International Journal of Bifurcation and Chaos, Vol. 9, No. 4 (1999) 723–733

COMPOSITION METHODS FOR THE SIMULATION OF ARRAYS OF CHUA'S CIRCUITS

YVES MOREAU and JOOS VANDEWALLE
Department of Electrical Engineering,
Katholieke Universiteit Leuven,
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium

Received February 21, 1998; Revised September 9, 1998

Composition methods are methods for the integration of ordinary differential equations arising from differential geometry, or more precisely, Lie algebra theory. We apply them here to the simulation of arrays of Chua's circuits. In these methods, we split the vector field of the array of Chua's circuits into its linear part and its nonlinear part. We then solve the elementary differential equation for each part separately — which is easy since the equations for the nonlinear part are all decoupled — and recombine these contributions into a sequence of compositions. This splitting gives rise to simple integration rules for arrays of Chua's circuits, which we compare to more classical approaches: fixed time-step explicit Euler and adaptive fourth-order Runge–Kutta.

1. Introduction

The standard methods for the integration of Ordinary Differential Equations (ODEs) are schemes such as the Euler method or the Runge–Kutta method. These schemes are straightforwardly applied to the integration of continuous-time recurrent neural networks, in particular Cellular Neural Networks (CNNs) consisting of arrays of Chua's circuits. However, physicists have recently introduced a completely different class of methods for the solution of ordinary differential equations: the composition methods [Forest & Ruth, 1990; Yoshida, 1990]. The spirit of these methods is that if the vector field of the differential equation is the superposition of some elementary vector fields, we can approximate its solution by composing the flows of the elementary contributions. Composition methods are particularly useful for numerically integrating ODEs when the equations have some special structure, of which we can take advantage [McLachlan, 1995]. The authors have recently introduced them in the field of neural networks [Moreau & Vandewalle, 1997].

To demonstrate further the relevance of composition methods to the field of neural networks, we present their application to the integration of the differential equations describing CNNs made out of an array of Chua's circuits. CNNs are effective for parallel signal processing and real-time simulation of phenomena described by partial differential equations. They have therefore been used for image processing and for the study of nonlinear spatio-temporal behaviors [Muñuzuri *et al.*, 1995]. Their main interest from the applied viewpoint is that a large array can be integrated into a monolithic VLSI chip and that CNNs can model a large number of practical systems — for example, in image processing.

We will first present the elements of dynamical system theory and Lie algebra theory necessary for the exposition of composition methods. We will then show how to use this theory for the integration of ordinary differential equations using composition methods. We will apply these methods first to a single Chua's circuit and then to an array of coupled Chua's circuits. We will compare

the performance of these methods with those of standard methods: fixed time-step explicit Euler and adaptive fourth-order Runge–Kutta. Not only do composition methods provide a new viewpoint on the dynamics of arrays of Chua's circuits, but they provide enhanced performance for their simulation.

2. Lie Algebra Theory

The CNNs that we consider here are arrays of Chua's circuits. These CNNs are described by ordinary differential equations $\dot{x}(t) = A(x(t))$ defined on the state-space $\mathbb{R}^n$, where the vector field A describes the dynamics of a number of nonlinear circuits evolving with a linear coupling. We write the ODE as $\dot{x}(t) = A(x(t))$ to follow the conventions of Lie algebra theory. We can write its solution, as a function of the initial condition x_0, in two forms: as a flow $x(t) = \Phi(x_0, t)$ (as is standard in the dynamical system literature) or as an exponential solution $x(t) = e^{tA}x_0$ (as in the Lie algebra literature [Arnold, 1989]). The latter notation should read: *"$x(t)$ is the image after time t of the initial condition x_0 under the flow of $\dot{x} = A(x)$"*. We stress that, in our notation, e^{tA} is a nonlinear transformation of the state-space and *not* a matrix as in the case of linear systems. The notation is also coherent with the basic properties of the exponential. For example (using the dot "$\cdot$" to denote the composition of maps), we have

$$e^{(t+s)A}x_0 = e^{sA} \cdot e^{tA}x_0 \,,$$

which is nothing but a restatement of the group property of the flow $\Phi(x_0, t + s) = \Phi(\Phi(x_0, t), s)$ [Irwin, 1980]. This property states that, for any initial condition, evolving according to the differential equation $\dot{x} = A(x)$ for $t + s$ units of time is equivalent to: First, evolving for units of time; then, starting from where we have arrived, evolving for another s units of time.

Lie algebra theory is an important tool in physics [Arnold, 1989; Bluman & Kumei, 1989] and an essential part of nonlinear system theory [Isidori, 1989]. It provides here the framework for the presentation of composition methods for ODEs. The Lie algebra we consider here is the vector space of all smooth vector fields, where we define a supplementary operation: the Lie bracket of two vector fields, which is again a vector field. Bracketing is a bilinear, anti-symmetric operation, which also satisfies the Jacobi identity [Arnold, 1989]:

$$\begin{aligned}
&[A, B] = -[B, A], && \text{antisymmetry}\,,\\
&[aA, B] = a[A, B], && \text{bilinearity}\,,\\
&[A, B + C] = [A, B] + [A, C], && \text{bilinearity}\,,\\
&[[A, B], C] + [[B, C], A] + [[C, A], B] = 0, && \text{Jacobi identity}\,.
\end{aligned} \tag{1}$$

Recalling that we take the product of exponentials to denote the composition of these maps, we can define the vector field $[A, B]$ as the Lie bracket of the vector fields A and B:

$$[A, B](x) = \left.\frac{\partial^2}{\partial s \partial t}\right|_{t=s=0} e^{-sB}e^{-tA}e^{sB}e^{tA}(x)\,.$$

The bracket $[A, B]$ is called the commutator of the vector fields, as it measures the degree of noncommutativity of the flows of the vector fields ($[A, B] = 0 \Leftrightarrow e^{tA}e^{tB} = e^{tB}e^{tA}$). As we define our vector fields on $\mathbb{R}^n$, we can further express the bracket as

$$[A, B]_i = \sum_{j=1}^{n} \left(B_j \frac{\partial A_j}{\partial x_i} - A_j \frac{\partial B_j}{\partial x_i} \right) .$$

