Petroleum and Coal

Article **Open Access**

Functional Analysis of Performance of Artificial Neural Networks (MPL) and (RBF) in predicting seismicity of tubes related to offshore drilling in one of Iran's oilfields

Sajjad Mozaffari[1, 2], Armin Hosseinian[1,2,3], Arash Pourabdol Shahrekordi[1, 2], Seyed Jamal Sheikhzakariaee[1], Mohammad Javad Mansouri[4]

[1] *Department of Petroleum and Chemical Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[2] *Young Researchers and Elite Club, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[3] *Department of Earth, Ocean and Atmospheric Sciences, University of British Columbia, Vancouver, Canada*
[4] *Oil Exploration Operations Company (OEOC), No 234, Taleghani St., Tehran, Iran*

Received April 4, 2019; Accepted July 21, 2020

Abstract

The prediction of a progressive pipeline proceeder is an important factor in an offshore drilling operation. In this research, the effect of two parameters of differential pressure and depth of well in conjunction with seven other parameters related to drilling fluids has been evaluated on the divergence rate of drilling pipes for some offshore operation fields in the Persian Gulf as a case study. The artificial neural networks were used for multilayer and radial base perceptron To identify the complex relationship between those parameters and also pipe trapping. Neural networks, with the imposition of the human mind, are able to identify this complex and obscure relationship, and eventually predict the objective function of the problem (differential divergence of pipes) in other cases. In this work, Particle Swarm Algorithms (PSO), Colonial Competition (ICA), and Genetic Algorithm (GA) will be employed as a training algorithm in both networks. The results then compared with each other for the accuracy of the algorithms. The outcomes of this study describe how each of the algorithms is being applied to investigate the related parameters for piping and fluid flow.

Keywords: *Artificial neural networks; Drilling; Particle swarm algorithm; Colonial competition algorithm; Genetic algorithm.*

1. Introduction

One of the most costly and time-consuming problems in offshore drilling operations is the pipeline prediction. It is an important factor which, needs an appropriate action to avoid some problems such as financial losses.

Investigations on tube trapping began in 1950. In 1985, Kingsboro and Hopkins performed a static analysis of pipe bends based on drilling parameters [1]. This was done by comparing the wells in which the pipes were pulled, and the wells were unencumbered. So that the parameters of each of the two wells were compared and then, according to the non-intercepted wells, were planned for drilling other wells. The two studied 221 parameters in 131 wells in Mexico and surveyed the probability of drilling pipes in the wells surrounding the pipeline. In 1994, Byjler and the priest analyzed this issue by creating a database for 22 drilling parameters in 73 wells without the Gulf of Mexico and 54 wells with a pipe problem [2].

In 1994, Howard, Waggler, using statistical techniques, was able to develop telescope bundle models [3]. This was done by experimenting in 100 wells in the Gulf of Mexico. These models were used to prevent pipe taps or release operations. Recently, by Halliburton Corporation, an application of neural networks for the diffusion of tubes has been published differently in the Gulf of Mexico [1].

The compressive strength mechanism of the drill pipes was expressed by Helmick and Langley in 1957 and by Utmansz in 1958 based on laboratory tests [2]. They explained that the pressure difference gauge occurs when the pipe stops against a high permeability layer. The coating of the drilling bit covers the surface of the pipe, and the pipe is kept by the pressure difference between the walls of the well and the formation. Their experiments also showed that the force required to move a flushed steel pipe would increase with increasing pressure and time difference [4] and [5]. It was costly and time consuming for experiments to simulate. It was also very difficult to meet the actual conditions of the laboratory. On the other hand, in most statistical relationships, there was a need to isolate different types of stuck, eliminating a large number of data due to scattering, and also searching very well for a suitable method [5].

So far, many methods for predicting and detecting drilling have been introduced. In all of the above methods, a reservoir with a particular formation is processed because the parameters of the drilling parameters and variety of mud used in each region are different from the formation [6]. Therefore, each method will only provide the desired response for the same region where the information is obtained from [7]. One of these predictive methods for pipe inflow is the use of an artificial intelligence method, which will be analyzed in this paper using data from one of the oil fields in Iran.

In this paper, the effect of 2 parameters related to drilling naming differential pressure and depth of well and 7 parameters related to drilling fluid known as fluid loss rate, solids content, the viscosity of drilling fluid filters, viscosity plastic, substrate delivery point, initial gel resistance, and gel resistance after 10 minutes were evaluated on the differential rate of drill pipes in 7 fields in the Persian Gulf. The artificial neural networks are used for multilayer and radial base perceptron to identify the complex relationship between these parameters and the trapping of pipes. Neural networks, with the imposition of the human mind, are able to identify this complex and obscure relationship, and eventually predict the objective function of the problem (differential divergence of pipes) in other cases. In this regard, Particle Swarm Algorithms (PSO) and Colonial Competition (ICA) and Genetic Algorithm (GA) will be used as a training algorithm in both networks. To provide data on training and testing of artificial neural network, data from 30 wells in Soroush, Nowrooz, Abu Dhar, Foruzan, Salman, Dena, and Dorood fields were studied in Iran.

In this study, 20% of the available data was tested, and the rest of the data (80%) was allocated to the network education department. The advantage of a neural network is the direct learning of data without the need to estimate their statistical characteristics. The neural network, regardless of any initial hypothesis and previous knowledge of the relationships between the parameters studied, is able to find the relationship between the set of inputs and outputs to predict each output corresponding to the desired input. Such algorithms can be used to study the number of attachments. In one sentence, the tightness of the pipes can be defined as the forces inside the well that prevent the pipe from rotating or exiting the well. One of the main problems in digging oil chains is pipe seals. After grabbing a string in drilling operations, there is a lot of effort to pull it out. One of the well-known methods of drill-down liberation is the upward pull-up and downward weight-increasing method, which are somewhat time-consuming and costly, but in most cases, they release the drilling field from the well. The difficulty of the drilling field becomes more important when it is not possible to remove the drill from the well for some reason. In seaworthy operations, tubing can only increase the cost of a well by 30% [8].

