point of view, the definition of ABST given in the note[1] could be modified to mention matrix D as well, as follows.

Definition 3.1: System (N) is said to be ABST w.r.t. $\mathcal{S}$ if it possesses a GAS equilibrium point for every function $g \in \mathcal{S}$, every input vector I and any diagonal matrix $D > 0$.

Remark 3.2: From the proof of Theorem 2.1 above, it is clear that the condition $T \in \mathcal{D}_o$ ensures that system (N) is ABST w.r.t. $\mathcal{S}$ in the sense of the definition above. Notice also that absolute stability in the sense of this new definition guarantees stability when the elements of the matrix D are perturbed as well. In the neural circuit context, the elements of the matrix D depend on certain resistance and capacitance values which are subject to uncertainties in practical implementations, thus motivating the modification in the definition.

Analyzing the linearized version of system $(\tilde{N})$ in the neighborhood of the equilibrium point 0 we have

$$\dot{y} = (TG'(y) - D)y \qquad (L\tilde{N})$$

where $G'(y) = (dG_1(y_1)/dy_1\ ,\ dG_2(y_2)/dy_2, \ldots, dG_n(y_n)/dy_n)^T$ evaluated at the equilibrium $y = 0$, and where $y = x - x_e$. For convenience we rewrite $(L\tilde{N})$ in the form below

$$\dot{y} = (T - D_1)D_2 y \qquad (L\tilde{N}')$$

where $D_2 = G'(y)$ is a positive diagonal matrix for all $y \in \mathbb{R}^n$, and $D_1 = D(G'(y))^{-1}$ is also a positive diagonal matrix; note that $(G'(y))^{-1}$ is defined (and positive) for all $y \in \mathbb{R}$.

Taking $(L\tilde{N}')$ as a reference for our discussion we see that asymptotic stability of $(L\tilde{N}')$ provides necessary conditions for ABST of (N) in the sense of definition 3.1. This is so because local stability is necessary for global stability.

In this context we see that matrix $(T - D_1)D_2$ has to have roots with negative real parts for arbitrary diagonal matrices $D_1 > 0$ and $D_2 > 0$; and this has to do with the classes of matrices defined in the nomenclature. Indeed, the matrix T has to belong, simultaneously, to class $\mathcal{A}_o$ and $\mathbb{D}_o$, this is so because: assuming $D_2 = I$, clearly T has to belong to $\mathcal{A}_o$; on the other hand, assuming that $D_1 = \epsilon\ I$, with $\epsilon \to 0$, we conclude that T has necessarily to belong to $\mathbb{D}_o$. We denote this class by $\mathcal{I}_o := \mathcal{A}_o \cap \mathbb{D}_o$. Note that $\mathcal{I}_o \subset \mathcal{H}_o$ and $\mathcal{I}_o \subset \mathcal{P}_o$, i.e. $\mathcal{I}_o \subset \mathcal{H}_o \cap \mathcal{P}_o$.

Defining $\mathcal{I} := \mathcal{A} \cap \mathbb{D}$, in a similar manner to $\mathcal{I}_o$, it is known that the inclusion $\mathcal{D} \subset \mathcal{I}$ holds. It is also worth noting that, within the class of $\mathcal{Z}$ matrices, we have $\mathcal{H} = \mathcal{D} = \mathbb{D} = \mathcal{A} = \mathcal{I}$ and this is not true in general for matrices that are not in $\mathcal{Z}$ (see [1] for proofs of these facts).

From the fact that $T \in \mathcal{I}_o$ is a necessary condition for ABST and, by Theorem 2.1, $T \in \mathcal{D}_o$ is sufficient, it follows that the inclusion above is also true in the weak case, i.e., $\mathcal{D}_o \subset \mathcal{I}_o$.

As a result of the discussion above, we make the following conjecture.

Conjecture 3.3: $T \in \mathcal{I}_o$ is a necessary and sufficient condition for ABST of (N), where $\mathcal{I}_o := \mathcal{A}_o \cap \mathbb{D}_o$.