The last mathematical tool we need is the Baker–Campbell–Hausdorff (BCH) formula. This formula gives an expansion for the product of two exponentials of elements of the Lie algebra [Arnold, 1989]:

$$e^{tA}e^{tB} = e^{t(A+B)+\frac{t^2}{2}[A,B]+\frac{t^3}{12}([A,[A,B]]+[B,[B,A]])+\ldots}\,. \tag{2}$$

It should be interpreted as follows. Letting the flow of $\dot{x} = B(x)$ act on the initial condition of the system for t, and then — from where we have arrived — letting the flow of $\dot{x} = A(x)$ act for another time-step t is equivalent to letting the flow of $\dot{x} = ((A + B) + t/2[A, B] + t^2/12([A, [A, B]] + [B, [B, A]]) + \ldots)(x)$ act on the initial condition for t.

2.1. *Integration of ordinary differential equations by compositions*

Let us now try to solve ordinary differential equations using compositions. Suppose we want to solve the ODE $\dot{x}(t) = X(x(t))$ for a time-step of Δt. Since $x(t) = e^{\Delta t X} x_0$, the problem becomes that of building an approximation to $e^{\Delta t X}$. This problem has recently been the focus of much attention in the field of numerical analysis, especially for the integration of Hamiltonian differential equations [McLachlan, 1995]. The basic idea is that, if you can split the vector field X into elementary parts for which you can solve the differential equation directly, you can recombine these solutions to approximate the solution of the more complex system.

Suppose, in a first step, that the vector field X is the sum of two vector fields: $X = A + B$, where you can integrate A and B either analytically or much more easily than X. Then we can use the BCH formula to produce a first-order approximation to the exponential map:

$$\textbf{BCH}: \quad e^{tX} = e^{tA} e^{tB} + o(t^2)\,. \tag{3}$$

You can check this relation by multiplying the left- and right-hand sides of Eq. 2 by $e^{tX} (= e^{t(A+B)})$, and then expanding using the BCH formula itself (2). The first-order approximation (3) between the solution of A and B, and the solution of X is the essence of the method since it shows that we can approximate an exponential map (that is the mapping arising from the solution of an ODE) by composing simpler maps (Fig. 1).

By using the BCH formula to eliminate higher-order terms as we did for the first-order approximation, but on the composition of three terms, we can show that the following symmetric leapfrog scheme is second order:

$$\begin{aligned}\textbf{Leapfrog}: \quad e^{tX} &= e^{\frac{t}{2}A} e^{tB} e^{\frac{t}{2}A} + o(t^3)\,,\\ &= S(t) + o(t^3)\,.\end{aligned}$$

Using this leapfrog scheme as a basis element, we can build a fourth-order scheme:

$$\begin{aligned}\textbf{Fourth order}: \quad e^{tX} &= S(ct)S(dt)S(ct) + o(t^5)\,,\\ &= SS(t) + o(t^5)\,,\end{aligned} \tag{4}$$

with $c = -2^{1/3}/(2 - 2^{1/3})$ and $d = 1/(2 - 2^{1/3})$. Repeating the leapfrog strategy, Yoshida [1990] showed that it is possible to produce an approximation to e^{tX} up to any order:

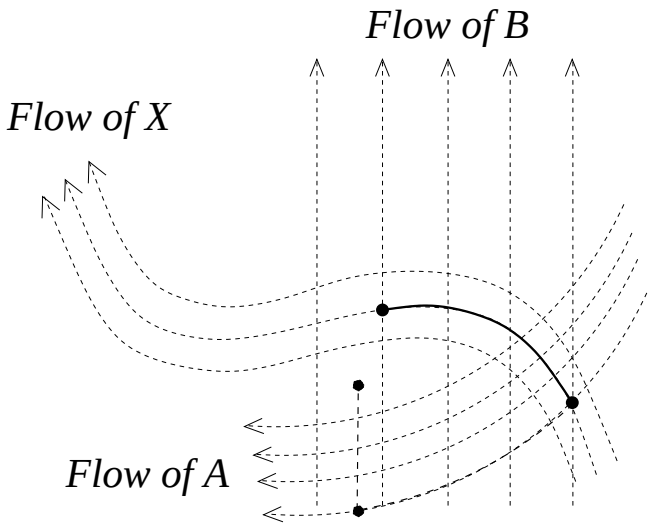

Fig. 1. First-order approximation by composition. The vector field X of the system $\dot{x} = X(x)$ is the sum of two vector fields A and B. The Baker–Campbell–Hausdorff formula gives a first-order approximation with $e^{\Delta t X}(x_0) \approx e^{\Delta t B} e^{\Delta t A}(x_0)$. The bold solid line represents the evolution of the system between 0 and Δt from the initial condition x_0. The dashed lines represent the evolution from x_0 to $e^{\Delta t A}(x_0)$ and from $e^{\Delta t A}(x_0)$ to $e^{\Delta t B} e^{\Delta t A}(x_0)$.

$$\begin{aligned}&\textbf{Arbitrary order}: \quad \text{There exists } k \text{ and}\\ &w_1, v_1, \ldots, w_k, v_k \text{ such that}\\ &e^{tX} = e^{w_1 tA} e^{v_1 tB} \cdots e^{w_k tA} e^{v_k tB} + o(t^{p+1})\,.\end{aligned} \tag{5}$$

This result is useful as long as t is sufficiently small to keep the higher-order terms negligible. If t becomes too large, the previous result offers no guarantees. But Suzuki [1993] showed that higher-order repeated-leapfrog schemes converge everywhere. Further, Forest and Ruth [1990] also showed that approximations can be built for more than two vector fields: If $X = \sum_{i=1}^{m} A_i$ then

$$\begin{aligned}&\text{there exists } k \text{ and } w_{ij}\ (i = 1, \ldots, m;\\ &j = 1, \ldots, k) \text{ such that}\\ &e^{tX} = \prod_{j=1}^{k} \prod_{i=1}^{m} e^{w_{ij} t A_i} + o(t^{p+1})\,.\end{aligned} \tag{6}$$

3. Composition Integrators

We present a short-list of different composition integrators (Table 1) for the case where the vector field can be split into two parts: $X = A + B$. These

Table 1. Formulas for the composition methods.

