DOE/BC/14894--7

APPLICATION OF ARTIFICIAL INTELLIGENCE TO RESERVOIR CHARACTERIZATION: AN INTERDISCIPLINARY APPROACH

(DOE Contract No. DE-AC22-93BC14894)

Submitted by

The University of Tulsa
Tulsa, OK 74104

Contract Date:	October 1, 1993
Anticipated Completion Date:	September 30, 1996
Government Award:	$240,540
Program Manager:	R.F. Gamble
Principal Investigators:	B.G. Kelkar
	D.R. Kerr
	L.G. Thompson
	S. Shenoi
Reporting Period:	January 1 - March 31, 1995

Contracting Officer's Representative

Mr. Robert E. Lemmon
Pittsburgh Energy Technology Center
P.O. Box 10940, M/S 141-L
Pittsburgh, PA 15236-0940

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

Objectives

This basis of this research is to apply novel techniques from Artificial Intelligence and Expert Systems in capturing, integrating and articulating key knowledge from geology, geostatistics, and petroleum engineering to develop accurate descriptions of petroleum reservoirs. The ultimate goal is to design and implement a single powerful expert system for use by small producers and independents to efficiently exploit reservoirs.

The main challenge of the proposed research is to automate the generation of detailed reservoir descriptions honoring all the available "soft" and "hard" data that ranges from qualitative and semi-quantitative geological interpretations to numeric data obtained from cores, well tests, well logs and production statistics. In this sense, the proposed research project is truly multi-disciplinary. It involves significant amount of information exchange between researchers in geology, geostatistics, and petroleum engineering. Computer science (and artificial intelligence) provides the means to effectively acquire, integrate and automate the key expertise in the various disciplines in a reservoir characterization expert system. Additional challenges are the verification and validation of the expert system, since much of the interpretation of the experts is based on extended experience in reservoir characterization.

The overall project plan to design the system to create integrated reservoir descriptions begins by initially developing an AI-based methodology for producing large-scale reservoir descriptions generated interactively from geology and well test data. Parallel to this task is a second task that develops an AI-based methodology that uses facies-biased information to generate small-scale descriptions of reservoir properties such as permeability and porosity. The third task involves consolidation and integration of the large-scale and small-scale methodologies to produce reservoir descriptions honoring all the available data. The final task will be technology transfer. With this plan, we have carefully allocated and sequenced the activities involved in each of the tasks to promote concurrent progress towards the research objectives. Moreover, the project duties are divided among the faculty member participants. Graduate students will work in teams with faculty members.

The results of the integration are not merely limited to obtaining better characterizations of individual reservoirs. They have the potential to significantly impact and advance the discipline of reservoir characterization itself.

Summary of Technical Progress

1. Decomposition of System

We have decomposed the overall system development into smaller component parts to allow us to focus on the expert knowledge required for that component. In addition, the decomposition will facilitate the implementation of the system and its validation and verification. The three component systems will be representative of how each of the experts in geology, geostatistics, and engineering characterizes the reservoir. Figure 1 describes a model for this breakdown. The concurrent development of these component systems fits into the development of the large and small scale aspects of the system as originally stated in the proposal. In Figure 1, each component system in the model is depicted as interfacing (through the bi-directional links) with a central repository of reservoir descriptions. Though, portions of these description will essentially be passed from component to component as more information is gathered (as shown by the bi-directional links in Figure 2), the model of a central repository is an accurate account of how the components are integrated, i.e., the final descriptions in the repository are consistent will all of the information given by the components systems. This system model allows us to develop the system using an Artificial Intelligence technique called a *blackboard system*, in which information is centrally located, i.e., on a blackboard, and experts take their turn to update, change, and correct the information on the blackboard.

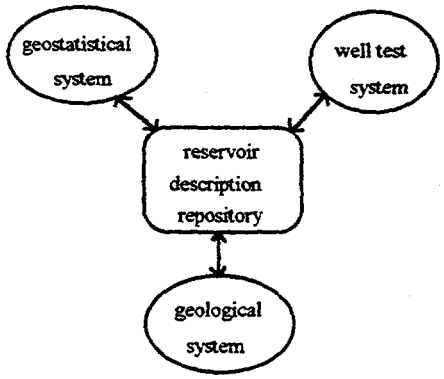


Figure 1: Expert System Decomposition

The entire system is now under development using the C/C++ programming language. Changes were made to this platform from the initial Kappa-PC development environment because Kappa-PC could not interface with C and Fortran code as it advertised. Kappa-PC code that has currently be developed is being ported to the C/C++ environment. In this report, we discuss the new development platform and the building of a customized expert system inference engine. We update the status of the geostatistical and geological systems. Finally, we introduce the concept of *wavelets* that we will believe

may be useful in integrating all of the systems into one. Both the well interpretation system and the correlation system await redevelopment in the new platform.

