*Journal of the Nigerian Association of Mathematical Physics*
*Volume* **19 (November, 2011), pp 61 – 68**


# A study on switched linear system identification using game-theoretic strategies and neural computing

***C. N. Njoku***

**Dept. of Computer Science**
**Federal University of Technology, Owerri, Nigeria.**

## *Abstract*

***This study deals with application of game-theoretic strategies and neural computing to switched linear system identification, wherein some of the subsystems may be in failed, standby, or working states. The controller is to detect failed subsystems, and switch standby and working subsystems to maintain stable system configuration. Its performance is based on a number of successful stabilizing controls before entire system failure. A strategy board game is redefined as switched linear system and used in the experimental study. Mathematical model of the system is derived from its physical description, and the controller is modelled as adaptive neural network. The analysis results to estimation of system parameters and identification of stabilizing switching control rules. The system trajectory is ascertained by simulation tests, wherein subsystem failure time intervals are varied. The study demonstrates feasibility of the non-classical methods in system identification from first principles. Besides prototype system identification, our methodology has applications in combinatorial switching control modelling and experimental validation of mathematical models of switched linear systems.***

## 1.0 Introduction

The basic problem of linear system identification is explained for continuous- and discrete- time types as follows. A linear continuous-time system is described by the basic equations

$$x'(t) = A(t)x(t) + B(t)u(t) \quad (1.1)$$

$$y(t) = C(t)x(t) + D(t)u(t) \quad (1.2)$$

where x is the state variable vector, u is the energizing input, y is output vector or system response, while A, B, C, and D are the associated coefficient matrices. In real-life, x may be physical quantities found in chemical, electrical, mechanical, thermal systems, etc.

Similarly a discrete-time linear system is described by the basic equations

$$x(k+1) = A(k)x(k) + B(k)u(k) \quad (1.3)$$

$$y(k) = C(k)x(k) + D(k)u(k) \quad (1.4)$$

where k denotes the time instants or sequences of events in the system.

For both types of system, identification is the process of determining coefficient matrices A, B, C, and D from the input/output data. When these are determined, mathematical methods can be applied to test for controllability, observability, and stability. Classical methods for determining the coefficient matrices include least squares, state space modelling, calculation of Markov parameters, etc [1–3]. Recently, due to the need to deal with uncertainties in identification problems, intelligent methods such as game-theoretic strategies, neural networks, fuzzy logic, and expert system technology are being integrated with classical methods. Particularly, intelligent methods are applied to nonlinear, reactive, and ill-conditioned systems [4–7]. For instance, game-theoretic strategies are used to investigate conditions for relaxing or tightening controls, while neural network is used to model controller structure to deal with nonlinear and adaptive aspects.

In this study, game-theoretic strategies and neural computing are applied to switched linear system identification. A board game called Jonda is redefined as switching system and used in the experimental study. Starting from the physical description, the system is analyzed using game strategies, linear switching control concepts, and neural computing to develop the analytical simulation model. To ascertain the validity of the estimated parameters and controller performance, the simulator output is analyzed, and system trajectory determined.

[1]Corresponding authors:-- , E-mail: cnnodimnjoku@yahoo.com, Tel. +2348059910428

## 2.0 Game strategies and neural computing in system identification

### 2.1 Game strategies in switched linear system identification

Game-theoretic strategies are used as optimization methods in control engineering, operations research, and other computer-aided design problems [8–11]. Like control problems, strategy games may be continuous-time or discrete-time systems. Continuous-time games are differential games such as pursuit-evasion and trajectory tracking games, while discrete games (e.g. board games) are formulated as difference equations. Qualitatively, a discrete strategy game can be defined as a 7-tuple:

$$G = (N, S, \delta, Q, X, Y, \Omega) \quad (2.1)$$

where

(1) N is the number of players;
(2) S is a set of strategies available to the players:
(3) $\delta$ is the set of rules that orders the game;
(4) Q is the set of stages or internal states;
(5) X is the finite set of admissible inputs.
(6) Y is the finite set of output, or system configuration;
(7) $\Omega$ is the payoff or performance objective function.