2. Detailed expression of an algorithm solving

The pipe clogging can lead to many problems, such as cutting drilling, residual operations, blocking wells, increasing non-useful life (NPT), and increasing the cost of developing a well. Therefore, it is important to recognize and predict the flow of pipes before occurrence and minimize the economic problems associated with them. In this regard, multi-layer perceptron (MLP) and radial base (RBF) artificial neural networks can be very useful and effective in predicting the objective function of pipe-laying and determining the non-linear relationship

between input parameters. It is possible to accurately predict the pipes to be considered before major problems, and thus avoid proper occurrence by making appropriate decisions using these networks, While the performance of these networks is heavily dependent on the learning algorithm, the choice of the most accurate and precise algorithm is also a topic that is very significant because it can play a significant role in making decisions. Therefore, the selection of the optimal algorithm is also essential and very important. It should be noted that the input values include the properties of rock and fluid, and the output is the differential rate of the tubes. Using the MLP and RBF would need to use some algorithm which is described in follow.

2.1. Optimization algorithms and practices genetic algorithm

Since the genetic algorithm is a random search method, it is difficult to provide a specific formula for its ending. The fitting of the population for a number of generations may have remained constant, suggesting that we have reached the final answer. One common way of ending the algorithm is to stop it after generating a certain number of generations. Because some operators need to know the total number of generations, this seems appropriate. After completion of the algorithm repetition to the number of generations given as input to the algorithm, the quality of the final solutions is considered, which, if the answers are not satisfied, the algorithm continues to a certain number, or from the beginning and By arranging a different initial population and possibly some new settings, it is implemented [10].

2.2. Particle swarm algorithm

For the first time in 1995, by Eberhart and Kennedy, the PSO term was inspired as an indeterminate search method for functional optimization. This algorithm is inspired by the collective movement of birds seeking food [9].

After finding the best values, the speed and location of each particle are updated using equations (1) and (2) [10].

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)] \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

The right side of equation (1) consists of three parts: the first part is the current velocity of the particle and the second and third parts of the change of the velocity of the particle and its rotation towards the best personal experience and the best experience of the group. If the first part is not taken into account in this equation, then the particle velocity is determined only by the current situation and the best experience of the particle and the best-accumulated experience. In this way, the best aggregate particle stays in place, and the others move toward that particle. In fact, the mass motion of particles without the first part of the equation (1) will be a process in which the search space gradually becomes smaller and a local search around the best particle forms. In contrast, if only the first part of equation (1) is taken into account, particles move their normal way to reach the boundary wall and perform a kind of global search [9].

2.3. Colonial competition algorithm

The colonial competition algorithm is a method in the field of evolutionary computing that addresses the optimal answer to various optimization problems. This algorithm provides an algorithm for solving mathematical optimization problems by mathematical modeling of the socio-political evolution process. In terms of application, this algorithm falls into the category of evolutionary algorithms. Like all algorithms in this category, the colonial competition algorithm is also the first set of possible solutions. The initial results of genetic algorithm called "chromosomes", in the particle swarm algorithm called "particle" and the imperialist competitive algorithm, also known as "country" known.

3. Outcome result analysis and discussion

The data on the divergence of tubes from 30 different wells of offshore fields in the Persian Gulf naming Soroush, Nowrooz, Abuzar, Forouzan, Salman, Dena, and Dorood were presented and presented in tabular form in Table 1. Later, using artificial neural network code and genetic

algorithms, colonial competition, and particle swarm, the objective function of the problem is addressed, and the corresponding accuracy is mentioned. Finally, the basic radial network performance was analysed, and the results are presented in the following sections.

Table 1. Input and output data used for differential pressure pipes for the training of artificial neural networks MLP and RBF

Differen-tial pres-sure (psi)	Well depth (m)	Fluffy fluid (cc/min)	Viscosity filter, (cP)	Percentage of solids	Viscose plastic (cP)(	Submission point (lb/100ft^2)	Primary gelatin (lb/100ft^2)	Jelly Power 10 Minutes (lb/100ft^2)	Pipe cap
700	3200	2.8	55	4	20	19	2	11	1
560	2500	1.5	62	13	50	25	7	10	1
230	1840	3.2	71	7	13	11	3	6	0
400	725	4.6	42	8	21	15	5	7	0
940	3325	3.3	81	9	42	16	3	4	1
821	2950	4.1	32	7	54	21	8	15	1
790	4113	2	69	14	49	20	1	11	1
261	1404	4.5	39	6	17	13	5	7	0
231	1021	2.9	57	8	16	24	4	8	0
756	3120	2.4	80	3	61	16	13	3	1
845	3150	3.7	42	4	31	22	10	14	1
250	1700	4.9	35	6	10	26	3	11	0
490	2540	3	50	1	43	27	13	5	1
831	2980	2.1	67	16	51	28	3	13	1
740	4053	1	61	19	52	17	6	8	1
132	941	5.1	37	7	25	13	5	9	0
150	1980	6	52	10	15	12	3	5	0
362	700	5	37	9	10	24	3	6	0
900	3790	1.3	83	3	17	28	1	14	1
766	2342	4.3	70	3	31	30	5	3	1
151	2000	5.4	32	9	12	10	4	5	0
268	1715	6	21	7	25	22	5	5	0
896	3910	4	47	16	52	20	1	17	1
933	3100	2.8	73	9	2	51	13	3	1
459	2934	3	40	9	20	37	15	4	1
685	2220	2.9	52	10	22	28	6	8	1
367	1915	5	36	9	11	12	4	6	0
146	850	2.8	58	7	23	25	4	8	0
582	2450	3	75	17	32	18	9	12	1
812	2302	2	49	15	14	29	11	13	1

