Proceedings of the Institute of Acoustics

AN EXPERT SYSTEMS APPROACH TO UNDERSTANDING SIGNALS AND SYSTEMS

A.M. Raper† and J.K Hammond‡

† Andersen Consulting, Arundel St, London, UK
‡ ISVR, Southampton University, Southampton, UK

INTRODUCTION

Achievement in signal processing is a matter of successful interpretation. The analyst is provided with a signal that has been recorded as the 'output' of an experiment. In our experience, this signal will be a time series where the variable under observation is some form of physical motion. This measurement contains a great deal of information concerning the process that created it. Unfortunately, the information is rarely obvious. So in order to learn about the process, the signal must be interpreted.

It appears that information is stored in a signal in two ways:

1) Numerically - the actual values of the signal at moments in time have a physical meaning as a reflection of reality. This data is suitable to being operated upon by algorithms.

2) Patterns - when the numerical data is presented pictorially, the individual values merge into patterns, patterns that can be recognised as symptomatic of known causes.

A good analyst is skilled at using both sources of information to aid interpretation. This may be as simple as performing a Fourier Transform on a time series and spotting a harmonic set. Background knowledge of the process can also be used. For instance, if a shock experiment has been conducted in a reverberant environment, one would look for echoes.

The computer is the basis for most signal processing operations today. The manipulation, storage and display of signals are computer abilities universally utilised. Yet, programs are not capable of finding or using the more quantative information discussed above, information that is an equally important part of signal interpretation.

This project aims to develop a program that will enable a user to gain an understanding of a signal and the process that caused it. This paper sets out a method to achieve this goal by using both a model representation of the process and expert system's programming techniques.

Since we intend to adopt a systems approach to the problem, time series coming from econometric or biological processes could be similarly treated. However, there are aspects of our work that are peculiar to mechanical systems.

AN EXPERT SYSTEMS SOLUTION

Expert systems are a programming method that lets computers mimic human problem solving ability. By observing how an expert tackles a problem, it is possible to turn expertise and experience into a computer usable form. This knowledge is usually expressed as rules but may include implicit strategies. Although a crude attempt at machine intelligence, it is a technique that has proved successful when the domain of interest is small and specialised. This means that an examination of how an analyst works forms the basis of the project:

When presented with a signal and some information on how that signal was created, the analyst draws on experience to hypothesise a simple model of the experiment. This model takes the form of components that are known to cause the patterns present in the signal or are suggested by the environment of the experiment.

A comparison takes place between the actual signal and the one generated by the model. This comparison shows to what extent the data conforms to the model, or conversely, to what extent the model mimics the experiment. Since the model is purely a mental one, generating a signal from it requires an act of imagination. This imagined signal will be described not by numbers but the patterns in it.

The comparison produces information on how the model and signal are dissimilar. Particular differences can suggest what is missing from the model, or is spurious in it. So based on the comparison, the analyst alters the model until it is 'sufficiently' accurate to be taken as a copy of the experiment.

The construction and alteration of the model by the analyst may have taken place subconsciously, and almost certainly without simulation, making it only possible to perform successfully with experience. If the program can work with a user to construct a model, simulation is continuous. This makes analysis more structured, but admittedly tedious to the experienced user.

From both a practical and conceptual viewpoint, the problem of designing such a program breaks down into five sections:

- Constructing and manipulating a model. This includes being able to generate data at the output or intermediate positions.
- Determining whether any processing should be performed on the measured signal before comparing it to the model's output.
- Determining whether the two signals are similar and describing their differences if they do not match.
- Choosing appropriate strategies for extracting information from the measured signal. It will depend on how the signal was created as to which techniques will be effective in the interpretation. It will also depend on the differences found in section iii) as to what information is required.
- Perform the signal processing operations suggested by the previous section.

CONSTRUCTING AND MANIPULATING A MODEL

Firstly a suitable notation and technique for a signal processing model needs to be found. We adopt the systems approach [1] since it is so widely accepted in signal processing texts.