2. Customized Inference Engine

Rules are an essential part of an expert system and are being used in different parts of the project. This includes the *Well test Interpretation System* and the *Marker bed Identification System*. These rules are presently coded in KAPPA-PC and use the features provided by KAPPA for functions like asserting facts and chaining. Porting the code from KAPPA-PC to C++ requires building these functions to be used in a C++ program to create and use these rules as in KAPPA-PC. A major component of a rule-based system is the inference engine. The inference engine takes care of the chaining process. This functionality is provided by KAPPA, which provides a convenient way use the inference engine with rules coded in KAPPA-PC. An inference engine was required along the same lines that could provide the same functionality as that in KAPPA, and could be used in a C program. It was decided to develop an inference engine in C++ that was best suited for the problem and provided the functionality required to implement the expert systems in this project. The following sections present the details of the inference engine developed.

2.1 Inference Engine Features

The inference engine was to be developed in C++ in the form of a library function that can be called from any C program. The following features were required:

1. Representation of rules: The user was to be given a fixed representation to be used to write rules. The rules consist of a condition and an action part. The structure of the rule was modeled along the lines of OPS-5 rules, which is a popular development environment for rule-based systems. Use of free variables (variables used in rules for the purpose of binding values) in the rules was also required. Another feature required in the rule-base was the ability to assign priorities to rules. This value is used to decide which rule fires first in case multiple rules are selected.
2. Assertion of facts: The user needs to assert known facts before the chaining process. Functions were required for this purpose.
3. Forward chaining: The forward chaining process starts with the known facts and selects rules to be fired successively till the goal is reached, or no more rules satisfy the known facts.
4. Backward chaining: The backward chaining process starts with a goal, and finds subgoals that need to be satisfied. The subgoals act as goals and the process is repeated to see if the subgoals match the known facts.

The rule representation and other components of the inference engine, including assertion of facts, are discussed in detail in the following sections. The forward chaining process has been implemented in C++. Forward chaining in its simplest form is an interactive program that performs a loop of substitution. It steps through the rule list until

it finds a rule in which premises match the fact or situation. The rule will then be used of "fired" to assert a new fact. The fact concluded as the result of that rule's firing is added to the knowledge base. Once a rule fires, it is prevented from firing again to avoid repeated firing of the same rule. This cycle of finding a matched rule, firing it, and adding the conclusion to the knowledge base is repeated until no more matched rules can be found. The following steps make the forward chaining process:

1. A fact is asserted.
2. The fact matches the premise of the rule.
3. The system computes the substitution that unifies the fact and the premise.
4. The substitution is applied to the conclusion of the rule.
5. This result is asserted and is available for further forward chaining.
6. Steps 1 through 5 are repeated.

As shown by the above steps, the implementation of the forward chaining process involves three elements, **unify**, **substitute**, and **stash**. The purpose of unify is to return a substitution that will make the fact unify with the premise in the rule, i.e., will test the compatibility of the fact and the premise. Unify can by designed in a narrow sense that exactly matches the fact and the premise. The use of free variables in the rules requires unify to be designed in a way that allows substitution of parts of the fact and the premise to make the fact and the premise compatible. For example, consider the following unification:

$$\text{Unify}((\text{well1.depth } ?x)\ (\text{well1.depth } 300)) \Rightarrow (?x\ 300)$$

The above two statements can be unified by setting variable *?x* to 300. This approach is used in the inference engine designed to unify the premise of a rule with the working memory. Once a variable is bound to a value, this value is used wherever the variable is used in the rest of the rule. *Substitute* is used to compute the substitution that unifies the fact and the premise; it performs a variable substitution on the proposition, as shown in the previous example. *Stash* is needed to incorporate the proposition (conclusion) into the knowledge base. The rule representation does not allow disjunction in the consequent of the rules. The literals in the consequent of the rules are thus in conjunction, and this requires all the literals to be asserted for further chaining.

Backward chaining reasoning is used when the user makes a query as to whether a certain fact is true and when there is a rule that can determine the query from known information in the knowledge base of from answers given by the user. In other words, backward chaining attempts to prove the hypothesis from the facts. The steps involved backward chaining are as follows:

1. A request is made to achieve a fact (the goal).
2. The goal does not match any known fact.
3. The goal matches the conclusion of a rule.
4. The system computes the substitution that unifies the goal with the conclusion.
5. The substitution is applied to the premise of the rule.
6. This result becomes a new goal of the system.

7. This new goal can do the following:
   match a fact in the knowledge base
   match a conclusion of a rule, leading to further backward chaining
   ask the user for the needed information
   fail, in which case the original goal fails
8. Steps 1 through 7 are repeated.

Forward chaining has been implemented this chaining procedure is used in most parts of the project. Backward chaining process is still to be implemented.

2.2 Inference Engine Components

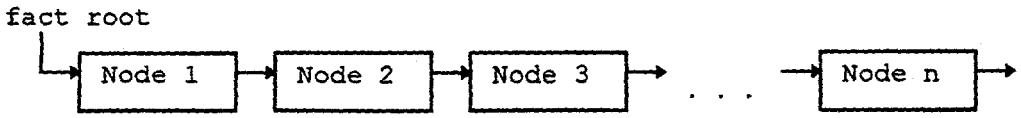
The *working memory* constitutes the facts known, and this information is used during the chaining process. The premise of a rule is matched with the working memory. The rule fires if it matches directly or with unification. Once a rule fires, the working memory is updated by the facts asserted by the conclusion of the rule. The working memory thus represent the state of the system at any time. In our implementation, the working memory if implemented as a fact base, and is discussed in detail in this section. The rule base is a collection of all the rules, and these used in the chaining process.