Order	Substeps	Error	Formula
2	3	0.070	$S(\Delta t) = e^{(\Delta t/2)A} e^{\Delta t B} e^{(\Delta t/2)A}$
2	5	0.026	$e^{a_1 \Delta t A} e^{b_1 \Delta t B} e^{a_2 \Delta t A} e^{b_1 \Delta t B} e^{a_1 \Delta t A}$ $a_1 = 0.1932$, $b_1 = 0.5$, $a_2 = 1 - 2a_1$
4	7	0.098	$S(w_1 \Delta t) S(w_2 \Delta t) S(w_1 \Delta t)$ $w_1 = 1.3512$, $w_2 = 1 - 2w_1$
4	11	0.033	$S(w_1 \Delta t) S(w_2 \Delta t) S(w_3 \Delta t) S(w_2 \Delta t) S(w_1 \Delta t)$ $w_1 = 0.28$, $w_2 = 0.6254$, $w_3 = 1 - 2w_1 - 2w_2$
4	11	0.0046	$e^{a_1 \Delta t A} e^{b_1 \Delta t B} e^{a_2 \Delta t A} e^{b_2 \Delta t B} e^{a_3 \Delta t A} e^{b_3 \Delta t B}$ $\times\ e^{a_3 \Delta t A} e^{b_2 \Delta t B} e^{a_2 \Delta t A} e^{b_1 \Delta t B} e^{a_1 \Delta t A}$ $a_1 = 0.0893$, $b_1 = 0.4$, $a_2 = -0.0973$, $b_2 = -0.1$, $a_3 = 1 - 2a_1 - 2a_2$, $b_3 = 1 - 2b_1 - 2b_2$

methods can be recursively used for the case of more than two vector fields. For example, in the case $X = A + B + C$, we can write $X = A + D$, where $D = B + C$. We then choose a composition method for D, which we use as basis step for the integration of X. The error constant associated to each method is the effective error constant defined in [McLachlan, 1995]. This error measure accounts for the fact that, for a given order, methods with more substeps maybe more precise, but they also require more computational work. This tradeoff would allow methods composed of fewer substeps to use a smaller step size, therefore increasing their precision. For a given order, even if the effective error constant decreases as the number of substeps increases, it may not be advantageous to use the method with the most substeps for finite time steps. This choice will depend on the system we are integrating, and on the error we require.

We present two types of methods: the symmetric methods and the symmetric methods composed of symmetric steps. The symmetric methods are presented directly as a product of exponentials, while the symmetric methods composed of symmetric steps are presented as a product of symmetric leapfrog steps $S(\Delta t)$. The leapfrog is the simplest method to implement, while the fourth-order symmetric method (error = 0.0046) is the most efficient. We therefore recommend the leapfrog for preliminary experiments and the fourth-order symmetric method (error = 0.0046) for final implementations. For the formulas of higher-order integrators, see [McLachlan, 1995]. A remark of fundamental importance is that these formulas are valid even without analytical solutions of $e^{\Delta t A}$ and $e^{\Delta t B}$. Indeed, we can replace these solutions in the following symmetric methods by any first-order integrator (e.g., $I + \Delta t A = e^{\Delta t A} + o(\Delta t^2)$) and still guarantee the same order of approximation. Similarly, in the symmetric methods composed of symmetric steps, we can replace the leapfrog $S(\Delta t)$ by any second-order symmetric integrator.

4. Application to CNNs

We will now apply this technique to arrays of Chua's circuits. Chua's circuit is a nonlinear electrical circuit. We can write it as a state-space model or ordinary differential equation of the form

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{bmatrix} -a & a & 0 \\ b & -b & 1 \\ 0 & -c & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} g(x) \\ 0 \\ 0 \end{pmatrix}, \quad (7)$$

with $g(x)$ a piecewise linear function: $g(x) = -(a/2)[(s_1 + s_2)x + (s_0 - s_1)(|x - \beta_1| - |\beta_1|) + (s_2 - s_0)(|x - \beta_2| - |\beta_2|)]$. This system can exhibit chaotic behavior for appropriate parameter values — for example, $a = 6.3$, $b = 0.7$, $c = 7$, $s_0 = -1.14286$, $s_1 = s_2 = -0.71429$, $\beta_1 = -1$ and $\beta_2 = 1$. If we split the vector field of (7) into its linear part and

its nonlinear part, we see that we can solve both parts explicitly. If we call A the matrix containing the parameters a, b, c, the solution for the linear part is $x(\Delta t) = e^{\Delta t A_1} x_0$, where the exponential is the classical matrix exponential. The solution for the nonlinear part is $x(\Delta t) = e^{tg}(x_0)$. In the simple case where $\beta_1 < 0 < \beta_2$, we can compute the positive part of the solution as follows:

$$\tau(x_0) = \begin{cases} \dfrac{1}{s_2} \ln\left(\dfrac{x_0 - \delta}{\beta_2 - \delta}\right), & x > \beta_2\,, \\ \dfrac{1}{s_0} \ln\left(\dfrac{x_0}{\beta_2}\right), & 0 < x < \beta_2\,; \end{cases}$$

$$x(\Delta t) = \begin{cases} e^{s_2(\tau(x_0)+\Delta t)}(\beta_2 - \delta)\,, & \tau(x_0) + \Delta t > 0\,, \\ e^{s_0(\tau(x_0)+\Delta t)}\beta_2\,, & \tau(x_0) + \Delta t \le 0\,; \end{cases} \tag{8}$$

where $\delta = \beta_2(1 - s_0/s_2)$. As $x = 0$ is a fixed point, trajectories starting at positive initial conditions always remain positive. For negative initial conditions, we have the same formula except that β_1 takes the place of β_2, s_1 takes the place of s_2, and all signs are reversed. Therefore, we can integrate the equations of Chua's circuit with a composition method.

