

# **APPLICATION OF NATURE-INSPIRED KNOWLEDGE MINING ALGORITHMS TO EMERGENT BEHAVIOUR DISCOVERY IN ECONOMIC MODELS**

David Al-Dabass

School of Computing and Informatics  
Nottingham Trent University  
Nottingham, NG11 8NS, UK  
Phone No. +44-781-041-5218  
Fax No. +44-115-848-8429  
Email: [david.al-dabass@ntu.ac.uk](mailto:david.al-dabass@ntu.ac.uk)

Published as:

Chapter XII in Handbook of Research on Nature-Inspired Computing, ed. Jean-Philippe  
Rennard.

ISBN 1-59140-984-5 h/c.

# **APPLICATION OF NATURE-INSPIRED KNOWLEDGE MINING ALGORITHMS TO EMERGENT BEHAVIOUR DISCOVERY IN ECONOMIC MODELS**

**Abstracts:** Economic models exhibit a multiplicity of behaviour characteristics that are nonlinear and time-varying. 'Emergent' behaviour appears when reduced order models of differing characteristics are combined to give rise to new behaviour dynamics. In this chapter we apply the algorithms and methodologies developed for nature-inspired intelligent systems to develop models for economic systems. Hybrid recurrent nets are proposed to deal with knowledge discovery from given trajectories of behaviour patterns. Each trajectory is subjected to a knowledge mining process to determine its behaviour parameters. The knowledge mining architecture consists of an extensible recurrent hybrid net hierarchy of multi-agents where the composite behaviour of agents at any one level is determined by those of the level immediately below. Results are obtained using simulation to demonstrate the quality of the algorithms in dealing with the range of difficulties inherent in the problem.

Keywords: knowledge acquisition, economic models, data mining.

## **INTRODUCTION**

Recurrent inference networks are introduced to represent knowledge bases that model dynamic intelligent systems. Through a differential abduction process, the causal parameters of the system behaviour are determined from measurements of its output to represent the knowledge embedded within (Al-Dabass et al, 2001). The use of dynamical knowledge mining processes ensures that knowledge evolution is tracked continuously (Al-Dabass et al, 2002-a). Meta-knowledge, defined in terms of the causal parameters of the evolution pattern of this first level knowledge, is further determined by the deployment of second level dynamical processes (Bailey et al, 1996). In data mining applications, for example, there is a need to determine the causes of particular behaviour patterns. Other applications include cyclic tendencies in stock values and sales figures in business and commerce, changes in patient recovery characteristics, and predicting motion instabilities in complex engineering structures (Al-Dabass et al, 2002-c). Full mathematical derivation is given together with simulations and examples to illustrate the techniques involved.

**Knowledge Models:** To understand and control the behaviour of economic systems, models that represent the knowledge embedded within these systems are formulated and used to acquire this knowledge from measurements. In data mining applications, for example, there is a need to determine the causes of particular behaviour patterns. Applications include cyclic tendencies in stock sales figures in business and commerce, sudden movements in share indices changes and, in no economic areas, patient recovery characteristics and predicting motion instabilities in complex engineering structures.

**Hybrid Inference Networks:** To represent the knowledge embedded within intelligent systems, a multilevel structure is put forward. By its very nature this knowledge is continually changing and need dynamic paradigms to represent and acquire its parameters from observed data. In a normal inference network the cause and effect relationship is static and the effect can be easily worked out through a deduction process by considering all the causes through a step-by-step procedure which works through all the levels of the network to arrive at the final effect. However, reasoning in the reverse direction, such as that used in

diagnosis, starts with observing the effect and working back through the nodes of the network to determine the causes.

Knowledge Mining for Stock Market Models: Work in this chapter extends these ideas of recurrent or dynamical systems networks to economic models where some or all the data within the knowledge base is time varying. The effect is now a time dependent behaviour pattern, which is used as an input to a differential process to determine knowledge about the system in terms of time varying causal parameters. These causal parameters will themselves embody knowledge (meta knowledge) which is obtained through a second level process to yield 2<sup>nd</sup> level causal parameters. These processes consist of a differential part to estimate the higher time derivative knowledge, followed by a non-linear algebraic part to compute the causal parameters.

## **ECONOMIC SYSTEM MODELLING AND SIMULATION USING HYBRID RECURRENT NETWORKS**

Numerous economic systems in practice exhibit complex behaviour that cannot be easily modelled using simple nets (Berndt & Clifford, 1996). In this part we re-cast this problem in terms of hybrid recurrent nets, which consist of combinations of static nodes, either logical or arithmetic, and recurrent nodes. The behaviour of a typical recurrent node is modelled as a second order dynamical system. The causal parameters of such a recurrent node may themselves exhibit temporal tendencies that can be modelled in terms of further recurrent nodes. Layers of recurrent nodes are added until a complete account of the behaviour of the system has been achieved (Al-Dabass et al, 2002-b). Algorithms are given to estimate the values of the parameters of these models from behaviour trajectories of intelligent systems. One novel aspect of the work lies in having a simple hierarchical 6<sup>th</sup> order linear model to represent a fairly complicated behaviour encountered in numerous real examples in finance, biology and engineering.

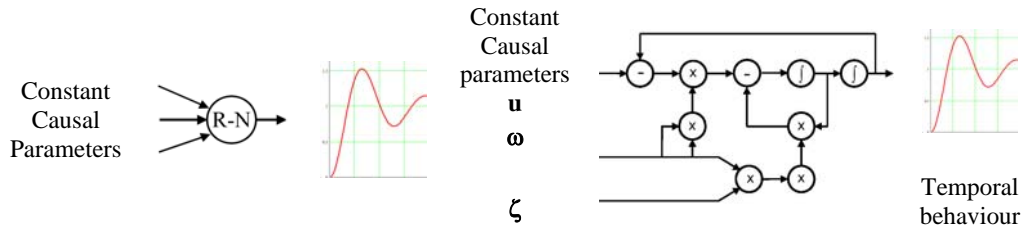
### **Hybrid Recurrent Network Models**

Many physical, economical and biological phenomena exhibit temporal behaviour even when the input 'causal' parameters are constant, Figure 1.

To model this oscillatory behaviour a second order integral hybrid model is proposed, shown in Figure 2. This model is based on the well known second order dynamical system which has the following form:

$$\omega^{-2} x'' + 2. \zeta. \omega^{-1}. x' + x = u \quad (1)$$

Where  $x$  is the output of the node and  $\omega$ ,  $\zeta$  and  $u$  are the natural frequency, damping ratio and input respectively, which represent the 3 causal parameters that form the input. To configure this differential model as a recurrent network, a twin integral elements are used to form a hybrid integral-recurrent net as shown in Figure 2.



**Figure 1.** A Recurrent Node (R-N) exhibits a temporal behaviour at the output despite having constant causal parameters.