If a control system is set up as two-player game scenario, the game object is the system to be controlled, the controller is player-1, while input sources such as the operator inputs, internal and external disturbances can be lumped together as player-2. Multiplayer games are used to study distributed control systems [12,13]. For most control system identification problems, N, X, and Y are known. The other aspects S, $\delta$, Q, and $\Omega$ may be partially known or entirely unknown and these form the identification problem space.

In a game-theoretic approach, the game-like control system is modelled as a mathematical object, while the control process is modelled as an algebraic combinatorial game. In control-theoretic form, a discrete-time game can be expressed as:

$$x(t+1) = f(x(t), u(t), c(t)) \quad (2.2)$$

$$y(t) = g(x(t), u(t), c(t)) \quad (2.3)$$

where $x(t) \in R^m$ is the state space vector, $u(t) \in R^n$ is the external input, c(t) is the controller input, y(t) is the output vector, f, and g are input and output functions, respectively.

The switching rule for the system can be defined by the equation [14–16]:

$$q(t+1) = \Phi(q(t), x(t), c(t)). \quad (2.4)$$

where q is the switching mode, $t \in [t_0, t_{max}]$, and $\Phi$ is a mapping function that orders the switching sequences. If switching takes place at time instants $t_0, t_1, \ldots, t_j, \ldots$ so that a subsystem is switched at time $t_j$, then the sequence is expressed as:

$$s = (q_0, t_0), (q_1, t_1), \ldots, (q_j, t_j), \ldots \quad \text{for } j = 0, 1, \ldots, \quad (2.5)$$

We note that for some adaptive or predictive switching control problems, the internal processes through which the controller decides on switching sequences can be considered as tic-tac-toe game strategy. Hence, game strategies can be applied in modelling the switching controller.

### 2.2 Neural network in switching controller model identification

We consider controller model identification as the problem of determining the controller mathematical structure and algorithms. For some problems, neural networks (NNs) can be constructed and adapted to approximate required system controller characteristics due to the features that include: (a) Suitability to many interconnected subsystems; (b) Well-defined mathematical structure, implicit parallelism, statistical learning, and ability to classify pattern; (c) Integration of suitable (non)linear output filtering functions to produce decision parameters; (d) The basic topologies and algorithms can be enhanced to suit adaptive, predictive, and optimal control strategies. Figure 2.1a shows a multilayer neural network with single output, while

Figure 2.1b shows a cascaded set of neural networks for multivariable control.

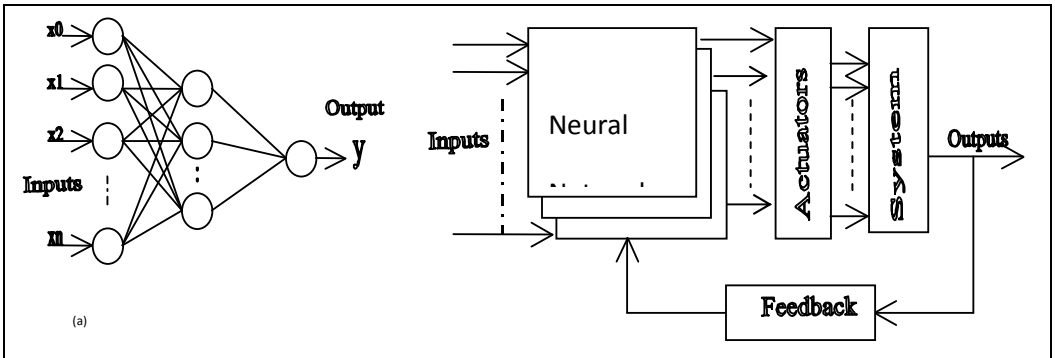


**Figure 2.1: Basic structure of neural network based control system**

NNs are suited to system identification, as they are not explicitly programmed, but have to be trained or adapted to particular problems. For control applications, the controller behaviour is established during the training process. The training is a statistical learning process, relating inputs, computed node link weights, output nodes' values, and desired (predefined) output values. Let the set of inputs be $X = (x_1, x_2, \ldots, x_m)$ and $Y = (y_1, y_2, \ldots, y_n)$ be outputs computed from various combinations of the inputs. The sets X and Y are related by matrix operator or node link weights $w_{ij}$, and node activation function that may be linear, sigmoid, hyperbolic tangent, gaussian, ramp, sign function, etc. Not only on nature of the problem, the actual mathematical relationship between X and Y depends also on the chosen network topology (e.g. multilayer perceptron, adaptive bidirectional associative memory, self-organizing maps, etc), and training algorithm. The mathematical formulations and training algorithms abound in the literature. A fully trained or adapted NN-based controller is ready for operation. Such controllers have been realized in hardware and software for industrial process control applications.

