Using Artificial Neural Nets To Identify the Well-Test Interpretation Model

Abdul-Aziz U. Al-Kaabi, SPE, and **W. John Lee,** SPE, Texas A&M U.

Summary. This paper presents a new approach, based on artificial neural net technology, to identify a preliminary well-test interpretation model from derivative plot data. Artificial neural nets can identify patterns from incomplete and distorted data and also eliminate the need for elaborate data preparation, such as smoothing.

Introduction

In a pressure-transient test, a signal of pressure vs. time is recorded. When this signal is plotted with specialized plotting functions, diagnostic plots, such as derivative[1] or Horner plots, are produced that often are used in the interpretation process. The signal on these plots is deformed and shaped by underlying mechanisms in the formation and wellbore. These mechanisms are the well-test interpretation model. The objective of this work is to identify these mechanisms from the signatures on the derivative plot.

Identifying the well-test interpretation model is described in the literature as an inverse problem.[2] The traditional way of solving an inverse problem is with inverse theory[3] techniques (e.g., regression analysis). A serious disadvantage of such techniques is that we have to assume an interpretation model. The inverse theory provides estimates of the model parameters but not of the model itself. Because more than one interpretation model can produce the same signal, this approach can lead to misleading results. We seek the model itself instead of its parameters in this study.

In this study, we trained a neural net simulator to identify the well-test interpretation model from the derivative plot. The neural net simulator can be part of a well-test expert system or a computer-enhanced well-test interpretation.

Literature Review

In 1988, Allain and Horne[4] used syntactic pattern recognition[5] and a rule-based approach to identify the well-test interpretation model automatically from the derivative plot. Their approach is based on transforming the derivative plot into a symbolic form. The symbols generated (UP, DOWN, etc.) are used by a rule-based system to identify the well-test interpretation model.

In 1989, Stewart and Du[6] presented a technique to transform the derivative plot into a symbolic form for use in a knowledge-based system to identify the well-test interpretation model. This approach relies on preprocessing test data to obtain a smooth derivative plot. The data are filtered with a preselected smooth parameter incorporated in approximating spline functions. Then, the smoothed derivative plot is transformed into symbols by first approximating the curve with linear and nonlinear segments based on preselected error thresholds and measurements of inflection points. The resulting segments then are used to obtain a detailed symbolic description of the derivative plot.

In this paper, we present a new method to identify a preliminary interpretation model. Our approach eliminates the need for preprocessing and writing complex rules. The approach is based on artificial neural net technology and is fully automatic because it does not depend on threshold choice and difficult-to-measure parameters, such as inflection points. The disadvantage of this procedure is that it is useful for model selection only; the analyst must estimate the model parameters using methods totally independent of the neural net. An advantage of the symbolic approach is that it can be used to estimate model parameters. Thus, each approach has strengths and limitations.

Artificial Neural Net Approach

The dominant artificial intelligence paradigms (e.g., rule-based systems) assume a symbolic approach to describe intelligence. Such paradigms assume that such means as rules or algorithms are available that can be encoded into a computer to solve a problem. The symbolic approach is applied successfully when the problem domain model (i.e., the solution) can be mapped into a procedure. The fundamental limitation of this approach is that it cannot be used to solve problems for which a solution procedure does not exist or cannot be found. The visual recognition problem is one that cannot be solved by an exact procedure. We normally solve a vision problem without thinking of a formal procedure to carry out the recognition task. This limitation in the symbolic approach is minimized with artificial neural nets.[7-11] The neural net approach does not assume a predefined algorithm to solve a problem; instead, it learns the solution model automatically by training on examples and their expected outputs.

Artificial neural nets are computing systems based on the belief that intelligence is achieved through interaction of large numbers of simple processing units (nodes),[8] a belief based on the current understanding of brain anatomy. The brain consists of billions of small processing units called neurons. A group of highly connected neurons specializes in solving certain cognitive tasks. Experts think that learning occurs through constructing internal representations of concepts by strengthening or weakening connections between neurons. Similarly, artificial neural nets learn tasks by changing the strength of links between nodes.

Artificial Neural Net System Architecture. Fig. 1 is a schematic of a multilayer artificial neural net. It consists of nodes arranged in layers. The first layer is an input layer, the second is a middle or intermediate layer, and the last layer is an output layer. The input in each node, except for the nodes in the output layer, is propagated throughout the entire net by connections called links. Each link has a weight with either a positive (strengthening) or a negative (weakening) value. The activation level in the intermediate and output layer nodes is measured by a threshold function usually called a squashing or activation function. The threshold function approaches the limiting value 1 (i.e., the node is very active) at a very large positive argument and approaches the limiting value 0 (i.e., the node is inactive) at a very small negative argument. This function is continuous and differentiable everywhere. Any function that has such characteristics can be used as a threshold function. The sigmoidal function **(Fig. 2)** frequently is used as a threshold function and has the form

$$O_j = 1/[1+\exp(-I_j)], \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where O_j = the output from a node in the j layer (Fig. 1) and I_j = the sum of the input signals to this node:

$$I_j = \sum_i w_{ji} O_i. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

This summarizes the calculation that occurs in a single node.

Learning is the most important computationally and memory-intensive task in a neural net operation. In a mathematical sense, learning is the process by which we can find a set of weights that produces the expected output when a net is presented within an input. In the learning phase, the net is given a set of examples (usually called a training set) where each training set consists of an input vector and a desired output vector. In this study, we used back-

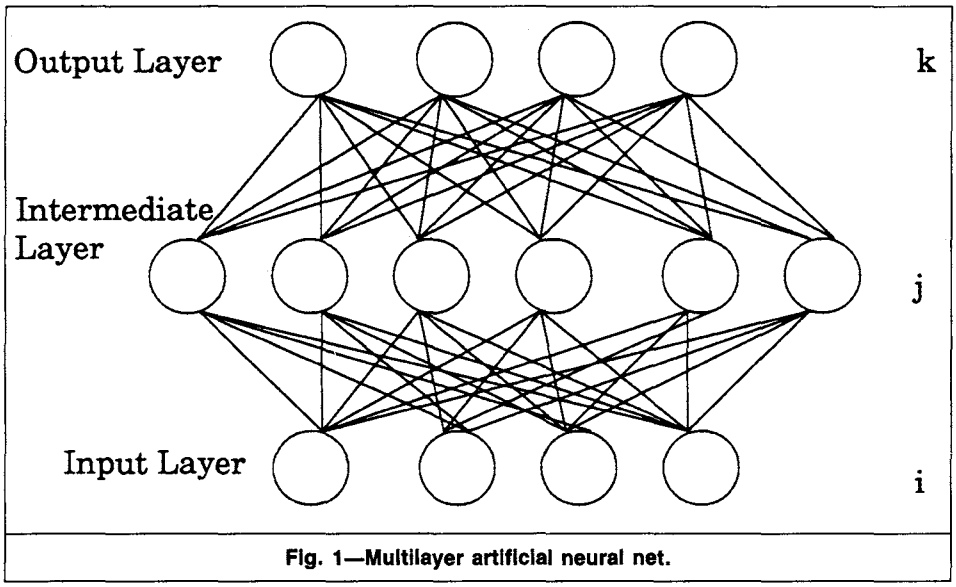


Fig. 1—Multilayer artificial neural net.

propagation as the learning algorithm. The key idea in this algorithm is that the error (difference between computed and actual output) at the output layer is propagated backward through the net with credit-assignment strategy. This means that the weight in each link is changed by an amount proportional to the error it brings to the output layer. As the error is propagated backward through the links between layers, weight changes are added or subtracted on the basis of the sign of the sigmoidal function derivative as in a gradient-descent method:

$$\Delta w_{kj} = \eta(t_k - O_k) f_k(I_k) O_j. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The left side of Eq. 3 is the weight change within a link between Layers k and j. Note that,on the right side of Eq. 3, the weight change is based on the output from a node in a previous layer (the node that caused the error). Eq. 3 is used to calculate weights in links pointing to the output layer only. Another form of Eq. 3 is used to calculate internal link weights.