4.1. *Arrays of Chua's circuits*

Moreover, this type of splitting applies to general CNNs. For example, we can have a CNN composed by an array of Chua's circuits (7) coupled together by some constant linear coupling. If we have n circuits having each three state variables, the equations describing the evolution of the system would be

$$\begin{pmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{z}_i \end{pmatrix} = \begin{bmatrix} -a & a & 0 \\ b & -b & 1 \\ 0 & -c & 0 \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \begin{pmatrix} g(x_i) \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \rho \Sigma_{j=1}^{n} \xi_{ij} x_j \\ 0 \\ 0 \end{pmatrix} \tag{9}$$

with $i = 1, \ldots, n$ and where the coupling term ξ_{ij} is equal to one if circuit i is connected to circuit j, and is equal to zero else. Observe that, however large the number of circuits is, the system is made of a large linear part (the first and third terms) and of a nonlinear part that consists of n identical decoupled piecewise linear equations. This splitting means that, once again, both the linear and the nonlinear part can be solved explicitly (8), and combined to find the solution of this large system of ODEs. If we call x_{tot} the total state of the system: $x_{\text{tot}} = (x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n)^T$, A the matrix representing the linear part of the dynamics of the Chua's circuits and the linear interconnections, and Γ the nonlinear part of the dynamics; we have the following ODE:

$$\dot{x}_{\text{tot}} = A x_{\text{tot}} + \Gamma(x_{\text{tot}})\,.$$

As a first toy problem, we present a simulation of the solution of such a CNN for the case of a two-by-two array of Chua's circuits. The circuits are placed on the corners of a square and symmetrically connected in a ring. We choose the value of the interconnection strength as $\rho = 0.1$. We realize the simulation in the Matlab environment. We use a second-order integrator: $x_{\text{tot}}(\Delta t) = e^{(\Delta t/2)A} e^{\Delta t \Gamma} e^{(\Delta t/2)A}(x_{\text{tot}})$. The time-step for the integration is $\Delta t = 0.1$. In Fig. 2, we see the evolution of the state of the four circuits. The four circuits have identical parameters but different random initial conditions, hence the behavior of the system is not symmetric.

An important point that comes under consideration is how to solve efficiently the one-dimensional nonlinear ODEs arising through the splitting of the vector field of the array of Chua's circuits. An analytical solution might not be difficult to compute for this piecewise linear ODE, but its implementation might be rather inefficient. Therefore, we present a method for computing the solution of a one-dimensional ODE using a lookup table.

4.2. *Table lookups for the solution of one-dimensional ODEs*

The main observation is that the behavior of a one-dimensional ODE is monotone between two fixed points. Let us write the ODE as $\dot{x} = A(x)$ with $x \in \mathbb{R}$, and its solution as $x(t) = \Phi(x_0, t)$. Suppose first that the ODE has no fixed points. Then, the trajectory passing through $x_0 = 0$ will cover the whole of $\mathbb{R}$. Define this trajectory as the function $\psi(t) = \Phi(0, t)$. Because of the group property of the flow ($\Phi(\Phi(x_0, t), s) = \Phi(x_0, t + s)$), we have that $\Phi(x_0, t) = \Phi(\Phi(0, \psi^{-1}(x_0)), t) = \Phi(0, \psi^{-1}(x_0) + t)$. So that $\Phi(x_0, t) = \psi(\psi^{-1}(x_0) + t)$. Thus, by storing

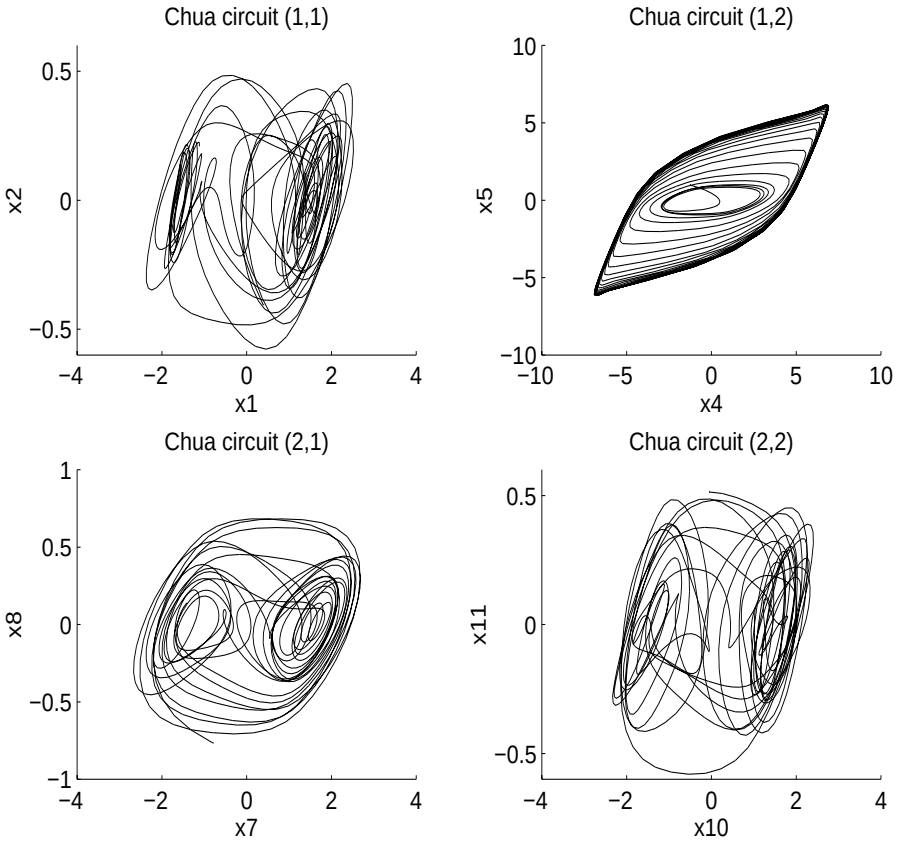

Fig. 2. Evolution of the state of the CNN array. Simulation by a leapfrog composition method. The array consists of four Chua's circuits in a ring. The circuits have identical parameters ($a = 6.3$, $b = 0.7$, $c = 7$, $s_0 = -1.14286$, $s_1 = s_2 = -0.71429$, $\beta_1 = -1$ and $\beta_2 = 1$) and the coupling term is $\rho = 0.1$. The initial conditions of the circuits are different and chosen at random. We plot the evolution of the x and y variables of each of the four circuits.

the solution ψ of $\dot{x} = A(x)$, with $x(0) = 0$, and its inverse in a lookup table, we are able to compute the solution of the one-dimensional ODE in just two table lookups.

