Lossless Cellular Neural Networks

A. Schlaffer[1], *J. A. Nossek*[1], *M. Tanaka*[2]

1)Institute for Network Theory and Circuit Design, Technical University of Munich, Arcisstrasse 21, D-80333 Munich, Germany
Tel: +4989 289-28 501, Fax: +4989 289-28 504,
E-Mail: nossek@nws.e-technik.tu-muenchen.de

2)Department of Electrical Engineering, Sophia University, 7-1 Kioicho, Chiyoda-ku, Tokyo 102, Japan
Tel: +81-3-3238-3878, Fax: +81-3-3238-3321,
E-Mail: tanaka@mamoru.ee.sophia.ac.jp

ABSTRACT:*Since their introduction, Cellular Neural Networks [4] have turned out to be useful architectures for the solution of many problems, e. g. in image processing or in the simulation of partial differential equations. Therefore, there have been several attempts to introduce cell circuits suitable for large-scale integration [3]. Up to now, all of these cells need energy and therefore power supply.*
Just recently attempts have been made to build up circuitry being able to work without an external energy supply by using the energy stored in the initial state [1]. This principle can provide two major advantages. First, since no or at least not much energy is dissipated during computation, the circuit does not produce much heat. Therefore, there are no "hot spots" in integrated circuits, which limit integration density and operation speed. Furthermore, since there is no need for a power supply, the absence of voltage supply lines supports a high integration density.
In this work an architecture for the realisation of a lossless CNN is proposed. Further on, since standard learning algorithms turn out to fail for lossless systems, a way to amend these is introduced.

1 Mathematical outline and circuit model

1.1 Mathematical outline

As described in [2], lossless systems can be described by the following differential equation:

$$\dot{x} = A(x) grad_x H(x); \tag{1}$$

This differential equation with a coupling matrix A, which depends on the state vector, leaves the *Hamiltonian* H, an energy function of the state vector, unchanged, if A is skew-symmetric for all x. Normally the additional condition of Jacobi-identity has to be fulfilled, too, but for the kind of Hamiltonian proposed later on it does not need to be taken into account.

Since the Hamiltonian H is preserved, the trajectory of a system with a given initial energy H_0 will never leave a surface defined by

$$H(x) = H_0; \tag{2}$$

This equation defines an n-dimensional surface in the state space, which has the dimension $n+1$. If we introduce coordinates on this reduced surface, we can map the given lossless differential equation with the state vector $x \in R^{n+1}$ into a normal differential equation with the state vector $y \in R^n$. The mapping is carried out by using a (nonlinear) mapping function r:

$$y = r(x); \tag{3}$$

By using the inverse of this mapping *for a specific given energy* H, here denoted by r_H^{-1}, we can map the reduced state vector y back into the lossless system's state vector x:

$$x = r_H^{-1}(y); \tag{4}$$

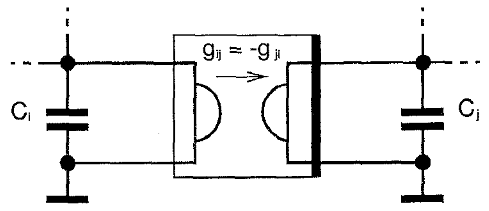

Figure 1: *Coupling of two states*

The given lossless differential equation (1) changes to the form:

$$\begin{aligned} \dot{y} &= f_H(y); \\ \dot{H} &= 0; \end{aligned} \tag{5}$$

By applying this procedure, *any lossless differential equation* can be turned into a normal differential equation with the order reduced by one. The other way round the same thing is possible: by using the inverse mapping *any differential equation* can be turned into a lossless differential equation with the grade augmented by one but with in principle the same behaviour.

1.2 Circuit model for lossless systems

To get a circuit model for the differential equation (1), we need to realize the coupling matrix $A(x)$ as well as the Hamiltonian $H(x)$.

The Hamiltonian H denotes the energy stored in the system. To enable the uncritical realization in a CNN of such a Hamiltonian, two demands have to be fulfilled: Since the energies of the states should be independent, the energy H of the system has to be the sum of the energies stored in the states. Further on, in a CNN all of the cell cores have to be equal; thus the dependence of the energy of a certain cell on its state has to be the same for all cells; thus: $H = h(x_1) + h(x_2) + \cdots$; To realize this postulated single-state Hamiltonian $h(x)$, we can, for example, use nonlinear capacitors with the characteristic $q = c(u)$.