**Figure 2.** Hybrid integral-recurrent net to model the temporal behaviour of the node in Fig. 1

### Structure of the Hybrid Integral-Recurrent Net

The net shown in Figure 2 is a direct representation of equation-1 and can be derived as follows:

- i) By multiplying both sides by  $\omega^2$  we get:

$$x'' + 2. \zeta. \omega. x' = \omega^2. (u - x) \tag{1-A}$$

OR

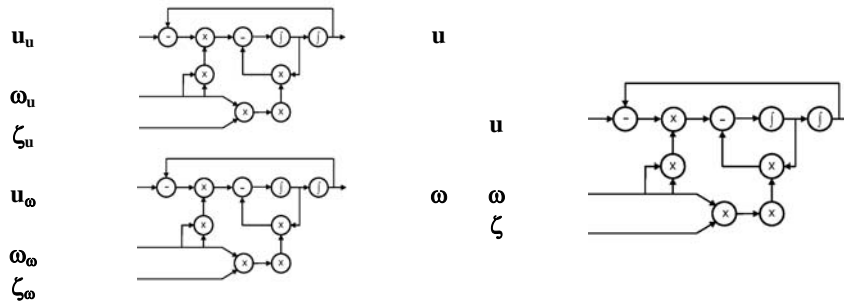
$$x'' = \omega^2. (u - x) - 2. \zeta. \omega. x' \tag{1-B}$$

- ii) The output of the net  $x$  is fed back to the first subtraction node on the left; as the input from the left of this node is  $u$  the output is  $(u - x)$ .
- iii) The middle input (to the whole net) from the left is  $\omega$ ; it is fed as 2 separate inputs to the multiplication node  $x$  to form  $\omega^2$  at its output, shown with an up arrow feeding as the lower input of the multiplier node above it, which is the second node from the left in the top chain of nodes.
- iv) The output of this multiplier node is therefore  $\omega^2. (u - x)$ , i.e. the RHS of equ. 1-A.
- v) The bottom input from the left (to the whole net) is  $\zeta$  which is fed as the lower input to the first of the two multipliers in the bottom chain of 2 nodes, - as the top input to this node is  $\omega$  the output is  $\zeta. \omega$ . which is multiplied by 2 in the 2<sup>nd</sup> node in the chain to produce  $2. \zeta. \omega$ .
- vi) The last node on the right in the top long node chain is an integrator node that generates  $x$  as stated in ii) above. As it is an integrator node, the input to it must therefore be the derivative of  $x$ , i.e.  $x'$ . This is multiplied by the output of the right node in the bottom 2-node chain (which is  $2. \zeta. \omega$ ) to produce  $2. \zeta. \omega. x'$ , which is the 2<sup>nd</sup> term in the LHS of equ. 2-1-A or the 2<sup>nd</sup> term on the RHS of equ. 2-1-B.
- vii) By subtracting this output from the output of the middle node in the top row, we get the full RHS of equ. 1-B, i.e.  $\omega^2. (u - x) - 2. \zeta. \omega. x'$ .
- viii) As the output of the second integrator from the right (in the top chain) is the first derivative of  $x$ ,  $x'$ , the input to this integrator node must be  $x''$ , i.e. the LHS of equ. 1-B.

- ix) Simply connecting the output of the middle node of the top chain (which is  $\omega^2 \cdot (u - x) - 2 \cdot \zeta \cdot \omega \cdot x'$ ) into the input of the 2<sup>nd</sup> integrator from the right ( $x''$ ) will just complete the equation.

### Models of Hierarchical Recurrent Nodes

The output trajectory of the system may be more complex than can be represented by a simple second order differential model. In this case each causal parameter may itself be modelled as having a dynamical behaviour, which may or may not be oscillatory. One such case is where two of the 3 causal parameters have 2<sup>nd</sup> order dynamical characteristics, as shown in Figure 3.



**Figure 3.** Two of the causal parameters of the final node have temporal behaviour modelled as 2<sup>nd</sup> order hybrid integral recurrent nets.

The 2<sup>nd</sup> order model of a node in a given layer in the hierarchy is given by equation 1 above. Starting with the final, output, node let both  $u$  and  $\omega$  have their own 2<sup>nd</sup> order dynamics. The input  $\mathbf{u}$  is the output of the following 2<sup>nd</sup> order system:

$$\omega_u^{-2} \mathbf{u}'' + 2 \cdot \zeta_u \cdot \omega_u^{-1} \cdot \mathbf{u}' + \mathbf{u} = u_u \quad (2)$$

The natural frequency  $\omega$  is the output of the following 2<sup>nd</sup> order system:

$$\omega_\omega^{-2} \omega'' + 2 \cdot \zeta_\omega \cdot \omega_\omega^{-1} \cdot \omega' + \omega = u_\omega \quad (3)$$

Thus the behaviour trajectory is generated by the following 6<sup>th</sup> order vector differential equation (using Runge Kutta in Mathcad for this example).

First Order Vector Form: To provide a simulation output of the node trajectory, the 2<sup>nd</sup> order equation is converted to a 2<sup>st</sup> order vector differential equation that can be easily computed.

$$x1 = x, \text{ and } x2 = x'$$

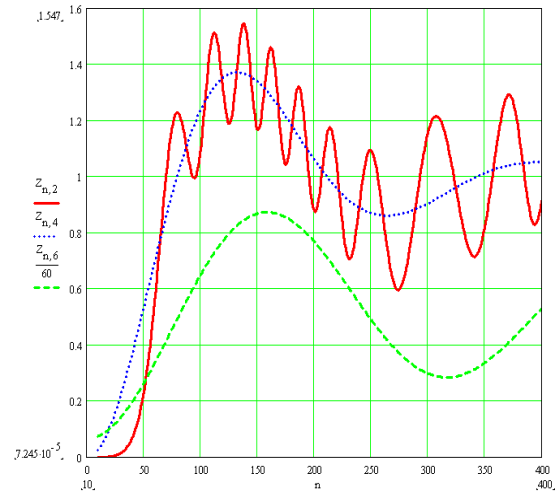
.

.

.

$$D(t, x) := \begin{bmatrix} x_2 \\ x_5 \cdot x_5 \cdot x_3 - 2 \cdot z \cdot x_5 \cdot x_2 - x_5 \cdot x_5 \cdot x_1 \\ x_4 \\ (w_u \cdot w_u \cdot u) - 2 \cdot z_u \cdot w_u \cdot x_4 - w_u \cdot w_u \cdot x_3 \\ x_6 \\ (w_\omega \cdot w_\omega \cdot \omega) - 2 \cdot z_\omega \cdot w_\omega \cdot x_6 - w_\omega \cdot w_\omega \cdot x_5 \end{bmatrix}$$

**Figure 4.** The derivative vector for generating the economic system output using subsystems for u and omega.



**Figure 5.** Simulated trajectory of a hierarchical recurrent node (oscillatory trace), with 2 variable inputs: u (upper trace) and omega (lower trace)

In Figure 4,  $x_1$  and  $x_2$  represent  $x$  and  $x'$ ,  $x_3$  and  $x_4$  represent  $u$  and  $u'$ , and  $x_5$  and  $x_6$  represent  $\omega$  and  $\omega'$  respectively. To generate the trajectory shown in Figure 5, the following values were used: for the  $u$  subsystem,  $u$  started from 0 aiming at  $u_u = 1$  at a rate of  $\omega_u = 5$  rad/s with  $\zeta_u = 0.3$ . For the  $\omega$  subsystem,  $\omega$  started from 4 rad/s aiming at  $u_\omega = 32$  rad/s at a rate of  $\omega_\omega = 4$  rad/s with  $\zeta_\omega = 0.1$ . Figure 5 shows the resulting compound trajectory of  $x$  (oscillatory trace), together with the trajectories for  $u$  (upper trace) and  $\omega$  (lower trace).