Rule base: The rule base consists of all the rules defined by the user. The user defines the rules in a specific format. The rule is parsed to create an internal representation. It is represented as a linked list. Rule parsing and the internal representation of the rules is discussed in more detail elsewhere in the report.

Fact base: The fact base is the collection of variables used in the rules and their values. The user is required to declare all the variables used in the rules. These variables, along with the values (if known), is stored in the fact base. This fact base also acts as the working memory of the system. The fact base is implemented as a linked list for fast access.

```
name  : string
value : string
type  : character
ptr   : pointer
next  : pointer
```

A single node in the fact base



Fact base represented as a linked list of nodes

A single node in the linked list that makes up the fact base consists of the following elements. The *name* field is a character string and holds the name of the fact asserted. This can either be a variable used in the program, or an object/slot pair. The *value* field holds the value of the variable. This is also represented as a character string to maintain consistency between types and easier comparisons. The character string representing the value is later converted to the appropriate types at the end of the chaining process. To identify the correct type of the variable, the *type* field is used. This is a single character denoting the type of the variable. *Ptr* is a pointer to the actual location in memory where the variable is stored. This information is required to update the variable at the end of the chaining process. Without the actual address of the variable, it was not possible to update the variable. The *next* field is a pointer to the next node in the linked list.

All the above fields, except *next*, are specified by the user before the chaining process. All variables needed in the chaining process are entered in the fact base. Variables that are used in the chaining process without being asserted in the beginning are left with a NULL value in the *value* field. The functions developed in C++ to handle the rule base, fact base, and their usage is discussed in more detail in the following section.

2.3 Rule Structure

Two primary components of a rule are the premise and the conclusion. The premise is the condition under which the rule is said to "fire". It is a proposition that is not assumed to be true given an unknown environment, whose truth value will be occasionally evaluated. The conclusion of the rule is the result of firing the rule. The firing of the rule frequently affects the knowledge base through the addition or modification of facts. The rule premise and action can contain conjunction (**and**) and disjunction (**or**) phrases. Presently only conjunctions are allowed in the premise and the action. Free variables are allowed in the right-hand side of the literals in the premise and the actions. The following is the grammar used for the rules:

```
rule := IF antecedent THEN consequent
antecedent := literal | literal logic_op literal
literal := variable predicate var_val
var_val := free_variable | value

consequent := literal | literal logic_op literal
literal := variable = var_val
var_val := free_variable | value

logic_op := AND                                  (to be extended)
predicate := == | != | < | > | <= | >=
free_variable := ?variable
variable := identifier
value := alnum (alnum)*   (value is represented as a string)
                          (alnum is alphanumeric characters)
```

The above grammar is being extended to allow disjunctions in the premise. It is also being extended to allow a less restrictive use of free variables, and allow function calls in the premise and action of the rule. The functions developed to write rules is discussed in the following sections.

2.4 Inference Engine Functions

This section presents the C++ functions developed for different parts of the inference engine, and their usage. The following sections also represent the steps to be followed to create a rule base and run the inference procedure.

2.4.1. Writing rules

The first step in the process is to create rules for the rule base. The *WriteRule* function has been developed for this purpose. The function prototype is as follows:

```
WriteRule(char *rule, int priority
```

The argument for this function is a character string that represents the rule. The rule consists of an antecedent and a consequent. Presently it allows only conjunctions in the antecedent and the consequent, and this is being extended. The antecedent and the consequent may consist of any number of literals connected by conjunctions. The second argument indicated the priority associated with the rule. This value is used in selecting rules in case of a conflict. The following example shows how this function is used.

```
WriteRule("IF well1.group == vertically_fractured AND well1.porosity == single
          THEN well1.group = vertically_fractured_single_porosity_system");
```

Free variables may be used in the rules as shown by the following example:

WriteRule("IF well1.depth == ?x AND well2.depth == ?x THEN print(equal depth wells)");

The rules written using the above function are parsed and an internal representation of the rules is created. This is discussed in more detail later in this report.

2.4.2. Assertion of facts

Facts need to be asserted before the chaining process. These are the facts that are used by the inference engine to match rules. As rules are matched and fired, the consequents of those rules are used to assert other facts. The way the inference engine is implemented, the user if required to declare all the variables used in the rule base. This is done to maintain a reference to the memory location of the variables, to be used to update facts at the end of the chaining process. The function *assert* is used to assert variables. The variables that do not need to be asserted at the beginning of the chaining process are still asserted, but their value field is left empty. This provides the inference engine with a reference to the variable without specifying any value. The function prototype is as follows:

Assert(char *name, void *ptr, char type, TYPE value);

The first argument is the name of the variable. *Ptr* is the pointer to the memory location of the variable. *Type* represents the type of the variable. The final argument if the value of the variable. This argument can be of any type since this function is implemented as an overloaded function. Specifying a value as the last argument asserts the variable with that value. Keeping this field empty would just add the variable in the fact base without asserting any value. All variables are entered into the fact base whose internal representation is shown in the previous section.