Appendices A and B show the mathematical derivation of and programming steps for the backpropagation learning algorithm, respectively. In Appendix C, we discuss some practical considerations used during training. **Fig. 3** is a schematic of the learning cycle in an artificial neural net.

Well-Test Interpretation Model Identification by Artificial Neural Net. Because of the nature of neural nets, segmentation and symbolic transformation are eliminated. The net examines the whole curve simultaneously and identifies models that could have caused the signal presented by the curve.

System Description. The neural net recognition system in **Fig. 4** is used to identify the well-test interpretation model. The recognition system consists of a data input module and two trained artificial nets. Two nets are used in the recognition process because it was difficult to arrive at one set of weights that would ensure correct classification of all models considered. The second net is used whenever the output in the third node of the first net is >0.5 or when the activation level in the second node is the highest.

Data Input. The input layer in each net consists of 60 nodes that accept 30 data points (x,y) from a derivative curve. This implies that the derivative curve should be sampled for 30 points **(Fig. 5).** The output layer of the first net has 16 nodes. Each node corresponds to a specific model class (Fig. 4). In this study, the first net is trained to identify seven different model classes, which implies that only seven nodes in the output layer are active. The rest of the nodes in the output layer can be used to incorporate new models. The output layer of the second net has two nodes, which

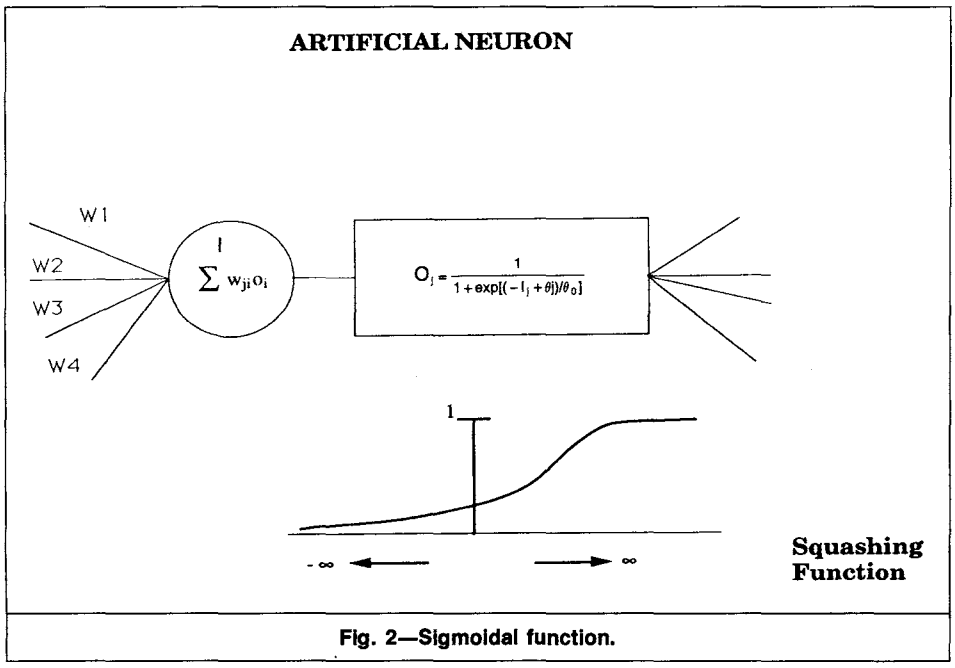


Fig. 2—Sigmoidal function.

correspond to two model classes (Fig. 4). The intermediate layer in each net has 120 nodes. The input curve is normalized between 0 and 1 by

$$x_i' = x_i / \sqrt{\Sigma (x_i)^2} \quad \ldots\ldots (4)$$

and $$y_i' = y_i / \sqrt{\Sigma (y_i)^2}. \quad \ldots\ldots (5)$$

Any normalization scheme can be used. The derivative curve must be normalized to avoid saturating the net. Saturation is the stage when the net stops learning because of very small sigmoidal function derivatives. Very small or zero derivatives occur when the argument of the sigmoidal function is large **(Fig. 6).** Note that the derivative curve on the logarithmic scale is shifted to the right or up by a value equal to the absolute value of the largest negative number on the axis on which the shift is done. This is necessary to scale all data points from 0 to 1. The output layer nodes are assigned the values 1 (firing) or 0 (silent), depending on whether the node represents the model. However, for computational accuracy, we used 0.9 and 0.1 instead of 1 and 0 because the sigmoidal function is 1 at positive infinity and 0 at negative infinity.

Training. Training is the stage when the net learns the recognition task by adjusting the weights in the links between nodes created by processing representative examples (input and output pairs). In the training set, we used models that included various combinations of inner boundary conditions (wellbore storage and skin or stimulation), reservoir types (homogeneous or dual porosity), and outer boundary conditions (infinite acting, closed outer boundary, and constant-pressure outer boundary). This phase is repeated until the net performance reaches a satisfactory level (i.e., recognition of all examples in the training set and at least 90% of test examples). Test examples are examples that are not included in the training set but that belong to the model classes that the net is trained to classify. Testing the net performance on new examples is important to ensure generalization. Generalization is the ability of the net to recognize patterns for which the net was not specifically trained. If generalization performance is poor, the net will act as a lookup table, which will limit its practical value. If the net identifies a test example with a high activation level (larger than 0.8), it is not added to the training set. If the identification was not successful or was poor, then the test example is added to the training set and the net is trained on this new set. The training set had 34 examples when training sessions were stopped.

The derivative computation took flow history effects into account by use of the derivative as a function of superposition time.

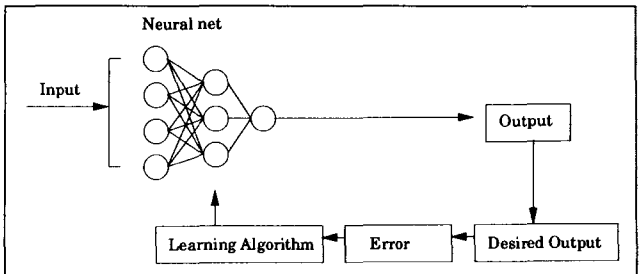


Fig. 3—Schematic of the learning cycle in an artificial net.

Each net is trained incrementally by increasing the training set size gradually until the net learns all model classes. The weights at which the net converges in each stage are used as the beginning weights for the next stage. A learning rate of 0.2 and a momentum value (explained in Appendix A) of 0.9 are used in the training sessions. The learning session is considered successful when the maximum error (Appendix A) in the output layer is <0.01.

To reduce the scaling effect (sensitivity to pattern size) during training, each derivative curve in the training set is presented to the net in different sizes (large, small, and medium). This was achieved by dividing the data points on the x and y axes after normalization by constant factors (2 and 3, respectively). An alternative method is to present the net with many examples of the same model class.

Examples

In this section, we present two examples to illustrate the application of our method.

Example 1. The first example is a buildup test in a severely damaged gas well. **Tables 1 and 2** show test data, and **Figs. 7 and 8** show diagnostic plots.

