01 Jan 1998

A Hybrid System for Well Test Analysis

Edward A. May

Cihan H. Dagli
Missouri University of Science and Technology, dagli@mst.edu

Recommended Citation

E. A. May and C. H. Dagli, "A Hybrid System for Well Test Analysis," *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1998.
The definitive version is available at https://doi.org/10.1109/IJCNN.1998.682280

A Hybrid System for Well Test Analysis

Edward A. May
Smart Engineering Systems Laboratory
University of Missouri-Rolla
eamay@umr.edu

Cihan H. Dagli
Smart Engineering Systems Laboratory
University of Missouri-Rolla
dagli@umr.edu

Abstract

Petroleum well test analysis is a tool for estimating the average properties of the reservoir rock. It is a classic example of an inverse problem. Visual examination of the pressure response of the reservoir to an induced flow rate change at a well allows the experienced analyst to determine the most appropriate model from a library of generalized analytical solutions. Rock properties are determined by finding the model parameters that best fit the observed data. This paper describes a framework for a hybrid network to assist the analyst in selecting the appropriate model and determining the solution. The hybrid network design offers significant advantages by reducing training time and allowing incorporation of both symbolic and numeric data. The network structure is described and the advantages and disadvantages compared to previous approaches are discussed.

1. Introduction

The well test is one of the primary diagnostic tools used in the evaluation of the productive capacity of oil and gas wells. It is a very important tool because it one of the few ways that engineers can actually "see" into a reservoir located many hundreds or thousands feet below the earth's surface. The results of the test can directly affect the profitability of the well and aid in decision making about the future of the well and the need for additional wells in a given reservoir.

Our understanding of the reservoir is limited to analytical models that have been derived to explain the response of wells over time. These models are based on equations of fluid flow and heat transfer, and include not only the reservoir, but also inner and outer boundary conditions. While the reservoir model itself can tell us the maximum potential of the well, the boundary conditions are perhaps even more important. From these boundary conditions the engineer can determine the size of the reservoir and the extent to which the formation was damage caused by the drilling process.

The well test is a classic example of an inverse problem. An input signal is applied to the reservoir, in the form of a change in the flow rate, and the pressure response is measured for a pre-determined period of time. From this response the model must be inferred [1]. If the model selected is incorrect, predictions of future response are also likely to be incorrect. Selection of the model is further complicated by a high degree of similarity of the test responses between dissimilar models and the presence of noise in the test response.

The earliest work involving intelligent logic used a rule-based blackboard architecture to examine the data [2]. While very robust, it required a large number of rules for a relatively limited number of models. In a fully developed form it would require an unacceptably large number of rules and the run time would be excessive.

Neural networks of various types, including feed-forward networks using backpropagation [3], a hybrid approach combining symbolic and backpropagation methods [4], a network incorporating the sequential predictive probability method [5], and a higher-order neural network architecture [6] have been implemented to attempt to identify the proper well test interpretation model. Although these implementations have been generally successful, there remains a need to further improve the time it takes to perform the analysis and train the network. The ability to discriminate between highly similar model responses was limited in the neural network and probability approaches, but was generally good in the applications incorporating symbolic and rule-based logic. Also, since the test only provides a small window of the overall reservoir response, the network must be able to handle responses where early or late time data are missing and where the key features occur at different times, and with different magnitudes, during the test.

May and Dagli [7] recently examined an application of the Hausdorff-Voronoi Network (HaVNet) developed by Rosandich [8]. This network offers significant training time reductions compared to feed-forward designs and allows for multiple, dissimilar, representations of a given model to be attached to a single class. This feature greatly

0-7803-4859-1/98 $10.00©1998 IEEE

improves the run time of the network. In this application the HaVNet proved successful in determining the model in noise-free cases, but was increasingly unsuccessful as the noise level increased.

This paper describes a framework in which a modified version of the HaVNet architecture is employed to classify the sequence of features that occur in the test. The data is filtered to remove the noise and a feature extraction algorithm is employed. Several rule-based systems are employed to guide the user through the data entry and to resolve any uncertainty in the HaVNet analysis. Rule-based and non-linear regression techniques are used to find the parameters that best fit the selected model. A modular approach is used to enhance the ease of operation of the network. This application illustrates the advantages of applying appropriate elements of computational intelligence to small parts of a problem as opposed to a single-architecture approach.