## 3.0 The system identification problem formulation

Figure 3.1 shows the structure of a tic-tac-toe board game called Jonda [17,18]. Consider a switched linear system comprising subsystems (nodes) connected as shown in the figure. In the series-parallel configuration, each horizontal, vertical, and diagonal row is a set of specially related nodes as obtains in electrical power system, industrial process control, oil/gas pumping,
etc. It is assumed that the square nodes have greater capacity than the circular ones.

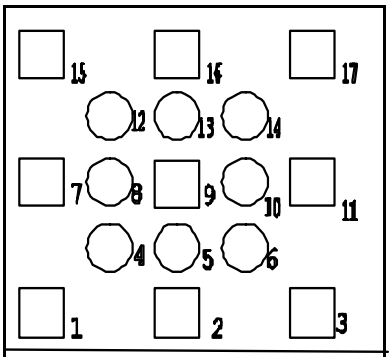


**Figure 3.1: Jonda board game**

The status of the whole system depends on the nodes such that:

(1) A node may be in any of the states: *failed* (-1), standby/redundant (0), or working (1);
(2) Working node failure is random and it is assumed that one node fails at a time instant;
(3) Failed nodes are restored to normalcy within specified downtime;
(4) A working node may be switched to standby after a specified maximum continuous working time;
(5) After a minimum rest time, a standby node may be switched to working mode to stabilize the system;
(6) Restored failed nodes are switched to standby mode;
(7) System failure occurs if all nodes in a row fail;
(8) Operational stability requires balanced configuration of standby and working nodes;
(9) Switching is based on temporal parameters such as working time, standby time, etc.
(10) Optimal switching control strategies are required to prevent system failure.

The problem to be addressed is to:

(a) Analyze the system using game- and control-theoretic methods;
(b) Formulate analytical models of the system and its controller;
(c) Identify the system parameters and switching control strategies
(d) Simulate the system and evaluate its dynamic behaviour and the controller performance.

In the rest of this paper, the term "the system" refers to the abstracted switching system as described above.

## 4.0 Modelling the switched linear system

### 4.1 System nodes configuration space

From the possible node status values {-1,0,1} defined in section 3, information about the system state at any time is determined by the configuration of the 17 node values. That is, with each configuration we can identify the various system conditions and required control strategies. The number of possible configurations highlights the complexity of this system identification problem. Let the total nodes be N, failed nodes F, standby nodes S, and working nodes W. Then, the total system configuration is determined by the permutation P:

$$P(N, F, S, W) = N!/( F !*S!*W!) \quad (4.1)$$

Suppose as part of the design decision, 4 failed and 4 standby nodes are tolerable, then equation (4.1) evaluates to $P(17,4,4,9) = 1.7017 \times 10^6$. Depending on the switching control policies, within this data pattern space are those that result to system conditions such as optimally safe, minimally safe, deadlock, failure threat, complete failure, etc. In the next section, these possible conditions are defined with some time scale attached to node records. For instance, deadlock occurs when no stabilization switching control can be safely performed and this indicates failure of multiple sets of subsystem due to, say, delayed failed node restoration. Predictive switching control strategies that will check such occurrences have to be identified.

### 4.2 System safety factors and control policy specifications

Assuming a true competitive situation where player-1 is the controller and player-2 is an intelligent adversary that induces node failures in attempt to defeat the control strategies. The system safety factors and control policy are determined by analyzing the effects of node switching, including time constraints on the various nodes.

Let node time variables be downtime $t_d$, standby time $t_s$, and working time $t_w$. The pre-definable times are $t_{dmax}$, $t_{smin}$, and $t_{wmax}$. The system conditions are defined algebraically as shown in Table 4.1, where asterisk (*) denotes node state in the set {-1,0,1}.