The derivative curve is sampled first for 30 data points and normalized between 0 and 1. The normalization procedure should be consistent with that used to normalize the training set examples. The normalized derivative curve is used as input to the neural nets. **Fig. 9** shows the input curve and the activation levels in the output layer. The highest activation level (0.982) indicates that the interpretation model is one in which the well response is from a well in

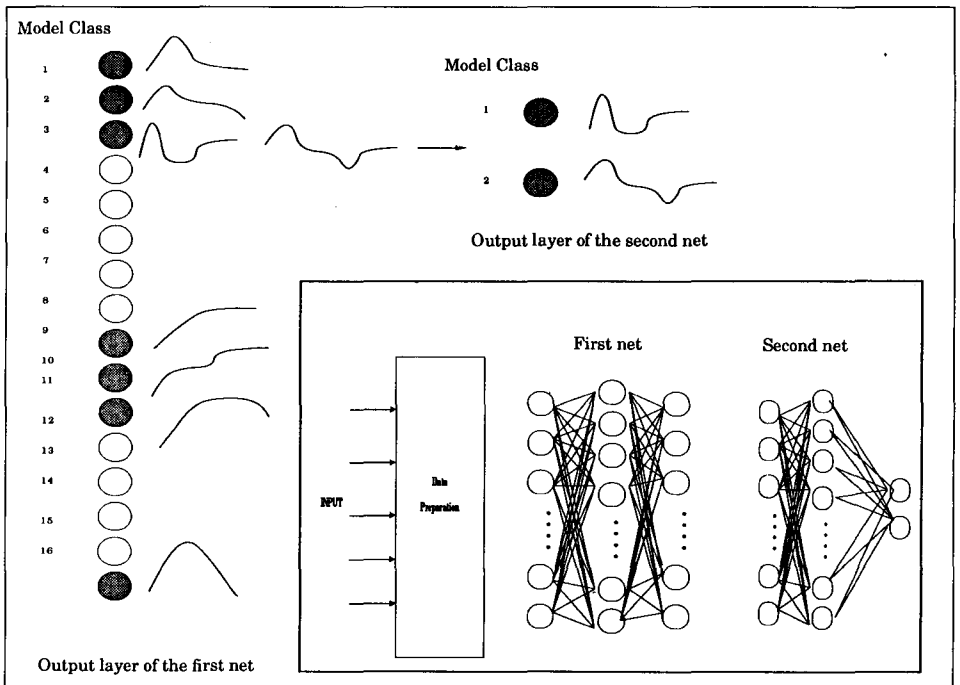


Fig. 4—Schematic of the recognition system showing output layer arrangements.

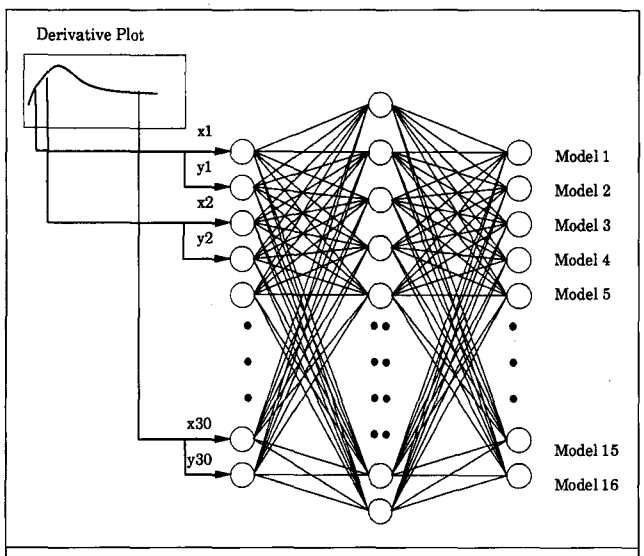


Fig. 5—Input of a derivative plot into the artificial neural net.

a homogeneous infinite-acting reservoir with wellbore storage and skin.

The second node in the ouput layer shows some activity (0.698), which indicates a response from a well affected by wellbore storage and skin in a homogeneous reservoir with a closed or constant-pressure outer boundary. This second possibility could be caused by the shallow downward trend of the last few points on the derivative plot, which suggests that the neural net approach is capable of ranking the models that could have caused the observed response on the basis of the distinctness of the features (signatures) that characterize each model.

The interpretation model suggested by the system is a preliminary model and should be verified with other information sources, such as other diagnostic plots, geologic information, well log data, and history matching, as Ref. 12 discusses.

The history-matching results indicate that the parameters obtained from analyzing the infinite-acting model accurately describe the actual pressure history of the well **(Fig. 10). Table 3** shows the interpretation model parameters.

An interesting feature of this approach is its very high tolerance to *random* noise (i.e., noise that is not correlated with the observed response). **Fig. 11** is the same input derivative curve distorted by random noise. The net was able to identify the interpretation model successfully despite the noise, as the activation levels indicated. Such performance is not guaranteed when dealing with very noisy field data. This result can be explained by the systematic (nonrandom) nature of noise in field data. Systematic noise produces false patterns (signatures) that confuse the net during recognition. We elaborate on this subject in the Recommendations for Future Work section.

Example 2. This example is a buildup test from a gas well completed in the eastern Devonian shales. Tables 1 and 2 in Ref. 12 contain the test data, and the diagnostic plots are presented in Figs. 5 and 6.[12]

Fig. 12 shows the input curve and the activation levels in the output layer. The highest activation level (0.954) indicates that the recorded response can be produced by either a heterogeneous (dual-porosity) model with stimulation in an infinite-acting reservoir (Model 1) or a homogeneous model with stimulation and a linear discontinuity (Model 2). Both models produce similar responses. More evidence now is needed to find the correct alternative. As explained in Ref. 12, Model 1 is more probable from geologic evidence.

The parameters obtained from history matching the test data with an analytical solution for a dual-porosity reservoir accurately

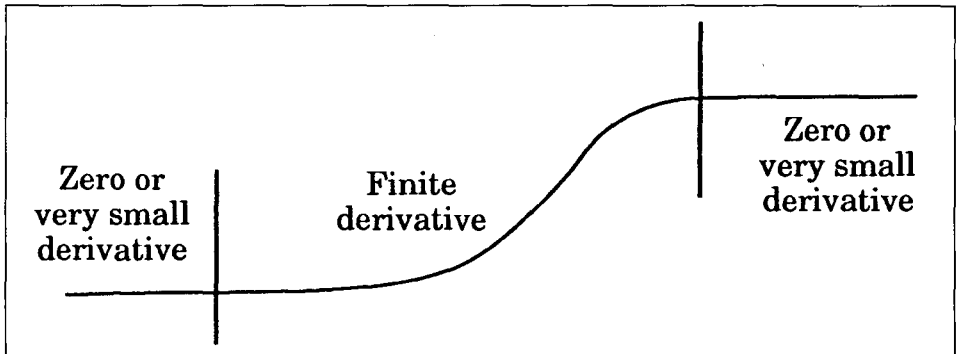


Fig. 6—Schematic of a sigmoidal function showing regions of derivative values.

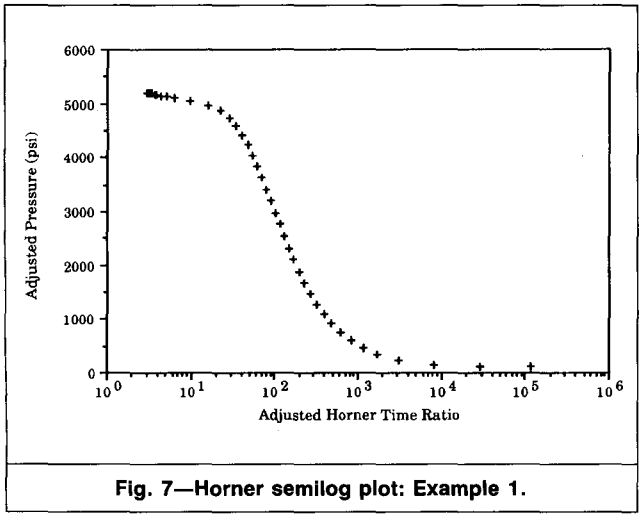


Fig. 7—Horner semilog plot: Example 1.