However, obtaining a characterization of this class (algebraic or otherwise), would, in our opinion, be a hard problem since, so far, not even a characterization of class $\mathbb{D}$ is available for the general case. In terms of complexity, the characterization problem may well be NP- or co-NP-complete, since the problem of detecting whether a given matrix is in $\mathcal{P}$ has been shown to be co-NP-complete [7]. On the other hand, testing whether a matrix is in $\mathcal{D}$ or not has polynomial complexity [8].

References

[1] D. Hershkowitz, "Recent directions in matrix stability," *Linear Algebra and its Applicat.*, vol. 171, pp. 161–186, 1992.

[2] A. Berman and D. Shasha, "Inertia preserving matrices," *SIAM J. Matrix Analysis and Applicat.*, vol. 12, pp. 209–219, Apr. 1991.

[3] E. Kaszkurewicz and A. Bhaya, "On a class of globally stable neural circuits," *IEEE Trans. Circuits Syst. I*, vol. 41, pp. 171–174, Feb. 1994.

[4] M. Forti, "On global asymptotic stability of a class of nonlinear systems arising in neural network theory," *J. Differential Equations*, vol. 113, pp. 246–264, Oct. 1994.

[5] S. Persidskii, "Problem of absolute stability," *Automation and Remote Contr.*, vol. 30, no. 12, pp. 1889–1895, Dec. 1969.

[6] M. Forti, A. Liberatore, S. Manetti, and M. Marini, "On absolute stability of neural networks," in *Proc. 1994 IEEE Int. Symp. Circuits Systems*, Apr. 1994, pp. 241–244.

[7] G. E. Coxson, "The P-matrix problem is co-NP-complete," *Mathematical Programming*, vol. 64, no. 2, ser. A, pp. 173–178, Apr. 1994.

[8] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory.* Philadelphia, PA: SIAM, 1994.

Learning a Simple Recurrent Neural State Space Model to Behave Like Chua's Double Scroll

Johan A. K. Suykens and Joos Vandewalle

***Abstract*—In this letter, we present a simple discrete time autonomous neural state space model (recurrent network) that behaves like Chua's double scroll. The model is identified using Narendra's dynamic backpropagation procedure. Learning is done in "packets" of increasing time horizon.**

I. Introduction

The last ten years, Chua's circuit has become a paradigm for studying chaos [11]. This simple electrical circuit is able to produce complex behavior and to bifurcate from order to chaos [5]. Moreover, Chua's circuits have been used recently as cells within cellular neural networks instead of classical neurons, leading to phenomena such as, e.g., spiral waves [4], [13], [14]. On the other hand, cellular and generalized cellular neural networks are in itself also able to produce double scroll or n-double scroll-like behavior, respectively, [1], [9], [15], [17]. The latter results were described in continuous time, and design of the networks is based upon mathematical analysis and insight.

In this letter, we propose neural state space models as a general recurrent network architecture, and more specifically, we investigate the problem of learning such a network to behave like a double scroll. The neural state space model can be considered as some general purpose architecture in order to represent or emulate the behavior of some given system and learn its behavior from examples. Like this is done in the CNN universal machine [14], one could also think of arrays of neural state space models but resulting into an overall architecture that is discrete in time as well as in space.

Manuscript received March 24, 1995. This paper was recommended by Associate Editor H.-D. Chiang.

The authors are with the Department of Electrical Engineering, ESAT-SISTA, Katholieke Universiteit Leuven, B-3001 Leuven (Heverlee), Belgium.

IEEE Log Number 9413219.