2.4.3. Chaining procedure

Once all the rules are written and the necessary facts asserted, the chaining function may be called. The function for the forward chaining process is *ForwardChain()*. This function does not take any arguments and uses the existing fact base and the rule base. Rules are selected by matching the conditions of the rules with the fact base (i.e. working memory), and using the rule priority if required. Rules are fired by executing the consequents and asserting the corresponding facts.

The forward chaining process continues till no more rules match the working memory. At this point the chaining process stops, and the fact base is updated. The new values that result from the chaining process are updated in the corresponding memory location. The forward chaining process is to be extended to allow goal directed chaining. The code is also being extended to handle backward chaining.

2.5 Conclusion

Porting code from KAPPA-PC to C/C++ required us to convert the rules written in KAPPA-PC to be converted to C. Also the unavailability of an inference engine required us to develop our own code in C++ to suit the purpose. Code for the inference engine was thus developed in C++. This has been written in the form of C library functions and can be called from any C program by including the required headers and linking the library. It presently provides the basic functionality required for our problem. This code is to be extended to handle more complicated features like goal directed chaining, backward chaining, and more sophisticated conflict resolution schemes. Code written for the expert systems in KAPPA-PC is presently being ported to C++.

3. Geostatistical System

The work undertaken during this period focused on the flow simulation part of the simulated annealing (SA) algorithm. As mentioned before, the inclusion of flow simulation has a severe impact on the execution time of the algorithm -- even when the LTFD (Laplace Transform Finite Difference)[1] methodology is used. The areas of investigation included an analysis of the algorithm, a comparison of matrix solvers, an analysis of one of the default parameters used, upscaling and the two-scale approach. Below are further details of this work.

3.1. Analysis of Flow Simulation Algorithm

It was previously reported that the results from the Laplace Transform Finite Difference (LTFD) simulator showed good agreement when compared with the results from the ECLIPSE-100 (ECL)[2] simulator. While this is true in a general sense, we found that when multiple production rates are used, the pressure responses from the LTFD simulator are "smoother" than the ECL results at the times when the rates are changed. This was determined to be caused by using the Laplace transform. Even though we attempted to correct for the variable rates, our modification could not completely correct these effects. However, recent work[3] has resulted in a possible solution to this problem. The ability to include the use of this solution in our work is being studied.

3.2. Matrix Solvers

We considered matrix solvers from the SLAP,[4] ITPACK[5] and Templates[6] packages. Some solvers hold promise of faster execution times. However, because of the specialized nature of those approaches e.g. red-black ordering, they have not yet been tested. For example, red-black ordering implies parallelization of the code -- something which we are not considering at present. From ITPACK the solvers considered included the Reduced System Conjugate Gradient (RSCG) and the Symmetric Successive Overrelaxation Conjugate Gradient (SSORCG). We have not implemented the Templates code because of time constraints and the apparent lack of benefit to be derived from

pursuing it. The table below summarizes the results obtained with the packages tested. A 224-block dataset was used for the testing.

Package	Method	RunTime (CPU Sec)
ITPACK	Jacobi Conjugate Gradient (JCG)	14096
	Symmetric Successive Overrelaxation Semi-Iteration (SSORSI)	44084
SLAP	Conjugate Gradient (CG)	13824
	Generalized Minimum Residual (GMRES)	22045
	Bi-Conjugate Gradient Squared (BCGS)	14283

Based on the above results, we have selected the SLAP CG code for use. Discussions with a professor in the Mathematics Department have also suggested that CG may be the method of choice for our problem. Further, he has suggested that the usage of a symmetric storage format for the coefficient matrix may not result in significant gains in the matrix solution time but the storage requirements are reduced. Of course, further efficiencies may be realized by considering the specialized storage systems alluded to above.

3.3. Default Parameter: NVAL

NVAL is the parameter used to represent the number of pressure values per well used for matching in the flow simulation part of the objective function of the simulated annealing algorithm. This is defined in the following equation:

$$\text{Flow Simulation Part for Iteration, k} = wgt2 * \frac{1}{E_2^0} \sqrt{\sum_{i=1}^{NDIFF} \left[\frac{L\left[\Delta p(i)\right]_s^k}{L\left[\Delta p(i)\right]_0} - 1 \right]^2}$$

where

wgt2 =weighting applied to the flow simulation part

$$E_2^0 = \sqrt{\sum_{i=1}^{NDIFF}\left[\frac{L\left[\Delta p(i)\right]_s^0}{L\left[\Delta p(i)\right]_0} - 1\right]^2} = \text{Initial Flow Simulation Energy}$$