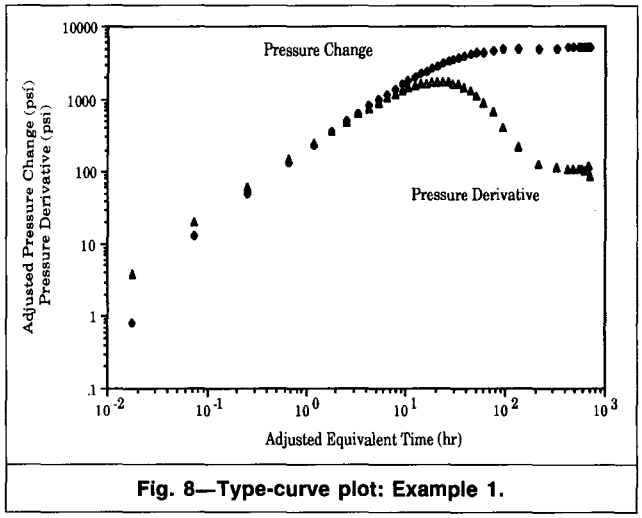


Fig. 8—Type-curve plot: Example 1.

TABLE 1—PRESSURE BUILDUP DATA, EXAMPLE 1

Time (hours)	Pressure (psi)	Derivative (psi)	Time (hours)	Pressure (psi)	Derivative (psi)
0.02	107.65	3.83	23.73	3,197.30	1,737.69
0.07	119.58	20.02	26.74	3,412.70	1,708.73
0.25	156.94	61.88	30.25	3,625.80	1,653.84
0.68	242.27	145.78	34.28	3,835.30	1,566.92
1.20	345.11	243.38	39.00	4,037.70	1,441.65
1.82	465.06	356.20	44.99	4,231.90	1,289.53
2.53	601.47	482.02	52.30	4,415.90	1,095.87
3.36	752.51	625.70	62.15	4,586.20	889.41
4.30	917.56	750.56	76.36	4,736.20	662.70
5.32	1,094.20	877.73	99.06	4,861.70	413.15
6.53	1,281.80	1,026.09	141.33	4,960.30	225.23
7.83	1,477.30	1,167.53	240.75	5,038.60	129.54
9.23	1,681.10	1,291.80	388.95	5,093.20	112.18
10.72	1,889.30	1,418.27	516.02	5,124.30	109.85
12.40	2,103.80	1,516.32	625.94	5,145.80	108.18
14.27	2,319.60	1,594.00	721.87	5,160.80	107.29
16.25	2,538.10	1,654.57	806.27	5,172.60	106.39
18.44	2,758.40	1,706.28	881.07	5,182.20	103.38
20.97	2,976.90	1,740.07	947.81	5,189.70	73.47

TABLE 2—RESERVOIR, FLUID, AND WELL PROPERTIES, EXAMPLE 1

Porosity, fraction	0.150
Net pay thickness, ft	25
Gas viscosity, cp	0.0203
Gas FVF, RB/Mcf	0.855
Gas-law deviation factor	0.924
Total compressibility, psi $^{-1}$	0.000123
Water saturation	0.50
Wellbore radius, ft	0.25
Rate, Mcf/D	190.0
Production time, hours	1,200.0

TABLE 3—INTERPRETATION MODEL PARAMETERS, EXAMPLE 1

Reservoir type	Homogeneous
Permeability, md	0.115
Skin factor	23.96
Dimensionless wellbore storage coefficient	985.96

describe the actual pressure history of the well **(Fig. 13).** Node 11 in the output layer shows an activation level of 0.408, which is considered too low to be significant. The model associated with this node is for a stimulated well in a homogeneous reservoir with closed or constant-pressure outer boundary. Some features from this model are present in our example input curve. We also distorted the same input curve by introducing random noise. As in Example 1, the model was identified again despite this distortion.

Recommendations for Future Work

In this study, we showed that the neural net approach is effective in identifying the well-test interpretation model. In this section, we provide recommendations based on our experience with neural nets.

The performance described in this paper is from the examples we selected for our training set. Other training sets can provide better or worse performance, depending on how representative the training examples are of the pattern recognition space the system will identify. Therefore, we strongly recommend careful selection of training examples.

We noted that the approach in this paper has high tolerance to random noise but that it performs poorly with data affected by systematic noise. Because we do not know how systematic noise is distributed in test data, we recommend that neural nets be trained without examples affected by this type of error. In fact, we recommend use of high-quality data (analytically generated data are preferred) for the training examples so that the net will extract the most significant features of the curve used. Training the net with data affected by systematic noise can result in the net's extracting the wrong features.

As more models are included in the net decision space and more training examples are added to the training set, a general approach like the one presented here reaches a stage where the net cannot learn new models or where training time can become very long (days of CPU time). For this reason, modular design can be very attractive. Smaller nets (the size of the net reflects the number of links) can be trained to recognize one model. The smaller nets can be integrated to act as a larger net. In such a design, training and modifications are applied only to those nets that do not perform well. **Fig. 14** is a schematic of this modular approach.

In this study, the well-test interpretation model was identified from the derivative plot only. We recommend use of the approach summarized here to identify the well-test interpretation model, or at least part of the interpretation model, from other diagnostic plots, such as the semilog graph. The neural net can be trained to recognize such significant features as those caused by phase segregation, skin, and stimulation. We also can train a net to recognize problems in the recorded data, such as those from leaks.

Artificial neural nets are computationally intensive, which implies that they require fast computers during training. In this study, we used a personal computer with a math coprocessor and a 20-MHz accelerator. The simulator was written in C, which allows dynamic memory allocation.

Conclusions

This paper presented a new approach, based on artificial neural nets, to identify a preliminary well-test interpretation model automatically from a derivative plot. This approach can be part of a well-test expert system or a computer-enhanced well-test interpretation.

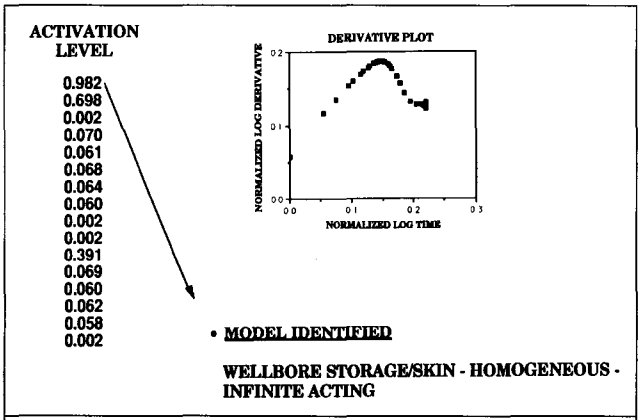


Fig. 9—Normalized derivative curve and activation levels in the output layer: Example 1.

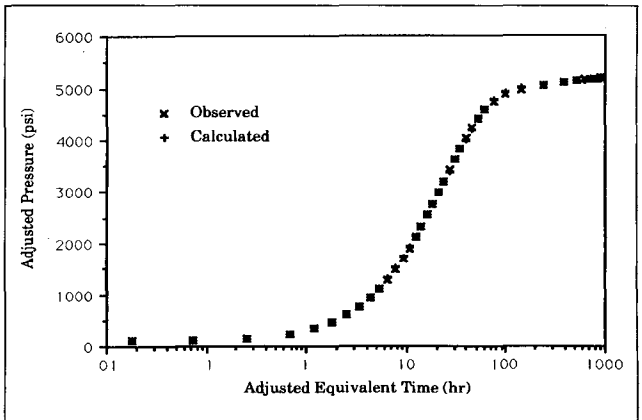


Fig. 10—Observed and calculated pressure data: Example 1.