**Table 4.1: System safety conditions and node set configuration**

| S/No | System Safety Condition | Node set configurations | | Time constraint/Explanation |
|---|---|---|---|---|
| | | 3-node row | 5-node row | |
| 1 | Optimally safe | {1,1,*} | {1,1,1,*,*} | $t_d \geq t_{dmax}$, $t_s \geq t_{rmin}$, $t_w < t_{wmax}$ |
| 2 | Minimally safe | {1,0,-1} | {1,1,-1,-1,-1} | $t_d \geq t_{dmax}$, $t_s \geq t_{rmin}$, $t_w < t_{wmax}$ |
| 3 | Working square and circular nodes ratio | See the explanation | See the explanation | Total ratio $\geq 0.5$ is preferred |
| 4 | Deadlock | Highly constrained switching | Highly constrained switching | Successive switching $\Gamma(1,0)$ and $\Gamma(0,1)$ involving few nodes due to several nodes with $t_d < t_{dmax}$, $t_s < t_{smin}$ |
| 5 | Single row failure chance | {-1, -1, 1} | {-1,-1,-1,-,1, 1} | $t_w < t_{wmax}$, $t_d < t_{dmax}$ for the W and F nodes |
| 6 | Multiple row failure chance | {-1, -1, 1} where {1} connects several node row | {-1,-1,-1,-,1, 1}, where {1} connects several node rows | Two or more row failure chances, where $t_w \leq t_{wmax}$, $t_d < t_{dmax}$ |
| 7 | System failure | {-1,-1,-1} | {-1,-1,-1,-1,-1} | $t_d < t_{dmax}$ for all failed nodes |

The node switching control policy can be summarized in the statement:

*Switch nodes to maintain at least minimum safe configuration by checking immediate and near-future occurrences of unsafe conditions such as (6) and (7) in Table 4.1.*

From the foregoing, switching control resulting to tolerable configurations (1) – (5) can be assigned weighted switching safety factors 1.0, 0.80, 0.60, 0.40, and 0.20 respectively. The intolerable conditions (6) and (7) are assigned 0.0, implying that the nodes cannot be switched. Derived in section 4.4.1 is a linear function for calculating system stability factors.

### 4.3 Prediction and controller performance measures

Two levels of prediction are identified for the control operations. The first level recognizes effects of failed nodes by analyzing system configurations based on control policy stated in section 4.2. The second level considers consequences of node switching by the controller to maintain safe operations. The second level is complicated by time and nodes pattern constraints, which may result to adverse effects such as unbalanced working node patterns, deadlocks, and near-future system failure chance.

Generally, performance objective of a predictive controller may be trajectory tracking or steering the system from initial state to stable state. The performance measure is usually defined as a cost function to be minimized in the control process. The performance of a game-theoretic system controller can be based on cost function related to payoff values, search cost, or time-optimal control. For our experimental system, the performance measure is defined in terms of:

(1) Number of successful predicted switching controls before system failure;
(2) Speed of the predictive switching control.

With performance measure (2) for instance, controllability of the system is observed within some time horizon. In this case the system trajectory is described by equation:

$p'(t) = f(p(t), u(t), c(t))$ (4.2)

where p, u, and c are respectively, successful predictions before system failure, node failures, and controller inputs. These depend on feasible times: $t_{dmax}$, $t_{smin}$, and $t_{wmax}$.

### 4.4 The neural network controller model identification

Based on the system data pattern, the controller structure is approximated by an adaptive bidirectional associative memory (ABAM) neural network model. ABAM is a general matrix-type operator for dynamic feedforward/feedback or static feedforward systems [19]. The model has two layers for input and output, and control memory that stores prototype data patterns for recognizing inputs. How the model is adapted to our system is discussed as follows.

#### 4.4.1 Input and output functions

In control applications the inputs to a neural network are usually real numeric values, which may have to be normalized to simplify computations. The types of possible output functions have been mentioned in section 2.2. For our system, the discrete ABAM model has output functions defined as $f(a) = \text{sign}(a)$. Generally, if u and v are respectively the input and output vectors, the state of the feedforward/feedback discrete system is described by the equations:

$$u_i(k+1) = f(a_{ui}(k)) = \begin{cases} 1 & \text{for } a_{ui}(k) > 0 \\ u_i(k) & \text{for } a_{ui}(k) = 0 \\ -1 & \text{for } a_{ui}(k) < 0 \end{cases} \quad (4.3)$$

