Monograph:
Billings, S.A. and yANG, Y.X. (1999) Identification of the Neighbourhood and CA Rules from Spatio-temporal CA Patterns. Research Report. ACSE Research Report 751 . Department of Automatic Control and Systems Engineering

Identification of the Neighbourhood and CA Rules from Spatio-temporal CA Patterns

S.A.Billings Y.X.Yang

Department of Automatic Control and Systems Engineering
University of Sheffield
Mappin Street, Sheffield S1 3JD
United Kingdom

Research Report No.751

July 1999

University of Sheffield

Identification of the Neighbourhood and CA Rules from Spatio-temporal CA Patterns

S.A.Billings Y.X.Yang

Dept. of Automatic Control and Systems Engineering

Univ. of Sheffield, Mappin Street, Sheffield S1 3JD, UK

Abstract

Extracting the rules from spatio-temporal patterns generated by the evolution of Cellular Automata (CA) usually produces a CA rule table without providing a clear understanding of the structure of the neighbourhood or the CA rule. In the present paper a new identification method based on using a modified orthogonal least squares or CA-OLS algorithm to detect the neighbourhood structure and the underlying polynomial form of the CA rules is proposed. The Quine-McClauskey method is then applied to extract minimum Boolean expressions from the polynomials. Spatio-temporal patterns produced by the evolution of one-, two- and higher-dimensional binary CA's are used to illustrate the new algorithm and simulation results show that the CA-OLS algorithm can quickly select both the correct neighbourhood structure and the corresponding rule.

1 Introduction

Cellular Automata (CA) represent an important class of models that evolve in time over a spatial lattice structure of cells. CA's have been applied in image processing [1], pattern recognition [2], digital circuit design [3] and robotics [4]. Many authors have demonstrated that relatively simple binary CA rules can produce highly complex patterns of behavior. These results illustrate the potential of CA's as a model class and suggest that it may be possible to model even very complex spatio-temporal behavior using CA models of a simple form. But very few studies have investigated how these rules can be extracted from observed patterns of spatio-temporal behavior.

Ideally the identification technique should produce a concise expression of the rule. This ensures that the model is parsimonious and can readily be interpreted either for simulation or hardware realization of the CA. Sequential and parallel algorithms for computing the local transition table were presented by Adamatskii [5], and Richards [6] introduced a method using Genetic Algorithms (GA's). However, no clear structure of the related neighbourhoods was obtained in either of these studies and the detection process was complicated and time consuming. Genetic Algorithms were also employed in [7] to determine the rules as a set of logical operators. Parsimonious local rules were found for low-dimensional CA's , but the identification was computationally demanding.

In the present study a totally new approach is adopted to identify both the neighbourhood and the CA rule from complex patterns of high-dimensional spatio-temporal behavior. Identifying the CA rule or model is considered as a two stage procedure. First the neighbourhood which defines the spatial interaction of the cells over a temporal window

is determined and then the rules that specify the values of these cells is estimated. Earlier studies [5], [6], [7] have attempted to devise solutions to these problems based on the logical rule base which defines binary CA's. But this involves determining the spatio-temporal rules as nonlinear combinations of cellular values.
In the present research this problem is avoided by exploiting the fact that the binary rules can be expressed as Boolean functions and showing that these can be exactly represented using simple polynomial models. The main advantage of this is that now the problem is mapped into a linear-in-the-parameters model. A modified orthogonal least squares algorithm, called the CA-OLS method is introduced which determines the neighbourhood and the unknown model parameters. The Quine-McClauskey algorithm can then be applied to extract the minimum Boolean expression to produce the final CA model. Mapping the problem into a polynomial model form, determining the structure and parameters, and then mapping back to a logical expression produces for the first time a powerful method for determining the rules of high-dimensional CA's in the form of a parsimonious model. This is achieved from just the observations of the data and no a priori information.
The remainder of the paper is organized as follows. In Section 2, the definition, characteristics and other relevant background information of one-, two- and higher-dimensional cellular automata are introduced. Section 3 introduces the crucial link between boolean expressions and polynomial model equivalents of binary CA rules. The CA-OLS technique is then derived to determine the neighbourhood and polynomial model parameters. Finally the Quine-McClauskey method is employed to extract the minimum Boolean expression from the polynomial model. Simulation results including four-dimensional CA's are presented in Section 4, and the conclusions are in Section 5.

2 Cellular Automata and the difficulties of CA identification

A cellular automaton is defined by three parts: a neighbourhood, a local transition rule and a discrete lattice structure consisting of a large number of cells which are occupied by states from a finite set of discrete values. The local transition rule updates all cells synchronously by assigning to each cell, at a given time step, a value which depends only on the neighbourhood.
Attention in this paper is restricted to binary cellular automata where the cells can only take binary values. Although binary CA's form one of the simplest classes of CA's, they have been the focus of most investigations and are capable of generating complicated patterns of global behavior and capturing the essential features of many complex phenomena.

2.1 CA neighbourhoods

The neighbourhood of a cell is the set of cells directly involved in the evolution of the cell. Sometimes this includes the cell itself. The neighbourhood structure varies depending on the construction of the cellular automata. Consider a one-dimensional 3-site CA for example. Denoting the cell at position j at time step t as $cell(j;t)$, then the neighbourhood

of $cell(j;t)$ could be a von Neumann neighbourhood illustrated in Figure 1 (a) or the two exotic neighbourhoods shown in Figure 1 (b) and (c) respectively. The neighbourhood can involve cells from different spatial and temporal scales. The exotic neighbourhood in Figure 1 (b) encompasses cells from the same temporal scale but different spatial scale than the cells in the von Neumann neighbourhood while the neighbourhood in Figure 1 (c) involves cells from the same spatial scale but different temporal scale from the cells in the von Neumann neighbourhood.

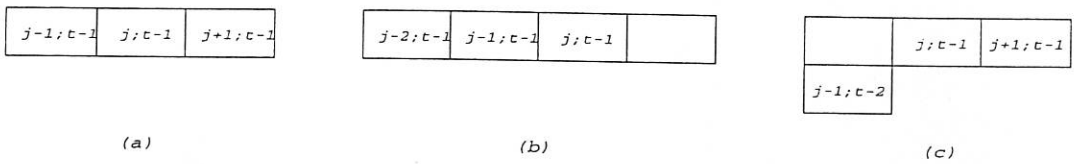


Figure 1: Examples of 3-site neighbourhoods for a one-dimensional CA (a) von Neumann neighbourhood, (b) and (c) exotic neighbourhoods

There are many more possible neighbourhood structures for two-dimensional cellular automata. The most commonly used are the 5-site von Neumann neighbourhood and the 9-site Moore neighbourhood, illustrated in Figure 2 (a) and (b).

Figure 2: Examples of neighbourhoods for a two-dimensional CA (a) 5-site von Neumann neighbourhood (b) 9-site Moore neighbourhood

The neighbourhood structures for higher-dimensional cellular automata are much more complicated and diverse than the one- and two-dimensional cases. Figure 3 shows a simple example of the neighbourhood for a three-dimensional cellular automata.

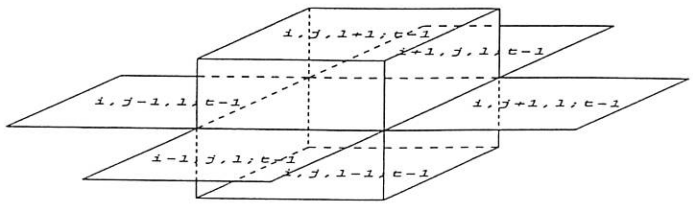


Figure 3: An example of the neighbourhood for a three-dimensional CA

Neighbourhoods for two- and higher-dimensional CA can also involve cells from temporal scales other than $t-1$. For example, a 5-site two-dimensional neighbourhood could take

the form $\{cell(i,j-1;t-3), cell(i,j-1;t-2), cell(i,j-1;t-1), cell(i+1,j;t-1), cell(i,j+1;t-1)\}$. Clearly the number of possible cells in a multi-dimensional CA over a range of temporal scales can be huge.

2.2 Local CA transition rules

Local transition rules can be defined in several equivalent ways. The most common method is to use a transition table analogous to a truth table where the first row describes the states of the neighbourhood and the second row indicates the next state of the cells. The rules are then labelled by specifying which neighbourhoods map to 0 and which to 1. The standard form of a 3-site one-dimensional rule R is shown below

000	001	010	011	100	101	110	111
r_0	r_1	r_2	r_3	r_4	r_5	r_6	r_7

where $r_i, i = 0, \cdots, 7$ indicating the next-states of the cells. Every component r_i corresponds to a coefficient 2^i which is essential in computing the numerical label associated with the rule. The numerical label D assigned to rule R above is therefore given by $D(R) = \sum_{i=0}^{2^3-1} r_i 2^i$, which is simply the sum of the coefficients associated with all the non-zero components.