## DERIVATIVE ESTIMATION USING RECURRENT NETWORKS

Based on models that describe the behaviour of complex natural and physical systems, a number of explicit static algorithms are developed to estimate the parameters of recurrent second order models that approximate the behaviour of these complex higher order systems (Ren et al, 1996) (Bovet & Crescenzi, 1994). These algorithms rely on the availability of the time derivatives of the trajectory. In this section a cascaded recurrent network architecture is proposed to 'abduct' these derivatives in successive stages (Cant et al, 2001). The technique is tested successfully on parameter tracking algorithms ranging from the constant parameter algorithm that only requires derivatives up to order 4 to an algorithm that tracks two variable parameters and requires up to the 8<sup>th</sup> time derivatives.

### Algorithm for Constant Parameters from Single Point Data

Consider using the 1<sup>st</sup> to 4<sup>th</sup> time derivatives at a single point. Given the second order system:

$$\omega^{-2} x'' + 2 \cdot \zeta \cdot \omega^{-1} \cdot x' + x = u \quad (4)$$

Differentiate with respect to t:

$$\omega^{-2} x''' + 2. \zeta. \omega^{-1}. x'' + x' = 0 \quad (5)$$

divide by  $x''$ :

$$\omega^{-2} x''' / x'' + 2. \zeta. \omega^{-1} + x' / x'' = 0 \quad (6)$$

and differentiate with respect to t again to give:

$$\omega^{-2}. [(x'' . x''' - x''^2) / x''^2] + 0 + [(x''^2 - x' . x''') / x''^2] = 0 \quad (7)$$

Expressions for estimated  $\omega$ , estimated  $\zeta$ , using (5), and estimated u result as follows:

$$E\omega^2 = [x'' . x''' - x''^2] / [x' . x'' - x''^2] \quad (8)$$

$$E\zeta = -[E\omega^{-2} x''' + x'] / [2. E\omega^{-1}. x''] \quad (9)$$

$$Eu = E\omega^{-2}. x'' + 2. E\zeta. E\omega^{-1} . x' + x \quad (10)$$

### High Order Algorithms

Assume that the first and higher time derivative of u to be non zero. For simplicity assume that both a and b (the coefficients of  $x''$  and  $x'$  to make symbol manipulation easier) to be constant and hence disappear on first differentiation. The extra information needed for  $u'$ ,  $u''$ ,  $u'''$  and  $u''''$  to be non zero is extracted from the 5<sup>th</sup>, 6<sup>th</sup>, 7<sup>th</sup> and 8<sup>th</sup> time derivatives of the trajectory. Only the case for the  $u'$  is shown here, the others for  $u''$  etc are simple extensions of the idea and are left as an exercise for the reader.

$$a.x'' + b.x' + x = u \quad (11)$$

Differentiate wrt to t and assume  $u'$  is non zero to give:

$$a.x''' + b.x'' + x' = u' \quad (12)$$

Differentiate again and set  $u'' = 0$  gives:

$$a.x'''' + b.x''' + x'' = 0 \quad (13)$$

Divide Equation 11 by  $x'''$  to isolate b:

$$a.x''''/x''' + b + x''/x''' = 0 \quad (14)$$

Differentiate again to eliminate b:

$$a.(x'''' . x'' - x''''^2)/x''''^2 + (x''^2 - x'' . x'''' )/x''''^2 = 0 \quad (15)$$

Re-arranging for a gives:

$$E(a) = (x'' . x'''' - x''''^2)/(x'''' . x'' - x''''^2) \quad (16)$$

Solve for  $b$  by substituting  $a$  from equ. 16 into equ. 14:

$$E(b) = -x''/x''' - a.x''''/x'''$$

which after substituting for  $a$  and manipulating gives:

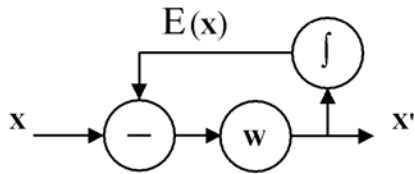
$$E(b) = (x'''.x'''' - x'''.x'''')/(x''''^2 - x'''.x''''') \quad (17)$$

We can now substitute these values for  $a$  and  $b$  into Equation 1 to solve for  $u$ ,

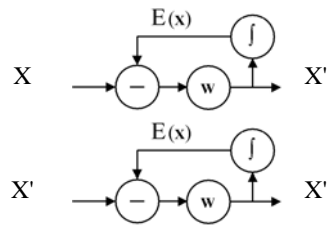
$$u = a.x'' + b.x' + x$$

### A Recurrent Architecture to Estimate Time Derivatives

The structure of each cell of the recurrent network is shown in Figure 6. The output of each cell feeds the input to the next one to generate the next higher order time derivative, see Figure 7. The output of the system and the cascade of 1<sup>st</sup> order recurrent network filters are simulated using the 4<sup>th</sup> order Runge-Kutta method in Mathcad. The derivatives vector is shown in Figure 8. Figure 9 shows a typical set of derivatives estimated from a damped oscillatory trajectory.



**Figure 6:** A single stage recurrent sub-net using an integrator in the feedback path to estimate the derivative  $x' = w(x-E(x))$ ; the net is a low pass filter with a cut off frequency  $w$ .

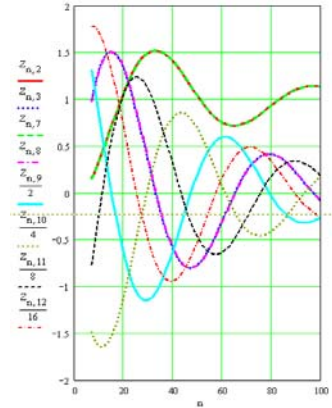


**Figure 7:** A 2<sup>nd</sup> order recurrent network to estimate 1<sup>st</sup> and 2<sup>nd</sup> time derivatives.

$$x := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad D(t, x) := \begin{bmatrix} x_2 \\ -\omega^2 \cdot x_1 - 2 \cdot \zeta \cdot \omega \cdot x_2 + \omega^2 \cdot u \\ G(x_1 - x_3) \\ G[G(x_1 - x_3) - x_4] \\ G[G[G(x_1 - x_3) - x_4] - x_5] \\ G[G[G[G(x_1 - x_3) - x_4] - x_5] - x_6] \\ G[G[G[G[G(x_1 - x_3) - x_4] - x_5] - x_6] - x_7] \end{bmatrix}$$

$Z := \text{Rkadapt}(x, t_0, t_1, N, D)$

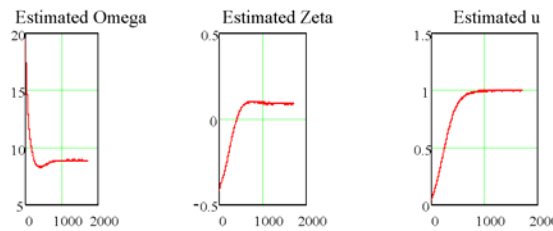
**Figure 8:** A cascade of 5 recurrent cells plus the 2<sup>nd</sup> order trajectory model.