NVAL =Number of summation terms per well

NWELL =Number of wells

NDIFF =*NWELL*NVAL*=number of terms used in the summation

$L[\Delta p]$ = the Laplace transform of the change in the flowing bottomhole pressure

To date, ten values were being used. Because the run time of one iteration in the algorithm is directly proportional to the value of NVAL, we decided to analyze the effect of the value of NVAL on the performance of the code. Values of NVAL ranging from five to twenty were used. The observations so far are:

- The lower the value of NVAL the shorter the cycle time, but, in some cases, more cycles are needed for convergence. This suggests that there is/are optimum values for NVAL. It seems that this value is about six or seven. However, a word of caution is that these results are with just one dataset so far.
- The relative "goodness of fit" of the distributions obtained using the different NVAL values is difficult to determine by visual inspection (i.e. whether one description is better than another) and, since we are still researching an appropriate *objective* measure of goodness of fit, has not yet been determined.

3.4. Upscaling

It was previously stated that the production behavior of upscaled grids agreed well with the production behavior of the "true" grid scale. This is true in the sense that the upscaled responses "tracked" the true-scale responses with a small -- but discernible -- difference. A more detailed analysis was undertaken in which the relative errors defined as:

$$\text{Relative Percentage Error} = \frac{\text{"true" scale } p_{wf} - \text{upscaled } p_{wf}}{\text{"true" scale } p_{wf}} x\ 100$$

It was found that at very early time, the errors were quite large -- of the order of 40% at a time of about 1.5 minutes, but decreasing to about 10% at a production time of about 0.1 day. However, the error did not decrease much below 10% except at very late times. These results were found to be about the same for both the geometric average and the renormalization upscaling methods. In view of the tolerance required of the objective function of the SA algorithm, such errors are not acceptable. We are currently

considering various options -- based on the suggested reasons for the differences -- for obtaining better matching between the true and the upscaled grids. Two of these are:

- local grid refinement[8]
- consideration of the flow geometry in grid design; this implies using cylindrical or radial geometry in the near-well regions.[9]

3.5. Two-Scale Approach

As we have repeatedly noted, the most computationally-expensive part of the simulated annealing algorithm (in which a two-part -- variogram and flow simulation -- objective function is considered) is the flow simulation. The larger the number of grid blocks used -- or the finer the scale of gridding -- the longer the flow simulation takes. The idea therefore, was to "speed up" the execution of the algorithm by performing the flow simulation at a coarser scale while we undertake the variogram analysis at the finer scale. As noted in the previous section, upscaling is the major problem to be solved to be able to use this approach. While the upscaled distribution "resembles" the true distribution, matching the flowing bottomhole pressures obtained from flow simulation at the different scales is the problem (two upscaling factors were used: 9 -> 1 and 25 -> 1). The results however have been encouraging, in that the descriptions obtained are similar to the true image, but still somewhat "diffuse". It is anticipated that correction of some of the problems we face may improve the description match we get.

4. Geological System: Marker Bed Identification

In this section, we update the progress on the geological model with respect to marker bed identification. The log facies description system and the correlation system are currently be evaluated using more complex examples. In addition, these systems need to be ported to the new development platform.

4.1 Introduction

In order to identify log facies and correlate stratigraphic units within wells across a reservoir, a reference point must be available that identifies the interval of observation for such analysis to take place. Marker bed identification provides the beginning and ending interval depths for this analysis.

A marker bed is a specific unit of formation that is widely distributed and laterally stable across an area. Marker beds can be traced universally between different continents, regionally across a whole basin, locally in a field-scale area, or for a very limited area of interest of some formation interval. The scope of study extends to marker bed identification in a local field-scale area.

The focus of our study is to identify the main marker beds that are common across all the wells within a field and their corresponding beginning and ending depths. While a marker bed can be easily identified in the logging curves by the naked eye of an expert, the prototypical expert system requires the processes, rules, and experiences be captured in order to arrive at the same conclusions as the expert.

4.2 The Approach

In order to identify marker beds across a field, we have limited our domain to a field consisting of three wells. Our three wells are dispersed across the field. Two wells lie at either ends of the field, and the third well lies in-between the first two wells.

Our approach to identifying marker beds involves two steps. These two steps were described in detail in the last quarterly report. To briefly summarize these steps, we first identify all the potentials marker beds in a single well. This is determined by applying a set of heuristics that characterize marker beds to the gamma ray, resistivity, and sp logs of a single well. These heuristics were derived by experts in the field and are available in Figure 1. Second, we apply a cross-correlation algorithm to the gamma ray or resistivity log of one well with the gamma ray or resistivity log of the potential marker bed of another well. Positions of high correlation between these two logs should show areas where the real marker beds lie across these two wells.