In general systems theory, a system is a collection of objects united by some form of interaction. Each object is characterised by a set of measurable attributes which might be dependent on each

other. The relationship between the attributes characterises the object. Those variables that can be altered by an experimenter are the inputs, and those that can be observed but not directly altered are the outputs. $(u_1, \ldots, u_k)$ is the input vector **u** and $(y_1, \ldots, y_m)$ is the output vector **y**. By varying **u** over all its allowed values, and observing **y** the experimenter can derive a set of relations between y_i and u_i as shown in Figure 1.

$$\Psi^{(1)}(u_1, u_2, \ldots, u_k, y_1, y_2 \ldots, y_m) = 0$$
$$\Psi^{(2)}(u_1, u_2, \ldots, u_k, y_1, y_2 \ldots, y_m) = 0$$
$$\Psi^{(n)}(u_1, u_2, \ldots, u_k, y_1, y_2 \ldots, y_m) = 0$$

$$\text{or } \Psi(u, y) = 0$$

Figure 1: Equation set for attribute relationships

A notation for a system is to use boxes joined by leads. Leads to a box denote terminal variables, and are labeled with arrows if the object is orientated ie) if the distinction is made between inputs and outputs. The term black box is often used to talk about objects since it indicates that they can only be dealt with (effected or observed) through their terminal variables. If a system is comprised of several objects, $\Lambda_1, \Lambda_2, \ldots, \Lambda_N$ the interactions are represented as constraints on the terminal variables. The joining of a terminal of Λ_i to one from Λ_j signifies that the corresponding terminal variables are constrained to be equal for all t.

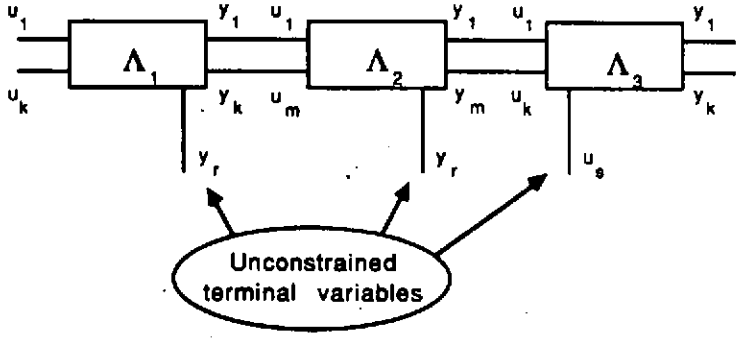

Figure 2: Multiple input-multiple output system

In signal processing we adopt a more implicit notation where the attribute relationships are limited to physical ones, namely filtering. This is shown in figure 3.

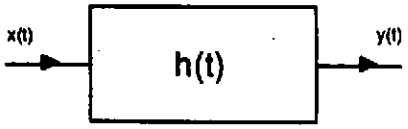

Figure 3: Single input-single output filter

If we are dealing with time series, the relationship between x and y, assuming that the system is linear and time invariant, is

$$y = h * x \text{ - convolution}$$

or in the frequency domain

$$Y = H . X \text{ - multiplication}$$

So the characteristic that defines the object is the impulse response function h, or equivalently the transfer function, H, depending on which domain is relevant.

An object such as the one shown in figure 3 is called a filter. Other objects needed to build simple systems are sources, sinks and adders.

A source is an object with no input but contains a procedure which can generate an output signal.

A sink just has an input and allows data to flow into it. This notion of data flowing through the model is entirely conceptual as the signal processing software only works with static data files. Having sinks therefore is no more than a tidy way to terminate a model.

An adder is simply a specialised filter that sums all its inputs to form one output.

To make the model work, the laws of causality must be at its core. When a signal is generated or given, it must be propagated through the model. To this end the signal processing routines of addition, multiplication and convolution operate between components.

To make the model useful, it must be displayed on the screen and the user must have the ability to inspect any part of it. To look at the signal 'flowing' between two components, or the characteristic of a component the user simply clicks the mouse on the link.

The advantage of this approach is that the user is working with a program that has a physically meaningful interface, and that whole systems can be manipulated with no need for programming skills or remembering housekeeping functions.