1057–7122/95$04.00

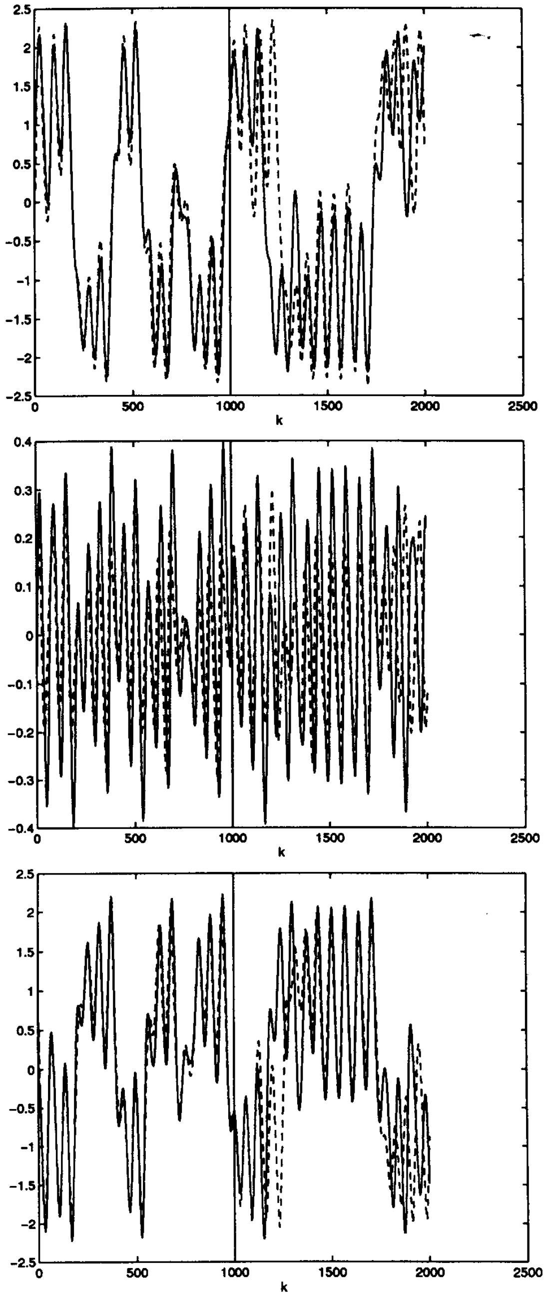

Fig. 1. A simple recurrent neural network emulator for Chua's double scroll using neural state space models. First, second, and third state variable of Chua's circuit through discrete time k is shown at the top, middle, and bottom, respectively. The first 1000 data points were used for training (before the vertical line). Due to the chaotic nature of the system, the error becomes larger behind the vertical line (Full line = data to be tracked from simulation of Chua's circuit by means of a trapezoidal integration rule with constant step length; Dashed line = behavior of the identified neural state space model).

This letter is organized as follows: In Section II, we discuss neural state space models, together with dynamic backpropagation. In Section III, a neural state space model is trained to behave as Chua's double scroll.