Figure 1: Heuristics applied to identify a potential maker bed

Top 10% gamma ray, top 10% spontaneous potential, and low 10% resistivity
Top 10% gamma ray, top 10% spontaneous potential, and low 30% resistivity
Top 10% gamma ray, top 10% spontaneous potential, and low 50% resistivity
Top 10% gamma ray, top 30% spontaneous potential, and low 10% resistivity
Top 10% gamma ray, top 30% spontaneous potential, and low 30% resistivity
Top 10% gamma ray, top 30% spontaneous potential, and low 50% resistivity
Top 10% gamma ray, top 50% spontaneous potential, and low 10% resistivity
Top 10% gamma ray, top 50% spontaneous potential, and low 30% resistivity
Top 10% gamma ray, top 50% spontaneous potential, and low 50% resistivity
Top 30% gamma ray, top 10% spontaneous potential, and low 10% resistivity
Top 30% gamma ray, top 10% spontaneous potential, and low 30% resistivity
Top 30% gamma ray, top 10% spontaneous potential, and low 50% resistivity
Top 30% gamma ray, top 30% spontaneous potential, and low 10% resistivity
Top 30% gamma ray, top 30% spontaneous potential, and low 30% resistivity
Top 30% gamma ray, top 30% spontaneous potential, and low 50% resistivity
Top 30% gamma ray, top 50% spontaneous potential, and low 10% resistivity
Top 30% gamma ray, top 50% spontaneous potential, and low 30% resistivity
Top 30% gamma ray, top 50% spontaneous potential, and low 50% resistivity

4.3 Results

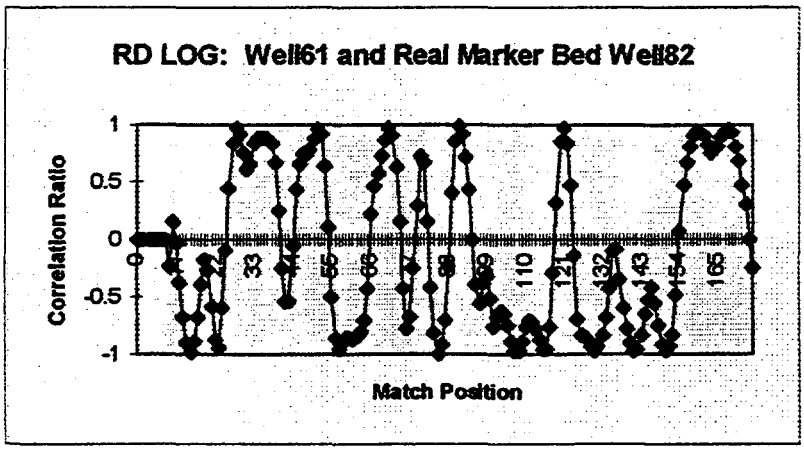
The results of executing the first step of our approach, identifying the potential marker beds from a set of heuristics, was successful. Applying the heuristics resulted in identifying as series of potential marker beds which included among them the main marker beds of that particular well. Using these potential marker beds and their corresponding depths, we then applied the cross-correlation algorithm. The output of the cross-correlation algorithm are correlograms. High correlations on these correlograms indicate instances where the main marker beds lie in the two wells being compared.

To give an example of this cross-correlation technique, assume we are comparing only two wells in a field. The cross-correlation algorithm was applied to the entire gamma ray log of the first well and a potential marker bed of the second well. This process was repeated for the gamma log of the first well and the gamma ray logs of all the potential marker beds of the second well. This process was also repeated for the resistivity logs.

Figure 2 shows a resulting correlogram from applying the cross-correlation algorithm to a resistivity log of Well61 and a potential marker bed (which happens to be the main marker bed) of Well82. Note that the match positions along the x-axis contain a set of corresponding depth intervals created in a separate file. High correlations in several points of the correlogram indicate that the real marker bed of Well61 can lie at any of those match positions or depth intervals where the high ratios exist.

Results of the correlograms, as the one in Figure 2, indicate that a series of high correlations exist between both the gamma ray and resistivity logs of the well and potential marker bed compared. Since the depth intervals under consideration of our wells only contain two main marker beds, the correlogram should indicate roughly two sets of depth intervals with high correlation ratios. On the contrary, our results show more than two sets of depth intervals with high correlation values. As a result, our current approach to identifying the main marker beds across a set of wells is not sufficient.

Figure 2: An example correlogram



4.4 Conclusion

The results of our approach indicate that it alone will not suffice in identifying the main marker beds. Our approach does reduce the search of main marker beds from a series of wells by identifying the potential marker beds within a single well. While our approach will identify potential marker beds within a single well, it still needs to be modified to identify the main marker beds across a set of wells. The experts are currently considering extensions to this approach as well as other alternatives.

4.5 Future Work

Several alternative courses have been considered for the marker bed identification problem. For example, given a gamma ray log, experts are considering calculating derivatives to determine areas of sudden changes in slope. Such sudden changes in slope indicate areas where main marker beds may lie. By expanding our current approach and incorporating other main marker bed characteristics such as sudden slope change, we hope to narrow down the margins of which marker beds may be the main marker beds across a series of wells. Currently our research group is examining a set of characteristics similar to the one mentioned above to incorporate into our marker identification approach.