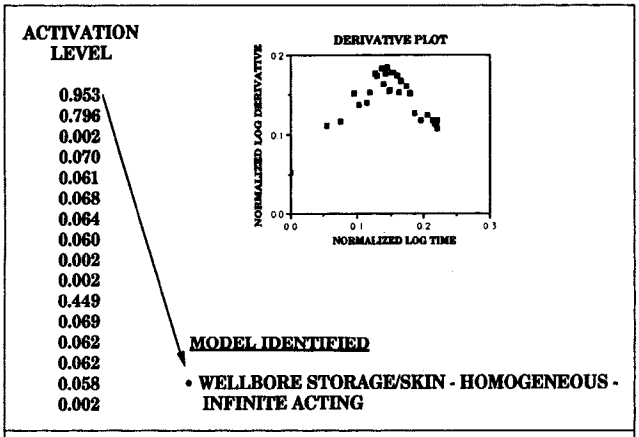


Fig. 11—Distorted normalized derivative curve and activation levels in the output layer: Example 1.

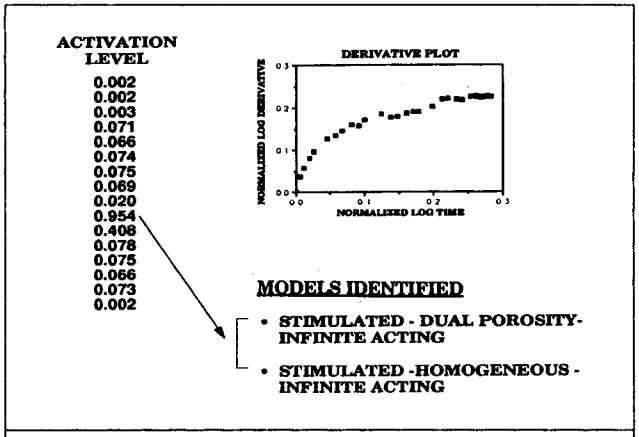


Fig. 12—Normalized derivative curve and activation levels in the output layer: Example 2.

Nomenclature

E = error
$f_k(I_k)$ = sigmoidal function derivative
I = input
n = number of iteration
O = calculated output
t = expected ouptut
w = weight
x_i = coordinate of the ith data point before normalization
y_i = coordinate of the ith data point before normalization
α = momentum
δ = derivative of E with respect to I (Eq. A-8)
η = learning rate constant
θ = threshold

Subscripts

i,j = layers
ij = between Layers i and j
ji = between Layers j and i
jk = between Layers j and k
kj = between Layers k and j
P = Pattern P

Superscript

$'$ = after normalization

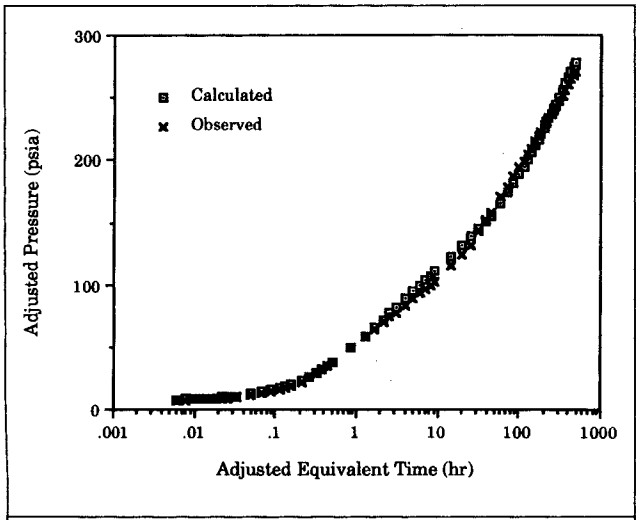


Fig. 13—Observed and calculated pressure data: Example 2.

Acknowledgments

We thank King Fahd U. of Petroleum & Minerals for scholarship support. We also thank the Crisman Inst. for Petroleum Reservoir Management at Texas A&M U. for supporting this research.

References

1. Clark, D.G. and Von Golf-Racht, T.D.: "Pressure Derivative Approach to Transient Test Analysis: A High-Permeability North Sea Reservoir Example," *JPT* (Nov. 1985) 2023.
2. Gringarten, A.C.: "Computer-Aided Well Test Analysis," paper SPE 14099 presented at the 1986 Intl. Meeting on Petroleum Engineering, Beijing, March 17–20.
3. Menke, W.: *Geophysical Data Analysis: Discrete Inverse Theory*, Academic Press Inc., New York City (1989).
4. Allain, O.F. and Horne, R.N.: "Use of Artificial Intelligence in Well-Test Interpretation," *JPT* (March 1990) 342.
5. Fu, K.S.: *Syntactic Pattern Recognition and Applications*, Prentice-Hall Inc., Englewood Cliffs (1982).
6. Stewart, G. and Du, K.F.: "Feature Selection and Extraction for Well-Test Interpretation Model Selection by an Artificial-Intelligence Approach," paper SPE 19820 presented at the 1989 SPE Annual Technical Conference and Exhibition, San Antonio, Oct. 8–11.
7. Pao, Yoh-Han: *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley Publishing Co., Reading, PA (1989).

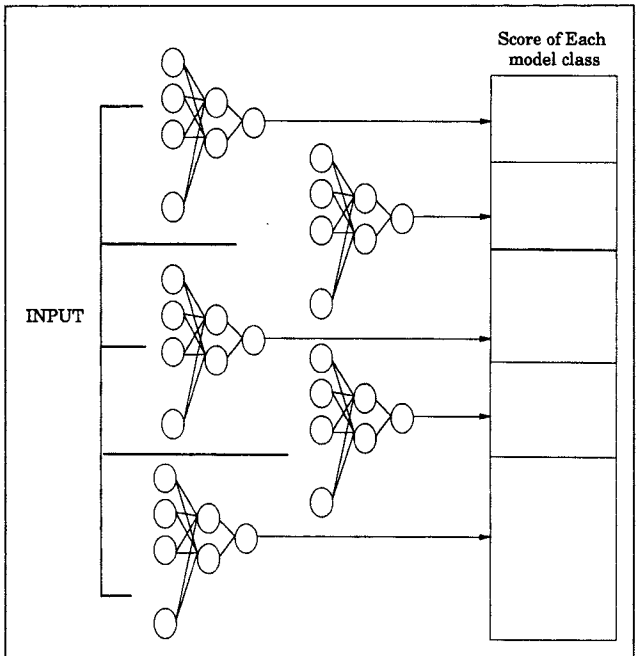


Fig. 14—Schematic of a modular recognition system with neural nets.

8. Wasserman, Philip: *Neural Computing: Theory and Practice*, Van Nostrand Reinhold Co., New York City (1989).
9. McClelland, J. and Rumelhart, D.: *Exploration in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercise*, The MIT Press, Cambridge, MA (1989).
10. Rumelhart, D., McClelland, J., and the PDP Research Group: *Parallel Distributed Processing: Exploration in the Microstructure of Cognition-Foundation*, The MIT Press, Cambridge, MA (1986) Chap. 1.
11. Rumelhart, D., McClelland, J., and the PDP Research Group: *Parallel Distributed Processing: Exploration in the Microstructure of Cognition—Psychological and Biological Models*, The MIT Press, Cambridge, MA (1986) Chap. 2.
12. Al-Kaabi, A.U., McVay, D.A., and Lee, W.J.: "Using An Expert System To Identify the Well-Test Interpretation Model," *JPT* (May 1990) 654.