2. Overview of the HaVNet network

The HaVNet network architecture was designed by Rosandich [8] for robotic vision applications. The network gets its name from the use of the Hausdorff distance as a measure of similarity between patterns, and because it employs a learned version of the Voronoi surface to perform the comparison.

2.1. Network architecture

The HaVNet neural network behaves as a binary pattern classifier. The network takes as inputs two-dimensional binary patterns, employs feed-forward processing, and produces an analog output value. A single analog output value is generated by each node, with the value indicating the level of match between the input pattern and the class represented by that node. The HaVNet neural network consists of three layers, the plastic layer, the Voronoi layer, and the Hausdorff layer. An overview of the architecture is shown in Figure 1(a). The plastic layer contains neurons with weights that are trained during the learning process. The Voronoi layer serves to measure the distance between individual points in the input and learned patterns, and the Hausdorff layer uses information from the Voronoi layer to compute the overall level of similarity between the input pattern and the learned pattern. Figure 1(b) shows a detailed diagram of the architecture for a single node. The node is shown in a configuration for one-dimensional inputs for reasons of clarity. In the actual network, the input pattern, plastic layer, and Voronoi layer are all two-dimensional.

2.2. Network learning

Learning in the HaVNet architecture is conducted off-line and in a supervised manner by presenting examples of each class to the network during a training phase. The network is informed *a priori* of the class to which each training pattern belongs. The weight matrix for each node is initialized to zero.

When node n is trained on input pattern m, the change in each of the weights is computed as follows:

$$\Delta w^{n}_{(x+\delta),(y+\delta)} = a^{m}_{x,y}\alpha\left(1 - w^{n}_{(x+\delta),(y+\delta)}\right) \qquad (1)$$

The normalizing subsystem weights are adjusted in a similar manner:

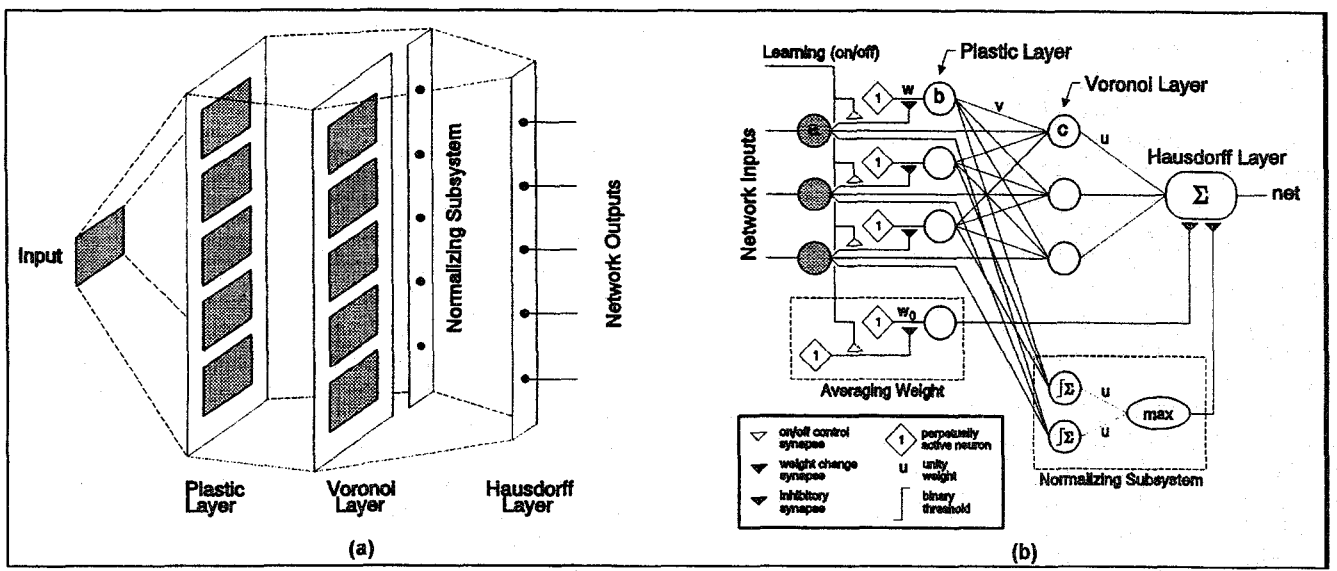

Figure 1. HaVNet architecture (from Rosandich [8])

$$\Delta w_0^m = \alpha\left(1 - w_0^n\right) \tag{2}$$

where a is the input matrix, α is the learning rate, and w is the current value of the weight matrix.