Suppose now that the ODE has k fixed points: $z_1, \ldots, z_k$. Set also $z_0 = -\infty$ and $z_{k+1} = \infty$. Define $v_0 = z_1 - 1$, $v_1 = (z_1 + z_2)/2, \ldots, v_{k-1} = (z_{k-1} + z_k)/2$, $v_k = z_k + 1$. Define $\psi_i(t) = \Phi(v_i, t)$, then we know that this trajectory $\psi_i(t)$ will cover the whole of the interval $]z_i, z_{i+1}[$. Thus, for all $x_0 \in]z_i, z_{i+1}[$: $\Phi(x_0, t) = \psi_i(\psi_i^{-1}(x_0) + t)$ for all $i = 0, \ldots k$. We can thus store the function ψ_i^{-1} in a lookup table and the function ψ_i in another lookup table consisting of k subtables (one per interval). This procedure gives us a powerful way to compute repeated integrations of a one-dimensional ODE.

5. Simulations

To show that the composition method is an efficient integration method, we ran a simulation of a spiral wave on an array of 50×50 Chua's circuits (9) as shown in [Pivka, 1995] and [Muñuzuri *et al.*, 1995]. The appropriate set of parameters is

$$\begin{aligned} &a = 10\,, && b = 1\,, && c = 0.3014987\,,\\ &s_1 = 0.078573\,, && s_2 = 55\,, && s_0 = -1.25719\,,\\ &\beta_1 = -1\,, && \beta_2 = 0.023744\,. \end{aligned}$$

These parameters correspond to a bistable cell with two stable equilibria: $P^- = [-1.238, 1.238]$, $P^+ = [0.02385, -0.02385]$. The appropriate initial condition is described as follows. All the x and y variables are in the rest state except for a front of excited cells. The z variable has a circular gradient of excitation, which is smoothly decreasing. This initial condition is described in details in [Pivka, 1995].

We have compared the composition method to a Runge–Kutta adaptive time-step method and a simple Euler fixed-step method. We have chosen a fourth-order composition method with different time-steps ($\Delta t = 0.01$, $\Delta t = 0.02$, $\Delta t = 0.05$), a