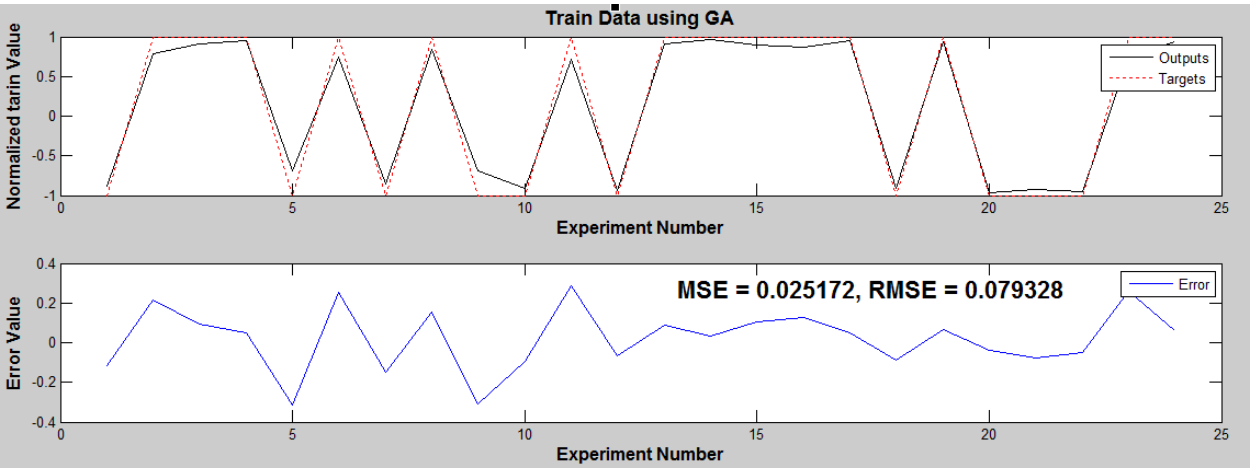

Figure 1. Normalized data on actual data (Target) and network output (Output) of pipe holders for data on Soroush, Nowrooz, Abuzar, Foruzan, Salman, Dena and Dorood fields, and error in each Experiment with GA algorithm

3.1. Analysis of data on the divergent diameters of tubes with ICA algorithm

This section examines the role of an artificial neural network trained by the ICA algorithm on the research results. The number of primary countries, the number of primary colonial countries, the number of decades, and the rate of the revolution were estimated at 100, 10, 20, and 30, respectively. As previously mentioned, in order to predict the differential frequency of tubes in the marinas, the artificial neural network trained with the colonial competition algorithm used data from 30 wells in seven fields in the Persian Gulf as input for consideration were taken. 80% of the data were randomly used as training data, the result of which is shown in Figure 2.

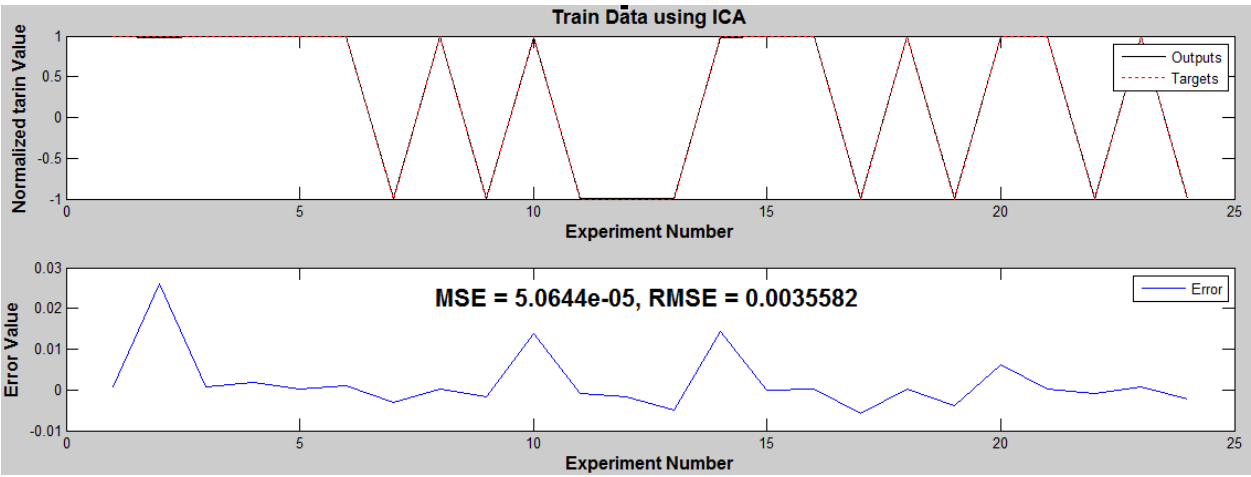

Fig. 2. Normalized data. Target and Output data. Taps for data for Soroush, Nowrooz, Abuzar, Foruzan, Salman, Dena, and Dorood fields and error in each. Experiment with the ICA algorithm

3.2. Analysis of data and results of prediction of radial artificial neural network (RBF)

In this section, we study the performance of the base radial network, which is trained with different training algorithms and is analyzed under different scenarios of network accuracy.

As already mentioned, the radius-based function (RBF) is a function whose value is related to the input distance from a central point. So, the general form of this function can be written as follows [10]:

$$y(x) = \sum_{j=1}^{N} (w_j \varphi_j)(\|x - x_j\|) \quad (3)$$

where: $y(x)$: weighted by an appropriate coefficient; w_i: each associated with a different center; x_j: differentiable with respect to the weight. With open

$$\varphi_r = f(x) = \begin{cases} e^{\left(\frac{1}{1-(\delta r)^2}\right)} & , \ r < \frac{1}{\delta} \\ x & , \ otherwist \end{cases}$$

where j φ is a different inverse function, the most important of which is the Gaussian function.

As defined by the definition of the rbf function, the distance between all the data with the specified centers (x0) is an issue that matters in this type of network. Additionally, the correct selection of weights (w_j), as well as kernels (centers), is one of the most important issues in these networks.

In this section, we discuss the role of the trained artificial neural network with the PSO algorithm trained by the particle swarm algorithm. To do this, the newrb command is used in MATLAB software that generates the network as follows:

Network = newrb (P, T, Goal, Spread, MN, DF)

In which P represents the input data vectors (pipe holders for wells in Soroush, Nowrooz, Abuzar, Foruzan, Salman, Dena and Dorood fields), T represents the output data obtained from the fields (snug or non-stop pipes), GOAL represents the theoretical error that specifies the end of the grid, Spread represents the amount of dispersion of the rbf functions, which indicates more dispersion numbers, MN represents the maximum number of nerve for basic radial network training, which is at most equal to the number Problem inputs (9 inputs) and DF represent the number of repetitions in which the error network is displayed.

3.3. Analyzing the data on the trapping rate with the PSO algorithm

The results of the basic radial network results are discussed in follow by using the particle swarm algorithm. As previously mentioned, in order to predict the flow rate of the pipes by an artificial neural network trained by the particle swarm algorithm, The wells of Soroush, Nowruz, Abuzar, Foruzan, Salman, Dena, and Dorood fields were considered as inputs.