2.3 Difficulties in CA identification

Many studies of CA have focussed on demonstrating that relatively simple CA rules can produce complex patterns of behavior. This demonstrates the potential of CA's as a model class but shows that this can only be realized if the simple underlying rules can be determined from observed spatio-temporal behavior.
When identifying CA rules the only a priori knowledge will be the spatio-temporal patterns produced by the evolution of the CA. Realistically the neighbourhood structure including the size will be unknown and this means that the possible combinations can number into the hundreds of millions. It may be possible to find simple CA rules by searching through the rule space when only one- or two-dimensional CA's with very small neighbourhoods are involved. But as the neighbourhood size, or the dimensionality, or both increase the combinational possibilities become huge. The few authors that have studied this problem [5], [6] have therefore focussed on a very limited class of low-dimensional CA's. But there is a clear need to develop procedures which can operate on observed data from CA's over higher-dimensional spatial and temporal neighbourhoods with no a priori information.

3 Identification using the CA-OLS method

In the present study the problem of searching for the neighbourhood and then the parameter values associated with a nonlinear logical model will initially be mapped into an equivalent polynomial representation. While this relationship is well known it has not previously been appreciated that the polynomial model can be collapsed to a very simple structure with integer parameters, even for high-dimensional and complex CA's .

Using this model form and introducing a modified orthogonal least squares routine, called the CA-OLS method, both the CA neighbourhood and the unknown polynomial model parameters can easily be determined. The equivalent polynomial model is then mapped back to a minimal logical expression to yield the final parsimonious CA model. The steps associated with this new procedure are introduced below.

3.1 Boolean form of CA rules

The local rule for an binary cellular automaton may also be considered as a Boolean function of the cells within the neighbourhood. For a one-dimensional CA, denote the state of the cell at position j at time step t as $s(j;t)$ and the states of the cells within the neighbourhood of cell j at previous time steps as $\mathbf{N}(j;|t)$ where $|t$ represents time steps before t. The one-dimensional CA can then be represented by

$$s(j;t) = f(\mathbf{N}(j;|t)) \tag{1}$$

where f is the Boolean form of the local transition rule.
Two different ways of constructing Boolean rules are currently available. One formulation produces Boolean rules using only the NOT, AND and OR logical operators and rules for all one-dimensional CA with 3-site neighbourhoods are listed in [8]. The Boolean form of $Rule30$, for example, is

$$\begin{aligned} s(j;t) \;=\; & (s(j-1;t-1) * \bar{s}(j;t-1) * \bar{s}(j+1;t-1)) \\ & || \, (\bar{s}(j-1;t-1) * s(j;t-1)) \, || \, (\bar{s}(j-1;t-1) * s(j+1;t-1)) \end{aligned} \tag{2}$$

where $\bar{}$, $*$ and $||$ denote NOT, AND and OR operators respectively.
The alternative formulation uses only the NOT, AND and XOR operators instead. Lists of Boolean expressions of even number one-dimensional 3-site CA rules based on this formulation can be found in [9]. Using these operators $Rule30$ can also be represented as

$$s(j;t) = s(j-1;t-1) \oplus s(j;t-1) \oplus (\bar{s}(j;t-1) * s(j+1;t-1)) \tag{3}$$

where $\oplus$ denotes the XOR operator.
Note that on the right side of equation (3) logical terms are produced by connecting the 'states' or the 'NOT states' of the cells within the neighbourhood using AND operators which are then combined by XOR operators. It can easily be observed that every one-dimensional binary rule can be reformulated into a Boolean form which follows this principle.
Furthermore, note that $\bar{a} = 1 \oplus a$, $0 \oplus a = a$. Hence all one-dimensional binary cellular automata can be represented by a Boolean function with only AND and XOR operators. For example a CA with an n-site neighbourhood of $\{cell(j+1;t-1), \cdots, cell(j+n;t-1)\}$ can be expressed in the form

$$s(j;t) = a_0 \oplus a_1 s(j+1;t-1) \oplus \cdots \oplus a_N(s(j+1;t-1) * \cdots * s(j+n;t-1)) \tag{4}$$

where a_i $(i = 0, \cdots, N,\ N = 2^n - 1)$ are binary numbers and $a_i = 1$ indicates that the following term is included in the Boolean function while $a_i = 0$ indicates that the following term is not included.
Note that the number of possible expressions in equation (4) is 2^{2^n} which is exactly the number of all possible one-dimensional rules with that particular neighbourhood. This implies that the representation in equation (4) is unique, one set of $\{a_i, i = 0, \cdots, N\}$ corresponds to one and only one CA rule.
Equation (1) can be extended to higher-dimensional CA's. For a three-dimensional CA, this would be denoted as

$$s(i, j, l; t) = f(\mathbf{N}(i, j, l; |t))$$

where $s(i, j, l; t)$ is the state of the cell at position (i, j, l) at time step t and $\mathbf{N}(i, j, l; |t)$ represents the states of the cells within the neighbourhood of $cell(i, j, l; t)$.
The unified expression in equation (4) can also be extended to multi-dimensional CA's. For example any three-dimensional CA with a 5-site neighbourhood $\{cell(i-1, j, l; t-1), cell(i, j, l; t-1), cell(i, j+1, l; t-1), cell(i, j, l-1; t-1), cell(i, j, l+1; t-1)\}$ can be represented by a Boolean expression

$$s(i, j, l; t) = a_0 \oplus a_1 s(i-1, j, l; t-1) \oplus \cdots \oplus a_{31}(s(i-1, j, l; t-1) * \cdots * s(i, j, l+1; t-1)) \quad (5)$$

Extending this further, every CA with an n site neighbourhood $\{cell(x_1; |t), \cdots, cell(x_n; |t)\}$ may be written as

$$s(x_j; t) = a_0 \oplus a_1 s(x_1; |t) \oplus \cdots \oplus a_N(s(x_1; |t) * \cdots * s(x_n; |t)) \quad (6)$$

where $N = 2^n - 1$ and $cell(x_j; t)$ is the cell to be updated.
Equation (6) is important because it significantly reduces the complexity of CA identification by using a reduced set of logical operators. The difficulty in identifying multi-dimensional CA's is also decreased because the higher-dimensional CA rules are reduced to an equation which depends on the size of the neighbourhood not the dimensionality.

3.2 Polynomial form of CA rules

Every CA with an n site neighbourhood can be reformulated from a truth table to a Boolean function of the form of equation (6). However the model to be identified is defined in terms of AND and XOR operators and is therefore nonlinear in the parameters. But it is often advantageous to reconfigure the nonlinear model to be linear in the parameters if this is possible. This will be investigated below for CA's.

Table 1. Relationship between NOT, AND, XOR and $+, \times$

a	$\bar{a}$	$1-a$	a_1	a_2	$a_1 * a_2$	$a_1 \times a_2$	a_1	a_2	$a_1 \oplus a_2$	$a_1 + a_2 - 2a_1 \times a_2$
0	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1	1
			1	0	0	0	1	0	1	1
			1	1	1	1	1	1	0	0

If a, a_1, a_2 are binary integer variables taking the values 0 and 1 for true and false respectively, then there is an exact polynomial representation of each of the logical functions

$$\bar{a} = 1 - a, \quad a_1 * a_2 = a_1 \times a_2, \quad a_1 \oplus a_2 = a_1 + a_2 - 2a_1 \times a_2$$

This is obvious from the truth table Table 1.

Therefore all CA rules can be represented by exact polynomial expressions. The one-dimensional von Neumann *Rule*30, for example, can be written as $s(j;t) = \sum_{i=1}^{13} b_i$, where $b_1 = s(j-1;t-1)$; $b_2 = s(j;t-1)$; $b_3 = s(j+1;t-1)$; $b_4 = -2s(j-1;t-1) \times s(j;t-1)$; $b_5 = -2s(j-1;t-1) \times s(j+1;t-1)$; $b_6 = -s(j;t-1) \times s(j+1;t-1)$; $b_7 = 2s(j-1;t-1) \times s(j;t-1) \times s(j+1;t-1)$; $b_8 = -2s^2(j;t-1) \times s(j+1;t-1)$; $b_9 = -2s(j;t-1) \times s^2(j+1;t-1)$; $b_{10} = 4s(j-1;t-1) \times s^2(j;t-1) \times s(j+1;t-1)$; $b_{11} = 4s(j-1;t-1) \times s(j;t-1) \times s^2(j+1;t-1)$; $b_{12} = 4s^2(j;t-1) \times s^2(j+1;t-1)$; $b_{13} = -8s(j-1;t-1) \times s^2(j;t-1) \times s^2(j+1;t-1)$.