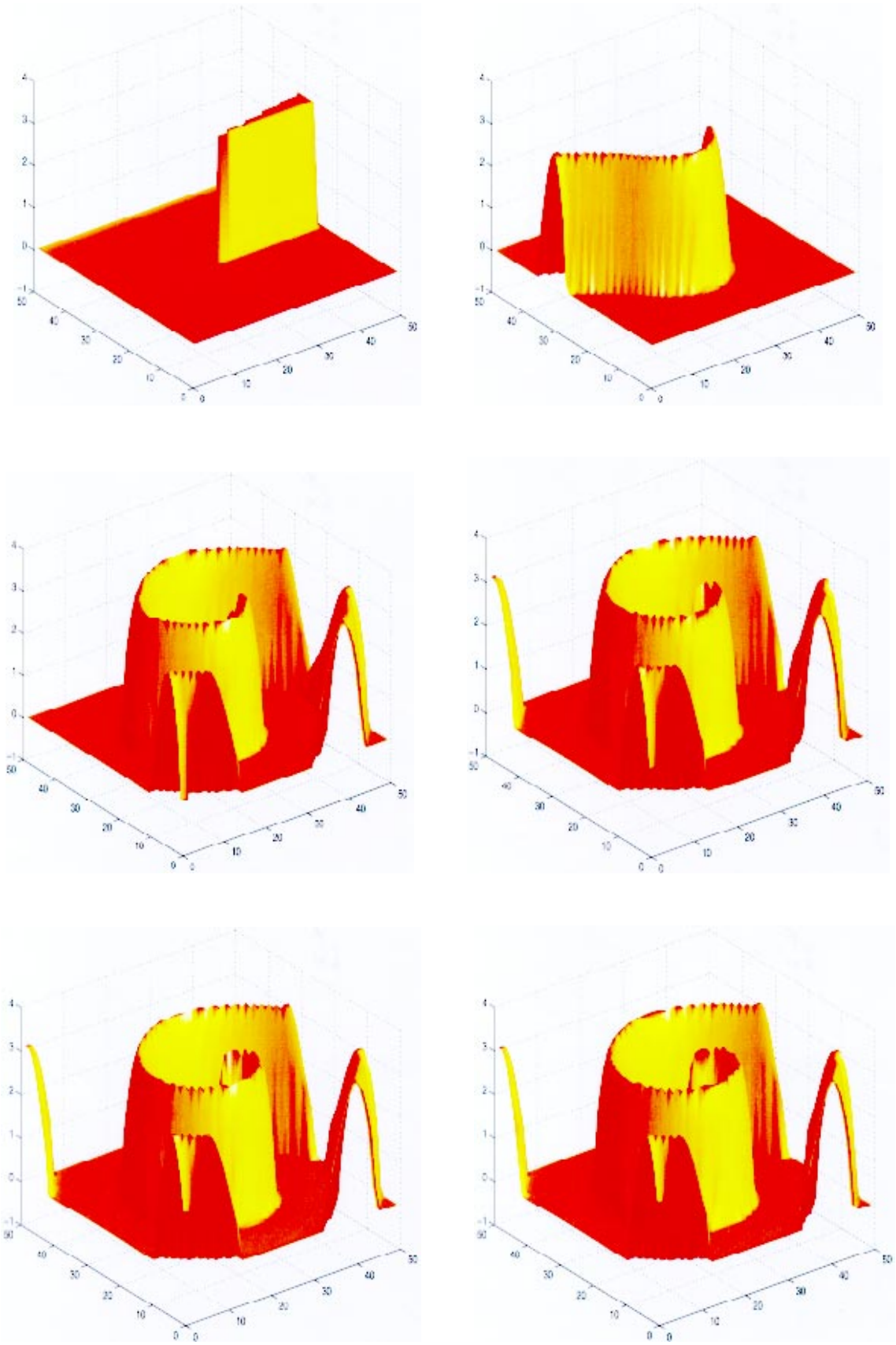

Fig. 3. Snapshot of the spiral wave for the Euler method ($t_0 = 0$, $t_1 = 20$, $t_2 = 40$, $t_3 = 60$, $t_4 = 80$, $t_5 = 100$). The picture represents the activity of the x variable on a 50×50 grid of Chua's circuits. We do not present the activity of the y and z variables.

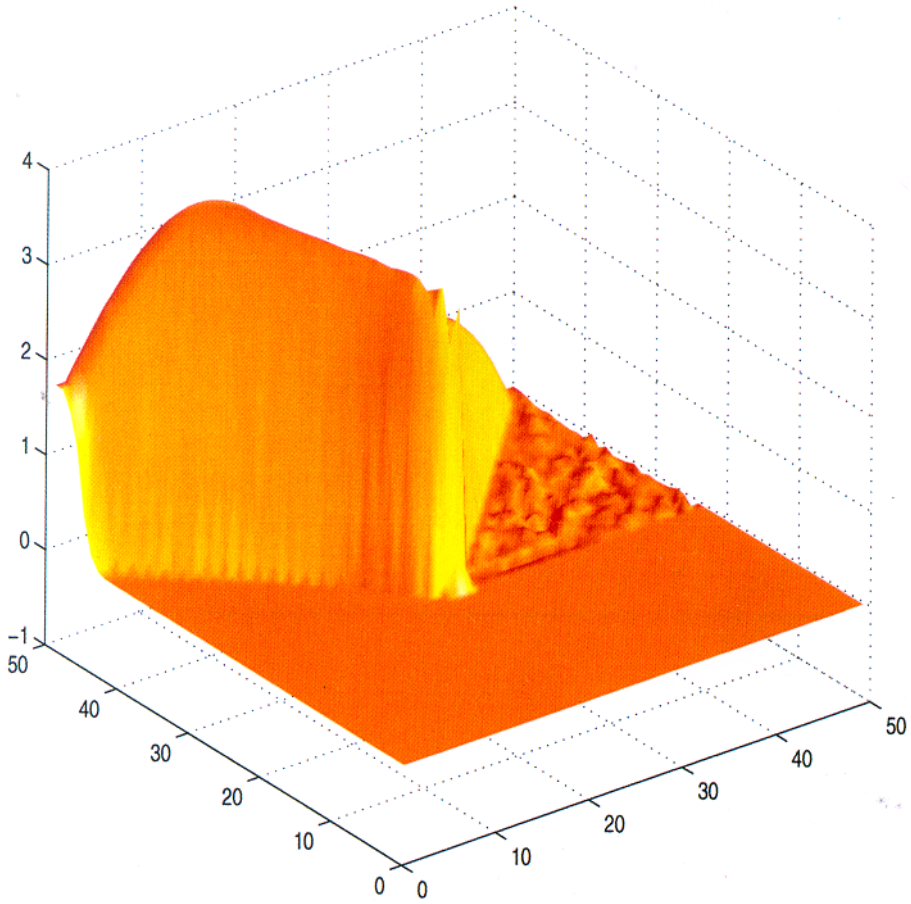

Fig. 4. Snapshot of the integration for the Euler method with $\Delta t = 0.005$ at time $t = 10$. With this step size, the method becomes unstable and cannot give accurate results.

first-order Euler method with different time-steps ($\Delta t = 0.001$, $\Delta t = 0.002$, $\Delta t = 0.005$), and a fourth-order Runge–Kutta method (the time-step is automatically adaptively chosen). In a previous experiment [Muñuzuri *et al.*, 1995], a fixed-step Euler method with time-step $\Delta t = 0.0005$ had been used.

We found that it was possible to increase the time-step of the Euler method to $\Delta t = 0.002$ while keeping sufficient accuracy. We found that the method was then about twice as fast as the adaptive Runge–Kutta method. We show in Fig. 3 snapshots of the activity of the x variable on a grid of 50×50 Chua's circuits at times $t_0 = 0$, $t_1 = 20$, $t_2 = 40$, $t_3 = 60$, $t_4 = 80$, $t_5 = 100$ for the Euler method. We should not forget that there are two other variables y and z for each Chua's circuit that also evolve over time.

If we try to increase the time-step of the Euler method, we see that instabilities appear for $\Delta t = 0.005$, which make the integration unreliable as we can see in Fig. 4 where we show snapshots of the activity of the x variable at $t = 10$.