5. Wavelets

There are many different kinds of wavelets. One can choose between smooth wavelets, compactly supported wavelets, wavelets with simple mathematical expressions, wavelets with simple associated filters, etc. The most simple is the Haar wavelet. Wavelets have many desirable properties which include linearity, completeness, unitary, appropriateness to problem at hand, no redundancy or at least "useful" redundancy, and they utilize fast algorithms.

Unlike sines and cosines, which define a unique Fourier transform, there is not one single unique set of wavelets; in fact, there are infinitely many possible sets. One of the distinctions between the different sets of wavelets are the different trade-offs between how compactly they are localized in space and how smooth they are. Wavelets have compact support because they are localized in time which makes them zero outside of a certain interval.

The Discrete Wavelet Transform (DWT) consists of applying a wavelet coefficient matrix hierarchically, first to the full data vector of length N, then to the "smooth" vector of length $N/2$, then to the "smooth-smooth" vector of length $N/4$, and so on until only a trivial number of "smooth-...-smooth" components (usually 2) remain[17]. This procedure is sometimes called a pyramidal algorithm. The output of the DWT consists of these remaining components and all the "detail" components that were accumulated along the way.

A wavelet transform is essentially a family of orthogonal functions that separates a function or a signal into distinct frequency packets that are very localized in the spatial

domain. In other words, a projection of a function or a discrete data set onto a wavelet space tells us how the function behaves in a certain scale of measurement at any point in space. Wavelet transform is a multiresolution framework and, thus, it is well suited for analyzing non-stationary data. Because wavelets have compact support, it is easy to apply this transform to large data sets with limited computer resources[15].

5.1 Fourier Versus Wavelet

The Fourier transform works under the assumption that the original time-domain function is periodic in nature. As a result, the Fourier transform has difficulty with functions having transient components, that is, components localized in time. This is especially apparent when a signal has sharp transitions. Another problem is that the Fourier transform of a signal does not convey any information pertaining to translation of the signal in time. Applications that use the Fourier transform often work around the first problem by windowing the input data so that the sampled values converge to 0 at the endpoints. Attempts to solve the second problem, such as the development of the short-time Fourier transform have met with marginal success. The computational complexity of the Fast Fourier Transform (FFT) is $O(n \log_2(n))$. For the fast wavelet transform (FWT), the computational complexity goes down to $O(n)$.

5.2 Compression

The data being used in this project is constructed by using an exponential covariance function with no sine or cosine components. This data is then run through a flow simulator and specific characteristics are found. Once the data has been compressed using a wavelet transform, then it will be uncompressed and sent back through the flow simulator. If the uncompressed data demonstrates enough of the same characteristics as the original data set, then the wavelet compression was successful and be used in a practical sense.

A general two dimensional transformation can compress data. This would be a functional compression and the reconstruction is not exact, but its accuracy is measured with respect to an error norm like square sums. A compromise between compression ratio and characteristic retrieval is inevitable. Using a 2D wavelet or wavelet packet transform to decorrelate the data along with one of the Daubechies wavelets should do the best job at concentrating the data's energy content into the lower coefficients for a given filter length. After the transform, the coefficients below a given threshold will be zeroed out. The threshold is set to trade-off between the degree of compression and the degradation of the reconstructed data set. The thresholding can be followed by quantization (scalar and/or vector) and a Huffman or algebraic compression. Decompression performs the above operations in reverse.

The FBI has standardized the use of wavelets in digital fingerprint image compression. The compression ratios are on the order of 20:1, and the difference between

the original image and the decompressed one can be told only by an expert[18]. This type of compression works by discarding coefficients below a certain threshold.

After successfully compressing the data, the next step is to capture the "global" reservoir features on a large scale grid system and then fill in the fine scale features using wavelets. For example, once a fractal is characterized, then it can be reproduced on any scale. So, once a reservoir feature has been characterized, we should be able to reproduce it at any scale.

5.3 Purpose

The purpose of using wavelets is to reduce the flow of simulation computing cost for the implementation of a two part simulated annealing objective function[12]. The local nature of wavelets and their ability to localize in time-frequency or phase space, along with a natural scheme for fast computation, has made wavelets the approach of choice in certain applications[16]. Traditional methods, such as spectral methods involving frequency representation of data, usually assume stationarity. These methods, therefore, extract only the average information and hence are not suitable for analyzing data with isolated or deterministic discontinuities, such as faults or fractures in reservoir rocks[15]. The study of nonstationary signals, where transient events appear that cannot be predicted (even statistically with knowledge of the past), necessitates techniques different from Fourier analysis. These techniques, which are specific to the nonstationarity of the signal, include wavelets of the "time-frequency" type and wavelets of the "time-scale" type. "Time-frequency" wavelets are suited, most specifically, to the analysis of quasi-stationary signals, while "time-scale" wavelets are adapted to signals having a fractal structure[13].

5.4 Expert Advice

There is a multitude of information on wavelets and wavelet transforms. There is not much information on the practical application of wavelets though. Most of the information we have obtained has been through contacting wavelet experts via Internet.