The model also acts as controller of the program. The signal processing knowledge has been structured so that it maps on to the model. The task of the program is to find the parameters that describe each of its components. As the knowledge to do this is local to each component, the interface can control the execution of knowledge based routines.

DETERMINING HOW TO PRESENT THE SIGNALS

The best way to determine whether or not the model is accurate is not always to simply compare the output of the model with the measured signal. More information might be available if the signals were compared in the frequency domain or if a filtering operation was performed to both prior to comparison. If a processing strategy has just been performed it will dictate how its results should best be seen.

COMPARING TWO SIGNALS

A signal from a mechanical system tends to be very complex, yet an analyst will be able to study it and hypothesise the features of the system that caused it. This is because the human eye is adept at picking out patterns in noise. Having identified a pattern, the analyst can use experience to deduce the physical characteristic that might have caused this pattern. An example of other work taking this approach is that of Milios [2]. Working in the acoustic tracking of helicopters, it is the spotting of sets of harmonic peaks in a spectrum that guides his program. The peaks are found and classified into sets automatically and not by using human eyesight.

As well as using patterns in signals to guess which components might be in the model, two signals can be compared by matching the patterns in one with those in the other. This information can then be returned to a module that would determine the implications of the mismatch for the model. The problem of determining when the model is accurate enough is solved simply by the model producing a signal that has the same patterns as the measurement. This avoids having to use statistical methods or correlations which would be prone to exaggerating small errors.

CHOOSING A STRATEGY FOR INTERPRETATION

Depending on the current state of the model, one particular strategy for interpretation will be more suitable than another. To choose the right strategy depends on knowing what the measurement suggests is present in the experiment. It also depends on knowing how accurate the model is. Again, this information can be found but it takes experience to use it.

PERFORMING SIGNAL PROCESSING

A commercial signal processing package (DATS) is available that takes a lot of the computational burden away from the programmer. As it maintains named data files, there is no need to be concerned about the representation of the data or storing details such as sampling rate, number of points. When data files are put through algorithms they are not altered, but new ones are created as outputs. There is an interactive graphics module that displays signals and allow interrogation of data values and graphic data editing. The package can be driven interactively or in a batch mode where several modules are strung together in a program to carry out a particular signal processing strategy.

IMPLEMENTATION

Since the aim is to produce this program on a computer, it is worth examining how the programming might take place. It was noted earlier that the design of the program broke down into five problem areas. In terms of programming style, there are just four.

i) The model construction and manipulation interface and functions

ii) The signal comparison interface

iii) The signal processing routines

iv) The knowledge based programs to link the above three utilities together so that the program runs as closely as possible to the analyst's behaviour.

THE MODEL UTILITY

In order to be readily acceptable, a graphical interface has been developed for the model. A collection of connected objects can be drawn by the user, or a selection made from a pre-defined library and displayed. Using a mouse pointing device, the user indicates which area of the model is of interest. The following protocol determines the program's behaviour.

i) Choosing a link between objects, means that the data file representing that signal should be displayed. If it does not exist, it is automatically created.

ii) If a filter is selected using a particular mouse key, the data file representing the transfer function is displayed.

iii) If a filter is selected using another key, a procedure for creating the transfer function is activated.

iv) If a source is indicated, a procedure that creates its output is activated.

If any creation procedure is activated that wishes to use a data file already in existence, the old version is deleted. Any change made is then propagated through the model so that it remains consistent. This has the effect that the model has no memory of how it is changing. Nothing can be undone, nor can a technique be performed and a check made to see if it has improved the model. However, any procedure can specifically save old versions of the data files for its own purposes.