In contrast, we can use a much larger time-step for the composition method because of its good conservation properties. This larger time-step comes at the price of a larger number of operations per iteration since the method contains a number of substeps. But as a whole, the fourth-order composition method with a time-step of $\Delta t = 0.01$ is already as fast as the Euler method with a time-step of $\Delta t = 0.001$ for the same accuracy. At $\Delta t = 0.02$ for the composition method, the integration still proceeds normally and the method is two times as fast as the Euler method. This simulation is displayed in Fig. 5. In the case of a simulation for a 50-by-50 array ($N = 50$) and a time span of $T = 100$ seconds, the simulation for the Euler method with $\Delta t = 0.002$ takes 290 seconds for a C code running on a Sun Ultra-1 workstation. In contrast, the simulation for the fourth-order composition method with $\Delta t = 0.02$ takes

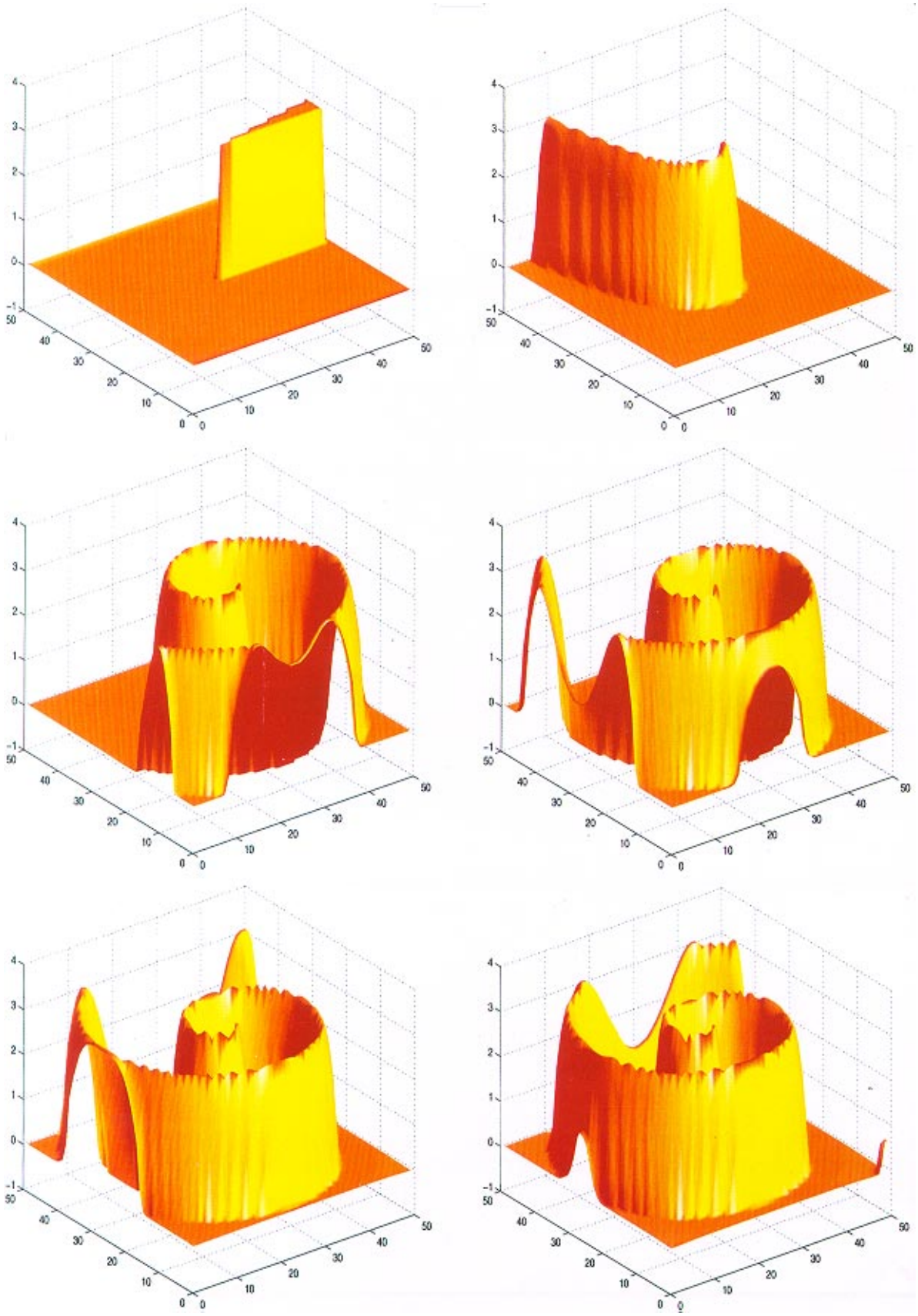

Fig. 5. Snapshot of the spiral wave for the composition method ($t_0 = 0$, $t_1 = 20$, $t_2 = 40$, $t_3 = 60$, $t_4 = 80$, $t_5 = 100$). The composition method uses a time step much larger than the Euler method.