But this equivalent expression will involve as many parameters as the number of possible combinations of all the cells within the neighbourhood and little will be gained by using such a representation.

However using the Principle of Duality and Absorption in Boolean Algebra [11] where for every binary variable a, $a \times a = a$, considerable simplification can be achieved. Therefore terms in the form of $s^{l_1}(j-1;t-1)s^{l_2}(j;t-1)s^{l_3}(j+1;t-1)$ where l_1, l_2, l_3 are integers can all be reduced to one term $s(j-1;t-1)s(j;t-1)s(j+1;t-1)$. Consequently applying the Principle of Duality and Absorption to all the terms results in a new expression for all one-dimensional CA's with von Neumann neighbourhood of the form

$$\begin{aligned} s(j;t) \;=\; & \theta_1 s(j-1;t-1) + \theta_2 s(j;t-1) + \theta_3 s(j+1;t-1) + \theta_4 s(j-1;t-1) \times s(j;t-1) \\ & + \theta_5 s(j-1;t-1) \times s(j+1;t-1) + \theta_6 s(j;t-1) \times s(j+1;t-1) \\ & + \theta_7 s(j-1;t-1) \times s(j;t-1) \times s(j+1;t-1) \end{aligned}$$

where the parameters $\theta_1, \cdots, \theta_7$ can only take integer values and $s(j-1;t-1), s(j;t-1), s(j+1;t-1)$ are binary variables.

Applying this to equation (6) shows that a general polynomial expression of all binary CA rules with an n-site neighbourhood $\{cell(x_1;|t), \cdots, cell(x_n;|t)\}$ can be expressed by the exact polynomial expression

$$s(x_j;t) = \theta_1 s(x_1;|t) + \cdots \theta_n s(x_n;|t) + \cdots + \theta_N s(x_1;|t) \times \cdots \times s(x_n;|t) \qquad (7)$$

where $N = 2^n - 1$ and $cell(x_j;t)$ is the cell to be updated. Using this important observation the number of parameters to be identified can be substantially reduced. It can also be seen that the most important factor is the size of the neighbourhood n, not the order of the dimension. For example, a two-dimensional CA rule with a 5-site neighbourhood may have a simpler polynomial expression than a one-dimensional CA rule with an 8-site neighbourhood.

These are important observations which surprisingly have not previously been exploited and which together with the CA-OLS algorithm introduced below provide a new and powerful method of reconstructing the CA model even for high-dimensional CA's.

3.3 Identification using CA-OLS

A CA can be viewed as a nonlinear dynamical system. Although the system has a spatio-temporal structure, a single time series can be measured at a single lattice site or a spatial series can be measured at a fixed time and traditional methods can be applied to model either. However Diks [10] showed that studying only a time series or a spatial series from a spatio-temporal system without any knowledge of the system can easily lead to the incorrect conclusion that there is no spatio-temporal structure. For a full characterization of the system structure time and space have to be considered simultaneously. Determination of the spatial and temporal span of the neighbourhood is therefore very important in identifying CA models.

In practice the neighbourhood structure will be unknown and it is necessary to extend the assumed neighbourhood to a more general case which encompasses cells from different spatial and temporal scales. Hence a set of models which are over-specified on both the spatial and temporal spans will be introduced as the model set. For a three-dimensional CA the model set can be defined as

$$\begin{aligned} s(i,j,l;t) &= f(s(i+i_1, j+j_1, l+l_1; t-1), \cdots, s(i-i_2, j-j_2, l-l_2; t-1); \cdots; \\ &\quad s(i+i_1, j+j_1, l+l_1; t-h), \cdots, s(i-i_2, j-j_2, l-l_2; t-h)) \end{aligned} \tag{8}$$

where i_1, i_2, j_1, j_2, l_1 and l_2 denote the maximum space scale the three-dimensional CA could possibly span and h denotes the maximum time scale the three-dimensional CA could possibly span. Reducing the dimensions in equation (8) will yield models for one- and two-dimensional CA's as special cases while increasing the dimensions will produce models for four-, five- and higher-dimensional CA's.

Finding the neighbourhood can now be defined as determining just the relevant or significant terms in equation (8) for the three-dimensional case and analogously for other dimensions. The neighbourhood can therefore be thought of as equivalent to the model structure in nonlinear system identification. Using equation (8) and the three dimensional case as an example the neighbourhood must be determined from a set of $2^{(i_1+i_2+1)(j_1+j_2+1)(l_1+l_2+1)h}-1$ possible candidate model terms. Consider for example a very simple three-dimensional CA where $i_1 = i_2 = j_1 = j_2 = l_1 = l_2 = h = 1$, this produces $2^{27} - 1 = 134,217,727$ candidate terms. This clearly shows the complexity of the task even for a simple three-dimensional case. Higher dimensions produce even more frightening numbers and clearly show why there are no existing solutions to these important problems. To overcome these problems the new CA-OLS algorithm is introduced below.

Initially using the three-dimensional case to illustrate the method, denote the states of the neighbourhood $\{s(i+i_1, j+j_1, l+l_1; t-1), \cdots, s(i-i_2, j-j_2, l-l_2; t-1), \cdots, s(i+i_1, j+j_1, l+l_1; t-h), \cdots, s(i-i_2, j-j_2, l-l_2; t-h)\}$ in equation (8) as $\{u_1, \cdots, u_n\}$, where the size of the neighbourhood $n = (j_1+j_2+1)(i_1+i_2+1)(l_1+l_2+1)h$. Then expanding equation (8) into the polynomial form shown in (6) yields

$$s(i,j,l;t) = \theta_1 u_1 + \cdots + \theta_n u_n + \cdots + \theta_N u_1 \times \cdots \times u_n \tag{9}$$

where $N = 2^n - 1$, and $\theta_1, \cdots, \theta_N$ are integer parameters to be identified. Note that equation (9) can be readily extended from the three-dimensional case to be valid for all binary CA's.
The CA-OLS algorithm is derived by applying a modified Gram-Schmidt orthogonal procedure to equation (9). The CA-OLS algorithm is given in the Appendix.
The simple three-dimensional example above shows the number of possible candidate terms can be excessive but simulations by many authors show that often complex CA patterns can be produced using simple models. If the appropriate terms that are significant can be selected therefore the remainder can be discarded without any deterioration in model precision or prediction accuracy and a concise CA model can be obtained. One way to determine which terms are significant or which should be included in the model can be derived as a by-product of the CA-OLS estimation algorithm and is very simple to implement. From the Appendix the quantity $[ct]$ is defined as

$$[ct]_d = \frac{\tilde{\theta}_d^2 \times \sum_{t=1}^{M} e_d^2(t)}{\sum_{t=1}^{M} s^2(i,j,l;t)}$$

and measures the contribution that each candidate term makes to the updated state $s(i,j,l;t)$ and provides an indication of which terms to include in the model. Using the term contribution $[ct]$ the candidate model terms can be ranked in order of importance and insignificant terms can be discarded by defining a value of $[ct]$, below which terms are considered to contribute a negligible reduction in the mean-squared error. The threshold value of $[ct]$ for the CA model can be set to 0 because the polynomial model is not an approximation but an exact representation of the CA rules. The threshold value is set to 0 to ensure that sufficient terms are included and the prediction errors are reduced to zero. Notice that the forward-regression orthogonal algorithm [12] is used in the Appendix, this provides a $[ct]$ test which is independent of the order of inclusion of terms in the model. The structure of the neighbourhood is therefore defined by retaining only the significant $[ct]$ terms and the CA rule can then be computed by linearly combining all the selected terms with the estimated parameters.