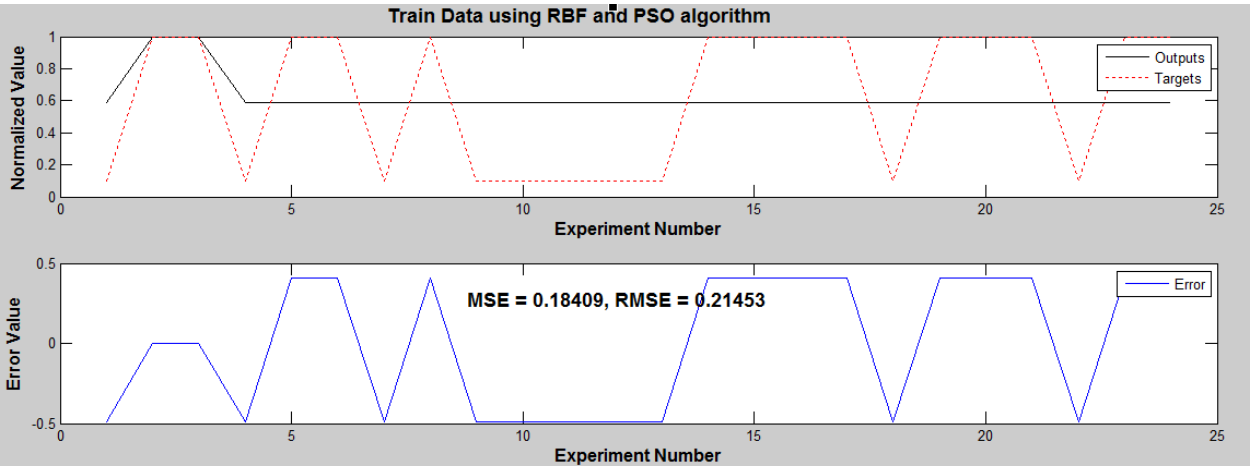

Fig. 3. Normalized data. Target and Output data for pipe data rate for Soroush, Nowrooz, Abuzar, Foruzan, Salman, Dena and Dorood fields, and error in each Experiment with PSO algorithm and RBF network

3.4. Analyzing the data with the GA algorithm

As in the previous state, the condition for the termination of the network was selected using a genetic algorithm, reaching a certain number of courses (number of generations equivalent to 50 courses). In this way, 20% of the rounded input data is considered as a test for the trained network with the GA algorithm, and the rest of the data is assigned randomly to the network education section. Therefore, for the basic radial network, the GA algorithm was attempted to improve the network structure and, finally, a network that was selected to have the lowest RMSE associated with the test data. The parameters of the survey and optimization

in this network were spread (kernel functions dispersion) and the number of neurons needed, which eventually reached 90 and 4, respectively. Optimization parameters were selected using the genetic algorithm, such as the table for the mlp network.

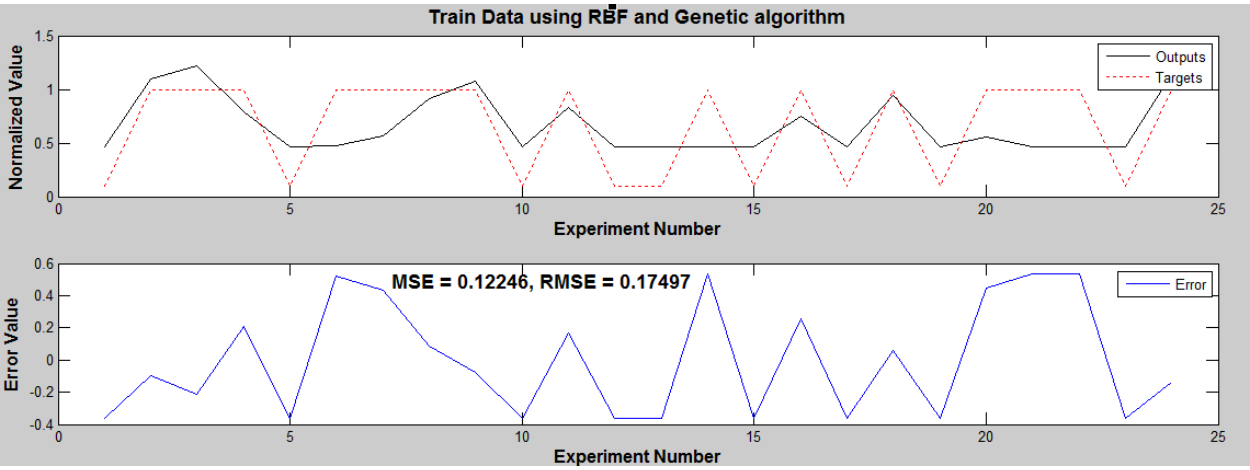

Fig. 4. Normalized data. Target and Output data of pipe holders for Soroush, Nowrooz, Abuzar, Foruzan, Salman, Dena and Dorood fields and error in each Experiment with GA algorithm and RBF network

4. Results

The results obtained from artificial neural networks belonging to artificial neural networks of the multilayer and radial base perceptron showed that it should be the basis for decision making based on network testing and validation data, not training data, since data from the Overtrained Education section Have been.

Undoubtedly, one of the important factors in the accurate prediction of MLP artificial neural networks was the use of randomly based algorithms. In this case, the possibility of trapping the network in the local minima will be reduced, and the results will be better targeted to the actual values.

The accuracy of about 100% obtained from the PSO algorithm indicates the reliability of the neural network in the performances of this algorithm. In other words, the use of the simulator in other scenarios is unnecessary and can be used instead of networks.

Particle swarm algorithms were more accurately able to predict pipeline data in offshore wells than colonial and genetic competition algorithms so that the RMSE error of this algorithm was 0/001 for test data. While this amount was 0/009 and 0/004 for genetic algorithms and colonial competition, respectively. Therefore, the function of the network under the use of genetics is much weaker than the colonial competition algorithm.

Since the number of generations and the initial population for all algorithms are a key parameter in optimizing the structure and accuracy of the network, in this thesis, the initial population (particles for PSO and countries for ICA) all algorithms are equal and equal to 100 was considered.

References

[1] Bataee M, and Mohseni S. January. Application of Artificial Intelligent Systems in ROP Optimization: a Case Study. In SPE Middle East Unconventional Gas Conference and Exhibition 2011, SPE-140029-MS.

[2] Beach WH. Kennametal Inc, 1988. Erosion resistant cutting bit with hardfacing. U.S. Patent 4,725,098.