Appendix A—Algorithm Derivation

In this appendix, we summarize the mathematical derivation of the backpropagation learning algorithm.[7] Then we explain the step-by-step learning procedure. We use Fig. 1 to clarify the derivation procedure.

Backpropagation Learning Rule. The following derivation is for links connected to the outer layer (e.g., links between Layers i and k). Later, a similar derivation procedure will be used to derive an expression for changing weights in links not connected to the output layer (e.g., links between Layers i and j).

Define the net input to a node in Layer j from Pattern P as

$$I_{Pk}=\sum_j w_{jk}O_{pj}. \quad \text{(A-1)}$$

The output, O_{Pj}, from a node in the middle (hidden) Layer j because of Pattern P is defined as

$$O_{Pj}=1/[1+\exp(-I_{Pj})]. \quad \text{(A-2)}$$

Similarly, the output from a node in the output Layer k caused by Pattern P is defined as

$$O_{Pk}=1/[1+\exp(-I_{Pk})]. \quad \text{(A-3)}$$

Define error as

$$E=\frac{1}{2}\sum_P\sum_k (t_{Pk}-O_{Pk})^2. \quad \text{(A-4)}$$

The division by two is for mathematical convenience (as explained later, this 2 will be cancelled by a 2 that results from taking derivatives). Eq. A-4 is the objective function, which we will minimize during learning iterations.

Convergence (learning) is achieved by changing the weights by an incremental value, Δw_{ij}. This value is defined as proportional to E by

$$\Delta w_{kj}=-\eta(\partial E/\partial w_{kj}). \quad \text{(A-5)}$$

The derivative on the right side of Eq. A-5 can be written

$$\frac{\partial E}{\partial w_{kj}}=\frac{\partial E}{\partial I_{Pk}}\frac{\partial I_{Pk}}{\partial w_{kj}}. \quad \text{(A-6)}$$

By taking the derivative of Eq. A-1, we obtain

$$\frac{\partial I_{pk}}{\partial w_{kj}}=\frac{\partial \sum_j w_{kj}O_{Pj}}{\partial w_{kj}}=O_{Pj}. \quad \text{(A-7)}$$

Now, define

$$\delta_{Pk}=-(\partial E/\partial I_{Pk}). \quad \text{(A-8)}$$

By using Eqs. A-6 through A-8 in Eq. A-5, we obtain

$$\Delta w_{kj}=\eta\delta_{Pk}O_{Pj}. \quad \text{(A-9)}$$

Eq. A-7 gives the required weight change within a link between a node in Layer j and a node in the output Layer k. However, we need to find a computable form for δ_{Pk}. If we use the chain rule, we can write

$$\delta_{Pk}=-\frac{\partial E}{\partial I_{Pk}}=-\frac{\partial E}{\partial O_{Pk}}\frac{\partial O_{Pk}}{\partial I_{Pk}}. \quad \text{(A-10)}$$

From the definition of E (Eq. A-4), we obtain

$$\partial E/\partial O_{Pk}=-(t_{Pk}-O_{Pk}), \quad \text{(A-11)}$$

and, from the definition of O_{Pk} (Eq. A-3), we obtain

$$\partial O_{Pk}/\partial I_{Pk}=f_{Pk}(I_{Pk}), \quad \text{(A-12)}$$

where the right side of Eq. A-12 is the derivative of the squashing function. Substituting Eqs. 11 and 12 into Eq. A-10, we obtain

$$\delta_{Pk}=(t_{Pk}-O_{Pk})f_{Pk}(I_{Pk}). \quad \text{(A-13)}$$

Substituting Eq. A-13 into Eq. A-9, we obtain

$$\Delta w_{jk}=\eta(t_{Pk}-O_{Pk})f_{Pk}(I_{Pk})O_{Pj}. \quad \text{(A-14)}$$

Eq. A-14 gives the weight change in a link between Layer j and an output layer. η usually is called the learning rate and takes any value between 0 and 1. The higher the learning rate, the higher the weight changes. We recommend that the learning rate be kept at low values (e.g., 0.1 or 0.2) during early stages of learning and that it be increased as the net begins to converge.

A modified derivation procedure similar to the one outlined previously can be used to find an expression for the weight change in a link not connected to the output layer (e.g., links between Layers i and j). Δw_{ij} is defined as

$$\Delta w_{ij}=-\eta(\partial E/\partial w_{ij}). \quad \text{(A-15)}$$

The derivation that follows is similar to that for the case when the link is connected to the output layer.

$$\Delta w_{ij}=-\eta(\partial E/\partial I_{Pj})(\partial I_{Pj}/\partial w_{ij}) \quad \text{(A-16)}$$

$$=-\eta(\partial E/\partial I_{Pj})O_{Pi} \quad \text{(A-17)}$$

$$=\eta[-(\partial E/\partial O_{Pj})(\partial O_{Pj}/\partial I_{Pj})]O_{Pi} \quad \text{(A-18)}$$

$$=\eta[-(\partial E/\partial O_{Pj})]f_{Pj}I_{Pj}O_{Pi} \quad \text{(A-19)}$$

$$=\eta\delta_{Pj}O_{Pi}, \quad \text{(A-20)}$$

where $\delta_{Pj}=-(\partial E/\partial O_{Pj})f_{Pj}(I_{Pj})$. (A-21)

The partial derivative $\partial E/\partial O_{Pj}$ cannot be evaluated directly because E is based on output in the output layer only. Consider the following indirect approach:

$$-\frac{\partial E}{\partial O_{Pj}}=-\sum_k\frac{\partial E}{\partial I_{Pk}}\frac{\partial I_{Pk}}{\partial O_{Pj}}=\sum_k\left(-\frac{\partial E}{\partial I_{Pk}}\right)\frac{\partial}{\partial O_{Pj}}\sum_j w_{kj}O_{Pj}. \quad \text{(A-22)}$$

Recall that

$$-(\partial E/\partial I_{Pk})=\delta_{Pk}. \quad \text{(A-23)}$$

Substituting Eq. A-23 into Eq. A-22 and evaluating the partial derivative on the right side of Eq. A-22, we obtain

$$-(\partial E/\partial O_{Pj})=\sum_k\delta_{Pk}w_{kj}. \quad \text{(A-24)}$$

Substituting Eq. A-24 into Eq. A-21, we obtain

$$\delta_{Pj}=f_{Pj}(I_{Pj})\sum_k\delta_{Pk}w_{kj}. \quad \text{(A-25)}$$

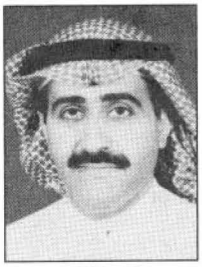
Finally, substituting Eq. A-25 into Eq. A-20, we obtain

$$\Delta w_{ij}=\eta O_{Pj}f_{Pj}(I_{Pj})\sum_k\delta_{Pk}w_{kj}. \quad \text{(A-26)}$$

Eq. A-26 implies that the weight change in an internal link is based on the weight change in a higher layer (in this case, Layer k), which

Authors

Lee



Al-Kaabi

W. John Lee holds the L.F. Peterson chair in Petroleum Engineering at Texas A&M U. He also is executive vice president of S.A. Holditch and Assocs. Previously, he was a technical adviser at Exxon Co. U.S.A. and a senior research specialist at Exxon Production Research Co. Lee holds BChE, MS, and PhD degrees in chemical engineering from Georgia Tech. He is a member of the Career Guidance and Archie Scholarship committees and the Emerging/Peripheral Technical Committee. Lee is a Distinguished Service Award winner and a member of the National Academy of Engineering. Abdul-Aziz U. Al-Kaabi is assistant professor and head of the Petroleum Engineering Section at King Fahd U. of Petroleum and Minerals (KFUPM) Research Inst. He holds BS and MS degrees in petroleum engineering from KFUPM in Dhahran, Saudi Arabia, and a PhD degree in petroleum engineering from Texas A&M U.

also shows that the error is propagated from the output layer back to all the links in the net. In the next section, we describe the step-by-step training procedure using a backpropagation algorithm as the learning rule.