The model storage needs to be programmed as several linked data bases. Each component in the model has information attached to it. This information includes all the necessary details for determining the component's behaviour, name and position in the graphical interface. By objects having named input and output links, the structure of the model can be determined. The design of these small data bases and the way in which consistency is ensured in the overall system is based on a technique of structure information called frames. First proposed by Minsky [3] they have remained the property of the ai programmer, yet are perfectly relevant in a case such as this. A frame is a data structure that aims to group together pieces of information about an object. Relationships and facts are represented by attribute values of the object. A frame has slots (attributes), slots have facets and facets have values. The terminology of frames is illustrated by the figure 4:

The frame can take a heavy computational burden by obeying protocols for data insertion and retrieval. These data access procedures provide a buffer between the programmer and the underlying structure so that, for instance, it is not important to remember which order the slots are in. Other facilities for the programmer, such as sensitive slots where any added information triggers a "demon" procedure, can be readily provided. A typical frame for the model is shown in figure. 5. Note that if no facet name is given it is assumed to be one confusingly named "value".

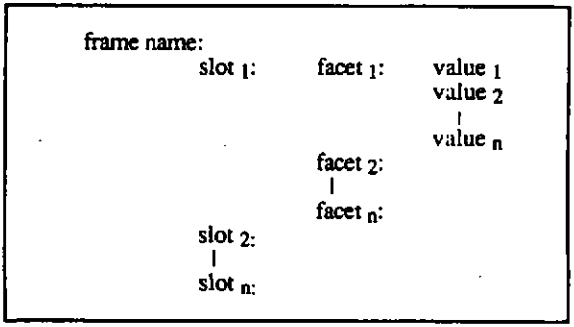

Figure 4: The terminology of a frame

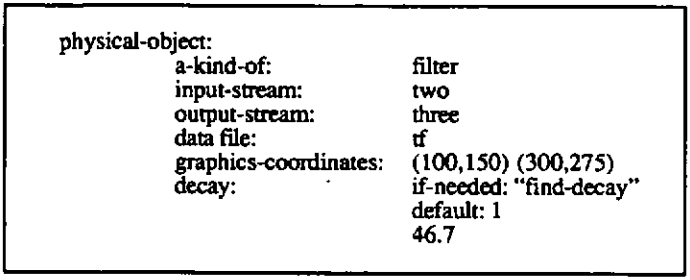

Figure 5: The frame representing the physical object in the model.

There are two possible methods for linking the frames together to form a system.

To make the slots sensitive to having information added to them. Causality operations then run as soon as a value is placed in a particular place, starting a chain reaction. An example of how this may be exploited is when the program inserts a value into the number-of-shocks slot in the excitation-source frame and a procedure is automatically activated that creates new slots if the number is greater than one.

To have procedures acting on named slots. In this method the information required is assumed to have been stored in the correct place, and a procedure to link all the relevant frames together can be called explicitly. An example of this approach is shown in the frame in figure 5 where the input and output stream slots must contain the names of the links between objects in the model.

In order to perform the causality properly the model also needs to be able to run several signal processing routines. Namely file addition, multiplication, convolution and display.

THE SIGNAL COMPARISON FACILITY

This is currently being developed and does not feature in the application discussed later. The implication of not having a signal comparison is that only one strategy can be followed, and that it is up to the user to determine when the model is complete.

THE SIGNAL PROCESSING

The usual interface to DATS has been dismantled, and individual modules are called by the knowledge based routines. DATS has no control over parameters passed to the routines, except in its interactive display module, and returns to the calling program when finished. The calling program can not access the data files other than through DATS routines. This discipline guards against errors.

EXPERT SYSTEMS PROGRAMMING

The knowledge based parts of the program are all controlled by small knowledge sources. These knowledge sources contain procedures linked into groups around the component whose parameters they are designed to find. Each procedure is an analysis technique that will find model parameters in certain conditions. These conditions are encapsulated by the rest of the system and its environment. By attaching conditions to each technique and making a control mechanism search the model database to try and satisfy these conditions, the technique is applied at the appropriate time. The user indicates interest in a particular knowledge source through the graphical interface.

A simple rule interpreter has been written that searches frame structures in order to ascertain facts. Thus the model data bases and the quantative signal representation are consistent with the format of the rules as the rules are represented as frames as well. For an example rule see figure 6.

rule 3:

a-kind-of:	rule
conditions:	(the value of signal-to-noise-ratio is high)
	AND
	(the value of periodicity is low)
procedure:	visual-epoch-detection
times-used:	0

Figure 6. An example frame that is the condition on a procedure running.