If the Hamiltonian is of a construction of the given type, we can map the differential equation to one corresponding to linear capacitors. Therefore, we have to introduce new coordinates $\tilde{x}$ with the characteristic:

$$H = h(x_1) + h(x_2) + \cdots = \frac{1}{2}(\tilde{x}_1^2 + \tilde{x}_2^2 + \cdots) = \tilde{H}; \tag{6}$$

This new Hamiltonian $\tilde{H}$ corresponds to the energy stored in linear capacitors. Using this new state variables we can, under the condition that $h(x)$ is invertible, describe any Hamiltonian system with a single-state Hamiltonian by a lossless system with linear capacitors.

The main problem, therefore, will be to realize an adequate, state-dependent coupling matrix. The main criteria for the matrix A to define a lossless system is the skew-symmetry. If we use capacitors to realize the Hamiltonian as proposed before, such a coupling matrix can easily be realised by connecting the states by nonlinear gyrators, as shown in Fig. 1. The gyrator is described by the following conductance matrix:

$$i = \begin{bmatrix} 0 & g(u) \\ -g(u) & 0 \end{bmatrix} u; \tag{7}$$

with the transconductance g depending on the voltage u. This dependence can be of several types, e. g. a polynomial or periodic function on the states.

The parameters that can be changed during learning are the coefficients of these coupling functions.

A gyrator between the nodes i and j realises both couplings a_{ij} and a_{ji} at one time. A band structured matrix like that known from standard CNNs results, with the difference that its main diagonal is empty and it is skew-symmetric.

2 Learning algorithms for lossless CNN

2.1 Standard RBP

The idea of RBP (Recurrent Backpropagation) [5] is quite simple. First, an error function e is defined, denoting the distance between the actual equilibrium x_∞ and the desired equilibrium d:

$$e(x_\infty) := \frac{1}{2} \|x_\infty - d\|_2^2; \tag{8}$$

Then, after simulating the system to determine the actually reached equilibrium x_∞, we change the controllable parameters of the system (denoted by the *parameter vector* p) using a gradient descent method:

$$p[k+1] = p[k] - \mu \frac{\partial e}{\partial p}; \tag{9}$$

Here μ denotes the learning rate, which can be given as a constant or, in refined methods, be chosen advantageously in every learning step. The gradient $\frac{\partial e}{\partial p}$ has to be computed from the equilibrium point x_∞:

$$\frac{\partial e}{\partial p} = \frac{\partial e}{\partial x_\infty} \frac{\partial x_\infty}{\partial p}; \tag{10}$$

Using the error function definition (8), this comes to:

$$\frac{\partial e}{\partial p} = (x_\infty - d) \frac{\partial x_\infty}{\partial p}; \tag{11}$$

In further examination, only a single element α of the parameter vector p is considered. Of course, this does not limit the scope of the conclusions. The main problem will be to determine the gradient $\frac{\partial x_\infty}{\partial \alpha}$. To do so, we have to consider the equilibrium condition derived from the differential equation (1):

$$A(x_\infty)\, x_\infty = 0; \tag{12}$$

If we derive this for α, we get:

$$\frac{\partial A}{\partial \alpha} x_\infty + (B + A) \frac{\partial x_\infty}{\partial \alpha} = 0; \tag{13}$$

with the matrix B denoting

$$B = [A_{x_1} x_\infty, A_{x_2} x_\infty, \cdots]; \tag{14}$$

However, if equation (13) was solvable, it would demand the matrix $A + B$ to be invertible. This is not the case since all column vectors of A as well as of B are orthogonal to the equilibrium vector x_∞.

For A this can be derived from the equilibrium condition (12), which means that all row vectors of A are orthogonal to x_∞. Since A is skew symmetric, this implies that the column vectors, which are simply the negative row vectors, are also orthogonal to x_∞.

If we regard the column vectors of B, we can see that all of them are simply the product of a derivative A_{x_i} and x_∞. Since all of the derivatives are skew symmetric, all of these products are orthogonal to x_∞.

Since x_∞ is neither an element of $span(A)$ nor of $span(B)$, it is likewise not an element of $span(A+B)$. Therefore, the sum matrix $A + B$ degenerates and there exists no solution for equation (13) generally. If numerical algorithms are applied, they will operate very slowly and inaccurately near the equilibria, and thus the direct applying of RBP to lossless systems will fail. Since most other learning algorithms for supervised learning will in some step make use of the gradient $\frac{\partial x_\infty}{\partial \alpha}$, those will fail, too.

The reason for this failure is the fact that equilibria in lossless systems do not depend only on the parameters p of the system but also, of course, on the initial state. Since this initial state is never reflected in the equations we regarded, the gradient $\frac{\partial x_\infty}{\partial p}$ cannot be derived from these.