[3] Atashpaz-Gargari E, and Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In Evolutionary computation, 2007. CEC 2007. IEEE Congress on (pp. 4661-4667), DOI:10.1109/CEC.2007.4425083.

[4] Becker WM, Kleinsmith LJ, Hardin J, and Raasch J. 2003. The world of the cell (Vol. 6). San Francisco: Benjamin Cummings, ISBN-13: 978-0805393934.

[5] Bilgesu HI, Altmis U, Ameri S, Mohaghegh S, and Aminian K. A new approach to predict bit life based on tooth or bearing failures. In SPE Eastern regional meeting 1998. SPE-51082-MS.
[6] Shahrekordi AP, Behnood M, Hosseinian A, Mozaffari S, Sheikhzakariaee SJ. A Sensitivity Analysis for the Effective Parameters Disparity on Pressure and Pressure Derivative of Well-Testing in A Horizontal Well. Petroleum & Coal, 2019; 61(4): 749-756.
[7] Bilgesu HI, Tetrick LT, Altmis U, Mohaghegh S, and Ameri S. A new approach for the prediction of rate of penetration (ROP) values. In SPE Eastern Regional Meeting. Society of Petroleum Engineers 1997.
[8] Arehart RA. Drill Bit Diagnosis Using Neural Network. Proc. the SPE Annual Technical Conference and Exhibition 1989, SPE-19558-PA.
[9] Desai NJ, and Sue JA. ReedHycalog LP, 2007. Fixed cutter drill bit with non-cutting erosion resistant inserts. U.S. Patent 7,237,628.
[10] Theodoridis S, Koutroumbas K. Pattern Recognition (Fourth Edition), Academic Press 2008, ISBN: 9781597492720.

***Attachments**:*

***Artificial neural network code**:*

```
%% Start of Program
clc
clear
close all
%% Data Loading
Data = xlsread('Mansoori.xlsx');

X = Data(:,1:end-1);
Y = Data(:,end);

DataNum = size(X,1);
InputNum = size(X,2);
OutputNum = size(Y,2);

%% Normalization
MinX = min(X);
MaxX = max(X);

MinY = min(Y);
MaxY = max(Y);

XN = X;
YN = Y;
for ii = 1:InputNum
   XN(:,ii) = Normalize_Fcn(X(:,ii),MinX(ii),MaxX(ii));
end

for ii = 1:OutputNum
   YN(:,ii) = Normalize_Fcn(Y(:,ii),MinY(ii),MaxY(ii));
end

%% Test and Train Data
TrPercent = 80;
TrNum = round(DataNum * TrPercent / 100);
TsNum = DataNum - TrNum;

R = randperm(DataNum);
trIndex = R(1 : TrNum);
tsIndex = R(1+TrNum : end);
```

```
Xtr = XN(trIndex,:);
Ytr = YN(trIndex,:);

Xts = XN(tsIndex,:);
Yts = YN(tsIndex,:);

%% Network Structure
pr = [-1 1];
PR = repmat(pr,InputNum,1);
MSEtsmin=100;
kkmin=100;
kkkmin=100;

for kk=6:6;
for kkk=6:6;
Network = newff(PR,[kk kkk OutputNum],{'tansig' 'tansig' 'tansig'});

%% Training
Network = TrainUsing_PSO_Fcn(Network,Xtr,Ytr);

%% Assesment
YtrNet = sim(Network,Xtr')';
YtsNet = sim(Network,Xts')';


MSEtr = mse(YtrNet - Ytr)
MSEts = mse(YtsNet - Yts)
if MSEts < MSEtsmin
   MSEtsmin=MSEts;
   kkmin=kk;
   kkkmin=kkk;
end

end
end
MSEts = MSEtsmin
kk = kkmin
kkk = kkkmin
RMSEts=sqrt(MSEts)/2*100

trainPerformance = perform(Network,Ytr,YtrNet)
testPerformance = perform(Network,Yts,YtsNet)
errors = gsubtract(Yts,YtsNet);

%% Display

figure(1)
plot(Ytr,'r-')
hold on
plot(YtrNet,'bo')
legend('Targets','Outputs')
    title('Real train data Vs Network train data' )


hold off

figure(2)
plot(Yts,'-or')
hold on
plot(YtsNet,'-sb')
```

```
legend('Targets','Outputs');
   title('Real test data Vs Network test data' );
hold off

figure(3)
t = -1:.1:1;
plot(t,t,'b','linewidth',2)
hold on
plot(Ytr,YtrNet,'ok')
legend('Targets','Outputs');
   title('YtrNet Vs Ytr');
hold off

figure(4)
t = -1:.1:1;
plot(t,t,'b','linewidth',2)
hold on
plot(Yts,YtsNet,'ok')
legend('Targets','Outputs');
   title('Yts Vs YtsNet' );
hold off


plot(Yts-YtsNet, 'linewidth',3)
plotfit(Network,Ytr,YtrNet)
plotfit(Network,Yts,YtsNet)
figure, plotregression(Yts,YtsNet)
figure, plotregression(Ytr,YtrNet)
figure, ploterrhist(errors)

PlotResults(Ytr,YtrNet,'Train Data');

PlotResults(Yts,YtsNet,'TestData');
```