**Figure 9:** A typical set of time derivatives estimated from the trajectory of an oscillatory 2<sup>nd</sup> order dynamical system

## Results and Discussion

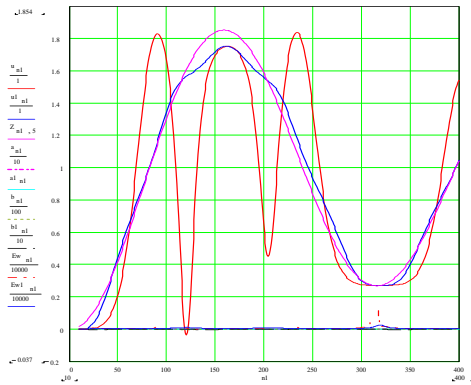
**First Algorithm using Constant Parameters:** This algorithm uses a single time point and four higher order time derivatives. The filter cascade provide a continuous estimate of the 1<sup>st</sup> to 4<sup>th</sup> time derivative  $x'$ ,  $x''$ ,  $x'''$  and  $x''''$ . This provides a continuous estimate of all parameters at each point on the trajectory. The result of the estimation are given in Figure 10; which shows fast and accurate convergence.



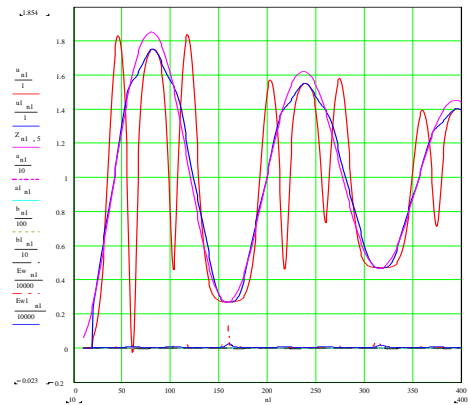
**Figure 10:** Estimated constant omega, zeta and u

**Discussion:** Estimated values for constants parameters are very close to the desired set values. The derived algorithms estimate  $\omega$ ,  $\zeta$  and  $u$  for a good range of values:  $\omega$  from 1 to 10,  $\zeta$  between +/- (0.01 to 1), and  $u$  between +/- (0.5-40), and give accurate estimates. Estimation errors decreased as  $\omega$  increased, particularly for small  $\zeta$  (less than 0.5): where oscillation provided wide variation in the variables to decrease errors. The differences between the (simulated) system time derivatives ( $x$ ,  $x'$  and  $x''$ ) and their estimates from the filter cascade depended on  $G$  (the cut-off frequency): high  $G$  provided more accurate estimation of derivatives but made the algorithms prone to noise and vice versa. Another disadvantage of high  $G$  from the simulation point of view is that simulation time increases considerably due to the integration routine adapting to ever smaller steps. The algorithm provides fast convergence.

**Results For The Higher Order Algorithm:** Mathcad routines are set up to generate the input  $u$  as second order system with its own parameters of natural frequency, damping ratio and input. The input subsystem damping ratio was set to 0.05 to generate an oscillatory behaviour for long enough to test the parameter tracking algorithm thoroughly. The frequency of the input is set to 16 radians per second, one quarter of the frequency of the data natural frequency. The derivative generation cascade is increased by one to produce the fifth time derivative. The results are shown in Figure 11 below.



**Figure 11:** Results of the high order algorithm



**Figure 12:** Results of the high order algorithm for one second integration time.

The actual input is the smooth trace, which gives approximately one and one quarter cycles over a period of half a second as expected, i.e. 16 radians/s = 2.546 Hertz. The large jagged trace shows the results from the previous constant  $u$  derivation algorithm which is failing completely to track the input parameter. The 3rd trace shows the result of the new algorithm, which is managing to track the input much more closely; however it starts to diverge slightly near the peak of the cycle but then returns to track it well right down and round the lower trough of the input trajectory.

To check the quality of tracking as time progresses, a second set of results, Figure 12, is obtained with integration time extended to one second to give two and a half cycles. It is clear that tracking remains stable. It is interesting to note that the old algorithm while completely failing to track the upper half of the input trajectory it seems to track it well during its lower half but not as well as the new algorithm.

## KNOWLEDGE MINING IN HYBRID INFERENCE NETS

**Deduction and Abduction in Inference Networks.** To engineer a knowledge base to represent economic systems, a multilevel structure is needed. By its very nature the knowledge embedded within these systems is continually changing and needs dynamic paradigms to represent and acquire their parameters from observed data. In a normal inference network the cause and effect relationship is static and the effect can be easily worked out through a deduction process by considering all the causes through a step-by-step procedure which works through all the levels of the network to arrive at the final effect. On the other hand, reasoning in the reverse direction, such as that used in diagnosis, starts with observing the effect and

working back through the nodes of the network to determine the causes; this is termed knowledge mining.

Dynamical Knowledge Mining Processes. These ideas are applied here to recurrent or dynamical systems networks where some or all of the data within the knowledge base is time varying. The effect is now a time dependent behaviour pattern, which is used as an input to a differential abduction process to determine the knowledge about the system in terms of time varying causal parameters. These causal parameters will themselves embody knowledge (meta knowledge) which is obtained through a second level mining process to yield 2<sup>nd</sup> level causal parameters. These mining processes consist of a differential part to estimate the higher time derivative knowledge, followed by a non-linear algebraic part to compute the causal parameters.

### **Hierarchical Causal Parameters with Temporal Behaviour**

The output trajectory of the system may be more complex than can be represented by a simple second order differential model. In this case each causal parameter is itself modelled as having a dynamical behaviour, which may or may not be oscillatory. One such case is where two of the 3 causal parameters have 2<sup>nd</sup> order dynamical characteristics, as was shown in Figure 3.

### **Knowledge Mining Algorithms**

Several explicit algorithms for the 3 usual parameters characterising the behaviour of second order models have been derived (Al-Dabass et al 1999-a) based on information available from the systems time trajectory. Leaving the 2<sup>nd</sup> order model in its 2<sup>nd</sup> time derivative form and using 3 points on the trajectory, each providing position, velocity and acceleration, a set of 3 simultaneous algebraic equations were solved to yield estimates of input, natural frequency and damping ratio. An online dynamical algorithm was then configured to combine estimates of the trajectory time derivatives with these explicit static non-linear functions to provide continuous parameter estimation in real time.

Multipoint Algorithms: Several algorithms are easily derived to estimate values of causal parameters using as many points from the trajectory as necessary to form a set of simultaneous algebraic equations. The parameters to be estimated form the unknown variables and the trajectory values and their time derivatives form the constant parameters of these equations, (Al-Dabass et al, 1999-b, 2002-a & 2002-b).

Algorithm 1: Three-Points in  $x$ ,  $x'$  and  $x''$ . Consider estimating  $\omega$ ,  $\zeta$  and  $u$  using three sets of  $x$ ,  $x'$  and  $x''$ :

$$\omega^{-2} x_1'' + 2. \zeta. \omega^{-1}. x_1' + x_1 = u \quad (18)$$

$$\omega^{-2} x_2'' + 2. \zeta. \omega^{-1}. x_2' + x_2 = u \quad (19)$$

$$\omega^{-2} x_3'' + 2. \zeta. \omega^{-1}. x_3' + x_3 = u \quad (20)$$

Subtracting (19) from (18) and (20) from (18) gives:

$$\omega^{-2}.( x_1'' - x_2'' ) + 2. \zeta. \omega^{-1}.( x_1' - x_2' ) + ( x_1 - x_2 ) = 0 \quad (21)$$

$$\omega^{-2}.( x_1'' - x_3'' ) + 2. \zeta. \omega^{-1}.( x_1' - x_3' ) + ( x_1 - x_3 ) = 0 \quad (22)$$