2.2 Modified RBP for lossless CNN

As we have seen, normal RBP fails due to an insufficient error definition. To overcome these difficulties, we have to use a different definition of the learning target. The approach proposed here is simply to work on the system after a reduction of order as outlined in section 1.1. By using a mapping as proposed in 1.1 we get the reduced system described by (5):

$$\dot{y} = f_H(y); \tag{15}$$

with the nonlinear function $f_H(y)$ derived from the initial system equations by using the reduction mapping $r(x)$ and its inverse for a certain energy $r_H^{-1}(y)$:

$$f_H(y) = \frac{\partial r}{\partial x} A\big(r_H^{-1}(y)\big)\, r_H^{-1}(y); \tag{16}$$

With a new error definition for RBP, which denotes the distance between the reached equilibrium and the desired equilibrium *in the reduced state variable y*

$$e_y(y_\infty) := \frac{1}{2}\,\|y_\infty - d_y\|_2^2; \tag{17}$$

and the new desired equilibrium

$$d_y = r(d); \tag{18}$$

we have a new optimisation problem with a normal system, which now is solvable. As was demonstrated for standard RBP in (11), the error gradient comes to:

$$\frac{\partial e_y}{\partial p} = (y_\infty - d_y)\,\frac{\partial y_\infty}{\partial p}; \tag{19}$$

Once again, we have to determine $\frac{\partial y_\infty}{\partial p}$.

In principle, from this point we could continue merely simulating and learning the reduced system. In the end, the parameters p found this way can be used directly in the lossless system. However, the explicit determination of $f_H(y)$ is very difficult for larger systems, though of course feasible in principle. To avoid this explicit determination, we can at first continue simulating the initial system to determine the actually reached equilibrium x_∞, which is numerically advantageous since the reduced system contains nonlinearities of a kind that require much time to compute, such as sine functions or fractions. Next we have to determine the gradient $\frac{\partial y_\infty}{\partial \alpha}$, which can be derived from the equilibrium condition

$$f_H(y_\infty) = 0; \tag{20}$$

by deriving for α:

$$\frac{\partial f_H}{\partial \alpha} + \frac{\partial f_H}{\partial y}\frac{\partial y}{\partial \alpha} = 0; \tag{21}$$

which leads to:

$$\frac{\partial y}{\partial \alpha} = -\frac{\partial f_H}{\partial \alpha}\left(\frac{\partial f_H}{\partial y}\right)^{-1}; \tag{22}$$

The two gradients $\frac{\partial f_H}{\partial \alpha}$ and $\frac{\partial f_H}{\partial y}$ can be derived from (16):

$$\frac{\partial f_H}{\partial \alpha} = \frac{\partial r}{\partial x}\frac{\partial A}{\partial \alpha} x_\infty; \tag{23}$$

$$\frac{\partial f_H}{\partial y} = \frac{\partial r}{\partial x}\left(\frac{\partial A}{\partial x}\frac{\partial r_H^{-1}}{\partial y} x_\infty + \frac{\partial r}{\partial x} A \frac{\partial r_H^{-1}}{\partial y}\right); \tag{24}$$

As a last step, we update the parameter vector using the gradient according to (9).

The modification carried out with RBP as shown above can be applied to any learning algorithm. Whenever the use of an error function is recommended, this error function should be formulated for the reduced system.

For example, for Backpropagation Through Time (BTT) [6], if the system is not near an equilibrium, the algorithm would work correctly without the reduction of order; however, if the system state approaches an equilibrium, the condition of the system matrix A becomes worse and worse. This leads to bad results and slow convergence if numerical algorithms are applied. Nevertheless, the order can be reduced, and as was shown for RBP, the problem of degeneration of the system matrix will not appear any more.

3 Example: Two-dimensional system

If we regard a two-dimensional, lossless system with linear capacitors, it is described by the following differential equation:

$$\dot{x} = \begin{bmatrix} 0 & g(x) \\ -g(x) & 0 \end{bmatrix} x; \tag{25}$$

We now apply a polar coordinate mapping, which for a two-dimensional system has the form

$$\varphi = r(x) = \arctan\left(\frac{x_1}{x_2}\right); \tag{26}$$

with the inverse

$$x = r_H^{-1}(\varphi) = \sqrt{H} \begin{matrix} \cos\varphi \\ \sin\varphi \end{matrix} \Bigg]; \tag{27}$$

By doing so, the reduced system takes the form

$$\dot{\varphi} = f_H(\varphi) = g(x) = g(\sqrt{H}\cos\varphi, \sqrt{H}\sin\varphi); \tag{28}$$