COMPUTING

The ability to combine all four types of programming has come from the use of LISP as the programming language. However, some sections of the program are written in a similar language to LISP called POP11. This is because the version of LISP used is not a complete implementation of Common LISP and is itself written in POP11, so communication with files or graphics is more

easily accomplished in POP11. This mixing of languages has not been a problem since LISP can call POP11 and they pass parameters back and forward within the tool POPLOG. POPLOG and DATS are running on a VAX 11/750 under VMS. The base for much of the LISP programming is the textbook by Winston [4].

APPLICATION

The ideas discussed previously are currently being implemented in the mechanical shock domain. An example case study is presented. Computer prompts are shown *in this typeface* with the users replies **in bold**. The following text and diagrams show the progression from default model parameters to the final selection.

The signal has been recorded from a system subject to shock excitation. The aim of the program is to assist an operator determine details of the shock excitation, structure and environment of the system. Firstly a few details are asked of the user:

How many signals were recorded ? **1**
Is the recording a response or input ? **response**
What is the filename of the signal ? **19218**
What sort of recording is it ? **acceleration**
Is the environment reverberant ? **no**
How many shocks occur during the excitation ? **unknown**
In that case which of the following is closest ?
one
five
ten
fifty
loads ? **ten**
Is the signal to noise ratio high enough for you to see the peaks ? **yes**
Are the shocks periodic ? **nearly**
Are the shocks close enough to cause overlapping ? **yes**

The measurement under investigation is shown in Figure 7:

Next the model that the computer proposes using is displayed, see Figure 8.

The user may commence by pointing the cursor at the excitation source object. This activates a knowledge base that will determine which method is most suitable to find the impulse details.

Activating the Excitation Source Knowledge Base
Visual Epoch Detection

The measurement will now be displayed. There should be 10 obvious peaks, but if not just pick out the clear ones. To pick a peak, position the cursor and click the mouse. When all have been selected, press return to leave the display.

Having identified the peaks an impulse train containing these peaks is created, and can be viewed by clicking the mouse on the link leaving the excitation source. See figure 9:

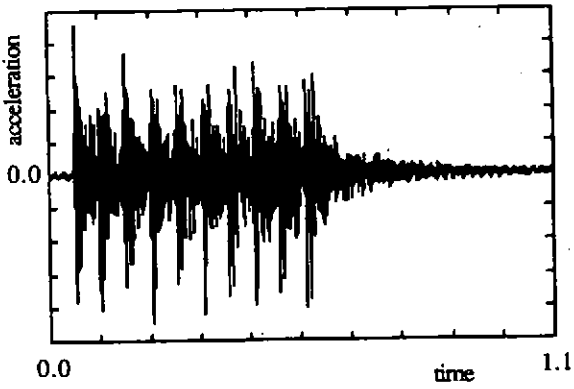

Figure 7: The measured shock response

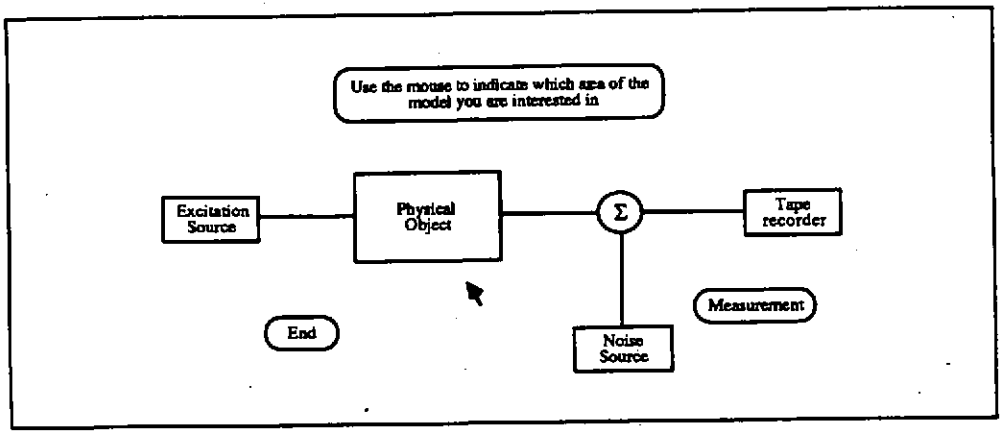

Figure 8: A snapshot of the user interface

If the next mouse movement at the interface is to try and display the output of the model, the program will have to find the physical object impulse response function and the noise, convolve and add these signals before displaying the required signal.