Divide (21) by  $( x_1' - x_2' )$  and (22) by  $( x_1' - x_3' )$  gives:

$$\omega^{-2} \cdot (x_1'' - x_2'') / (x_1' - x_2') + 2 \cdot \zeta \cdot \omega^{-1} + (x_1 - x_2) / (x_1' - x_2') = 0 \quad (23)$$

$$\omega^{-2} \cdot (x_1'' - x_3'') / (x_1' - x_3') + 2 \cdot \zeta \cdot \omega^{-1} + (x_1 - x_3) / (x_1' - x_3') = 0 \quad (24)$$

and subtracting gives:

$$\omega^{-2} \cdot [(x_1'' - x_2'') / (x_1' - x_2') - (x_1'' - x_3'') / (x_1' - x_3')] + [(x_1 - x_2) / (x_1' - x_2') - (x_1 - x_3) / (x_1' - x_3')] = 0 \quad (24-A)$$

Using the following notations:

$$\Delta 12 = (x_1 - x_2), \quad \Delta' 12 = (x_1' - x_2'), \quad \Delta'' 12 = (x_1'' - x_2'')$$

$$\Delta 13 = (x_1 - x_3), \quad \Delta' 13 = (x_1' - x_3'), \quad \Delta'' 13 = (x_1'' - x_3'')$$

we get expressions for estimated  $\omega$ , estimated  $\zeta$ , using (21), and estimated  $u$ :

$$E\omega^2 = [\Delta'' 13 \cdot \Delta' 12 - \Delta'' 12 \cdot \Delta' 13] / [\Delta 12 \cdot \Delta' 13 - \Delta 13 \cdot \Delta' 12]$$

$$E\zeta = [-E\omega^{-2} \cdot \Delta'' 12 - \Delta 12] / [2 \cdot E\omega^{-1} \cdot \Delta' 12]$$

$$Eu = E\omega^{-2} \cdot x_1'' + 2 \cdot E\zeta \cdot E\omega^{-1} \cdot x_1' + x_1$$

Algorithm 2: Two-Points and One Extra Derivative. Consider using two sets of  $x$ ,  $x'$ ,  $x''$  and  $x'''$ .

$$\omega^{-2} x_1'' + 2 \cdot \zeta \cdot \omega^{-1} \cdot x_1' + x_1 = u \quad (25)$$

$$\omega^{-2} x_2'' + 2 \cdot \zeta \cdot \omega^{-1} \cdot x_2' + x_2 = u \quad (26)$$

Subtracting (26) from (25) and dividing by  $(x_1' - x_2')$ :

$$\omega^{-2} \cdot (x_1'' - x_2'') / (x_1' - x_2') + 2 \cdot \zeta \cdot \omega^{-1} + (x_1 - x_2) / (x_1' - x_2') = 0 \quad (27)$$

Differentiating (27) with respect to  $t$  gives:

$$[\omega^{-2} [(x_1' - x_2') \cdot (x_1''' - x_2''') - (x_1'' - x_2'')^2] / (x_1' - x_2')^2 + 0 + [(x_1' - x_2')^2 - (x_1 - x_2) \cdot (x_1'' - x_2'')] / (x_1' - x_2') = 0 \quad (28)$$

Using the following notations:

$$\Delta 12 = (x_1 - x_2), \quad \Delta' 12 = (x_1' - x_2'),$$

$$\Delta'' 12 = (x_1'' - x_2'') \text{ and } \Delta''' 12 = (x_1''' - x_2''')$$

We get expressions for estimated  $\omega$ , estimated  $\zeta$ , using (27), and estimated  $u$ :

$$E\omega^2 = [(\Delta' 12) \cdot (\Delta''' 12) - (\Delta'' 12)^2] / [(\Delta' 12)^2 - \Delta 12 \cdot \Delta'' 12]$$

$$E\zeta = [-E\omega^{-2} \cdot \Delta'' 12 / \Delta' 12 - \Delta 12 / \Delta' 12] / [2 \cdot E\omega^{-1}]$$

$$Eu = E\omega^{-2} \cdot x_1'' + 2 \cdot E\zeta \cdot E\omega^{-1} \cdot x_1' + x_1$$

## Single Point Algorithms

In this section we relax the constant parameter condition by assuming a linear time variation, i.e. constant first derivative but zero second and higher time derivatives of parameters. As may be expected, more information is needed for this new case, which is to be extracted from the system output trajectory by obtaining higher time derivatives. Explicit functions of the parameters are still possible as well as those of their first time derivatives. A set of 3 equations, one for each parameter, is formulated and numerically computed in real time together with the state estimation vector observer to yield continuous trajectories of the parameters. This is a different technique to that of augmenting the state derivative vector with the parameter derivatives,- instead of driving these derivatives with some function of the error between the system and model output, we provide an explicit function that should aid successful and speedy convergence to actual parameter values and provide continuous tracking. This should hold even when the parameters are changing rapidly compared to the system's natural frequency or time constant.

These are classified according to the order of the parameter variation used in the derivation, i.e. constant, first order polynomial (constant  $u'$  but  $u''=0$ ), 2<sup>nd</sup> order polynomial (constant  $u''$  but  $u'''=0$ ) etc .

Algorithm 3: Constant Parameters. Consider using the 1<sup>st</sup> to 4<sup>th</sup> time derivatives at a single point. Given the second order system:

$$\omega^{-2} x'' + 2. \zeta. \omega^{-1}. x' + x = u \quad (29)$$

Differentiate with respect to t and divide by  $x''$ :

$$\omega^{-2} x''' / x'' + 2. \zeta. \omega^{-1} + x' / x'' = 0 \quad (30)$$

and differentiate with respect to t again to give:

$$\omega^{-2} . [(x'' . x'''' - x''^2) / x''^2] + 0 + [(x''^2 - x' . x''') / x''^2] = 0 \quad (4-15)$$

We get expressions for estimated  $\omega$ , estimated  $\zeta$ , using (3-14), and estimated  $u$ :

$$\begin{aligned} E\omega^2 &= [x'' . x'''' - x''^2] / [x' . x''' - x''^2] \\ E\zeta &= -[E\omega^{-2} x''' + x'] / [2. E\omega^{-1}. x''] \\ Eu &= E\omega^{-2} . x'' + 2. E\zeta. E\omega^{-1} . x' + x \end{aligned}$$

Algorithm 4: First Order Parameters. Let the first time derivative of  $u$  to be non zero. For simplicity assume that both  $a$  and  $b$  (the coefficients of  $x''$  and  $x'$  to make symbol manipulation easier) to be constant and hence disappear on first differentiation. The extra information needed for  $u'$  to be non zero is extracted from the 5<sup>th</sup> time derivative of the trajectory.

$$a.x'' + b.x' + x = u \quad (31)$$

Differentiate wrt to t and assume  $u'$  is non zero to give:

$$a.x''' + b.x'' + x' = u' \quad (32)$$

Differentiate again and set  $u'' = 0$  gives:

$$a.x'''' + b.x''' + x'' = 0 \quad (33)$$

Divide by  $x'''$  to isolate  $b$ :

$$a.x''''/x''' + b + x''/x''' = 0 \quad (34)$$

Differentiate again to eliminate  $b$ :

$$a.(x'''''.x'' - x''''^2)/x''''^2 + (x''''^2 - x'' \cdot x''''')/x''''^2 = 0 \quad (35)$$