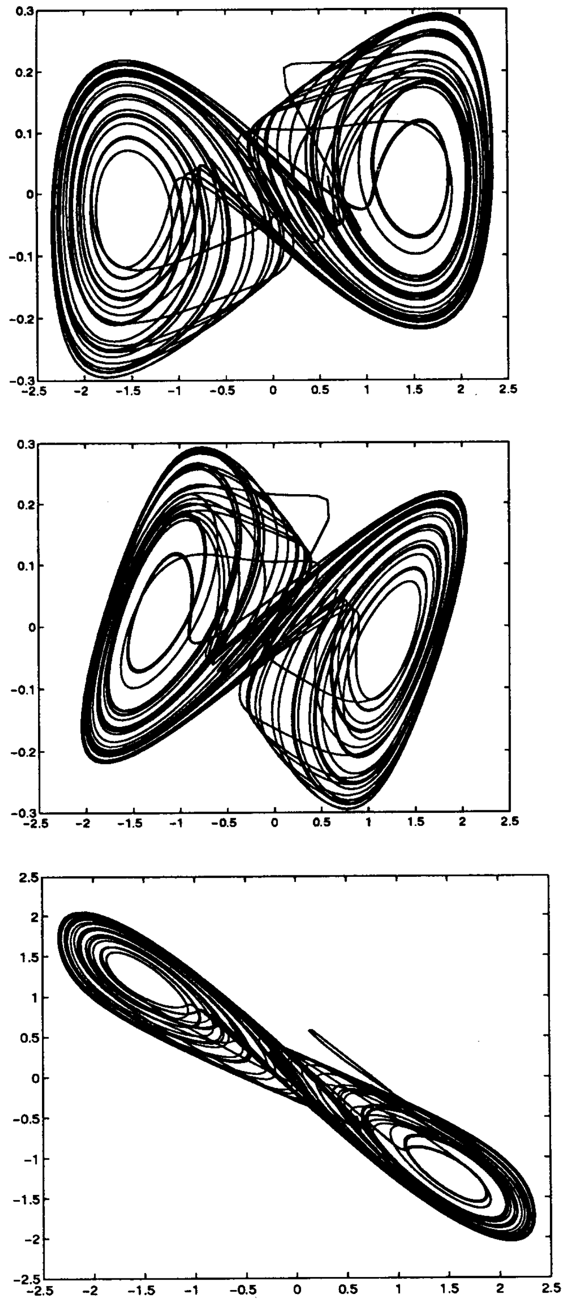

Fig. 2. Double scroll attractor, reconstructed by the third order autonomous neural state space model with three hidden neurons: (Top) (x_k, y_k). (Middle) (z_k, y_k). (Bottom) (x_k, z_k).

II. Neural State Space Models and Dynamic Backpropagation

We consider here discrete time autonomous nonlinear systems

$$x_{k+1} = f(x_k) \tag{1}$$

with state vector $x_k \in \mathbb{R}^n$ and $f(.)$ a continuous nonlinear mapping. Parametrizing $f(.)$ by a multilayer feedforward neural network with one hidden layer leads to the model

$$\hat{x}_{k+1} = W \tanh(V\hat{x}_k + \beta) \quad , \hat{x}_0 = x_0 \text{ given} \tag{2}$$

with interconnection matrices $W \in \mathbb{R}^{n \times n_h}$, $V \in \mathbb{R}^{n_h \times n}$, bias vector $\beta \in \mathbb{R}^{n_h}$, and n_h the number of hidden neurons. The hyperbolic tangent function $\tanh(.)$ is taken elementwise. Such a parametrization makes sense because any continuous nonlinear function can be approximated arbitrarily well on a compact interval by a multilayer perceptron with one or more hidden layers [6], [8], [10].

Remarks

- Nonautonomous models in state space form that are parametrized by multilayer perceptrons for doing nonlinear system identification are discussed in [16] and are called "neural state space models." Both process noise and measurement noise can be taken into account. The model (2) corresponds to the deterministic case with zero external input.
- Input/output models parametrized by neural networks such as NARX and NARMAX models are discussed, e.g., in [3]. These models correspond to feedforward networks. The neural state space model (2) on the other hand is recurrent.
- Concerning identifiability, the representation (2) is only unique up to a similarity transformation and sign reversals because the hidden nodes can be permuted and the sign of all the weights associated to a particular hidden node can be changed.

Suppose now N data $\{x_k\}_{k=0}^{N-1}$ are available and we are interested in minimizing the error between the state $\hat{x}_k$ of the model (2) and the given x_k. For prediction error algorithms, the aim is then to minimize the following cost function off-line:

$$\min_{\theta} J(\theta) = \frac{1}{2N} \sum_{k=1}^{N} [x_k - \hat{x}_k(\theta)]^T \, [x_k - \hat{x}_k(\theta)]. \tag{3}$$