The quantity δ is defined as the span of the Voronoi layer. The use of the Voronoi span causes the weight of the match to decrease as the compared point moves away from the learned point. The weight matrix of the normalizing subsystem serves as an indicator of the extent to which each node has received training.

Once the weight change is computed, the weights are updated as follows:

$$w_{(x+\delta),(y+\delta)}^{n(t+1)} = w_{(x+\delta),(y+\delta)}^{n(t)} + \Delta w_{(x+\delta),(y+\delta)}^{n(t+1)} \tag{3}$$

$$w_0^{n(t+1)} = w_0^{n(t)} + \Delta w_0^{n(t+1)} \tag{4}$$

where t is the number of training iterations.

During the learning phase, each training pattern is presented to the network in sequence, and the appropriate node is trained using the equations above. The learning rate determines the magnitude of the effect that each training pattern has on the trained weights.

2.3. Recognition

When a pattern is presented to the network for recognition, the response of a node n to an input pattern a^m is determined by first computing the output of the plastic layer:

$$b_{(x+i),(y+j)}^n = w_{(x+i),(y+j)}^n a_{x,y}^m \tag{5}$$

where:
- x = 1...X input x dimension
- y = 1...Y input y dimension
- i,j = $-\delta...\delta$ Voronoi layer span
- n = 1...N node number
- w^n = plastic layer weight for node n
- b^n = plastic layer output for node n

Next, the outputs from the Voronoi layer are computed as follows:

$$c_{x,y}^n = \max_j \left\{ \max_i \left\{ v_{i,j}, b_{(x+i),(y+j)}^n \right\} \right\} \tag{6}$$

where:
- v = Voronoi weight matrix
- c^n = Voronoi layer output for node n

The Voronoi weights are calculated by:

$$v_{i,j} = 1 - \frac{\sqrt{i^2 + j^2}}{\delta + 1} \quad -\delta \le i, j \le \delta \tag{7}$$

and are the same for all nodes.

Once the outputs from the Voronoi layer are computed, the response of the Hausdorff (output) layer is computed:

$$net^n = \frac{1}{w_0^n \eta^n} \sum_y \sum_x c_{x,y}^n \tag{8}$$

where η^n is the normalizing quantity for node n. This quantity is calculated by:

$$\eta^n = \max\left\{ p_a^m, p_w^n \right\} \tag{9}$$

where:

$$p_a^m = \sum_y \sum_x \phi(a_{x,y}) \tag{10}$$

$$p_w^n = \sum_y \sum_x \phi(w_{(x+\delta),(y+\delta)}^n) \tag{11}$$

and the function ϕ is the following binary threshold function:

$$\phi(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

2.4. Network Extensions

One of the strengths of the HaVNet architecture is the ability to train different aspects. In the original context, an aspect is defined as a characteristic two-dimensional view of a three-dimensional object. The plastic layer of the HaVNet neural network is expanded to include several aspect representations for each learned object rather than the single two dimensional representation used previously, and the learning process is modified to allow for the self-organization of the aspects. The recognition process of the expanded network is also modified to incorporate the multiple-aspect object representation. The concept of aspects is shown in Figure 2.

In a two-dimensional HaVNet the winner is the class with the highest activation above a given threshold. The final state remains unknown if no class exceeds the threshold. With aspects, this also includes intra-class competition, whereby the first aspect of a class to exceed the threshold is selected to represent the class in the inter-class competition.