3.4 Extracting the Boolean form of the CA rules

The polynomial form of the model can be determined using the orthogonal estimator which yields both the CA neighbourhood and the model parameters. Although the polynomial model can be used to directly reproduce the complex spatio-temporal patterns, hardware realization of the CA may not be straightforward based on the polynomial form and it is therefore important to extract the equivalent Boolean rules from the polynomial representation.
While it is straightforward to extract canonical forms [11] of Boolean functions from truth tables constructed on the basis of polynomial rules the canonical forms are often unwieldy and typically more operations than are necessary are involved. However, this problem can be solved by using the Quine-McCluskey [11] method to extract the parsimonious Boolean expressions from identified polynomials and this will be illustrated using the simulated examples.

The Boolean rules extracted using the Quine-McCluskey method involves NOT, AND and OR operators. To obtain rules employing NOT, AND and XOR instead, see details in Chapter 1 in [9].

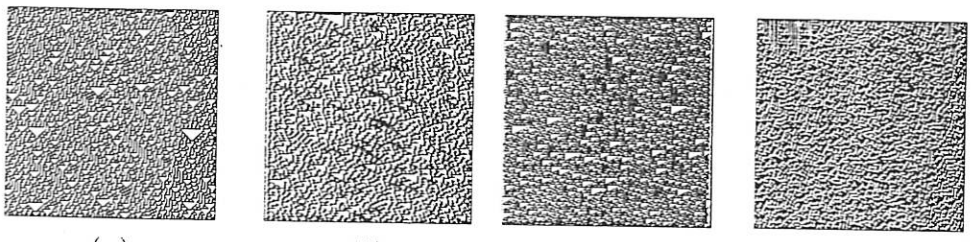
4 Simulation Studies

Four simulation examples are included to demonstrate the application of the new algorithm. Initially a simple one-dimensional example will be discussed to show all the steps involved in a transparent manner. More realistic two-, three- and four-dimensional examples will then be discussed.

4.1 Identification of one-dimensional 3-site CA Rule30

4.1.1 Complexity of spatio-temporal patterns produced by the evolution of one-dimensional CA

The spatio-temporal patterns generated by the evolution of $Rule30$ on a 200×200 lattice with four different neighbourhoods, a von Neumann neighbourhood $\{cell(j-1;t-1), cell(j;t-1), cell(j+1;t-1)\}$, a left-shift neighbourhood $\{cell(j-2;t-1), cell(j-1;t-1), cell(j;t-1)\}$, a right-shift neighbourhood $\{cell(j;t-1), cell(j+1;t-1), cell(j+2;t-1)\}$ and a temporal-shift neighbourhood $\{cell(j-1;t-2), cell(j;t-1), cell(j+1;t-1)\}$ are shown in Figure 4 (a), (b), (c) and (d) respectively.



(a) $Rule30(01111000)$ (b) $Rule30$ $left-shift$ (c) $Rule30$ $right-shift$ (d) $Rule30$ $temporal-shift$

Figure 4: Evolution of the one-dimensional CA Rule30 with four different neighbourhoods (a) a von Neumann neighbourhood (b) a left-shift neighbourhood (c) a right-shift neighbourhood and (d) a temporal-shift neighbourhood

An initial inspection of Figure 4 (a), (b), (c) and (d) shows that the structure of the neighbourhood corresponds to the pattern produced. The randomly distributed triangle structures in Figure 4 (b) are simply the left half of the triangles in Figure 4 (a), whereas Figure 4 (c) is composed of the right half of the triangles in Figure 4 (a). The patterns demonstrate the difference among these three neighbourhoods. The pattern in Figure 4

(d) is produced by operating *Rule*30 on a temporal-shift neighbourhood which involves cells from both time steps $t-1$ and $t-2$. However the blurred image and the rotated triangles that this produces barely shows any resemblance to the patterns in Figure 4 (a), (b) or (c).

Because of the special construction of cellular automata which are synchronously updated using the same Boolean function over the whole lattice, the data points that are available are redundant for identification purposes and can be extracted in two ways as shown in Figure 5 (a) and (b) respectively.

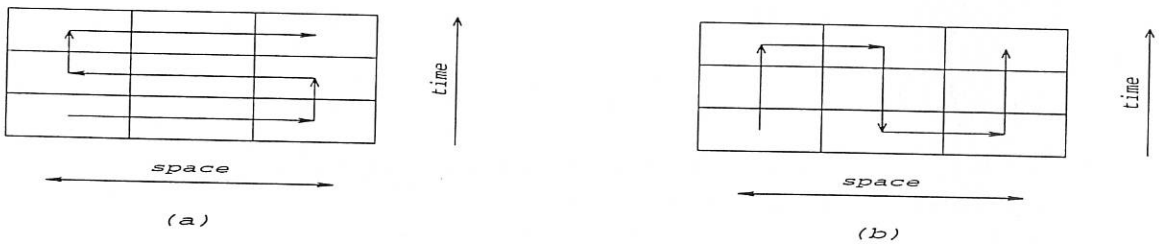


Figure 5: Extracting data points from one-dimensional CA patterns

In Figure 5 (a), data points are extracted row by row/space-wise, while in Figure 5 (b), data points are extracted column by column/time-wise. Since each cell is synchronously updated under the same Boolean function, a change of the rows or columns when extracting the data is not important.

Assume initially that the largest possible neighbourhood is defined by $\{cell(j-2;t-1), cell(j-1;t-1), cell(j;t-1), cell(j+1;t-1), cell(j+2;t-1), cell(j+1;t-2)\}$ and hence define the neighbourhood vector

$$a(t) = [\ s(j-2;t-1) \quad s(j-1;t-1) \quad s(j;t-1) \quad s(j+1;t-1) \quad s(j+2;t-1) \quad s(j+1;t-2)\ ]^T \quad (10)$$

The candidate model term set (MT) will initially be constructed as

$$MT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ & & \vdots & & & \\ 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ & & \vdots & & & \\ 5 & 6 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 \\ & & \vdots & & & \\ 4 & 5 & 6 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 0 & 0 \\ & & \vdots & & & \\ 3 & 4 & 5 & 6 & 0 & 0 \\ 1 & 2 & 3 & 4 & 5 & 0 \\ & & \vdots & & & \\ 2 & 3 & 4 & 5 & 6 & 0 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

where $1, 2, 3, 4, 5, 6$ denote the rows in the neighbourhood vector. For example, an entry

of '4' represents the fourth row in equation (10) and is therefore associated with $cell(j+1;t-1)$, and so on. The full model set MT consists of $N = 63$ terms/rows. Each row in this model represents a candidate term which corresponds to a term $s_d, d = 1, \cdots, N$ in equation (13) in the Appendix. For example, the first row (1 0 0 0 0 0) represents $s(j-2;t-1)$ only while the last row (1 2 3 4 5 6) corresponds to a product of six states $s(j-2;t-1) \times s(j-1;t-1) \times s(j;t-1) \times s(j+1;t-1) \times s(j+2;t-1) \times s(j+1;t-2)$. Five hundred data points were extracted from the patterns in Figure 4 (a), (b), (c) and (d) respectively and these were used to fit the models. No a priori information regarding the neighbourhoods or rules was assumed. In the simulation the threshold for the term contribution $[ct]$ values for the four models were all chosen as 0 and the CA-OLS estimator searched through 63 possible candidate terms for each model. Finally four different models were selected which were associated with four different neighbourhoods. The models and the corresponding parameters θ are shown in models $1-(a)$, $1-(b)$, $1-(c)$ and $1-(d)$.

$$MT = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 4 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix}$$

Model $1-(a)$

$$MT = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix}$$

Model $1-(b)$

$$MT = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 5 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix}$$

Model $1-(c)$

$$MT = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 \\ 3 & 6 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 6 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \theta = \begin{bmatrix} 1.0000 \\ -2.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \end{bmatrix}$$

Model $1-(d)$

The terms in model $1-(b)$ represent the left-shift neighbourhood because all the model entries are selected from $\{cell(j-2;t-1), cell(j-1;t-1), cell(j;t-1)\}$. Notice how the CA-OLS algorithm has correctly selected only the appropriate three cells and discarded the remainder. Combining the terms with the corresponding parameters θ , the identified model describes the CA rule in the polynomial form

$$\begin{aligned} s(j;t) \;=\; & s(j-1;t-1) - 2s(j-2;t-1) \times s(j-1;t-1) + s(j-2;t-1) \\ & -2s(j-2;t-1) \times s(j;t-1) + s(j;t-1) + 2s(j-2;t-1) \\ & \times s(j-1;t-1) \times s(j;t-1) - s(j-1;t-1) \times s(j;t-1) \end{aligned} \tag{11}$$

The terms in model $1-(a)$ represent the von Neumann neighbourhood $\{cell(j-1;t-1), cell(j;t-1), cell(j+1;t-1)\}$ while the terms in model $1-(c)$ correspond to the right-shift neighbourhood $\{cell(j;t-1), cell(j+1;t-1), cell(j+2;t-1)\}$. The result in model $1-(d)$ covers entry 6 which in equation (10) represents a cell at time step $t-2$, $cell(j-1;t-2)$. Model $1-(d)$ therefore defines a temporal-shift neighbourhood involving $cell(j-1;t-2), cell(j,t-1)$, and $cell(j+1,t-1)$.