Here $\theta = [W(:); V(:); \beta]$ denotes the unknown parameter vector and "(:)" is a columnwise scanning of a matrix to a vector. The nonlinear optimization problem (3) has many local optima in general. In case we use a gradient based local optimization method, the gradient of the cost function is given by

$$\frac{\partial J}{\partial \theta} = \frac{1}{N} \sum_{k=1}^{N} [x_k - \hat{x}_k(\theta)]^T \, \frac{\partial \hat{x}_k(\theta)}{\partial \theta} \tag{4}$$

where the expression of $\frac{\partial \hat{x}_k(\theta)}{\partial \theta}$ is generated by a so-called sensitivity model, which is in itself a dynamical system because the network is recurrent (see [12]). Let us denote the model (2) as

$$\hat{x}_{k+1} = \Phi(\hat{x}_k; \alpha) \tag{5}$$

with α an element of the parameter vector θ. The sensitivity model becomes then

$$\frac{\partial \hat{x}_{k+1}}{\partial \alpha} = \frac{\partial \Phi}{\partial \hat{x}_k} . \frac{\partial \hat{x}_k}{\partial \alpha} + \frac{\partial \Phi}{\partial \alpha} \tag{6}$$

which is a dynamical system with state vector $\frac{\partial \hat{x}_k}{\partial \alpha} \in \mathbb{R}^n$ and input vector $\frac{\partial \Phi}{\partial \alpha} \in \mathbb{R}^n$. The Jacobian $\frac{\partial \Phi}{\partial \hat{x}_k} \in \mathbb{R}^{n \times n}$ is evaluated around the nominal trajectory. In order to write down the derivatives, let us take an elementwise notation for (2)

$$\hat{x}^i := \sum_j w^i_j \tanh(\sum_r v^j_r \, \hat{x}^r + \beta^j) \tag{7}$$

where $\{.\}^i$ and $\{.\}^i_j$ denote, respectively, the i-th element of a vector and the ij -th entry of a matrix. The time index k is omitted after introducing the assignment operator ":=." Defining $\rho^l = \sum_r v^l_r \, \hat{x}^r + \beta^l$ one obtains

$$\begin{aligned} \frac{\partial \Phi^i}{\partial w^j_l} &= \delta^i_j \tanh(\rho^l) \\ \frac{\partial \Phi^i}{\partial v^j_l} &= w^i_j \, (1 - \tanh^2(\rho^j)) \, \hat{x}^l \\ \frac{\partial \Phi^i}{\partial \beta^j} &= w^i_j \, (1 - \tanh^2(\rho^j)) \\ \frac{\partial \Phi^i}{\partial \hat{x}^r} &= \textstyle\sum_j w^i_j \, (1 - \tanh^2(\rho^j)) \, v^j_r . \end{aligned} \tag{8}$$

The steepest descent algorithm

$$\hat{\theta}_{t+1} = \hat{\theta}_t - \eta \, \frac{\partial J}{\partial \theta} \tag{9}$$

is called dynamic backpropagation and was introduced by Narendra and Parthasarathy [12]. Here η is the learning rate and $\hat{\theta}_t$ the t-th iterate. More advanced local optimization methods are, e.g., quasi-Newton and conjugate gradient algorithms [7].

III. Learning to Behave as a Double Scroll

Let us consider now Chua's circuit

$$\begin{cases} C_1 \frac{dv_{C_1}}{dt} &= G(v_{C_2} - v_{C_1}) - g(v_{C_1}) \\ C_2 \frac{dv_{C_2}}{dt} &= G(v_{C_1} - v_{C_2}) + i_L \\ L \frac{di_L}{dt} &= -v_{C_2} \end{cases} \tag{10}$$

where v_{C_1},v_{C_2}, i_L denote, respectively, the voltage across C_1 and C_2 and the current through L and $g(v_{C_1})$ is a piecewise-linear function consisting of two breakpoints

$$g(v_{C_1}) = m_0 v_{C_1} + 0.5(m_1 - m_0)|v_{C_1} + B_p| + 0.5(m_0 - m_1) \, |v_{C_1} - B_p|. \tag{11}$$

By setting the parameters $1/C_1 = 9$, $1/C_2 = 1$, $1/L = 7$, $G = 0.7$, $m_0 = -0.5$, $m_1 = -0.8$, $B_p = 1$, the double scroll attractor is obtained [5]. The ODE (10) was simulated by means of a trapezoidal integration rule with constant step length equal to 0.05 for initial state $x_0 = [0.9365 \;\; -0.0610 \;\; -0.1889]^T$. The training set or the orbit to be tracked consists of the first 1000 data points (corresponding to samples in the time interval [0, 50]).