Appendix B—Training Procedure

The following procedure is used to train a neural net with backpropagation as the learning algorithm.

1. Define a maximum node error (large negative number).
2. Randomize the weights in all the net links.
3. Select a pattern from the training set (a pattern includes input and output pairs).
4. Evaluate the outputs at the output layer by propagating the input from the input layer to the output layer in a feed-forward manner.
5. Calculate the error at the output layer. If the node error is larger than the maximum error, set the maximum error equal to the node error. Repeat for all nodes in a feed-forward manner.
6. Use the backpropagation procedure described in Appendix A to calculate the amount of weight change required in each link because of the current pattern. Do not apply any changes to the links yet.
7. Repeat Steps 3 through 6 until the net has been presented with all the patterns in the training set. During this process, accumulate the weight changes in each link caused by the different patterns. When the net is given all the examples in the training set, use the cumulative weight change for each link to change the weight of that link.
8. Evaluate errors at the output layer. If the maximum error is less than the desired error, stop; otherwise, repeat Steps 3 through 7.

Appendix C—Practical Considerations

The shape of the squashing function can be changed to increase the convergence rate for certain problems:[7]

$$O_{Pk}=\frac{1}{1+\exp[(-I_{Pk}+\theta_k)/\theta_0]}. \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}1)$$

θ_k is a bias parameter. If θ_k is positive, the squashing function is shifted to the left, whereas if θ_k is negative, the squashing function is shifted to the right along the horizontal axis. The parameter θ_0 is always positive and modifies the shape of the function. Large values of θ_0 result in a gently increasing function; low values of θ_0 result in a steeper function. The parameter θ_0 can be fixed for all the net nodes or can be variable.

Another popular modification to increase the learning rate of the backpropagation algorithm is the use of a momentum term. The change in the link weight with the momentum term added is defined as[7]

$$\Delta w_{ij}(n+1)=\eta\delta_{Pk}O_{Pj}+\alpha\Delta w_{ij}(n), \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (C\text{-}2)$$

where α takes values between 0 and 1.

SI Metric Conversion Factors

bbl	×	1.589 873	E−01	= m^3
cp	×	1.0*	E−03	= Pa·s
cycles/sec	×	1.0*	E+00	= Hz
ft	×	3.048*	E−01	= m
ft^3	×	2.831 685	E−02	= m^3
psi	×	6.894 757	E+00	= kPa

*Conversion factor is exact.

SPEFE

Original SPE manuscript received for review June 25, 1990. Revised manuscript received Jan. 19, 1993. Paper accepted for publication Sept. 1, 1992. Paper (SPE 20332) first presented at the 1990 SPE Petroleum Computer Conference in Denver, June 25–28.

Discussion of Using Artificial Neural Nets To Identify the Well-Test Interpretation Model

Kacheong Yeung, SPE, U. of Alberta, **Chayan Chakrabarty**, SPE, Golder Assocs. (U.K.) Ltd., and **Sherman Wu**, U. of Melbourne

Use of artificial neural nets (ANN's) to identify noisy and apparently unrecognizable patterns is common for many real-world problems, ranging from applications such as speech recognition to stock-market prediction. ANN approaches are often good candidates for recognizing patterns when rigid mathematical models do not exist or are insufficient to meet a full-scale identification requirement. Al-Kaabi and Lee's[1] proposal of using ANN's to identify the well-test interpretation model is appropriate because well-test data is often highly nonlinear and noisy. The purpose of this discussion is to present some of our results in a similar study and to suggest a simple technique that would enhance the use of ANN's in Al-Kaabi and Lee's approach.

The idea behind an ANN is quite simple: it is a program that classifies samples (patterns) by adjusting its parameters repeatedly until all the samples are correctly classified. When a large number of samples are used in the process (training), a test pattern can be identified if it resembles one of the training samples. However, if the test pattern does not resemble any of the training samples, the neural net would not classify the pattern in the corresponding class.

One of the main problems in interpreting well-test data with ANN's is that, for a particular model, we do not know when the pattern begins and when it ends, let alone the various parameters (k, C_D, s, etc.) that control the pattern shape. Because of this, even one model (with the same reservoir parameters) would require many samples of different beginning and endpoints to identify its class. Al-Kaabi and Lee introduced the scaling of the following forms to reduce this effect.

$$x_i' = \frac{x_i}{\sqrt{\sum (x_i)^2}} \quad \text{..... (D-1)}$$

$$\text{and } y_i' = \frac{y_i}{\sqrt{\sum (y_i)^2}}. \quad \text{..... (D-2)}$$

They further attempted to reduce the sensitivity to pattern size by training the same samples with different sizes, which is done by dividing the data points on the x and y axes after normalization by constant factors (e.g., by 2 and 3). However, we found this scaling method to be undesirable in our neural-net program. To illustrate this, consider a hypothetical well-test response model, Model A (**Fig. D-1**). The model has two similar patterns, one smaller in size with longer beginning and ending positions as compared with the other. Using the scaling method Al-Kaabi and Lee proposed, we obtain scaled versions of the two patterns (**Fig. D-2**). We see that after scaling, the shapes have been altered considerably. Thus, if Pattern 1 was used as a training set for Model A, the neural net probably would not identify Pattern 2 as Model A. The long beginning and ending positions of Pattern 2 shrink the shape relative to Pattern 1 because of the scaling used. Another consequence of the scaling that Al-Kaabi and Lee suggested is that the pattern spans are different. To eliminate this effect, we suggest normalization between 0 and 1; i.e.,

$$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \quad \text{..... (D-3)}$$

$$\text{and } y_i' = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}}. \quad \text{..... (D-4)}$$

This will ensure that all scaled patterns have the same spans in the x and y directions. However, this still does not solve the problem of shrinking (or expanding) a pattern having longer (or shorter) beginning and ending positions.

A widely applicable technique for efficient algorithms is the "divide-and-conquer" strategy, which consists of breaking a larger problem into smaller problems in such a way that, from the combined solutions to the smaller problems, one can easily construct a solution to the entire problem. In Fig. D-1, Model A has four distinct parts (two horizontal lines and two slope lines). If we modify Eq. 3 to be

$$x_i' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \left(\frac{1}{n}\right), \quad \text{..... (5)}$$

where n is the number of subparts, the resulting scaling of patterns in Fig. D-1 is identical (**Fig. D-3**). Here, we let $y_i'=0$ when the denominator becomes 0 in Eq. 4. From Fig. D-3, we see that if Pattern 1 is used as the training set for Model A, the neural net also can identify Pattern 2 as Model A because the scaled versions of the patterns are identical. For the well-test data, the natural divisions of subparts would be the local minima and maxima of the derivative plots.