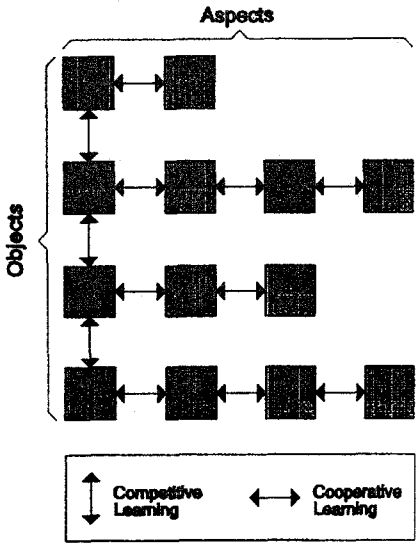

Figure 2. HaVNet aspects (from Rosandich [8])

3. Hybrid system design

The hybrid well test analysis package consists of five modules, each designed to isolate certain operations and provide an easy workflow. The five modules of the system are, in operational order, the 1) data entry module, 2) pre-processing module, 3) model identification module, 4) parameter estimation module, 5) graphical display and output module. A database is used to pass parameters from one module to the next, and maintains a set of state parameters that allow the user to go back to a previous module and make changes without corrupting downstream modules. If the system detects a significant change in a module, the user will be asked to reconfirm all previously defined downstream modules.

3.1. Data entry module

The data entry module is a graphical user interface used to guide the analyst through the entry of the test data and external data. In a well test, the user must provide not only the test data, which generally consists of pressure and flowrate gauge measurements taken form a wellbore, but also external data pertaining to the physical wellbore, the completion of the well to the producing formation, geologic data obtained from other specialized tests, and the physical properties of the produced fluids.

The control for the data entry module is a rule-based system using a forward-chaining inference engine. In general there some minimum amount of required data before the analysis can proceed. The type of test and the nature of the external data dictates the minimum amount required. The rules behind the interface constantly monitor the data being entered to insure that at least the minimum information based on the entries thus far has been entered. The user can stop and save the current data in order to address missing items, but workflow cannot proceed to the next module until there is enough data present to provide a valid analysis. Detailed error messages are presented so that missing items can be easily identified.

There is no attempt to validate the correctness of the external data, only to insure that it is present. Data validity is a major concern in well testing, however, since the test data can only partially validate the external data. These data must be validated by means appropriate to their particular methods of collection. Therefore validity of the external data, aside from a few routine checks, is beyond the scope of the well test analysis system. Validity of the gauge data is addressed in later modules.

3.2. Pre-processing module

The pre-processing module performs a number of mathematical transformations on the gauge data in order to prepare a relatively invariant pattern to the network for recognition. Well test data is highly susceptible to noise, which is divided into two types, systematic and random. Systematic noise is a result of the external data and can be used to help validate the overall analysis. Random noise, however, must be effectively removed. Therefore, the first step in the pre-processing is to apply a high-low bandpass filter over the data to minimize the amount of random noise.

The second step in the pre-processing phase is to carefully filter the systematic data, based on the external data provided. This takes the form of transformations with respect to the time and pressure response to account for fluctuations in the flow rate prior to and during the test. If the flow rate data is very good the entire effect of these variations can be removed and a constant flow rate response is generated. This is desirable since the constant rate response provides a well understood set of solutions and minimizes computational time. In reality, the true constant rate response is rarely achieved, but the uncertainty of the validity of the analysis can generally be greatly improved by performing this transformation.

Next, the transformed pressure versus time data is converted to "derivative" format. This format is used as it enhances the features in the data and makes pattern identification possible. Without transformation all pressure responses look alike.

A great body of work exists on the best way to calculate the derivative, and several method are included which the user can choose. The default transformation is that cited by Horne [1], which uses a numerical differentiation with respect to the logarithm of (transformed) time. A variable differentiation interval is allowed, with the default being the most commonly cited

value of 0.2. Using the defaults, the numerical differentiation becomes:

$$\left(\frac{\delta p}{\delta \ln(t)}\right)_i = \left[\frac{\ln\left(t_i/t_{i-k}\right)\Delta p_{i+j}}{\ln\left(t_{i+j}/t_i\right)\ln\left(t_{i+j}/t_{i-k}\right)} + \frac{\ln\left(t_{i+j}t_{i-k}/t_i^2\right)\Delta p_i}{\ln\left(t_{i+j}/t_i\right)\ln\left(t_i/t_{i-k}\right)} - \frac{\ln\left(t_{i+j}/t_i\right)\Delta p_{i-k}}{\ln\left(t_i/t_{i-k}\right)\ln\left(t_{i+j}/t_{i-k}\right)}\right] \quad (13)$$

subject to: $\ln(t_{i+j}) - \ln(t_i) \geq 0.2$
$\ln(t_i) - \ln(t_{i-k}) \geq 0.2$

In well testing the derivative is always positive. Any negative derivatives trigger the default interval to enlarge by one data point in each direction until a positive number is obtained.