$$v_j(k+1) = f(a_{vj}(k)) = \begin{cases} 1 & \text{for } a_{vj}(k) > 0 \\ v_j(k) & \text{for } a_{vj}(k) = 0 \\ -1 & \text{for } a_{vj}(k) < 0 \end{cases} \quad (4.4)$$

where

$$a_{ui} = \sum \omega_{ij} f(a_{vi}) + \Phi_j \qquad \text{for } i, j = 1 \ldots N \quad (4.5)$$

and

$$a_{vj} = \sum \omega_{ji} f(a_{uj}) + \theta_i \qquad \text{for } i = 1 \ldots N,\ j = 1..M \quad (4.6)$$

As usual, $\omega$ is the node link weight matrix, while $\Phi$ and $\theta$ are input and output biasing factors.

In more compact matrix notation,

$$u(k+1) = f(\omega v(k)) \quad (4.7)$$

$$v(k+1) = f(\omega^T u(k)) \quad (4.8)$$

Starting in a simple way, input/output process for our system is defined by the equation

$$v_{k+1} = v_k + u_k \quad (4.9)$$

where, at a time instant k, $u_k$ and $v_k$ are 17-element column vectors – denoting current input and previous system output, respectively. The new vector v produced is fed forward for prediction of the system conditions.

Let $c_k$ be the result of predicted switching control actions, then the output vector z, after control is expressed as:

$$z_k = v_k + c_k \quad (4.10)$$

where $c_k$ and $z_k$ are also 17-element column vectors.

Suppose inputs to the nodes are $(x_1, x_2, \ldots, x_n)$ and the link weights $(\omega_{11}, \omega_{12}, \ldots, \omega_{nn})$, then the two simplest ways of associating $x$ and $\omega$ to produce an outputs are $\varphi(\omega_{ij}x_i)$ and $\psi(\sum\omega_{ij}x_i)$, where $\varphi$ and $\psi$ are functions that determine output node values. In the network, controller actions resulting to the node transitions do not depend on the weighted sum of the node link strengths. Rather, it is a consideration of node pairs, depending on the system safety factors. Thus, the switched discrete-time linear system is described by the state variables as:

$$x(t+1) = \omega(t)x(t) + \beta u(t) \quad (4.11)$$

where at time t, x(t) is the node state variable vector, and $\omega(t)$ is a 17x17 matrix of the node link weights, u(t) is control input, and $\beta$ is the biasing factor. We note that equation (4.11) corresponds to basic equation (1.3). Switching control for stabilizing the system configuration involves finding appropriate switchable standby and working node pairs. The switching function for such node pairs is expressed as:

$f(x_i, x_j) = \omega_{ij}(x_i + x_j) + \beta u; \qquad i, j = 1, 2, \ldots, 17 \quad \text{and} \quad x_i \neq x_j \quad (4.12)$

The biasing factor $\beta$ is used to facilitate iterative search for switchable nodes that result to safe configuration as defined in Table 4.1. With the possible system safety conditions stated in section 4.2, the node link weights are in the range $0.0 \leq \omega_{ij} \leq 1.0$, so that $\beta = -0.20$. A node link weight of 0.0 implies that switching of $i^{th}$ and $j^{th}$ node pairs is either invalid or unsafe.

**4.4.2 Switching control rules formulation**

In section 2.1, it is mentioned that in game-theoretic approach, system control process is modelled as algebraic combinatorial game. For our system, the control rules are formulated based on some practical considerations. Suppose, by design policy, it is decided that the least switchable node times are 100%, 25%, and 80% of the $t_{dmax}$, $t_{smin}$, and $t_{wmax}$, respectively. Then, using the algebraic properties of the system nodes, we formulate the control rules as follows.

