then m_Subset_Pionts(FuzzySet,
Point) = X
            If Point = LBound(m_Subset_Pionts, 2) + 1 Then
               m_Subset_Pionts(FuzzySet, Point) = m_Subset_Pionts(FuzzySet, Point - 1)
               X = X - Interval
            End If
         'For other than the first subset
```

```
        ElseIf FuzzySet > LBound(m_Subset_Pionts) Then
           If Point <= LBound(m_Subset_Pionts, 2) + 1 Then
              m_Subset_Pionts(FuzzySet, Point) = m_Subset_Pionts(FuzzySet - 1, Point +
2)
              X = X - Me.Interval
           Else
              m_Subset_Pionts(FuzzySet, Point) = X
           End If
        End If

        'Set the last two points in the last subset equal
        If FuzzySet = UBound(m_Subset_Pionts) And Point =
UBound(m_Subset_Pionts, 2) Then
           m_Subset_Pionts(FuzzySet, Point) = m_Subset_Pionts(FuzzySet, Point - 1)
        End If
        X = X + Me.Interval
     Next Point
  Next FuzzySet


  If SubSet_Type = Normal_SubSet Then
     Me.X_Points = m_Subset_Pionts()
  ElseIf SubSet_Type = Core_SubSet Then
     Call CoreShape(m_Subset_Pionts(), Percent)
  End If

End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub CoreShape(arr, Percent)

  Dim FuzzySet As Integer
  Dim Point As Integer
  Dim Location As Double

  For FuzzySet = LBound(arr) To UBound(arr)
     For Point = UBound(arr, 2) To UBound(arr, 2)

        Location = (arr(FuzzySet, LBound(arr, 2)) + arr(FuzzySet, Point)) * Percent
        arr(FuzzySet, Point - 2) = Location
        arr(FuzzySet, Point - 1) = arr(FuzzySet, Point - 2)

     Next Point
  Next FuzzySet

  Me.X_Points = arr
```

```
End Sub

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let X_Points(arr As Variant)
   mvarX_Points = arr
End Property
Public Property Get X_Points() As Variant
   X_Points = mvarX_Points
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Get Y_Values_NormalSet() As Variant
       Y(0) = 0: Y(1) = 1: Y(2) = 1: Y(3) = 0
       Y_Values_NormalSet = Y()
End Property
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Sub InitializingSlop(arr)

   'code to calculate different slops in each Fz set
   Dim FuzzySet                    As Integer
   Dim Point                       As Integer
   Dim vData
   Dim Y

   vData = arr
   Y = Y_Values_NormalSet

   ReDim m_Slop(Me.Subsets - 1, UBound(vData, 2) - 1) As Double

   For FuzzySet = LBound(vData) To UBound(vData)
      'Starting from the second point
      For Point = (LBound(vData, 2) + 1) To UBound(vData, 2)
         If Y(Point) = Y(Point - 1) Then
            m_Slop(FuzzySet, Point - 1) = 0
         'this Condition is when two X values are equal
         ElseIf vData(FuzzySet, Point) = vData(FuzzySet, Point - 1) Then
           m_Slop(FuzzySet, Point - 1) = 0
         Else
           'slop in the left of the Fz set always > 0 (increasing)
           'slop in the right of the Fz set always < 0 (decreasing)
           m_Slop(FuzzySet, Point - 1) = (Y(Point) - Y(Point - 1)) / _
                                (vData(FuzzySet, Point) - vData(FuzzySet, Point - 1))
         End If

      Next Point
```

```
    Next FuzzySet

   Me.Slops = m_Slop()

End Sub
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Public Property Let Slops(arr As Variant)
    mvarSlops = arr
End Property

Public Property Get Slops() As Variant
    Slops = mvarSlops
End Property

'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Public Property Let OutPut_Value(ByVal vData As Double)
    mvarOutPut_Value = vData
End Property
Public Property Get OutPut_Value() As Double
    OutPut_Value = mvarOutPut_Value
End Property
```