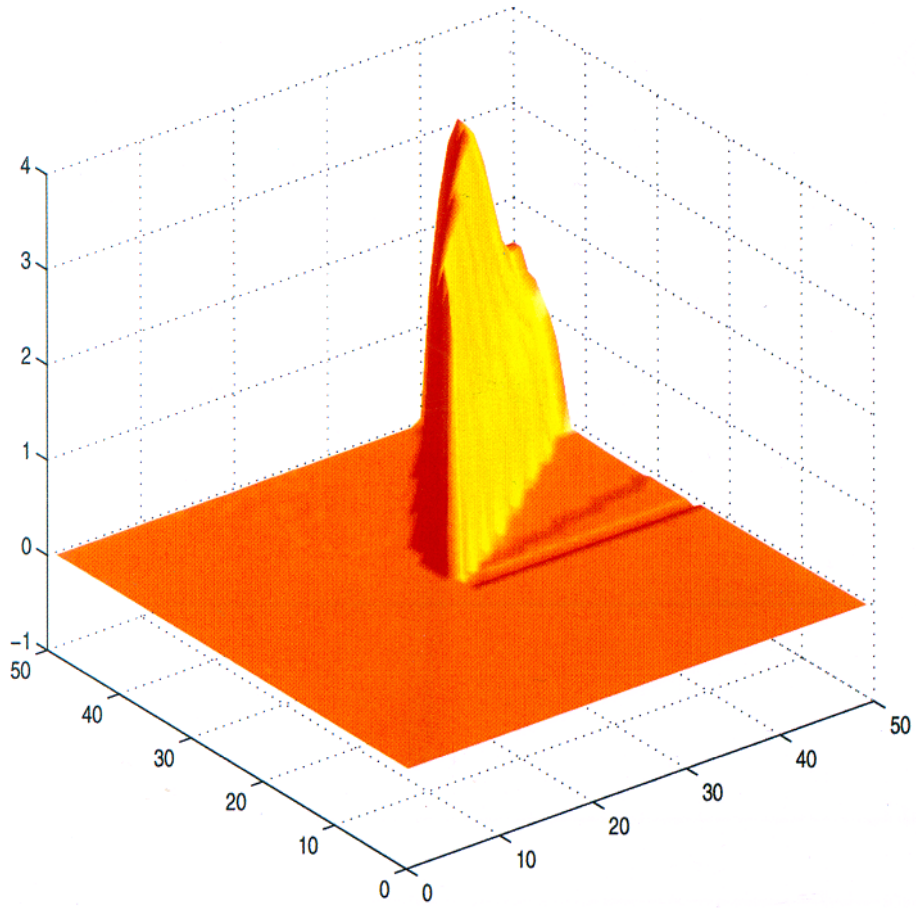

Fig. 6. Snapshot of the integration for the composition method with $\Delta t = 0.05$ at time $t = 10$. With this time step, the composition method becomes unstable and does not give accurate results.

141 seconds. We took care to write the codes for the two methods with similar parts written in a similar way to make the comparison fair.

At $\Delta t = 0.05$, instabilities appear and they are too severe to make the integration acceptable. We can see this behavior in Fig. 6.

Both methods are linear in $1/\Delta t$ so that taking a time-step twice as small leads to twice as long a computation; our experiments verified this ratio. In fact, we can estimate the complexity of both methods in terms of the number of multiplications and additions needed to perform the computation. The complexity is proportional to the time span T of the simulation and to the number of circuits (N^2) of the array. Per circuit and per time step, the complexity of the Euler method is 18 floating-point additions and 9 floating-point multiplications. In contrast, the fourth-order composition method is composed of 11 substeps. Six of these substeps account for the nonlinearity of the dynamics while five of the substeps account for the linear part of the dynamics. The complexity of the nonlinear substep is 5 floating-point additions and 1 floating-point multiplication. The complexity of the linear substep is 10 floating-point addition and 5 floating-point multiplications. Therefore, the complexity per circuit of a step of the composition method is $6 \times 5 + 5 \times 10 = 80$ floating-point additions and $6 \times 3 + 5 \times 5 = 43$ floating-point multiplications. This complexity is much higher than the complexity of the Euler method. However, we can use a time step for the composition method ($\Delta t_{\text{comp}} = 0.02$) that is ten times as big as the time step for the Euler method ($\Delta t_{\text{Euler}} = 0.002$). When we take this difference into account, we have that the total complexity of the composition method ($3N^2T/\Delta t_{\text{comp}}(80\ \text{FPadd} + 43\ \text{FPmult})$) is about twice as small as the total complexity of the Euler method ($3N^2T/(\Delta t_{\text{comp}}/10)(18\ \text{FPadd} + 9\ \text{FPmult})$).

Although, the two integration methods give results that are not numerically completely identical,

all the phenomena that have been described for the spiral wave in an array of Chua's circuits are preserved. The differences are mainly due to the truncation errors because we have tried to speed up both methods to a maximum. For smaller timesteps both methods again coincide.

6. Conclusions

We have presented a new class of integration rules for arrays of Chua's circuits: the composition methods. We derived these methods from Lie algebra theory through the use of the Baker–Campbell–Hausdorff formula. Not only do these methods shed new light on the dynamics of arrays of Chua's circuits, but they provide enhanced performances for the simulation of these arrays. The simplicity of our integration rules makes them good candidates for efficient software implementation. They could also be used to design the templates of cellular neural networks.

Acknowledgments

Joos Vandewalle is a full Professor at the K. U. Leuven. Yves Moreau is a research assistant of the F. W. O. Vlaanderen. This work is supported by several institutions:

1. The Flemish Government: Concerted Research Action GOA-MIPS (Model-based Information Processing Systems)
2. The Belgian State, Prime Minister's Office — Federal Office for Scientific, Technical and Cultural Affairs — Interuniversity Poles of Attraction Programme: (IUAP P4-02): Modeling, Identification, Simulation and Control of Complex Systems
3. The European Commission: ESPRIT project DICTAM (Dynamic Image Coding Using Tera-Speed Analogic Visual Microprocessors)
4. The FWO Research Communities: ICCoS (Identification and Control of Complex Systems) and Advanced Methods for Mathematical Modeling.

References

Arnold, V. [1989] *Mathematical Methods of Classical Mechanics* (Springer-Verlag, NY).

Bluman, G. & Kumei, S. [1989] *Symmetries and Differential Equations* (Springer-Verlag, NY).

Forest, E. & Ruth, R. [1990] "Fourth-order symplectic integration," *Physica* **D43**, 105–117.

Irwin, M. [1980] *Smooth Dynamical Systems* (Academic Press, NY).

Isidori, A. [1989] *Nonlinear Control Theory* (Springer-Verlag, NY).

McLachlan, R. I. [1995] "On the numerical integration of ordinary differential equations by symmetric composition methods," *SIAM J. Sci. Comput.* **16**(1), 151–168.

Moreau, Y. & Vandewalle, J. [1997] "System modeling using composition networks," in *Computer Intensive Methods in Control and Signal Processing*, eds. Warwick, K. & Karny, M., (Birkhauser, Boston) pp. 119–140.

Muñuzuri, A. P., Pérez-Muñuzuri, V., Gómez-Geistera, M., Chua, L. O. & Pérez-Villar, V. [1995] "Spatio-temporal structures in discretely-coupled arrays of nonlinear circuits: A review," *Int. J. Bifurcation and Chaos* **5**(1), 17–50.

Pivka, L. [1995] "Autowaves and spatio-temporal chaos in CNNs. Part I: A tutorial," *IEEE Trans. Circuits Syst. I: Fundamental Theor. Appl.* **42**(10), 638–649.

Suzuki, M. & Yamauchi, T. [1993] "Convergence of unitary and complex decompositions of exponential operators," *J. Math. Phys.* **34**(10), 4892–4897.

Yoshida, H. [1990] "Construction of higher order symplectic integrators," *Phys. Lett.* **A150**, 262–269.