Let x be is a vector of the system configuration, and $x_i$ and $x_j$ be node pairs to be considered in the switching transition $\Gamma(x_i, x_j)$. The arithmetic $(x_i + x_j)$ for $x_i \neq x_j$ and $i \neq j$, results to the data set {-1,0,1}. At start-up, $\omega_{i,j}$ is initialized to 0.0, and thereafter the values are calculated depending on the system configuration. The following are examples of the control rules:

$$\text{if } (x_i = -1 \text{ and } t_d/t_{dmax\,(i)} \geq 1.0) \text{ then } x_i = 0;\ t_{s\,(i)} = t_{smin} \qquad (4.13)$$

$$\text{if } (x_i + x_j = 1 \text{ and } t_s/t_{smin\,(i)} \geq 0.25 \text{ and } t_w/t_{wmax\,(j)} \geq 0.80) \text{ then } \omega_{i,j} = 1.0 \qquad (4.14)$$

Equations (4.13) and (4.14) are the basic node weights calculations for which no training or adaptation is required. Considering equation (4.12), a sample adaptable switching control rule for working-to-standby transition $\Gamma$, is defined as follows:

$$\text{If } 0.20 \leq f(x_i, x_j) \leq 1.0 \text{ then } \Gamma(W, S) \qquad (4.15)$$

Similar transitions rules can be defined for failed-to-standby $\Gamma(F, S)$, failed-to-working $\Gamma(F, W)$, etc. A set of rules defined by equation (4.15) corresponds to switching rule of equation (2.4), and this will produce switching sequences corresponding to equation (2.5). The switching control that results to unsafe configuration indicates system failure due to inaccurate predictions. Thus, accuracy of prediction is the basis of the controller performance evaluation, and it is a measure of completeness of the control parameters and rules.

With the preceding formulations and switching control rules, the controller model has been identified. The controller performance and system trajectory are determined through simulation tests reported in section 5.

## 5.0 Simulation and performance evaluation

The simulation has the following two major objectives:

(1) Discovering complementary low-level control strategies required to handle induced unsafe system configurations. For instance, it is discovered that delay in state and control is required in certain system states in order to satisfy some safety constraints.

(2) Evaluating the identified system behaviour as depicted by its time-based trajectories.

To achieve these objectives, the system is tested on random node failure and ordered node failure modes. As a form of worst-case testing, the latter uses knowledge of current system configuration to induce node failures aimed at defeating the control strategies.

As the system behaviour depends on its temporal parameters $t_d$, $t_{dmax}$, $t_s$, $t_{smin}$, $t_w$, $t_{wmax}$, and mean time before failure (MTBF), it would require several simulation tests involving various combinations of these parameter values to determine the optimal time values. As the parameters $t_{dmax}$, and $t_{wmax}$ are connected with reliability (in terms of availability and resilience), they are used as independent variables in separate simulation tests. Taking the unit time as a second, the system is tested based the on the subsystems' time parameters: $t_{dmax} = 2.0$, $t_{smin} = 0.5$, $t_{wmax} = 1.0$, and $0.1 \leq \text{MTBF} \leq 1.0$. With these, the simulation output includes count of successful switching controls before system failure, switching time, and speed.

Presented in Table 5.1 is a sample of the simulation output. Data labels with subscript (1) are for random node failure and those with subscript (2) are for ordered node failure. The selected sample simulation output is based mostly on those with both high switching control counts and fast switching speed. The graphs are presented in Figures 5.1, 5.2, and 5.3.

## 5.1 Performance evaluation

Referring to the identified system behavioural graphs presented in Figures 5.1, 5.2, and 5.3, it can be observed that:

(1) Both random and ordered node failure test modes give similar behavioural graphs.

(2) With the decision on number of failed, standby, and working nodes, and temporal parameters, $t_{dmax} = 2.0$, $t_{smin} = 0.5$, $t_{wmax} = 1.0$, the whole system tends to work for a very long time without failure if MTBF = 1.00, as particularly indicated by Figure 5.1.

(3) Figure 5.3 shows that the switching control speed is almost sinusoidal, which is similar to real-life situations where intelligent search for optimal decision parameters is like traversing hills, valleys, and plateaux.

(4) Considering the dimension of the abstracted identification problem, the system behaviour and controller performance are both reasonable and consistent with known control system theory as the system behavioural graphs are apparently classifiable as exponential growths, as shown by Figures 5.1 and 5.2.

On the whole, the result shows that our game theoretic abstraction and neural computing has yielded provable and valid system model.