In each case the CA-OLS algorithm has correctly determined the appropriate neighbour-

hood. Notice that the parameters θ in models $1-(a)$, $1-(b)$, $1-(c)$ and $1-(d)$ are all exactly the same but that each operates on a different neighbourhood and hence produces a different CA pattern.
The measured output and the Model Predicted Output (MPO) for the von Neumann neighbourhood are compared in Figure 7 where the MPO is defined by

$$\hat{s}_m(j;t) = \hat{f}(\hat{s}_m(j+j_1;t-1),\cdots,\hat{s}_m(j-j_2;t-1);\cdots;\hat{s}_m(j+j_1;t-h),\cdots,\hat{s}_m(j-j_2;t-h))$$

The MPO is a more strict criteria for evaluating the performance of the estimator than the One-Step Ahead prediction (OSA) which is defined as

$$\hat{s}_m(j;t) = \hat{f}(s_m(j+j_1;t-1),\cdots,s_m(j-j_2;t-1);\cdots;s_m(j+j_1;t-h),\cdots,s_m(j-j_2;t-h))$$

The comparison in Figure 6 clearly shows that the measured output and the predicted output using the estimated model are almost coincident. The dashed line follows the solid line without any deviation. The variance is virtually zero because the polynomial expression is not an approximation of the CA rules but an equivalent representation. The MPO's for other neighbourhoods also match the corresponding measured outputs exactly. For simplicity, the comparisons are not shown in this paper.

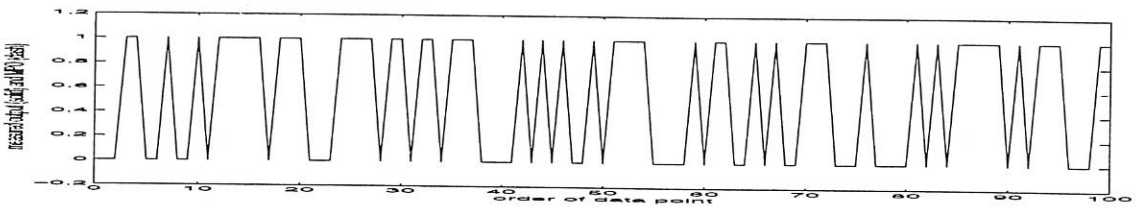


Figure 6: Comparison of the measured (solid line) and the model predicted output (dashed line) for $Rule30$ with a von Neumann neighbourhood

The Quine-McClauskey [11] method was then used to extract the minimum Boolean expression from the estimated polynomial form for $Rule30$ with the left-shift neighbourhood. The steps in this analysis are given below:

(i) Expanding the polynomial in (11)to the full disjunctive normal form [11] yields

$$\begin{aligned} s(j;t) \;=\; & \bar{s}(j-2;t-1)*\bar{s}(j-1;t-1)*s(j;t-1) \;\oplus \\ & \bar{s}(j-2;t-1)*s(j-1;t-1)*\bar{s}(j;t-1) \;\oplus \\ & \bar{s}(j-2;t-1)*s(j-1;t-1)*s(j;t-1) \;\oplus \\ & s(j-2;t-1)*\bar{s}(j-1;t-1)*\bar{s}(j;t-1) \end{aligned} \tag{12}$$

(ii) The numerical representation of these forms, grouped according to the number of

1's, is

$$\begin{array}{ll} \textit{Group (1) with one 1} & \begin{cases} \bar{s}(j-2;t-1) * \bar{s}(j-1;t-1) * s(j;t-1) & 001 \\ \bar{s}(j-2;t-1) * s(j-1;t-1) * \bar{s}(j;t-1) & 010 \\ s(j-2;t-1) * \bar{s}(j-1;t-1) * \bar{s}(j;t-1) & 100 \end{cases} \\ \textit{Group (2) with two 1's} & \bar{s}(j-2;t-1) * s(j-1;t-1) * s(j;t-1) \qquad 011 \end{array}$$

(iii) Comparing terms in adjacent groups, the following combinations can be found where * represents the dropped variable

$$\begin{cases} 0 & * & 1 \\ 0 & 1 & * \end{cases}$$
$$1 \quad 0 \quad 0$$

For example, combining 001 in Group (1) and 011 in Group (2) yields 0*1, while combining 010 in Group (1) and 011 in Group (2) produces 01*.
Again comparing adjacent groupings shows that now all the numerical expressions cannot be combined with any of the others and hence all are prime implicants.

(iv) The original four terms are numbered as 1), 2), 3), 4) and the prime implicants are lettered as $A : \bar{s}(j-2;t-1) * s(j;t-1)$, $B : \bar{s}(j-2;t-1) * s(j-1;t-1)$, $C : s(j-2;t-1) * \bar{s}(j-1;t-1) * \bar{s}(j;t-1)$. A cross $\times$ is placed in the (j,i) position of the table if the ith term was involved in the formation of the jth prime implicant. Table 2 then shows the contribution of each term to the prime implicant.

Table 2. Prime implicant chart for *Rule*30

	1)	2)	3)	4)
A	(×)		×	
B		(×)	×	
C				(×)

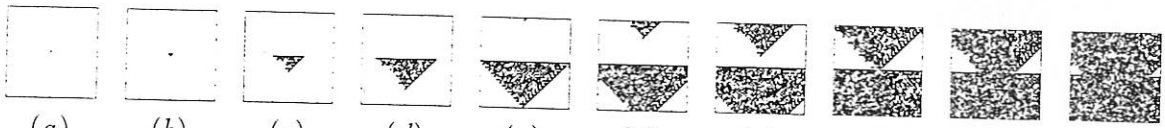
(v) The essential crosses, which are defined as the only cross in the column, are bracketed and the columns deleted are A-1) and A-3), B-2) and B-3) and C-4). Thus all columns are deleted at this stage and hence the function may be represented by the OR combination of A, B and C, or

$$\begin{aligned} s(j;t) \;=\; & \bar{s}(j-2;t-1) * s(j;t-1) \oplus \bar{s}(j-2;t-1) * s(j-1;t-1) \oplus \\ & s(j-2;t-1) * \bar{s}(j-1;t-1) * \bar{s}(j;t-1) \end{aligned}$$

which corresponds exactly to the entry of the Boolean expression for *Rule*30 in Wolfram [8]. Similarly applying the Quine-McClauskey method to models $1-(a)$, $1-(c)$ and $1-(d)$ will produce correct results respectively.

4.2 Identification of a two-dimensional 5-site CA rule

The spatio-temporal patterns produced by a two-dimensional CA evolution on a 200×200 lattice with a von Neumann neighbourhood are illustrated in Figure 7.



(a) t=0 (b) t=1 (c) t=10 (d) t=20 (e) t=30 (f) t=40 (g) t=50 (h) t=60 (i) t=70 (j) t=80

Figure 7: Evolution of a 2-D CA rule with a 5-site von Neumann neighbourhood

Assume initially that the largest possible neighbourhood for this two-dimensional rule is the 9-site Moore neighbourhood. Define the neighbourhood vector as

$$\begin{aligned} a(t) \;=\; & [\; s(i-1,j-1;t-1) \;\; s(i-1,j;t-1) \;\; s(i-1,j+1;t-1) \;\; s(i,j-1;t-1) \;\; s(i,j+1;t-1) \\ & s(i+1,j-1;t-1) \;\; s(i+1,j;t-1) \;\; s(i+1,j+1;t-1) \;\; s(i,j;t-1) \;]^T \end{aligned}$$

The initial model was constructed as

$$MT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

where 1, 2, 3, 4, 5, 6, 7, 8, 9 represent $cell(i-1,j-1;t-1), cell(i-1,j;t-1), cell(i-1,j+1;t-1), cell(i,j-1;t-1), cell(i,j+1;t-1), cell(i+1,j-1;t-1), cell(i+1,j;t-1), cell(i+1,j+1;t-1), cell(i,j;t-1)$ respectively.

$$MT = \begin{bmatrix} 4 & 7 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 7 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 5 & 9 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \theta = \begin{bmatrix} -1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ -2.0000 \\ -1.0000 \\ 1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \\ 2.0000 \\ -2.0000 \\ -1.0000 \\ -1.0000 \\ 1.0000 \end{bmatrix}$$

Model $2-(a)$

One thousand data points were extracted from the CA patterns in Figure 7 and the

threshold for the $[ct]$ cutoff was set to 0. The CA-OLS estimator produced a model with only 17 rows after searching through the whole model set of $N = 2^9 - 1 = 511$ candidate terms.
The identified model $2-(a)$ clearly shows that the CA-OLS algorithm has correctly selected cells 2, 4, 5, 7, 9 which correspond to the von Neumann neighbourhood $\{cell(i-1,j;t-1), cell(i,j-1;t-1), cell(i,j;t-1), cell(i,j+1;t-1), cell(i+1,j;t-1)\}$. The Model Predicted Output MPO superimposed on the measured output over the first 100 data points is illustrated in Figure 8.