Re-arranging for  $a$  gives:

$$a = (x'' \cdot x'''' - x''''^2)/(x'''' \cdot x'' - x''''^2) \quad (36)$$

Solve for  $b$  by substituting  $a$  from equation 36 into equation 34:

$$b = -x''/x''' - a.x''''/x'''$$

Substituting for  $a$  and manipulating gives:

$$b = (x'' \cdot x'''' - x'''' \cdot x''''')/(x''''^2 - x'' \cdot x''''') \quad (37)$$

We can now substitute these values for  $a$  and  $b$  into equation 31 to solve for  $u$ ,

$$u = a.x'' + b.x' + x$$

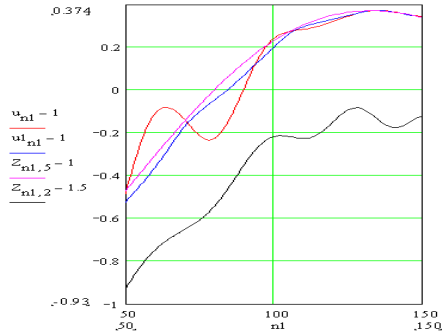
## Results and Discussion

**Algorithm 3 using Constant Parameters:** This algorithm uses a single time point but two further time derivatives compared to Algorithm-1. The filter cascade is increased by one again to provide a continuous estimate of the 4<sup>th</sup> time derivative  $x''''$ . The separation problem disappears altogether now to provide a continuous estimate of all parameters at each point on the trajectory. Program 3 [Al-Dabass et al, 1999-a] was run and the result of the estimation shows fast and accurate convergence.

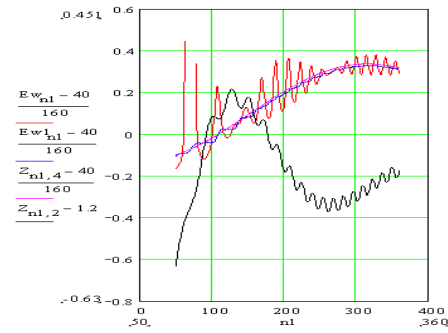
**Discussion.** Estimated values for constants parameters were close to the desired set values. The derived algorithms estimated  $\omega$ ,  $\zeta$  and  $u$  for a good range of values:  $\omega$  from 1 to 10,  $\zeta$  between +/- (0.01 to 1), and  $u$  between +/- (0.5-40), and gave accurate estimates. Estimation errors decreased as  $\omega$  increased, particularly for small  $\zeta$  (less than 0.5): where oscillation provided wide variation in the variables to decrease errors. The differences between the (simulated) system time derivatives ( $x$ ,  $x'$  and  $x''$ ) and their estimates from the filter cascade depended on  $G$  (the cut-off frequency): high  $G$  provided more accurate estimation of derivatives but made the algorithms prone to noise and vice versa. Another disadvantage of high  $G$  from the simulation point of view is that simulation time increased considerably due to the integration routine adapting to ever smaller steps. The algorithms provided progressively faster convergence with Algorithm-3 being the fastest to converge.

**Results:** Further results to those shown earlier are obtained to test the case when one or two of the input causal parameters were changing. Figure 13 shows the results when  $u$  is changing and the result of tracking it using algorithms 3 and 4, - algorithm 3 results showing large oscillations while those for algorithm 4 show much smoother tracking. Figure 14 shows the results of both algorithms tracking the other input  $\omega$ . Again algorithm 4 is producing a far smoother estimate than the large oscillatory output of algorithm 3.

**Comments:** For a given range of parameters the algorithms worked well, being able to estimate the two causal parameters  $u$  and  $\omega$  with their temporal behaviour, i.e. track them while they are changing. The 1<sup>st</sup> order algorithm worked better than the constant one, Figure 13 shows a comparison of the two algorithms tracking  $\omega$ . Algorithms of higher order than 1<sup>st</sup> showed marginal improvement but in certain cases showed a deteriorating behaviour, Figure 14 shows a 3<sup>rd</sup> order algorithm deviating quite markedly from the true trajectory compared to a 1<sup>st</sup> order algorithm. This is likely to be due to an accumulation of errors in higher derivative values used in the former algorithm.



**Figure 13.** The  $u$  causal parameter (smooth trace) being tracked using algorithm-3 (top sinusoid-like trace) and algorithm-4 (gentle wavy immediately below  $u$  trace). The black trace is the node output trajectory.



**Figure 14.** The  $\omega$  causal parameter (smooth trace) being tracked using algorithm-3 (top jagged trace) and the new algorithm (entwined with  $\omega$  trace). The bottom trace is the node output trajectory.

## KNOWLEDGE MINING FOR ECONOMIC SIGNAL PROCESSING APPLICATIONS

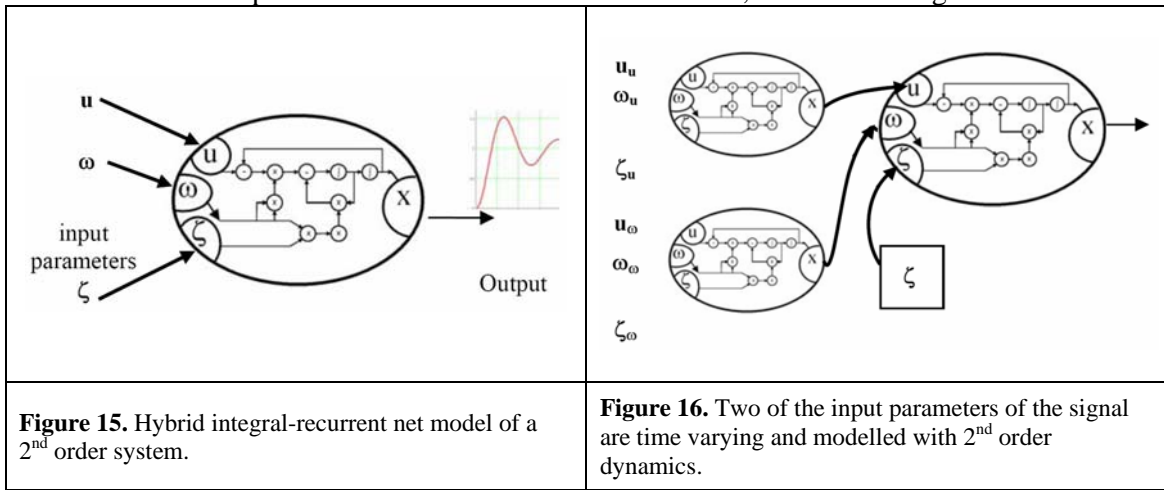
A special sixth order dynamical model is proposed to simulate the behaviour of complex signals. The model consists of a two layer hierarchy of second order dynamics two of whose parameters are themselves second order. Given the trajectory of the actual complex signal, a recurrent hybrid algorithm is derived to estimate the parameters of the model. Results show good performance of the algorithm in tracking the model parameters online. Suggestions for future directions are given

The algorithms derived earlier combine estimates of a given trajectory time derivatives, using data from several points on the trajectory, with explicit static non-linear functions to provide continuous parameter estimation in real time. For time varying parameters, the time separation between the points on the trajectory directly influences the estimation accuracy. This due to the assumption of constant parameters used in the derivation is no longer valid, and accuracy deteriorates with increasing rate of parameter variation. This is termed the separation effect, which can only be eliminated if all data is obtained from a single time point.

An algorithm was derived and proved, as expected, to be the most successful in coping with high rates of parameter variation. Accurate tracking of parameters when two of the parameters were varying simultaneously still proved difficult. The constant parameters assumption in the derivation is seen as the fundamental cause here.