**Table 5.1: Identified system simulation test output**

| MTBF | CtrlCnt(1) | CtrlTime(1) | CtrlCnt/Sec(1) | CtrlCnt(2} | CtrlTime(2) | CtrlCnt/Sec(2) |
|---|---|---|---|---|---|---|
| 0.10 | 104 | 4.84 | 21.4876 | 39 | 1.65 | 23.6364 |
| 0.15 | 767 | 26.75 | 28.6729 | 22 | 0.82 | 26.8293 |
| 0.20 | 598 | 20.82 | 28.7224 | 20 | 0.71 | 28.1690 |
| 0.25 | 894 | 38.72 | 23.0888 | 71 | 3.35 | 21.1940 |
| 0.30 | 776 | 24.44 | 31.7512 | 157 | 8.35 | 18.8024 |
| 0.35 | 1121 | 36.85 | 30.4206 | 296 | 11.59 | 25.5007 |
| 0.40 | 1565 | 51.63 | 30.3118 | 257 | 10.06 | 25.5467 |
| 0.45 | 2834 | 92.77 | 30.5487 | 203 | 9.93 | 20.4289 |
| 0.50 | 4514 | 208.77 | 21.6219 | 408 | 13.75 | 29.6485 |
| 0.55 | 5178 | 249.09 | 20.7877 | 587 | 29.60 | 19.8344 |
| 0.60 | 6863 | 332.79 | 20.6226 | 604 | 24.09 | 25.0856 |
| 0.65 | 6706 | 327.91 | 20.4507 | 844 | 40.81 | 20.6748 |
| 0.70 | 8751 | 405.57 | 21.5770 | 612 | 24.77 | 24.7073 |
| 0.75 | 9505 | 485.10 | 19.5939 | 888 | 43.29 | 20.5167 |
| 0.80 | 9701 | 484.88 | 20.0070 | 1203 | 59.49 | 20.2273 |
| 0.85 | 11927 | 553.54 | 21.5468 | 1534 | 63.94 | 23.9912 |
| 0.90 | 15907 | 695.19 | 22.8815 | 1865 | 96.48 | 19.3335 |
| 0.95 | 15831 | 1424.43 | 11.1139 | 4395 | 212.63 | 20.6674 |
| 0.99 | 18146 | 912.42 | 19.8878 | 7683 | 373.69 | 20.5601 |

Figure 5.1: System switching control counts versus Subsystem MTBF

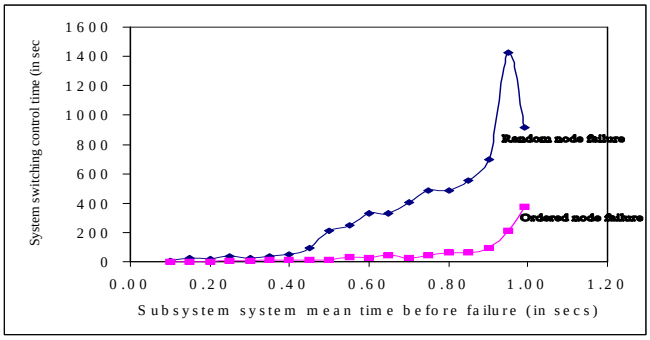


Figure 5.2: System switching control time versus Subsystem MTBF

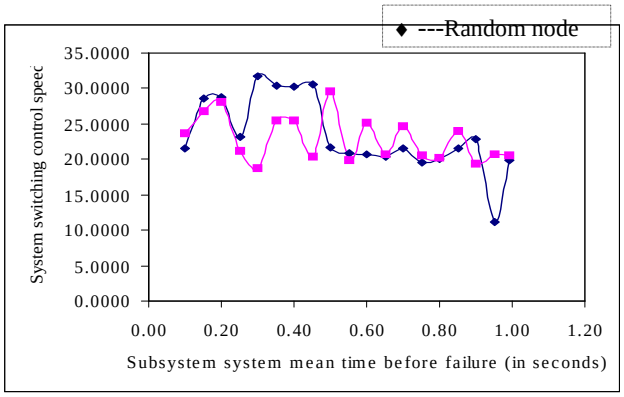


Figure 5.3: System switching control speed versus subsystem MTBF

## 6.0 Conclusion

System identification problems occur in biological, soil, econometric, engineering systems design, etc. Currently, intelligent methods are being employed to overcome some of the limitations of the classical methods. In this bootstrapping experiment, game strategies and neural computing have been explored in prototype switched linear system identification. To be practical, a tic-tac-toe board game called Jonda is redefined as switched linear system and used as experimental object. From first principle abstractions, the analytic model is identified and the controller structure approximated by an adaptive neural network. The system is simulated to ascertain its behaviour based on random and ordered node failure modes. Analysis of the simulation output gives classifiable behavioural graphs and reasonable performance.