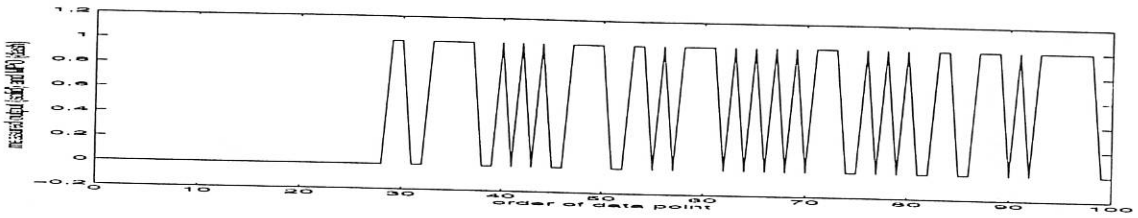


Figure 8: Comparison of measured (solid line) and model predicted output (dashed line) for the 2-D CA rule in Figure 7

Applying the Quine-McClauskey method to model $2-(a)$, the following final prime implicants were obtained

$A : \bar{s}(i,j;t-1) * \bar{s}(i,j+1;t-1) * s(i-1,j;t-1), B : \bar{s}(i,j-1;t-1) * s(i,j+1;t-1) * \bar{s}(i-1,j;t-1)$
$C : \bar{s}(i,j;t-1) * s(i,j+1;t-1) * \bar{s}(i-1,j;t-1), D : \bar{s}(i+1,j;t-1) * s(i,j;t-1) * \bar{s}(i-1,j;t-1)$
$E : \bar{s}(i,j-1;t-1) * s(i,j;t-1) * \bar{s}(i,j+1;t-1), F : \bar{s}(i+1,j;t-1) * s(i,j-1;t-1) * \bar{s}(i,j+1;t-1)$
$G : s(i,j-1;t-1) * \bar{s}(i,j;t-1) * \bar{s}(i,j+1;t-1), H : \bar{s}(i+1,j;t-1) * s(i,j-1;t-1) * \bar{s}(i-1,j;t-1)$
$I : s(i,j-1;t-1) * \bar{s}(i,j;t-1) * \bar{s}(i-1,j;t-1), J : s(i,j-1;t-1) * \bar{s}(i,j;t-1) * \bar{s}(i,j+1;t-1)$
$K : s(i,j-1;t-1) * \bar{s}(i,j+1;t-1) * s(i-1,j;t-1), L : s(i+1,j;t-1) * \bar{s}(i,j+1;t-1) * s(i-1,j;t-1)$
$M : s(i+1,j;t-1) * \bar{s}(i,j-1;t-1) * s(i,j;t-1)$

Finally therefore the Boolean expression of this 5-site two-dimensional rule is the OR combination of all the above $A-M$ 13 items.

4.3 Identification of higher-dimensional CA's

4.3.1 Identification of a three-dimensional CA

Assume initially the neighbourhood for a three-dimensional cellular automaton with null boundary conditions, that is the states of the boundary cells are permanently zero, covers $\{cell(i-1,j,l;t-1), cell(i+1,j,l;t-1), cell(i,j-1,l;t-1), cell(i,j+1,l;t-1), cell(i,j,l-1;t-1), cell(i,j,l+1;t-1), cell(i,j,l;t-1)\}$. This in turn defines the neighbourhood vector as

$$a(t) = [\ s(i-1,j,l;t-1) \quad s(i+1,j,l;t-1) \quad s(i,j-1,l;t-1) \quad s(i,j+1,l;t-1) \quad s(i,j,l-1;t-1) \quad s(i,j,l+1;t-1) \quad s(i,j,l;t-1)\ ]^T$$

The initial model was constructed as

$$MT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & \vdots & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

where 1, 2, 3, 4, 5, 6, 7 represent $\{cell(i-1,j,l;t-1), cell(i+1,j,l;t-1), cell(i,j-1,l;t-1), cell(i,j+1,l;t-1), cell(i,j,l-1;t-1), cell(i,j,l+1;t-1), cell(i,j,l;t-1)\}$ respectively. The $[ct]$ threshold was set as 0 and after searching through a set of $2^7 - 1 = 127$ candidate terms, the CA-OLS estimator produced a model with 53 rows and associated integer parameters.

$$MT = \begin{bmatrix} 3 & 6 & 4 & 1 & 2 & 2 & 2 & 1 & 5 & 3 & 1 & 1 & 1 & 2 & 1 & 3 & 2 & 1 & 1 & 3 & 3 & 5 & 2 & 3 & 1 & 1 & 2 \\ 0 & 0 & 5 & 3 & 0 & 3 & 4 & 2 & 6 & 5 & 2 & 2 & 3 & 3 & 4 & 4 & 6 & 2 & 3 & 6 & 5 & 0 & 3 & 4 & 5 & 3 & 5 \\ 0 & 0 & 0 & 4 & 0 & 0 & 5 & 3 & 0 & 6 & 5 & 3 & 4 & 5 & 5 & 6 & 0 & 3 & 4 & 0 & 0 & 0 & 4 & 5 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 6 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 5 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 1 & 2 & 2 & 4 & 2 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 3 & 4 & 1 & 1 \\ 3 & 4 & 2 & 3 & 4 & 0 & 4 & 2 & 4 & 2 & 3 & 3 & 2 & 3 & 3 & 2 & 2 & 2 & 4 & 2 & 3 & 4 & 4 & 6 & 3 & 2 \\ 4 & 0 & 3 & 4 & 0 & 0 & 5 & 4 & 6 & 0 & 6 & 0 & 4 & 5 & 4 & 3 & 3 & 4 & 0 & 3 & 4 & 6 & 5 & 0 & 6 & 6 \\ 6 & 0 & 4 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 5 & 6 & 5 & 4 & 5 & 0 & 0 & 4 & 6 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

$$\theta = \begin{bmatrix} 1.0000 & 1.0000 & -1.0000 & 1.0000 & 1.0000 & -1.0000 & -1.0000 & -1.0000 & -1.0000 \\ 2.0000 & 2.0000 & 1.0000 & 3.0000 & 1.0000 & 1.0000 & 2.0000 & -1.0000 & -1.0000 \\ -3.0000 & -1.0000 & -2.0000 & 1.0000 & 1.0000 & 3.0000 & -1.0000 & 2.0000 & -1.0000 \\ -3.0000 & -2.0000 & -4.0000 & 3.0000 & -2.0000 & 1.0000 & 2.0000 & -2.0000 & 1.0000 \\ -1.0000 & 1.0000 & -1.0000 & -2.0000 & -1.0000 & -3.0000 & 4.0000 & -2.0000 & 2.0000 \\ -1.0000 & 4.0000 & -3.0000 & 2.0000 & -2.0000 & -1.0000 & 1.0000 & 1.0000 \end{bmatrix}^T$$

Model $3-(a)$

Inspection of model $3-(a)$ shows that only terms defined by 1, 2, 3, 4, 5, 6 are involved in the identified model which in turn represent the correct neighbourhood $\{cell(i-1,j,l;t-1), cell(i+1,j,l;t-1), cell(i,j-1,l;t-1), cell(i,j+1,l;t-1), cell(i,j,l-1;t-1), cell(i,j,l+1;t-1)\}$ shown in Figure 3.

Figure 9 shows that the model predicted output follows the measured output almost perfectly and demonstrates that the CA-OLS estimated CA model exhibits excellent performance even for this three-dimensional cellular automata. Applying the Quine-McClauskey method to the polynomial in model $3-(a)$ produced the correct Boolean expression of the OR combination of 28 prime implicants. For simplicity the prime implicants are not listed.