The fourth step in the pre-processing module is feature extraction. This is accomplished by fitting a small series of splines through the derivative data and then applying the split and merge method to identify the features. This combines the spline approach used by Stewart and Du [9] and the split and merge method incorporated by Al-Kaabi *et al* [2].

The features extracted from the data consist of a series of slopes of the derivative data. These can be positive or negative in sign. The slopes are then transformed into a 9 x 9 matrix with the order of occurrence of the feature being the x axis and the group to which the slope value belongs being the y axis. This representation is the input matrix presented to the HaVNet neural network.

3.3. Model identification module

The model identification module contains a modified HaVNet neural network and a rule-based system. When an input pattern is selected for recognition the HaVNet neural network evaluates the pattern with respect to its trained classes and aspects. As a single class is tested, the HaVNet performs as originally designed, testing each aspect until a threshold value is met. This threshold value can be set by the user and defaults to 0.9. If an aspect meets this threshold, processing proceeds to the next class. Once all classes have been evaluated, the winner is chosen. If none of the classes meet the threshold activation then there is no winner. This provides the "I don't know" feature necessary for well testing but lacking in many neural network applications. In addition, instead of returning only a single class, all classes which meet the threshold are returned as a vector. This is to allow the external data to be used to differentiate between highly similar models. The classes represent general model types. The aspects represent different sets of inner and outer boundary conditions. A new modification to the HaVNet architecture provides for only a window of the trained matrix to be analyzed. This allows for the time variance and prevents a null winner simply because the test did not run long enough. The input matrix can be up to 9 x 9, but due to the number of features actually present in the data the x dimension may be less than the maximum. Also, another layer of aspects is added to allow for cases where different sets of parameter values yield highly dissimilar patterns for any single model/boundary combination.

Another modification to the HaVNet architecture allows for multiple winners in the competition between classes. In this implementation, all classes and aspects exceeding the threshold are retained. These are then examined by a rule-based system which examines the external data to establish the validity of the class/aspect combination. When rules are fired the activation of the corresponding class or aspect is adjusted depending on the conclusion of the rule. Limiting the possible candidates minimizes the run time compared to previous expert system approaches.

Finally, the winner is chosen as that class/aspect from those passed from the HaVNet that has the highest activation. The user also has the facility to force the system to accept a non-winner if that is desired, such as for what-if type studies.

3.3.1. Network training. The training mode is accomplished by adding a new model to the model file and any new rules to the rule file. The system will then automatically recompile the appropriate executables to include the new entries. When a new model is entered the user will be prompted to minimum and maximum expected values for the parameters in the model. The system then generates a large number (default is 10,000) of simulated tests using a uniform distribution of the parameter data. These simulations are passed through the HaVNet in training mode and added to the trained patterns as needed to insure that each simulated testing pattern generates an activation above the threshold for the appropriate first-level aspect.

One of the features of the HaVNet architecture that is fully exploited in this system is the ability to learn in one iteration. By using a learning rate of 1.0 the HaVNet then the trained weights also become a binary matrix. Although this can result in a very large number of trained solutions being held in storage, the very small size and binary nature of the matrices provides for excellent execution speed in both training and runtime modes. The

structure of the model identification module is illustrated in Figure 3.