Activating the Physical Object Knowledge Base
Synthetic Impulse Response Generation

On the following display of the measurement, please use the cursors to indicate where the final signal starts (at the peak) and finishes or disappears into the noise floor.

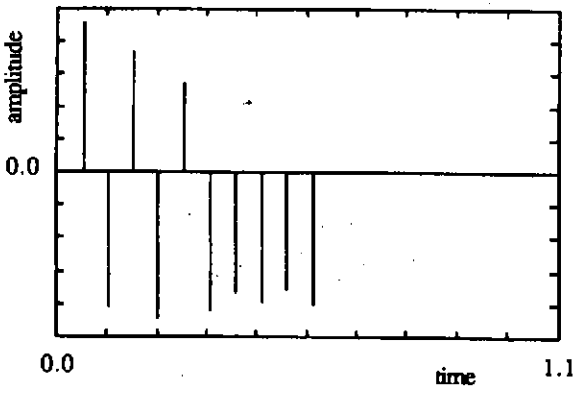

Figure 9: The visually located impulses

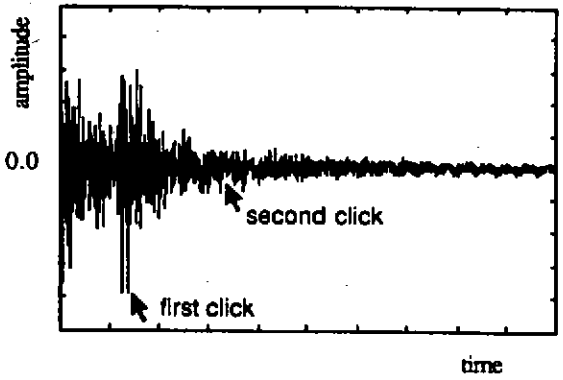

Figure 10: The user indicating the start and end of the final shock so that the program can calculate the decay.

Looking at the frequency response of the measurement may show a peak at the fundamental frequency of the object. Please use the cursors to indicate the major peak. Evidence of any more modes would be other peaks, these should also be indicated.

This method has assumed that the impulse response of the system can be modeled as a series of exponentially decaying sinewaves. The decay of the signal is calculated from the first display and the frequencies of the modes from the second. The modes are calculated and added together to become the impulse response function.

The impulse response found in this way is shown in figure 12. Then the program can create the output of the structure:

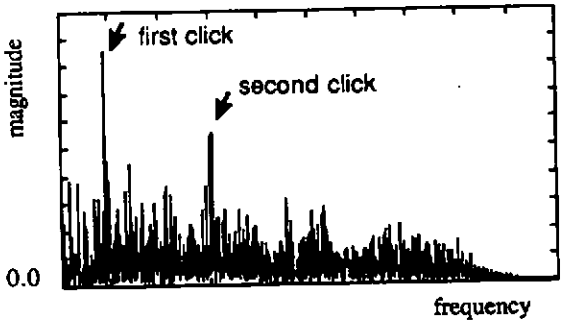

Figure 11: The frequency response of the measurement with the user's indication of the modes of vibration

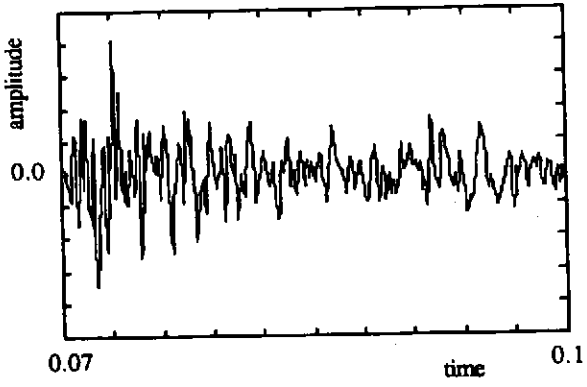

Figure 12: The estimated impulse response function of the physical object

CONVOLVE (ideal-input, tf) → object-response

The noise signal is assumed to be a gaussian process with the amplitude set by the user indicating the background noise level in the measurement again using the cursor on the signal display.