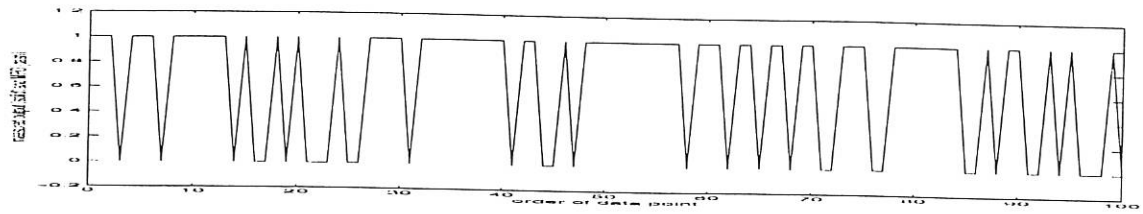


Figure 9: Comparison of measured (solid line) and model predicted output (dashed line) for a three-dimensional CA rule

4.3.2 Identification of a four-dimensional CA

Data extracted from a four-dimensional $12 \times 12 \times 12 \times 12$ cellular automaton with null boundary conditions will be used to illustrate the identification. The initial neighbourhood was assumed to encompass $\{cell(i-1,j,l,k;t-1), cell(i+1,j,l,k;t-1), cell(i,j-1,l,k;t-1), cell(i,j+1,l,k;t-1), cell(i,j,l-1,k;t-1), cell(i,j,l+1,k;t-1), cell(i,j,l,k-1;t-1), cell(i,j,l,k+1;t-1), cell(i,j,l,k;t-1)\}$. This in turn defines the neighbourhood vector as

$$a(t) = [\ s(i-1,j,l,k;t-1)\quad s(i+1,j,l,k;t-1)\quad s(i,j-1,l,k;t-1)\quad s(i,j+1,l,k;t-1)$$
$$s(i,j,l-1,k;t-1)\quad s(i,j,l+1,k;t-1)\quad s(i,j,l,k-1;t-1)\quad s(i,j,l,k+1;t-1)\quad s(i,j,l,k;t-1)\ ]^T$$

The initial model was constructed as

$$MT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & \vdots & & & & \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

where 1, 2, 3, 4, 5, 6, 7, 8, 9 represent $\{cell(i-1,j,l,k;t-1), cell(i+1,j,l,k;t-1), cell(i,j-1,l,k;t-1), cell(i,j+1,l,k;t-1), cell(i,j,l-1,k;t-1), cell(i,j,l+1,k;t-1), cell(i,j,l,k-1;t-1), cell(i,j,l,k+1;t-1), cell(i,j,l,k;t-1)\}$ respectively. The $[ct]$ threshold was again set to 0 and the number of possible candidate models was $2^9-1 = 511$. The CA-OLS estimator produced a model with only 14 rows and the associated integer parameters given in model $4-(a)$.

$$MT = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 5 & 6 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \theta = \begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ -1.0000 \\ -1.0000 \\ 1.0000 \\ -1.0000 \\ -2.0000 \\ 1.0000 \\ 2.0000 \\ -1.0000 \\ -1.0000 \\ -1.0000 \\ 1.0000 \end{bmatrix}$$

Model $4-(a)$

A comparison of the measured output and the MPO is shown in Figure 10 and again these are virtually coincidental showing that the correct model of the CA has been estimated.

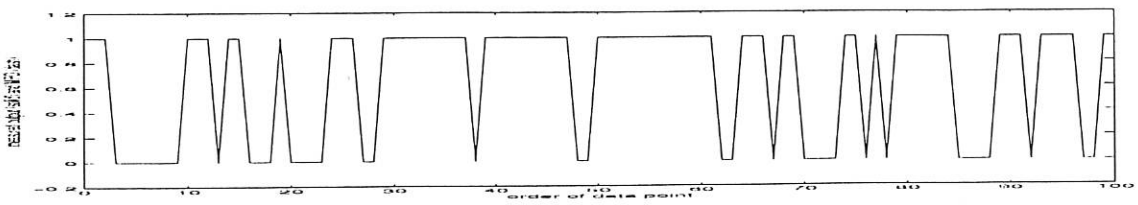


Figure 10: Comparison of measured (solid line) and model predicted output (dashed line) for a 4-D CA rule

Comparing the estimated model $4-(a)$ with model $2-(a)$ shows that the structure in model $4-(a)$ is much simpler than that in model $2-(a)$ although the former is extracted from a 4-D CA while the later is from a 2-D CA. The computation time for both are approximately the same. It can be seen that the efficiency of the CA-OLS estimator relies largely on the size of the neighbourhood, that is the number of cells within the neighbourhood, rather than the order or dimension of the CA and this can dramatically simplify the problem of identifying higher-dimensional cellular automata.
Applying the Quine-McClauskey method to the polynomial produced from model $4-(a)$ the correct Boolean expression of the OR combination of 9 prime implicants was obtained. For simplicity the prime implicants are not listed.

5 Conclusions

While many authors have demonstrated that simple CA models can produce complex spatio-temporal patterns few investigators have studied how to recover such models given only the data patterns. One possible solution to this important problem has been introduced in this study using the new CA-OLS estimator.
The new estimator exploits the observation that binary CA rules can be exactly represented as polynomial models which collapse to relatively simple forms even for high-dimensional CA's. This transforms the problem from a nonlinear-in-the-parameters to a linear-in-the -parameters formulation. The neighbourhood of the CA can then be determined using a modified orthogonal least squares estimator. Identifying the neighbourhood of the CA is critical if the underlying rules are to be estimated and it has been shown that the term contribution test is an efficient solution to this problem. Once the neighbourhood and the polynomial model parameters have been obtained the model can then be mapped back to a Boolean form using the Quine-McClauskey method.
The only information required is to set the range of the largest expected neighbourhood over which the algorithm searches for candidate model terms. The CA-OLS estimator

then searches through all the possible terms and discards all terms below the [ct] threshold to yield the estimated model. The model predicted output is used as a metric of performance to validate the model.
Several simulated examples show the power of the new approach and demonstrate for the first time how CA models can be extracted from data generated from high-dimensional CA systems.

6 Acknowledgment

S.A.Billings gratefully acknowledges that part of this work is supported by EPSRC. Y.X.Yang gratefully acknowledges that this work is supported by a scholarship from the University of Sheffield.

References

[1] G.Hernandez and H.J.Herrann, "Cellular Automata for Elementary Image Enhancement", *Graphical models and Image Processing*, vol.58, no.1, pp.82-89, 1996.

[2] P.Tzionas et al, "Design and VLSI implementation of a pattern classifier using pseudo 2D cellular automata", *IEE Proceedings: Circuits Devices and Systems*, vol.139, no.6, pp.661-668, 1992.

[3] E.Macii and M.Poncino, "Cellular automata models for reliablity analysis of systems on silicon", *IEEE Trans. Reliability*, vol.46, no.2, pp.173-183, 1997.

[4] S.Surka and K.P.Valavanis, "A cellular automata model for edge relaxation", *Journal of Intelligent and Robotic Systems*, vol.4, no.4, pp.379-391, 1991.

[5] A.I.Adamatskii, "Complexity of identification of cellular automata", *Automation and Remote Control*, vol.53, no.9, pt.2, pp.1449-1458, 1992.

[6] F.C.Richards, "Extracting cellular automaton rules directly from experimental data", *Physica D*, vol.45, pp.189-202, 1990.

[7] Y.X.Yang and S.A.Billings "Extracting Boolean rules from CA patterns"submitted to *IEEE Trans. System Man and Cybernetics Pt.B* for publication.

[8] S.Wolfram, *Cellular Automata and Complexity*, Addison-Wesley, 1994.

[9] B.H.Voorhees, *Computational Analysis of One-dimensional Cellular Automata*, World Scientific Series on Nonlinear Science, World Scientific, 1996.

[10] C.Diks et al. "Spatio-temporal chaos: a solvable model", *Physica D*, no.104, pp.269-285, 1997.

[11] R.Korfhage, *Logic and Algorithms*, John Wiley and Sons, 1966.

[12] S.A.Billings and M.J.Korenberg, "Identification of MIMO non-linear systems using a forward-regression orthogonal estimator", *Int. J. Control*, vol.49, no.6, 2157-2189, 1989.