We propose here a simple neural state space model with three hidden neurons and zero bias vector β in (2) for nonlinear system identification. Starting from random initial parameter vectors, it turns out that off-line learning of the complete given orbit of 1000 data points by means of a gradient based optimization method is extremely hard. In fact this does not come as a surprise because it is well-known that learning long-term dependencies with gradient descent is difficult

$$\begin{aligned} W &= \begin{bmatrix} 3.191701795026490e+00 & -3.961031505875602e+00 & -2.544300729387972e+00 \\ 6.302937463967251e-01 & 2.746315947131907e+00 & 8.024248038305574e-01 \\ -1.411085817901605e+00 & 8.436161546347900e+00 & 3.174868294146957e+00 \end{bmatrix} \\ V &= \begin{bmatrix} -2.446514424620466e-01 & 1.557093499163188e+00 & -6.192223155626144e-01 \\ 4.935534636051888e-01 & -9.111907179654775e-01 & 6.785493890595847e-01 \\ -1.711046461663226e+00 & 3.794433297552783e+00 & -2.105183558255941e+00 \end{bmatrix} \end{aligned} \tag{12}$$

(see [2]). Instead of applying global optimization schemes such as, e.g., multistart local optimization, simulated annealing, or genetic algorithms, we took a different approach here. The training data set was split into "packets" of increasing time horizon. The following *ad hoc* optimization procedure was applied then:

- Generate a random initial parameter vector θ_0 and put $a := 1$.
- Do while $a < a_{final}$

  a. $\theta^* = \arg\min_\theta J(\theta) = \frac{1}{2a\Delta}\sum_{k=1}^{a\Delta}[x_k - \hat{x}_k]^T[x_k - \hat{x}_k]$

  b. $\theta_0 := \theta^*$ and $a := a + 1$

  End

Here θ_0 is the starting point for the optimization problems and θ^* the local optimal solution. Hence, the time horizon $a\Delta$ is increased until all the N data points are consumed. The idea of this packets strategy is that one does not start learning new things before the previous parts of the orbit are memorized.

In order to identify the double scroll, we took $\Delta = 50$, $a_{final} = 20$. The danger for overfitting is not high because of the small scale neural network of only three hidden neurons. The following neural state space model was obtained and behaves like the double scroll, see (12) at the bottom of the previous page, for initial state $x_0 = [0.9365\ \ -0.0610\ \ -0.1889]^T$. A quasi-Newton method with BFGS updating of the Hessian and mixed cubic-quadratic line search was applied for the optimization with random initial parameter vector. The optimization was done with Matlab's optimization toolbox. The time consuming parts such as the simulation of the neural state space model and its sensitivity model were written in C code and called within Matlab, making use of Matlab's *mex* facility. The simulation results are shown in Figs. 1 and 2.

IV. Conclusion

In this paper, a discrete time recurrent nonlinear state space model was identified that behaves like Chua's double scroll. It is well known that learning recurrent networks is more difficult than feedforward networks, especially for systems with complicated dynamics such as chaotic systems. We avoided the use of global optimization methods and proposed a packet method with increasing time horizon (local optimization), which was successful in order to learn the double scroll trajectory, given some initial state. However, more research is needed in general on learning complex behavior by means of recurrent networks.

Acknowledgment

This work was carried out at the ESAT laboratory and the Interdisciplinary Center of Neural Networks (ICNN) of the Katholieke Universiteit Leuven, in the framework of the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology, and Culture (IUAP-17). The scientific responsibility rests with its authors.