Code of particle swarm algorithm:

```
%% Problem Statement
IW = Network.IW{1,1}; IW_Num = numel(IW);
LW1 = Network.LW{2,1}; LW1_Num = numel(LW1);
LW2 = Network.LW{3,2}; LW2_Num = numel(LW2);
b1 = Network.b{1,1}; b1_Num = numel(b1);
b2 = Network.b{2,1}; b2_Num = numel(b2);
b3 = Network.b{3,1}; b3_Num = numel(b3);
TotalNum = IW_Num + LW1_Num + LW2_Num+  b1_Num + b2_Num + b3_Num;

NPar = TotalNum;

VarMin = -1*ones(1,TotalNum);
VarMax = +1*ones(1,TotalNum);

CostFuncName = 'Cost_ANN_EA';

%% Algorithm's Parameters
SwarmSize = 100;
MaxIteration = 70;
C1 = 2; % Cognition Coefficient;
C2 = 4 - C1; % Social Coefficient;
%% Initial Population
GBest.Cost = inf;
```

```
GBest.Position = [];
GBest.CostMAT = [];
for p = 1:SwarmSize
    Particle(p).Position = rand(1,NPar) .* (VarMax - VarMin) + VarMin;
    Particle(p).Cost = feval(CostFuncName,Particle(p).Position,Xtr,Ytr,Network);
    Particle(p).Velocity = [];
    Particle(p).LBest.Position = Particle(p).Position;
    Particle(p).LBest.Cost = Particle(p).Cost;

    if Particle(p).LBest.Cost < GBest.Cost
        GBest.Cost = Particle(p).LBest.Cost;
        GBest.Position = Particle(p).LBest.Position;
    end
end

%% Start of Optimization
for Iter = 1:MaxIteration
    %% Velocity update
    for p = 1:SwarmSize
        Particle(p).Velocity = C1 * rand * (Particle(p).LBest.Position - Particle(p).Position) + C2 * rand *
(GBest.Position - Particle(p).Position);
        Particle(p).Position = Particle(p).Position + Particle(p).Velocity;

        Particle(p).Position = max(Particle(p).Position , VarMin);
        Particle(p).Position = min(Particle(p).Position , VarMax);

        Particle(p).Cost = feval(CostFuncName,Particle(p).Position,Xtr,Ytr,Network);

        if Particle(p).Cost < Particle(p).LBest.Cost
            Particle(p).LBest.Position = Particle(p).Position;
            Particle(p).LBest.Cost = Particle(p).Cost;

            if Particle(p).LBest.Cost < GBest.Cost
                GBest.Cost = Particle(p).LBest.Cost;
                GBest.Position = Particle(p).LBest.Position;
            end
        end
    end
    %% Display
    disp(['Itretion = ' num2str(Iter) '; Best Cost = ' num2str(GBest.Cost) ';']);
    GBest.CostMAT = [GBest.CostMAT GBest.Cost];
end

GBest.Position
plot(GBest.CostMAT)
Network2 = ConsNet_Fcn(Network,GBest.Position);
BestCost = GBest.Cost;
end
```

***Code of colonial competition algorithm**:*

```
function [Network2 BestCost] = TrainUsing_ICA_Fcn(Network,Xtr,Ytr)

%% Problem Statement
IW = Network.IW{1,1}; IW_Num = numel(IW);
LW1 = Network.LW{2,1}; LW1_Num = numel(LW1);
LW2 = Network.LW{3,2}; LW2_Num = numel(LW2);
b1 = Network.b{1,1}; b1_Num = numel(b1);
b2 = Network.b{2,1}; b2_Num = numel(b2);
b3 = Network.b{3,1}; b3_Num = numel(b3);
```

```
TotalNum = IW_Num + LW1_Num + LW2_Num+  b1_Num + b2_Num + b3_Num;

CostFuncExtraParams.Xtr = Xtr;
CostFuncExtraParams.Ytr = Ytr;
CostFuncExtraParams.Network = Network;

ProblemParams.CostFuncName = 'Cost_ANN_EA';    % You should state the name of your cost function
here.
ProblemParams.CostFuncExtraParams = CostFuncExtraParams;              % Reserved for the extra pa-
rameters in cost function. In normal application do not use it that is use [].
ProblemParams.NPar = TotalNum;                    % Number of optimization variables of your objec-
tive function. "NPar" is the dimention of the optimization problem.
ProblemParams.VarMin = [-1] ;                    % Lower limit of the optimization parameters. You
can state the limit in two ways. 1)   2)
ProblemParams.VarMax = [1];                   % Lower limit of the optimization parameters. You can
state the limit in two ways. 1)   2)

% Modifying the size of VarMin and VarMax to have a general form
if numel(ProblemParams.VarMin)==1
   ProblemParams.VarMin=repmat(ProblemParams.VarMin,1,ProblemParams.NPar);
   ProblemParams.VarMax=repmat(ProblemParams.VarMax,1,ProblemParams.NPar);
end

ProblemParams.SearchSpaceSize = ProblemParams.VarMax - ProblemParams.VarMin;

%% Algorithmic Parameter Setting
AlgorithmParams.NumOfCountries = 100;              % Number of initial countries.
AlgorithmParams.NumOfInitialImperialists = 40;     % Number of Initial Imperialists.
AlgorithmParams.NumOfAllColonies = AlgorithmParams.NumOfCountries - AlgorithmParams.NumOfIni-
tialImperialists;
AlgorithmParams.NumOfDecades = 80;
AlgorithmParams.RevolutionRate = 0.3;              % Revolution is the process in which the socio-politi-
cal characteristics of a country change suddenly.
AlgorithmParams.AssimilationCoefficient = 2;       % In the original paper assimilation coefficient is
shown by "beta".
AlgorithmParams.AssimilationAngleCoefficient = .5;  % In the original paper assimilation angle coeffi-
cient is shown by "gama".
AlgorithmParams.Zeta = 0.02;                    % Total Cost of Empire = Cost of Imperialist + Zeta *
mean(Cost of All Colonies);
AlgorithmParams.DampRatio = 0.99;
AlgorithmParams.StopIfJustOneEmpire = false;       % Use "true" to stop the algorithm when just one
empire is remaining. Use "false" to continue the algorithm.
AlgorithmParams.UnitingThreshold = 0.02;           % The percent of Search Space Size, which enables
the uniting process of two Empires.

%% Display Setting
DisplayParams.PlotEmpires = false;    % "true" to plot. "false" to cancel ploting.
if DisplayParams.PlotEmpires
   DisplayParams.EmpiresFigureHandle = figure('Name','Plot of Empires','NumberTitle','off');
   DisplayParams.EmpiresAxisHandle = axes;
end

DisplayParams.PlotCost = false;    % "true" to plot. "false"
if DisplayParams.PlotCost
   DisplayParams.CostFigureHandle = figure('Name','Plot of Minimum and Mean Costs','Num-
berTitle','off');
   DisplayParams.CostAxisHandle = axes;
end

ColorMatrix = [1   0   0  ; 0 1   0    ; 0   0 1    ; 1   1   0  ; 1   0 1    ; 0 1   1    ; 1 1 1      ;
```

```
            0.5 0.5 0.5; 0 0.5 0.5  ; 0.5 0 0.5  ; 0.5 0.5 0  ; 0.5 0 0    ; 0 0.5 0    ; 0 0 0.5    ;
            1   0.5 1  ; 0.1*[1 1 1]; 0.2*[1 1 1]; 0.3*[1 1 1]; 0.4*[1 1 1]; 0.5*[1 1 1]; 0.6*[1 1 1]];
DisplayParams.ColorMatrix = [ColorMatrix ; sqrt(ColorMatrix)];

DisplayParams.AxisMargin.Min = ProblemParams.VarMin;
DisplayParams.AxisMargin.Max = ProblemParams.VarMax;

%% Creation of Initial Empires
InitialCountries = GenerateNewCountry(AlgorithmParams.NumOfCountries , ProblemParams);

% Calculates the cost of each country. The less the cost is, the more is the power.
if isempty(ProblemParams.CostFuncExtraParams)
   InitialCost = feval(ProblemParams.CostFuncName,InitialCountries);
else
   InitialCost = feval(ProblemParams.CostFuncName,InitialCountries,ProblemParams.CostFuncExtraPa-
rams);
end
[InitialCost,SortInd] = sort(InitialCost);                % Sort the cost in assending order. The best
countries will be in higher places
InitialCountries = InitialCountries(SortInd,:);             % Sort the population with respect to their
cost.

Empires = CreateInitialEmpires(InitialCountries,InitialCost,AlgorithmParams, ProblemParams);

%% Main Loop
MinimumCost = repmat(nan,AlgorithmParams.NumOfDecades,1);
MeanCost = repmat(nan,AlgorithmParams.NumOfDecades,1);

if DisplayParams.PlotCost
   axes(DisplayParams.CostAxisHandle);
   if any(findall(0)==DisplayParams.CostFigureHandle)
      h_MinCostPlot=plot(MinimumCost,'r','LineWidth',1.5,'YDataSource','MinimumCost');
      hold on;
      h_MeanCostPlot=plot(MeanCost,'k:','LineWidth',1.5,'YDataSource','MeanCost');
      hold off;
      pause(0.05);
   end
end

for Decade = 1:AlgorithmParams.NumOfDecades
   AlgorithmParams.RevolutionRate = AlgorithmParams.DampRatio * AlgorithmParams.Revolution-
Rate;

   Remained = AlgorithmParams.NumOfDecades - Decade
   for ii = 1:numel(Empires)
      %% Assimilation;  Movement of Colonies Toward Imperialists (Assimilation Policy)
      Empires(ii) = AssimilateColonies(Empires(ii),AlgorithmParams,ProblemParams);

      %% Revolution;  A Sudden Change in the Socio-Political Characteristics
      Empires(ii) = RevolveColonies(Empires(ii),AlgorithmParams,ProblemParams);

      %% New Cost Evaluation
      if isempty(ProblemParams.CostFuncExtraParams)
         Empires(ii).ColoniesCost = feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosition);
      else
         Empires(ii).ColoniesCost = feval(ProblemParams.CostFuncName,Empires(ii).ColoniesPosi-
tion,ProblemParams.CostFuncExtraParams);
      end

      %% Empire Possession
```

```
        Empires(ii) = PossesEmpire(Empires(ii));

        %% Computation of Total Cost for Empires
        Empires(ii).TotalCost = Empires(ii).ImperialistCost + AlgorithmParams.Zeta * mean(Em-
pires(ii).ColoniesCost);

    end

    %% Uniting Similiar Empires
    Empires = UniteSimilarEmpires(Empires,AlgorithmParams,ProblemParams);

    %% Imperialistic Competition
    Empires = ImperialisticCompetition(Empires);

    if numel(Empires) == 1 && AlgorithmParams.StopIfJustOneEmpire
        break
    end

    %% Displaying the Results
    DisplayEmpires(Empires,AlgorithmParams,ProblemParams,DisplayParams);

    ImerialistCosts = [Empires.ImperialistCost];
    MinimumCost(Decade) = min(ImerialistCosts);
    MeanCost(Decade) = mean(ImerialistCosts);

    if DisplayParams.PlotCost
        refreshdata(h_MinCostPlot);
        refreshdata(h_MeanCostPlot);
        drawnow;
        pause(0.01);
    end

end % End of Algorithm
BestCost = MinimumCost(end)
BestIndex = find(ImerialistCosts == min(ImerialistCosts)); BestIndex = BestIndex(1);
BestSolution = Empires(BestIndex).ImperialistPosition;
Network2 = ConsNet_Fcn(Network,BestSolution);
```