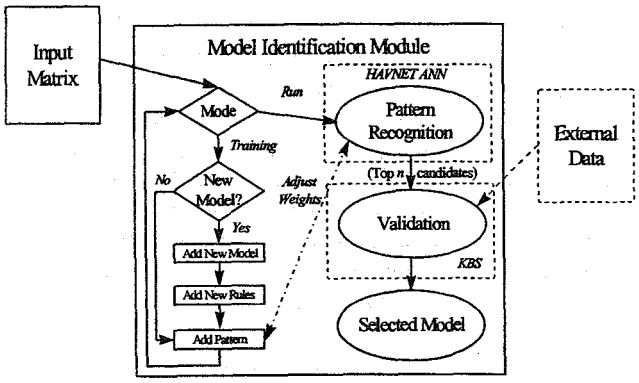

Figure 3. Model identification module design

3.4. Parameter estimation module

The parameter estimation module performs non-linear regression using the modified least absolute method. The first step in this module is to sample the data and reduce the number of point for the regression. The next step uses the HaVNet input matrix and the external data to identify the various flow regimes associated with the test. A rule-based system performs this task. The regimes are used to further validate the model selection.

The third step involves determining suitable initial guess values for the non-linear regression. A rule-based system is employed that examines the gauge data and flow regimes to determine the best initial guess for each parameter. Traditional straight-line methods are used to actually calculate the estimates using specialized plots.

The initial guess values and the gauge data are passed to the regression routine and final values are calculated. Confidence limits are calculated for each of the parameters.

The parameters and model are then cross-validated by generating a simulated test. This simulation is then passed through the system to that it produces the same result as the real data.

3.5. Graphical output module

The graphical output module provides a comfortable environment for the user to examine the results. The results are presented in various traditional well test forms, depending on which model is selected. A full range of conventional diagnostic plots is available. These plots show the original data, the simulation based on the final regressed data, and conventional straight-line methods using the final parameters. Specialized plots traditionally used for identification of specific features are also provided when appropriate.

4.0. Conclusions

The network described in this paper offers improved training and run time compared to single-architecture approaches. The network integrates both test and external to overcome the weaknesses inherent in purely numerical methods. The non-uniqueness problem is effectively handled by exploiting the strengths of both neural network and expert system architectures. While more complex than a single architecture, the modular design allows for efficient maintenance. Also, the network modifications allow for inconclusive results in the neural network.

5.0. References

[1] R.N. Horne, *Modern Well Test Analysis*, 2nd Edition, Petroway, Inc., Palo Alto, CA, 1995, pp. 1, 79-82.

[2] A.U. Al-Kaabi, D.A. McVay, and W.J. Lee, "Using an Expert System to Identify the Well Test Interpretation Model", *J. Petroleum Tech*, Mar 1990, pp. 654.

[3] A.U. Al-Kaabi and W.J. Lee, "Using Artificial Neural Networks to Identify the Well Test Interpretation Model", *Proc. 5th SPE Petroleum Computer Conf.*, Soc. of Petroleum Eng., Denver, 1990, pp. 71.

[4] O.F. Allain and O.P. Houzé, "A Practical Artificial Intelligence Application in Well Test Interpretation", *Proc. SPE European Computer Conf.*, Soc. of Petroleum Eng., Stavanger, Norway, 1992, pp. 245.

[5] T. Anraku and R.N. Horne, "Discrimination Between Reservoir Models in Well Test Analysis," *Proc. 68th Ann. Technical Conf. and Exhibition*, Soc. of Petroleum Eng., Houston, TX, 1993, pp. 113.

[6] A.O. Kumoluyi, T.S. Daltaban and J.S. Archer, "Identification of Well Test Models Using Higher-Order Neural Networks", *Proc. SPE European Petroleum Computer Conf.*, Soc. of Petroleum Eng., Aberdeen, U.K., 1994, pp. 185.

[7] E.A. May and C.H. Dagli, "Identification of the Well Test Interpretation Model Using the Extended HaVNet Neural Network", *Intelligent Eng. Systems Through Artificial Neural Networks*, Vol. 6, C.H. Dagli et al, ed., ASME Press, 1996, pp. 467-472

[8] R.G. Rosandich, "Artificial Vision: Three Dimensional Object Recognition Using Neural Networks," doctoral dissertation, Univ. of Missouri-Rolla, Dept. of Eng. Management, 1994.

[9] G. Stewart and K.-F. Du, "Feature Selection and Extraction for Well Test Interpretation by an Artificial Intelligence Approach", *Proc. SPE Int'l. Meeting on Petroleum Eng.*, Beijing, 1992, pp. 401.