References

[1] P. Arena, P. Baglio, F. Fortuna, and G. Manganaro, "Generation of n-double scrolls via cellular neural networks," submitted to *Int. J. Circuit Theory Applicat. (Special Issue Cellular Neural Networks)*, 1994.

[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, pp. 157–166, 1994.

[3] S. Chen, S. Billings, and P. Grant, "Nonlinear system identification using neural networks," *Int. J. Contr.*, vol. 51, no. 6, pp. 1191–1214, 1990.

[4] L. O. Chua, "Chua's circuit 10 years later," *Int. J. Circuit Theory Applicat.*, vol. 22, pp. 279–305, 1994.

[5] L. O. Chua, M. Komuro, and T. Matsumoto, "The double scroll family," *IEEE Trans. Circuits Syst.-I*, vol. CAS-33, pp. 1072–1118, 1986.

[6] G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Math. Contr., Signals Syst.*, vol. 2, pp. 183–192, 1989.

[7] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Chichester and New York: Wiley, 1987.

[8] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.

[9] C. Guzelis and L. O. Chua, "Stability analysis of generalized cellular neural networks," *Int. J. Circuit Theory Applicat.*, vol. 21, pp. 1–33, 1993.

[10] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[11] R. N. Madan, (Guest Ed.) "Special issue on Chua's circuit: A paradigm for chaos," *J. Circuits, Syst. Computers Part 1*, vol. 3, no. 1, Mar. 1993, also *J. Circuits, Syst. Computers Part 2*, vol. 3, no. 2, June 1993.

[12] K. S. Narendra and K. Parthasarathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 252–262, 1991.

[13] A. P. Perez-Munuzuri, V. Perez-Munuzuri, V. Perez-Villar, and L. O. Chua, "Spiral waves on a 2-D array of nonlinear circuits," *IEEE Trans. Circuits Syst.-I*, vol. 40, pp. 872–877, 1993.

[14] T. Roska and L. O. Chua, "The CNN universal machine: an analogic array computer," *IEEE Trans. Circuits Syst.-II*, vol. 40, pp. 163–173, Mar. 1993.

[15] J. A. K. Suykens and J. Vandewalle, "Generation of n-double scrolls (n = 1, 2, 3, 4, . . .)," *IEEE Trans. Circuits Syst.-I*, vol. 40, pp. 861–867, 1993.

[16] J. A. K. Suykens, B. De Moor, and J. Vandewalle, "Nonlinear system identification using neural state space models, applicable to robust control design," to appear in *Int. J. Contr.*

[17] F. Zou and J. A. Nossek, "Bifurcation and chaos in cellular neural networks," *IEEE Trans. Circuits Syst.-I*, vol. 40, pp. 166–173, 1993.

Experimental Control of Chaotic Behavior of Buck Converter

Gautam Poddar, Krishnendu Chakrabarty, and Soumitro Banerjee

***Abstract*—This letter reports a method for control of chaos in the dc-dc buck converter. The method differs from the existing ones and is particularly useful for piecewise linear systems with switching nonlinearity.**

I. Introduction

Power electronic circuits broadly fall into the category of piecewise linear systems with an overall nonlinear behavior contributed by discrete switching phenomena. It has been shown that such systems can exhibit deterministic chaos [1]–[3], especially in presence of a feedback loop. Since power electronic circuits with current or voltage feedback have wide industrial application, the control of chaos in such systems assume special importance.

The existing methods for controlling chaos (as surveyed in [4]) generally require complicated computation to be performed on-line or additional periodic reference voltage. The method proposed in this

Manuscript received March 24, 1995. This paper was recommended by Associate Editor H.-D. Chiang.

The authors are with the Department of Electrical Engineering, Indian Institute of Technology, Kharagpur, India 721302.

IEEE Log Number 9413220.