ADD (object-response, noise) → output

If the user now clicks in the "measurement" object a comparison between the model output and the the measurement is shown, see figure 13. An alternative mode of comparison is to generate a line drawing pattern of the model output. This is shown at the bottom of the figure .

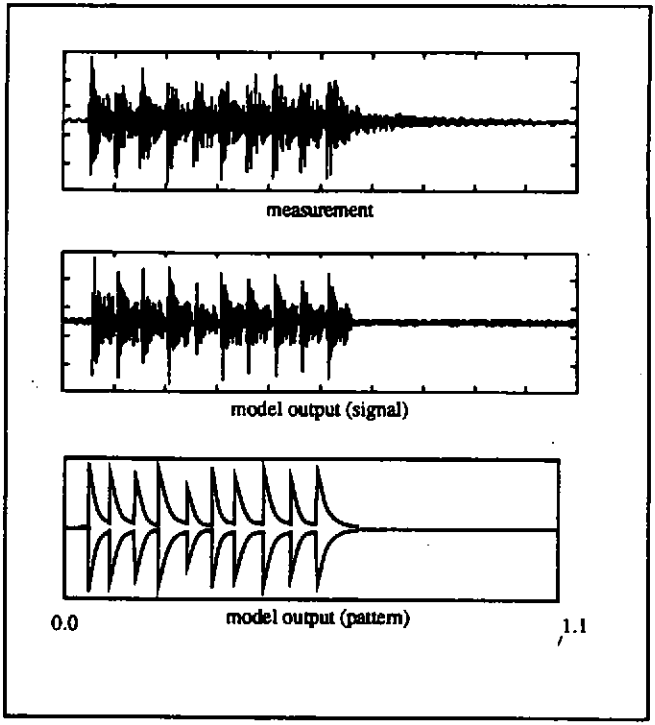

Figure 13: A comparison between the model output and the measured signal. Also shown is the line drawing pattern model output

At present, the mismatch information is left for the user to interpret but this is to be automated by associating confidence or probability factors with the parameters.

CONCLUSION

By putting a model at the foundation of a program the user is helped to come to an understanding of the system that created a signal. This is done using knowledge based techniques to fill out the parameters with the user's assistance. The strengths of this work are that the graphical interface is mapped to the control of the program and is familiar to the user. The model takes care of much of the humdrum signal processing such as propagating changes through the model in a causal manner. The project also demonstrates a way in which signal processing knowledge can be encoded into a program using a different architecture to previous knowledge based signal processing programs [5,6].

REFERENCES

1. L.A.Zadeh and C.A.Desoer 1963 Linear System Theory pp.102-103 New York: McGraw-Hill

2. E.E. Milios 1986 PhD Thesis, M.I.T., Cambridge Signal processing and Interpretation using Multilevel Signal Abstractions

3. M. Minsky 1963 In: The Psychology of Computer Vision (ed P.H.Winston) A Framework for Representing Knowledge New York: McGraw-Hill

4. P.H. Winston and B.K.P. Horn 1984 LISP Second Edition Massachusetts: Addison-Wesley

5. J.N. Maksym et al 1983 In:Issues in Acoustic Signal/Image Processing and Recognition (ed C.H.Chen) Machine Analysis of Acoustical Signals pp 95-112 Berlin:Springer-Verlag

6. H.P. Nii and E.A. Feigenbaum 1978 In:Pattern Directed Inference Systems Rule-Based Understanding of Signals pp 483-501 Academic Press