We have been in contact with Brani Vidakovic, co-author of Wavelets for Kids. She is currently examining our data set in order to forward any suggestions on what approach to take and which wavelet packages would prove useful. Even before seeing the data, she suggested looking into Nason's Wavethresh 2.2.

Wavethresh 2.2 was successfully downloaded from Internet, but there were problems with the installation. I contacted Guy P. Nason, co-author of Wavethresh at Bristol University in the United Kingdom. After several correspondences, it was finally discovered that the system is based on a program called S/S+. This is a statistical package for which the University of Tulsa no longer has a license; therefore, Wavethresh is unable to run on our system. Mr. Nason is quite interested in this application of wavelets and asked that we keep him advised as to the progress of the project.

There is an electronic publication called Wavelet Digest available on the Internet. A notice was posted to this publication stating the problem and asking for any suggestions. Through a series of different people, we came in contact with Christian Cenker, a professor in the Department of Statistics at the University of Vienna in Austria. He suggested the use of Daubechies S8 symmlet for compression because it is a compromise between length and smoothness. With this, only the n largest wavelet coefficients of the expansion need to be stored and a reasonable reconstruction of the function should be possible where n depends on the regularity and length of the signal. He also suggested the use of a wavelet package called WaveLab.

WaveLab is a library of MatLab routines for wavelet analysis, wavelet-packet analysis, cosine-packet analysis and matching pursuit. The library is free of charge over the Internet. Versions are provided for Macintosh, UNIX, and Windows machines. The material is copyrighted and requires advanced permission for any commercial use. There has also been problems with the installation of this package. These problems are currently being worked out through corresponding with the authors via the Internet.

5.5 Future Work

Through the available written material and the corresponding with experts, it has become quite clear that the only way to find the proper wavelet transform is trial and error and expert suggestions. As soon as WaveLab is up and running, we can being the experimentation. Data sets can be compressed using a certain transform, then uncompressed and run through the flow simulator. The flow simulator will determine whether or not the compress is acceptable. A successful compress is a trade off between the compression ratio and the amount of pertinent information retained. Once a successful compression has been found, then work can begin on the enhancement portion of this project.

REFERENCES

1. Moridis, G.J., McVay, D.A., Reddell, D.L. and Blasingame, T.A.: "The Laplace Transform Finite Difference (LTFD) Numerical Method for Simulation of Compressible Fluid Flow in Reservoirs", SPE 22888 presented at the 1991 Annual Technical Conference and Exhibition, Dallas, TX, Oct. 5-8

2. *ECLIPSE 100 - Black Oil Simulator*, ECL-Bergeson Petroleum Technologies, Inc., Oxfordshire, England (1990)

3. Chen, Chih-Cheng and Raghavan, Rajagopal: "An Approach To Handle Discontinuities by the Stehfest Algorithm", SPE 28419 presented at the SPE 69th Annual Technical Conference and Exhibition held in New Orleans, LA, U.S.A., 25-28 September 1994

4. Greenbaum, A. et al.: *Sparse Linear Algebra Package Version 2.0* Lawrence Liverpool National Laboratory, Liverpool Computing Center (1986)

5. Kincaid, David R., Respess, John R., Young, David M. and Grimes, Roger G.: *ITPACK 2C: A FORTRAN Package for Solving Large Sparse Linear Systems by Adaptive Accelerated Iterative Methods* freeware and documentation from netlib.

6. Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and van der Vorst, H.: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* freeware and documentation from netlib.

7. Hensley, Jeff: Personal Communication (Jan. 1995)

8. Lee, Jaedong: Personal Communication (Mar. 1995)

9. Kasap, Ekrem: Personal Communication (Apr. 1995)

10 Daubechies, Ingrid. *Ten Lectures in Wavelets.* Regional Conference Series in Applied Mathematics.

11 David, Guy. *Wavelets and Singular Integrals on Curves and Surfaces.* Lecture Notes in Mathematics 1465. Springer-Verlag, Berlin, Germany. 1991.

12 DOE Annual Report. October 1993 - October 1994.

13 Meyer, Yves. *Wavelets: Algorithms & Applications.* Society for Industrial and Applied Mathematics, Philadelphia. 1993.

14 Meyer, Yves. *Wavelets and operators*. Cambridge University Press, Cambridge. 1992

15 Panda, Manmath N. and Larry W. Lake. *Application of Wavelet Analysis to Reservoir Characterization*. The University of Texas at Austin.

16 Robinson, Sam L. and Peter F. Ryczek. *Wavelets*. The Mathematica Journal. Miller Freeman Publications. 1995.

17 Wavelet Transforms. William H. Press. Harvard-Smithsonian Center for Astrophysics Preprint No. 3184 (1991). Cambridge, MA 02138

18 Müller, Peter and Brani Vidakovic. *Wavelets for Kids : A Tutorial Introduction*. Duke University. 1991 AMS Subject Classification: 42A06, 41A05, 65D05.

19 Panda, Manmath N. and Larry W. Lake. *Application of Wavelet Analysis to Reservoir Characterization*. The University of Texas at Austin.