Genetic Algorithm Code

```
function [Network2] = TrainUsing_GA_Fcn(Network,Xtr,Ytr)

%% Problem Statement
IW = Network.IW{1,1}; IW_Num = numel(IW);
LW = Network.LW{2,1}; LW_Num = numel(LW);
b1 = Network.b{1,1}; b1_Num = numel(b1);
b2 = Network.b{2,1}; b2_Num = numel(b2);

TotalNum = IW_Num + LW_Num + b1_Num + b2_Num;

NPar = TotalNum;

VarLow = -1;
VarHigh = 1;
FunName = 'Cost_ANN_EA';

%% Algorithm Parameters
SelectionMode = 3; % 1 for Random, 2 for Tournment, 3 for ....
PopSize = 20;
MaxGenerations = 10;
```

```
RecomPercent = 15/100;
CrossPercent = 50/100;
MutatPercent = 1 - RecomPercent - CrossPercent;

RecomNum = round(PopSize*RecomPercent);
CrossNum = round(PopSize*CrossPercent);
if mod(CrossNum,2)~=0
   CrossNum = CrossNum - 1;
end

MutatNum = PopSize - RecomNum - CrossNum;

%% Initial Population
Pop = rand(PopSize,NPar) * (VarHigh - VarLow) + VarLow;

Cost = feval(FunName,Pop,Xtr,Ytr,Network);
[Cost Inx] = sort(Cost);
Pop = Pop(Inx,:);

%% Main Loop
MinCostMat = [];
MeanCostMat = [];

for Iter = 1:MaxGenerations
   %% Recombination
   RecomPop = Pop(1:RecomNum,:);

   %% CrossOver
      %% Parent Selection
      SelectedParentsIndex = MySelection_Fcn(Cost,CrossNum,SelectionMode);

      %% Cross Over
      CrossPop = [];
      for ii = 1:2:CrossNum
         Par1Inx = SelectedParentsIndex(ii);
         Par2Inx = SelectedParentsIndex(ii+1);

         Parent1 = Pop(Par1Inx,:);
         Parent2 = Pop(Par2Inx,:);


         [Off1 , Off2] = MyCrossOver_Fcn(Parent1,Parent2);

         CrossPop = [CrossPop ; Off1 ; Off2];
      end
   %% Mutation
   MutatPop = rand(MutatNum,NPar)*(VarHigh - VarLow) + VarLow;

   %% New Population
   Pop = [RecomPop ; CrossPop ; MutatPop];
   Cost = feval(FunName,Pop,Xtr,Ytr,Network);
   [Cost Inx] = sort(Cost);
   Pop = Pop(Inx,:);

   %% Display
   MinCostMat = [MinCostMat ; min(Cost)];
   [Iter MinCostMat(end)]
   MeanCostMat = [MeanCostMat ; mean(Cost)];
   subplot(2,1,1)
```

```
    plot(MinCostMat,'r','linewidth',2.5);
    xlim([1 MaxGenerations])
%     hold on
%     plot(MeanCostMat,':b','linewidth',2)
%     hold off

    subplot(2,1,2)
    plot(Pop(:,1),Pop(:,2),'rp')
    axis([VarLow VarHigh VarLow VarHigh])
    pause(0.05)

end
%% Final Result Demonstration
BestSolution = Pop(1,:);
BestCost = Cost(1);
Network2 = ConsNet_Fcn(Network,BestSolution);
```