### Sixth Order Models Of Compound Signals

Hierarchical Second Order Models: The signal trajectory is more complicated than can be represented by a simple second order differential model. In this case each parameter may itself be modelled as having dynamics, which may or may not be oscillatory. One such case is where two of the 3 parameters have 2<sup>nd</sup> order characteristics, as shown in Figure 15.



The 2<sup>nd</sup> order model of a node in a given layer in the hierarchy is given by:

$$\omega^{-2} x'' + 2. \zeta. \omega^{-1}. x' + x = u$$

To model complicated signals let both u and omega have their own 2<sup>nd</sup> order dynamics. The input **u** is the output of the following 2<sup>nd</sup> order system:

$$\omega_u^{-2} \mathbf{u}'' + 2. \zeta_u. \omega_u^{-1}. \mathbf{u}' + \mathbf{u} = \mathbf{u}_u$$

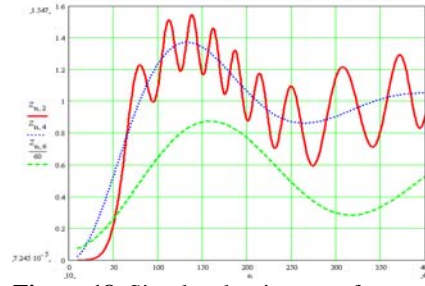
The natural frequency **ω** is the output of the following 2<sup>nd</sup> order system:

$$\omega_\omega^{-2} \boldsymbol{\omega}'' + 2. \zeta_\omega. \omega_\omega^{-1}. \boldsymbol{\omega}' + \boldsymbol{\omega} = \mathbf{u}_\omega$$

Thus the behaviour trajectory is generated by the following 6<sup>th</sup> order vector differential equation (using Runge Kutta in Mathcad for this example), see Figure 17.

$$D(t, x) := \begin{bmatrix} x_2 \\ x_5 \cdot x_5 \cdot x_3 - 2 \cdot z \cdot x_5 \cdot x_2 - x_5 \cdot x_5 \cdot x_1 \\ x_4 \\ (wu \cdot wu \cdot uu) - 2 \cdot zu \cdot wu \cdot x_4 - wu \cdot wu \cdot x_3 \\ x_6 \\ (ww \cdot ww \cdot uw) - 2 \cdot zw \cdot ww \cdot x_6 - ww \cdot ww \cdot x_5 \end{bmatrix}$$

**Figure 17.** Simulation vector of a 6<sup>th</sup> order trajectory (top 2 rows) with u (rows 3 and 4) and omega (rows 5 and 6) of the signal having 2<sup>nd</sup> order dynamics.



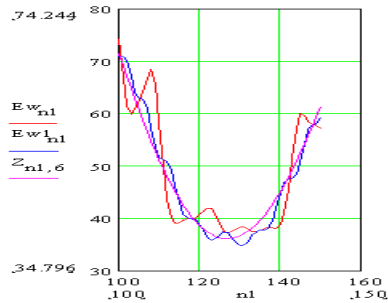
**Figure 18.** Simulated trajectory of a complex signal of a sixth (oscillating trace), with 2 variable inputs: u (top) and omega (bottom).

Where  $x_1$  and  $x_2$  represent the  $x$  and  $x'$ ,  $x_3$  and  $x_4$  represent  $u$  and  $u'$ , and  $x_5$  and  $x_6$  represent  $\omega$  and  $\omega'$  respectively. To generate the trajectory shown in Figure 18, the following values were used: for the  $u$  subsystem,  $u$  started from 0 aiming at  $u=1$  at a rate of  $\omega_u = 5$  rad/s with  $\zeta_u = 0.3$ . For the  $\omega$  subsystem,  $\omega$  started from 4 rad/s aiming at  $u_\omega = 32$  rad/s at a rate of  $\omega_\omega = 4$  rad/s with  $\zeta_\omega = 0.1$ . The resulting compound trajectory of  $x$  (red), together with the trajectories for  $u$  (blue dotted) and  $\omega$  (green dotted) are shown in the graph below.

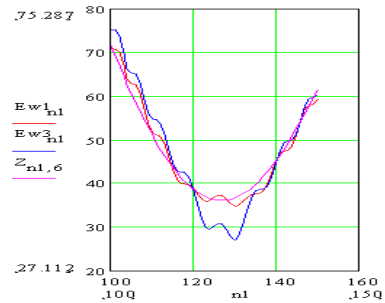
## Results and Discussion

Mathcad routines were set up to generate the input  $u$  as second order system with its own parameters of natural frequency, damping ratio and input. The input subsystem damping ratio was set to 0.05 to generate an oscillatory behaviour for long enough to test the parameter tracking algorithm thoroughly. The frequency of the input was set to 10 radians per second; the frequency of the main signal, on the other hand, started from 20 and aimed at 80 with a peak of about 130 radians. The derivative generation cascade was increased by one to produce the fifth time derivative.

Tracking Two Parameters: For a given range of parameters the algorithm worked well, being able to estimate the two input parameters  $u$  and  $\omega$  with their time varying behaviour, i.e. track them while they are changing. The 1<sup>st</sup> order algorithm worked better than the constant one, Figure 19 shows a comparison of the two algorithms tracking  $\omega$ . Algorithms of higher order than 1<sup>st</sup> showed marginal improvement but in certain cases showed a deteriorating behaviour, Figure 20 shows a 3<sup>rd</sup> order algorithm deviating quite markedly from the true trajectory compared to a 1<sup>st</sup> order algorithm. This is likely to be due to an accumulation of errors in higher derivative values used in the former algorithm.



**Figure 19.** Estimated Omega assuming constant parameters (Ew, very jagged trace) and first order parameters (Ew1, jagged trace) compared to actual (smooth trace).



**Figure 20.** Deterioration of accuracy of estimated Omega with higher order algorithms: 3<sup>rd</sup> order estimate (Ew3, very jagged trace) is worse than 1<sup>st</sup> order estimate (Ew1, jagged), actual (smooth).

## CONCLUSIONS

A model for hybrid logic nets was put forward to model the complex behaviour of economic systems. To estimate the values of the causal parameters a number of parameter knowledge mining algorithms were presented. Two of the algorithms used multiple points from the trajectory, 3 for the first algorithm and 2 points for the second. Two single point algorithms were presented: one that assumed constant parameters and used higher time derivatives of the trajectory (up to 4<sup>th</sup>), and a second algorithm that used additional information from a 5<sup>th</sup> time derivative of the trajectory to allow one of the parameters, the input parameter  $u$ , to have a non zero first order time derivative.

The fourth algorithm was tested for its ability to track the input parameter for a reduced order model. The test involved the generation of a lightly damped second order recurrent net. The results showed the algorithm maintaining good tracking over an extended period of time. This algorithm proved to be far superior to the third algorithm which relied on the assumption of constant input in the derivation.