Appendix

The CA-OLS algorithm

Consider the polynomial expression for three-dimensional CA's in equation (9) for example. Denote $(u_1, \cdots, u_n, \cdots, u_1 \times \cdots \times u_n)$ as $(s_1, \cdots, s_N)$. Equation (9) can then be written as

$$s(i,j,l;\tau) = \sum_{d=1}^{N} s_d(\tau) \times \theta_d = \mathrm{s}(\tau) \times \bar{\theta} \tag{13}$$

where τ indicates the order of the data point and

$$\bar{\theta} = \left[\begin{array}{cccc} \theta_1 & \theta_2 & \cdots & \theta_N \end{array} \right]^T$$

and

$$\mathrm{s}(\tau) = \left[\begin{array}{cccc} s_1(\tau) & s_2(\tau) & \cdots & s_N(\tau) \end{array} \right]$$

Equation (13) can also be represented in a matrix form as

$$\mathrm{s}(i,j,l) = \mathbf{S} \times \bar{\theta} \tag{14}$$

where

$$\mathrm{s}(i,j,l) = \left[\begin{array}{cccc} s(i,j,l;1) & s(i,j,l;2) & \cdots & s(i,j,l;M) \end{array} \right]^T$$

$$\mathbf{S} = \left[\begin{array}{cccc} \mathrm{s}^T(1) & \mathrm{s}^T(2) & \cdots & \mathrm{s}^T(M) \end{array} \right]^T = \left[\begin{array}{ccc} \mathrm{s}_1 & \cdots & \mathrm{s}_N \end{array} \right]$$

and M denotes the number of data points in the data set.
Matrix $\mathbf{S}$ can be decomposed as $\mathbf{S} = \mathbf{E} \times \mathbf{Q}$, where

$$\mathbf{E} = \left[\begin{array}{ccc} e_1(1) & \cdots & e_N(1) \\ \vdots & & \vdots \\ e_1(M) & \cdots & e_N(M) \end{array} \right] = \left[\begin{array}{ccc} \mathbf{e}_1 & \cdots & \mathbf{e}_N \end{array} \right]$$

is an orthogonal matrix,

$$\mathbf{E}^T \times \mathbf{E} = Diag \left[\begin{array}{ccc} \mathbf{e}_1^T \times \mathbf{e}_1 & \cdots & \mathbf{e}_N^T \times \mathbf{e}_N \end{array} \right]$$

and $\mathbf{Q}$ is an upper triangular matrix with unity diagonal elements

$$\mathbf{Q} = \begin{bmatrix} 1 & q_{12} & q_{13} & \cdots & q_{1N} \\ & 1 & q_{23} & \cdots & q_{2N} \\ & & \ddots & \ddots & \vdots \\ & & & 1 & q_{N-1N} \\ & & & & 1 \end{bmatrix}$$

Equation (14) can then be represented as

$$\mathrm{s}(i,j,l) = \mathbf{E} \times \mathbf{Q} \times \bar{\theta} = \mathbf{E} \times \bar{\tilde{\theta}} \tag{15}$$

where

$$\bar{\tilde{\theta}} = \mathbf{Q} \times \bar{\theta} = \begin{bmatrix} \tilde{\theta}_1 & \cdots & \tilde{\theta}_N \end{bmatrix}^T$$

Therefore, equation (13) can be written as

$$s(i,j,l;\tau) = \sum_{d=1}^{N} e_d(\tau) \times \tilde{\theta}_d \tag{16}$$

The contribution each term $\{\mathrm{s}_d, d = 1, \cdots, N\}$ in equation (14) makes to $\mathrm{s}(i,j,l)$ can then be calculated as

$$[ct]_d = \frac{\tilde{\theta}_d^2 \times \sum_{\tau=1}^{M} e_d^2(\tau)}{\sum_{\tau=1}^{M} s^2(i,j,l;\tau)} \tag{17}$$

The sum of all the $[ct]$ values will be unity so if $[ct]$ were multiplied by 100 this would give the percentage contribution that each term makes to $\mathrm{s}(i,j,l)$. The orthogonalization of $\mathbf{S}$ simplifies the term selection process and allows each relevant term to be added to the identified term set MT independently of other terms. The parameter vector $\bar{\theta}$ can then be estimated by computing each $\tilde{\theta}_d$ one at a time. However in the term selection process, $[ct]_d$ may depend on the order in which $s_d(\tau)$ enters equation (13). A change of the position of $s_d(\tau)$ in equation (13) may result in a change of the associated $[ct]_d$ value. Consequently simply orthogonalizing the columns in $\mathbf{S}$ into equation (15) in the order in which $s_d(\tau)$'s happen to appear in equation (13) may produce the wrong information regarding the corresponding contributions. To avoid this problem the following forward regression algorithm is used. This algorithm will forward add terms instead of forward deleting terms and will therefore disregard the order that $s_d(\tau)$ enters equation (13).

The forward regression CA-OLS algorithm is given by:

(i) Consider all the $s_d(\tau)$ as possible candidates for $e_1(\tau)$. For $d = 1, \cdots, N$, calculate

$$e_1^{(d)}(\tau) = s_d(\tau)$$

$$\tilde{\theta}_1^{(d)} = \frac{\sum_{\tau=1}^{M} e_1^{(d)}(\tau) s(i,j,l;\tau)}{\sum_{\tau=1}^{M} (e_1^{(d)}(\tau))^2}$$

$$[ct]_1^{(d)} = \frac{(\tilde{\theta}_1^{(d)})^2 \sum_{\tau=1}^{M} (e_1^{(d)}(\tau))^2}{\sum_{\tau=1}^{M} s^2(i.j,l;\tau)}$$

Find and denote the maximum of $[ct]_1^{(d)}$ as $[ct]_1^{(v)} = max\left\{[ct]_1^{(d)}, 1 \leq d \leq N\right\}$. The first relevant term $e_1(\tau)$ is selected as $e_1^{(v)}(\tau)$ and $\tilde{\theta}_1 = \tilde{\theta}^{(v)}$, $[ct]_1 = [ct]_1^{(v)}$. The corresponding $s_v(\tau)$ is then included in the identified model set MT.

(ii) All the $s_d(\tau)$, $d = 1, \cdots, N$, $d \neq v$ are considered as possible candidates for $e_2^{(v)}(\tau)$. For $d = 1, \cdots, N$, $d \neq v$, calculate

$$e_2^{(d)}(\tau) = s_d(\tau) - q_{12}^{(d)} e_1(\tau)$$

$$\tilde{\theta}_2^{(d)} = \frac{\sum_{\tau=1}^{M} e_2^{(d)}(\tau) s(i,j,l;\tau)}{\sum_{\tau=1}^{M} (e_2^{(d)}(\tau))^2}$$

$$[ct]_2^{(d)} = \frac{(\tilde{\theta}_2^{(d)})^2 \sum_{\tau=1}^{M} (e_2^{(d)}(\tau))^2}{\sum_{\tau=1}^{M} s^2(i.j,l;\tau)}$$

where

$$q_{12}^{(d)} = \frac{\sum_{\tau=1}^{M} e_1(\tau) s_d(\tau)}{\sum_{\tau=1}^{M} e_1^2(\tau)}$$

Find and denote the maximum of $[ct]_2^{(d)}$ as $[ct]_2^{(g)} = max\left\{[ct]_2^{(d)}, 1 \leq d \leq N, d \neq v\right\}$. The second term $e_2(\tau)$ is therefore selected as $e_2^{(g)}(\tau) = s_g(\tau) - q_{12}^{(g)} e_1(\tau)$ and $q_{12} = q_{12}^{(g)}$, $\tilde{\theta}_2 = \tilde{\theta}^{(g)}$, $[ct]_2 = [ct]_2^{(g)}$. The corresponding $s_g(\tau)$ is then included in the identified model set MT.

(iii) The procedure is terminated at the N_sth step either when $1 - \sum_{d=1}^{N_s} [ct]_d < C_{off}$ (desired tolerance), $N_s < N$ or when $N_s = N$.

(iv) From the selected orthogonal equation

$$s(i,j,l;\tau) = \sum_{d=1}^{N_s} e_d(\tau) \tilde{\theta}_d$$

it is then straightforward to calculate the corresponding N_s parameters θ using

$$\theta_{N_s} = \tilde{\theta}_{N_s}$$

$$\theta_m = \tilde{\theta}_m - \sum_{k=m+1}^{N_s} q_{mk} \theta_k, \quad m = N_s - 1, \cdots, 1.$$

List of Figures