In our study, we trained the ANN's with typical well-test models, such as infinite-acting homogeneous and double-porosity formations with and without wellbore storage and skin, etc. We found that

(28151)

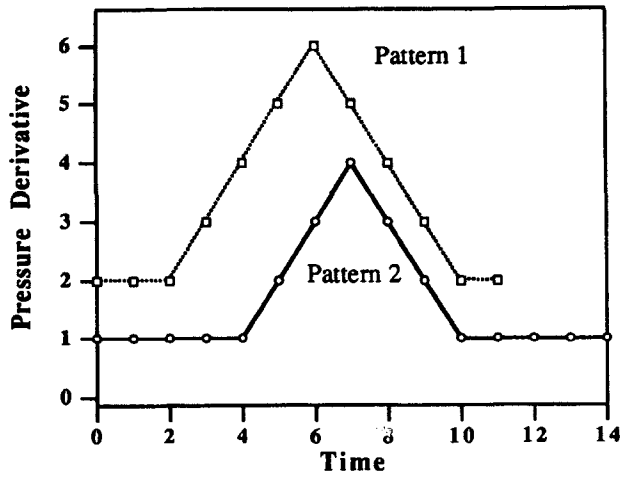


Fig. D-1—Two original patterns for Model A.

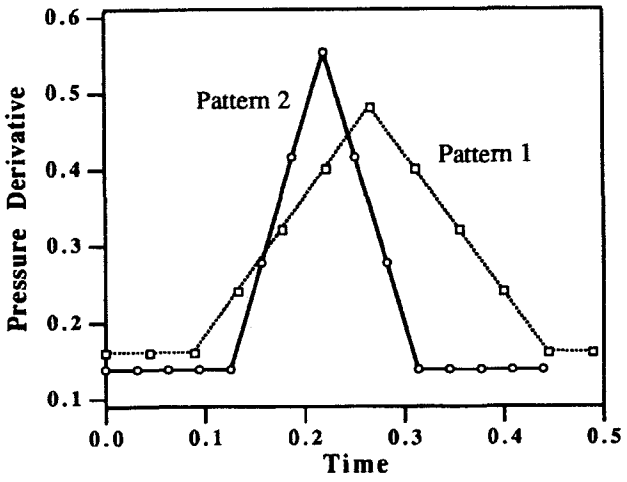


Fig. D-2—Two scaled patterns for Model A (Al-Kaabi and Lee's approach[1]).

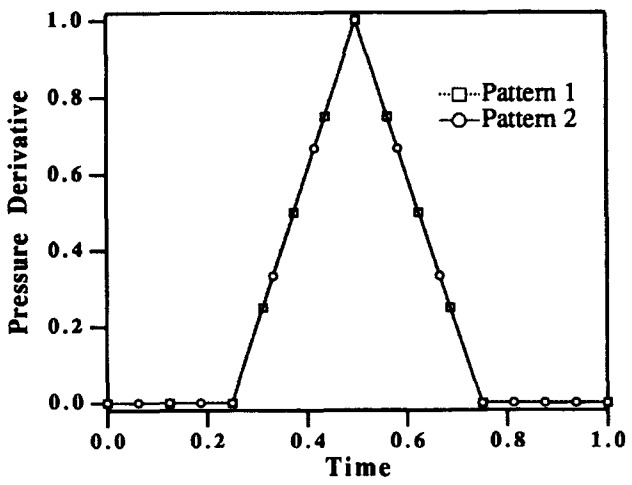


Fig. D-3—Two scaled patterns for Model A (divide-and-conquer approach).

after we had trained the ANN's for a particular well-test model, using Al-Kaabi and Lee's scaling approach, when the same well-test data was truncated by ≈5% of the original data, the ANN's would fail to identify the model. However, with our scaling approach, the truncation has little or no effect on model identification, unless the truncation was so large that a subpart of the pattern was truncated. Thus, using this proposed method of scaling, the training set is reduced significantly. And, as illustrated in our simple hypothetical example, it is more accurate and reliable.

As a final note, other ANN's are more efficient than the backpropagation neural network. For example, we have also programmed this problem with an adaptive logic network (ALN) with a unary encoding scheme; the ALN decreased the training time to minutes instead of the days needed for backpropagation. Other networks, such as the self-organized neural network, are also good candidates.

Acknowledgments

We thank W.W. Armstrong, U. of Alberta, for providing his ALN software, and Yun Wang, who programmed the ALN for the problem.

Reference

1. Al-Kaabi, A.U. and Lee, W.J.: "Using Artificial Neural Nets To Identify the Well-Test Interpretation Model," *SPEFE* (Sept. 1993) 233; *Trans.*, AIME, **295**.

(SPE 28151) **SPEFE**

Authors' Reply to Discussion of Using Artificial Neural Nets To Identify the Well-Test Interpretation Model

Abdulaziz U. Al-Kaabi, SPE, King Fahd U. of Petroleum & Minerals, and **W. John Lee,** SPE, Texas A&M U.

We thank Yeung *et al.* for their discussion[1] about our paper.[2] We agree with Yeung *et al.* that their proposed scaling method (see Eq. 5 in Ref. 1), when applied to patterns with distinct subparts such as the one shown in their Fig. D-1, represents an improvement on the method we proposed. This is particularly true because Yeung *et al.*'s method eliminates the need to train the artificial neural networks (ANN's) on different sizes (scales) of the same pattern of a specific interpretation model. We present the following comments for discussion and suggestions and for further improvement.

1. Yeung *et al.*'s scaling method requires that the number of subparts (segments) of the derivative curve, for which the well-test model will be recognized by a trained ANN, be identified before normalizing and scaling the derivative plot. Deciding on the number of such subparts in derivative plots generated from field well-test data can be difficult and subjective because of noise and interpreter bias. The outcome of pattern scaling will depend on the number of subparts used. The motivation behind using ANN's to identify a preliminary well-test model is to present a method without any interpreter subjectivity or bias.

To minimize such subjectivity in determining the number of subparts that reasonably subdivide a derivative plot, as proposed by Yeung *et al.*, we suggest looking at algorithms, such as the split-and-merge algorithm.[3,4] The objective of the split-and-merge algorithm is to find the minimum number of segments in a plot where, on each of the segments, the data points are approximated by a piecewise linear function with an error norm less than a prespecified quantity. Nonlinear functions can also be used in the previously mentioned algorithm to approximate each segment. It is important to realize that the segmentation that is produced by the split-and-merge algorithm depends on a prespecified error tolerance. Such an approach can give undesirable results when the derivative plot is noisy.

2. Yeung *et al.*'s proposed method assures that all normalized derivative plots of a specific well-test model will be identical only if the corresponding segments of the derivative plots have the same slopes and if the same number of subparts is used to normalize and scale each plot. The normalized and scaled plots using Yeung *et al.*'s method are not identical for similar models if the slopes of the subparts (segments) of their derivative plots are different. This difference in slopes could indicate the need to train the ANN's on more patterns to capture this variation in slopes. Otherwise, the learning may not be general.

The combined effect of using a different number of subparts in scaling patterns of similar models and having corresponding segments of different slopes could be significant. Understanding the extent of such an effect requires further investigation.

References

1. Yeung, K., Chakrabarty, C., and Wu, S.: "Discussion of Using Artificial Neural Nets To Identify the Well-Test Interpretation Model," *SPEFE* (June 1994).
2. Al-Kaabi, A.U. and Lee, W.J.: "Using Artificial Neural Nets To Identify the Well-Test Interpretation Model," *SPEFE* (Sept. 1993) 233; *Trans.*, AIME, **295**.
3. Pavlids, T. and Horowitz, S.L.: "Segmentation of Plane Curves," *IEEE Trans. on Computers* (Aug. 1974) **c-23**, No. 8, 860.
4. Al-Kaabi, A.U.: "Artificial Intelligence Approach to the Identification of the Well Test Interpretation Model," PhD dissertation, Texas A&M U., College Station (May 1990) 79.

(28165) **SPEFE**