**Biography of the author:** David Al-Dabass holds the chair of Intelligent Systems in the School of Computing & Informatics, Nottingham Trent University. He is a graduate of Imperial College, holds a PhD and has held post-doctoral and advanced research fellowships at the Control Systems Centre, UMIST. He is Fellow of the IEE, IMA and BCS and editor-in-chief of the International Journal of Simulation: Systems, Science and Technology. He currently serves as chairman of the UK Simulation Society and on the European Simulation Council for SCS Europe. For more details see his website: <http://ducati.doc.ntu.ac.uk/uksim/dad/webpage.htm>

## REFERENCES

Al-Dabass, D., Zreiba, A., Evans, D.J., & Sivayoganathan, K. (1999-a). Simulation of Three Parameter Estimation Algorithms for Pattern Recognition Architecture. UKSIM'99. Conference Proceedings, UK Simulation Society, St Catharine's College, Cambridge. 170-176. ISBN 0-905488-38-5.

- Al-Dabass, D., Zreiba, A., Evans, D.J., & Sivayoganathan, K. (1999-b). Simulation of Noise Sensitivity of Parameter Estimation Algorithms. Simulation'99 Workshop, UCL, London, 29 October 1999. 32-35.
- Al-Dabass, D. (2001-a). A Kalman Observer Computational Model for Metaphor Based Creativity. Panel paper, Workshop on Creative Systems, ICCBR2001, Simon Fraser University, Vancouver, 31 July, available online:  
<http://ducati.doc.ntu.ac.uk/uksim/dad/webpagepapers/Paper-1.doc>.
- Al-Dabass, D., Zreiba, A., Evans, D. J., & Sivayoganathan, K. (2002-a). Parameter Estimation Algorithms for Hierarchical Distributed Systems. *I. J. of Computer Mathematics*, 79(1), 65-88. ISSN 0020-7160.
- Al-Dabass, D., Evans, D. J., & Sivayoganathan, K. (2002-b). Derivative Abduction using a Recurrent Network Architecture for Parameter Tracking Algorithms. *IEEE 2002 Joint Int. Conference on Neural networks, World Congress on Computational Intelligence*, pp1570-1575, Hawaii, May 12-17.
- Al-Dabass, D., Zreiba, A., Evans, D.J., & Sivayoganathan, K. (2002-c). Parameter Estimation Algorithms for Hierarchical Distributed Systems. *I. J. of Computer Mathematics*. 79(1), 65-88. ISSN 0020-7160.
- Al-Dabass, D., Evans, D.J., & Sivayoganathan, K. (2002-d). Derivative Abduction using a Recurrent Network Architecture for Parameter Tracking Algorithms. *IEEE 2002 Joint Int. Conference on Neural networks, World Congress on Computational Intelligence, (IJCNN02)*, 12-17 May 2002, Honolulu, Hawaii, 1570-74. ISBN 0-7803-7278-6. ISBN 0-7803-7279-4 and ISBN 0-7803-7281-6 (CD-Rom).
- Al-Dabass, D., Evans, D. J., & Sivayoganathan, K. (2002-e). A Recurrent Network Architecture for Non-linear Parameter Tracking Algorithms. *2<sup>nd</sup> Int. Conference on Neural, Parallel, And Scientific Computations*, Morehouse College, Atlanta, U.S.A., August 07-10. 167-170. ISBN 0-9640398-2-6 (v.2).
- Al-Dabass, D. (2002-f). Modelling the Complexity of Concept Dynamics, *47th Meeting Of The International Society for the Systems Sciences*, 2<sup>nd</sup> – 6<sup>th</sup> August, Shanghai International Convention Centre, Shanghai, China.
- Al-Dabass, D., Evans, D.J., & Sivayoganathan, K. (2003). Signal Parameter Tracking Algorithms using Hybrid Recurrent Networks. *I. J. of Computer Mathematics*. 80(10), 1313 – 1322. ISSN 0020-7160 print, ISSN 1029-0265.
- Bailey, S., Grossman, R. L., Gu, L., & Hanley, D. (1996). A Data Intensive Approach to Path Planning and Mode Management for Hybrid Systems. Alur, R., Henzinger, T.A., & Sontag, E.: *Hybrid Systems III, Proceedings of the DIMACS Workshop on Verification and Control of Hybrid Systems*. Springer-Verlag, LNCS 1066.
- Berndt, D.J., & Clifford, J. (1996). Finding Patterns in Time Series: A Dynamic Programming Approach. *Advances in Knowledge Discovery and Data Mining*. Ed. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). AAAI Press/MIT Press. 229-248.

Bovet, D. P., & Crescenzi, P. (1994). Introduction to the Theory of Complexity. Prentice Hall. ISBN 0-13-915380-2.

Cant R., Churchill, J., & Al-Dabass, D. (2001). Using Hard And Soft Artificial Intelligence Algorithms To Simulate Human Go Playing Techniques. I. J. of Simulation, 2(1), 31-49. ISSN 1473-804x Online. ISSN 1473-8031 Print.

Cawley, P. (1984). The reduction of Bias Error in Transfer Function Estimates using FFT-based Analysers. Journal of Vibration, Acoustics, Stress and Reliability in Design, 29-35.

Close, C. M. & Fredrick, D. K., (1994). Modelling and Analysis of Dynamic Systems, 2<sup>nd</sup> edn. Wiley.

Dewolf, D., & Wiberg, D. (1993). An Ordinary Differential-Equation Technique for Continuous Time Parameter Estimation. IEEE Trans. Auto. Control. 38(4), 514-528.

Gersch, W. (1974). Least Squares Estimates of Structural System Parameters using Covariance Function Data. EEE Trans. on Auto. Control. 19(6).

Kailath, T. (1978). Lectures on Linear Least-Squares Estimation. CISM courses and lectures No. 140, Spring-Verlag, New York.

Kalman, R. (1960). A New Approach to Linear Filtering and Prediction Problems. Tans. Of SAME: Journal of Basic Eng., series D, 82, 35-45.

Mannila, H., Toivonen, H., & Verkamo, I. (1997). Discovery of Frequent Episodes in Event Sequences. Data Mining and Knowledge Discovery. 1, 259-289.

Man, Z. (1995). Parameter-Estimation of Continuous Linear Systems using Functional Approximation. Computers and Electrical Eng'g. 21(3), 183-187.

Ren, M., & Al-Dabass, D. (1995). An Associative Memory Artificial Neural Network System. ECAC'95-London: Proceedings of European Chinese Automation Conference, London, UK, 1995, pp.91-96.

Ren, M., Al-Dabass, D. , & Su, D. (1996). A Three-Layer Hierarchy for Representing Chinese Characters. Expert Systems 96, 6th Annual Technical Conference of the BCS Specialist Group on Expert Systems, Cambridge, UK, 1996, pp.137-146.

Ren, M., & Al-Dabass, D. (2001). Simulation Of Fuzzy Possibilistic Algorithms For Recognising Chinese Characters. I. J. of Simulation, 2(1), 1-13. ISSN 1473-804x Online. ISSN 1473-8031 Print.

Schank, R. C., & Colby, K. M., (editors) (1973). Computer Models of Thought and Language. Freeman & Co., ISBN 0-7167-0834-5.

Seveance, F. L. (2001). Systems Modelling and Simulation, An Introduction. Wiley.

## **Books:**

Baltagi, B. (2005). *Economic Analysis of Panel Data*. ISBN: 0470-0145-63.

Doyle, E. (2005). *The Economic System*. ISBN: 0470-8500-19.

Koop, G. (2005). *The Analysis of Economic Data*. ISBN: 0470-0246-82.

Papazoglou, M., & Ribbers, P. (2005). *e-Business: Organizational and Technical Foundations*. ISBN: 0470-8437-64.

Samuelson, W. (2005). *Managerial Economics*. ISBN: 0471-6636-2X.