The new error function to be minimized is

$$e_\varphi = \frac{1}{2}\left\|\varphi_\infty - \arctan\left(\frac{d_1}{d_2}\right)\right\|_2^2; \tag{29}$$

The equation (22) to compute the gradient $\frac{\partial\varphi_\infty}{\partial\alpha}$ in this case takes the form:

$$\begin{aligned} \frac{\partial\varphi_\infty}{\partial\alpha} &= -\frac{\partial g}{\partial\alpha}\left(\frac{\partial g}{\partial x}\frac{\partial x}{\partial\varphi}\right)^{-1} \\ &= -\frac{g_\alpha}{\sqrt{H}(g_{x_2}\cos\varphi_\infty - g_{x_1}\sin\varphi_\infty)}; \end{aligned} \tag{30}$$

with g_α and g_{x_i} denoting $\frac{\partial g}{\partial\alpha}$ resp. $\frac{\partial g}{\partial x_i}$. Since all of these gradients can in most cases be computed, we can now solve the optimisation problem. The learning algorithm was applied to a two-dimensional system with polynomial dependence of the coupling factor on the states

$$g(x_1, x_2) = a + b\,x_1 + c\,x_2; \tag{31}$$

Thus, the parameters to be learned, in the proposed formalism denoted by α, are here the coefficients of the polynomial, i. e. a, b and c. In the example, the given problem was to reach the state $d = [1,1]^T$ starting from the initial state $x_0 = [1,-1]^T$. Fig. 2 shows how the position of the reached equilibrium changes during the learning process. Finally, after approximately 500 learning cycles, the error is small enough to fulfill the break-with-sucess condition.

To simulate the behaviour of the unreduced, lossless system an explicit Euler integration rule was used. First, the integration was carried out using this rule, gaining an intermediate update $\tilde{x}[k+1]$

$$\tilde{x}[k+1] = x[k] + \tau\, f(x[k]); \tag{32}$$

Since this integration rule normally increases the length of the system vector and thus is not suitable for lossless systems, it was amended by an additional step which restored the length of the system vector after each iteration step, note that the length of the system vector $\sqrt{H}$ is known from the initial state.

$$x[k+1] = \sqrt{H}\,\frac{\tilde{x}[k+1]}{\|\tilde{x}[k+1]\|_2^2}; \tag{33}$$

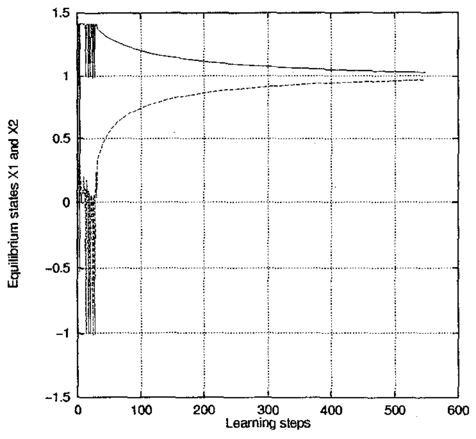

Figure 2: *Development of the equilibrium during the learning process*

So the main characteristic of losslessness of the system was not lost. The amended integration rule is fully explicit and therefore can be used advantageously for numerical simulations. It can be proved that this amended Euler integration rule will leave oscillating systems oscillating, which is an advantage for learning algorithms since the critical case of oscillation has to be taken into account.

References

[1] C. S. Lent, P. D. Tougaw, W. Porod, G. H. Bernstein: "Quantum cellular automata". *Nanotechnology 4* 1993.

[2] V. I. Arnol'd, S. P. Novikov (Eds.): "Integrable Systems". *Dynamical Systems IV*, Springer, 1980.

[3] J. L. Huertas, A. Rodriguez-Vazquez and S. Espejo: "Analog VLSI Implementation of Cellular Neural Networks". *CNNA '92 Proceedings*, pp.141-150, 1992.

[4] L. O. Chua, L. Yang: "Cellular neural networks: Theory". *IEEE Transactions on Circuits and Systems*, CAS-35, pp.1257-1272, Oct. 1988.

[5] F. J. Pineda: "Recurrent backpropagation and the dynamical approach to adaptive neural computation". *Neural Computation*, Vol.1, pp.161-172, 1989.

[6] A. J. Schuler, P. Nachbar and J. A. Nossek, L. O. Chua: "Learning State Space Trajectories in Cellular Neural Networks". *CNNA '92 Proceedings*, pp.68-73, 1992.