Basic artificial radial neural network code trained with GA

```
clc;close all;clear all
%% GA
Var_n=2;                    %Spread, MaxNeuron, DisplayAt
Range=[0.5,100;1,9];    %% rang of search
PopuSize=30;generation_n =10;
[Input]=MyGA(Var_n,Range,PopuSize,generation_n);

%% Training the best RBF network
Goal=0.001;
Spread=Input(1);     %%round
MaxNeuron=round(Input(2));  %%round
DisplayAt=1;  %%round

Data = xlsread('DPSTICK1.xlsx');

x = Data(:,1:end-1);
y = Data(:,end);

inputs = x';
targets = y';

nData=size(inputs,2);

Perm=randperm(nData);

pTrainData=0.8;
nTrainData=round(pTrainData*nData);
trainInd=Perm(1:nTrainData);
Perm(1:nTrainData)=[];
trainInputs = inputs(:,trainInd);
trainTargets = targets(:,trainInd);

pTestData=1-pTrainData;
nTestData=nData-nTrainData;
testInd=Perm;
testInputs = inputs(:,testInd);
testTargets = targets(:,testInd);

% Create and Train RBF Network
net = newrb(trainInputs,trainTargets,Goal,Spread,MaxNeuron,DisplayAt);
```

```
% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs);

% Recalculate Training, Validation and Test Performance
trainOutputs = outputs(:,trainInd);
trainErrors = trainTargets-trainOutputs;
trainPerformance = perform(net,trainTargets,trainOutputs);

testOutputs = outputs(:,testInd);
testError = testTargets-testOutputs;
testPerformance = perform(net,testTargets,testOutputs);

 PlotResults(targets,outputs,'All Data');
 PlotResults(trainTargets,trainOutputs,'Train Data');
 PlotResults(testTargets,testOutputs,'Test Data');
```

Basic artificial neural network code trained with pso

```
clc;close all;clear all
%% pso
NPar = 2;
VarMin = [0.5 1];
VarMax = [100 9];
[Input]=PSO_Count_General(NPar,VarMin,VarMax);

%% Training the best RBF network
Goal=0.001;
Spread=Input(1);     %%round
MaxNeuron=round(Input(2));  %%round
DisplayAt=1;  %%round

Data = xlsread('DPSTICK1.xlsx');

x = Data(:,1:end-1);
y = Data(:,end);

inputs = x';
targets = y';

nData=size(inputs,2);

Perm=randperm(nData);

pTrainData=0.8;
nTrainData=round(pTrainData*nData);
trainInd=Perm(1:nTrainData);
Perm(1:nTrainData)=[];
trainInputs = inputs(:,trainInd);
trainTargets = targets(:,trainInd);

pTestData=1-pTrainData;
nTestData=nData-nTrainData;
testInd=Perm;
testInputs = inputs(:,testInd);
testTargets = targets(:,testInd);

% Create and Train RBF Network
net = newrb(trainInputs,trainTargets,Goal,Spread,MaxNeuron,DisplayAt);
```

```
% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs);

% Recalculate Training, Validation and Test Performance
trainOutputs = outputs(:,trainInd);
trainErrors = trainTargets-trainOutputs;
trainPerformance = perform(net,trainTargets,trainOutputs);

testOutputs = outputs(:,testInd);
testError = testTargets-testOutputs;
testPerformance = perform(net,testTargets,testOutputs);

 PlotResults(targets,outputs,'All Data');
 PlotResults(trainTargets,trainOutputs,'Train Data');
 PlotResults(testTargets,testOutputs,'Test Data');
```

Artificial neural network radial base code trained with ICA

```
clc;
clear;
close all;

Data = xlsread('DPSTICK.xlsx');

x = Data(:,1:end-1);
y = Data(:,end);

inputs = x';
targets = y';

nData=size(inputs,2);

Perm=randperm(nData);

pTrainData=0.7;
nTrainData=round(pTrainData*nData);
trainInd=Perm(1:nTrainData);
Perm(1:nTrainData)=[];
trainInputs = inputs(:,trainInd);
trainTargets = targets(:,trainInd);

pTestData=1-pTrainData;
nTestData=nData-nTrainData;
testInd=Perm;
testInputs = inputs(:,testInd);
testTargets = targets(:,testInd);

% Create and Train RBF Network
Goal=0;
Spread=1;
MaxNeuron=100;
DisplayAt=1;
net = newrb(trainInputs,trainTargets,Goal,Spread,MaxNeuron,DisplayAt);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
```

```
performance = perform(net,targets,outputs);

% Recalculate Training, Validation and Test Performance
trainOutputs = outputs(:,trainInd);
trainErrors = trainTargets-trainOutputs;
trainPerformance = perform(net,trainTargets,trainOutputs);

testOutputs = outputs(:,testInd);
testError = testTargets-testOutputs;
testPerformance = perform(net,testTargets,testOutputs);

PlotResults(targets,outputs,'All Data');
PlotResults(trainTargets,trainOutputs,'Train Data');
PlotResults(testTargets,testOutputs,'Test Data');

% View the Network
% view(net);
```

To whom correspondence should be addressed: Dr. Sajjad Mozaffari, Department of Petroleum and Chemical Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran,
E-mail: Sajad.Mozafari@yahoo.com