Finally, this study has practically demonstrated the effectiveness of intelligent methods in system identification from first principles. Besides system identification, our approach can be found useful in prototype switching system control modelling and experimental validation of theoretical models of some switched linear systems.

## References

[1] Haverkamp, B. R. J. and Verhaegen M. G. H. (1997): SMI toolbox, state space model identification software for multivariable dynamical System, Delft University of Technology, 1st edition.

[2] Verhaegen, M. (1994): Identification of the deterministic part of MIMO state space models given in innovations form from input-output data, Automatica 30(1), pp. 61–74.

[3] Viberg, M. (1995): Subspace-based methods for the identification of linear time-invariant systems, Automatica, 31, no. 12, pp. 1835–1851.

[4] Basar T. and Bernhard P. (1995): $H_\infty$ optimal control and related minimax design problems: A dynamic game approach, 2nd Ed., Birkhauser, Boston.

[5] Hamda H. and Schoenauer M. (2000): Adaptive techniques for evolutionary topological optimum design; In Evolutionary Design and Manufacture, pp 123–136.

[6] Papavassilopoulos G. P. and Safonov M. G. (1989): Robust control design via game theoretic methods, Proceedings of the 28th Conference on Decision and Control, Tampa, Florida, pp. 382-387.

[7] Ram A., and Santamaria J. C. (1993): Multi-strategy learning in reactive control systems for autonomous robot navigation. Informatica 17(4):347-369.

[8] Basar T. and Olsder G. J. (1999): Dynamic noncooperative game theory, SIAM, Philadelphia.

[9] Chen H., Scherer C., and Allgöwer F. (1997): A game theoretic approach to nonlinear robust receding horizon control of constrained systems; Proceedings of the American Control Conference, Albuquerque New Mexico.

[10] Kannan R, and Iyengar S. S (2004): Game-theoretic models for reliable path-length and energy-constrained routing with data aggregation in wireless sensor networks, IEEE J. Selected Areas Comm., Vol. 22, No. 6, pp. 1141-1150.

[11] Grefenstette J., Ramsey C. and Schultz A. (1990): Learning sequential decision rules using simulation models and competition. Machine Learning 5, pp. 355-381.

[12] Lygeros J., Godbole D. N., and Sastry S. (1996): A game theoretic approach to hybrid system design. In R. Alur and T. Henzinger, editors, Hybrid Systems III, volume 1066 of Lecture Notes in Computer Science, pages 1–12. Springer-Verlag.

[13] Tomlin C. J., Lygeros, J. L., Sastry S. S. (2000): A game theoretic approach to controller design for hybrid systems, IEEE Proc., Vol. 88, pp. 949-970.

[14] Hocherman-Frommer J., Kulkarni S. R., Ramadge P. (1995): Supervised switched control based on output prediction errors," Proc. 34th IEEE Conf. Decision and Control.

[15] Xenofon D. Koutsoukos, Panos J. Antsaklis (2001): Design of stabilizing switching control laws for discrete and continuous-time linear systems using piecewise-linear Lyapunov Functions; ISIS Technical Report ISIS-2001-002.

[16] Zhivoglyadov P. V., Middleton R. H., Fu M. (2002): Localization based switching adaptive control for time-varying discrete-time systems, IEEE Trans. Automatic Control, 45(4): pp. 752-755.

[17] Njoku C. N. (2004): Towards the theory and applications of Jonda game; In The Journal of Computer Science and Its Applications; An International Journal of Nigeria Computer Society. Vol 10 (1); pp. 103–116.

[18] Njoku C. N. and Ayeni J. O. A. (2005): A study on AI-based modelling and simulation using Jonda game as experimental tool; In The Journal of Computer Science and Its Applications; An International Journal of Nigeria Computer Society, Vol.11(1); pp. 3–26.

[19] Luger G. F. and Stubblefield W. A. [1997]: Artificial Intelligence Structures and Strategies for Complex Problem Solving; Addison-Wesley Longman Inc. England.