# Computation of Delay-Free Nonlinear Digital Filter Networks: Application to Chaotic Circuits and Intracellular Signal Transduction

Federico Fontana, *Member, IEEE*, and Federico Avanzini

***Abstract*—A method for the computation of nonlinear digital filter networks containing delay-free loops is proposed. By preserving the topology of the network this method permits the inspection of all signals flowing across the loops. Furthermore, it enables to discretize analog filter networks described by nonlinear ordinary differential equation systems on a block-by-block basis, in such a way that *ad hoc* analog-to-digital maps can be individually applied to each filter. A nonlinear implicit system must be solved at every computation step using iterative methods, holding certain sufficient conditions for the existence of the solution. This condition is in algebraic relationship with the causality and structure of the network and its filtering blocks. The proposed method can be straightforwardly applied to the computation of several interconnected networks. This property adds modularity to the procedure, and enables to handle changes in the network topology. Examples are presented showing the applicability of the method to the computation of the Chua–Felderhoff *RLC* circuit and to the dynamic simulation of a known intracellular signal transduction model of circadian cycles in *Drosophila melanogaster*.**

***Index Terms*—Biological system modeling, circuit modeling, delay effects, digital filters, nonlinear systems.**

## I. Introduction

A continuous dynamic system which is mathematically described by a set of autonomous ordinary differential equations (ODEs) can be modeled as a network of smaller system blocks that exchange signals instantaneously. The networked perspective becomes attractive when it reflects the physical structure of the system and helps preserve its local properties. Examples of system networks are found in application fields ranging from traditional electronic engineering to the emerging field of computational biology, which often deals with networks that connect systems admitting an ODE-based description in terms of rate equations [1], [2].

In general, it is not true that a system network can be inspected using efficient numerical integrations, which decouple the computations across blocks: if the responses of two or more blocks depend on each other without temporal delay, then it is usually necessary to "lump" their computation together into a new system block. Conversely, every temporal delay occurring during the communication between causal blocks can be exploited to isolate parts of the integration procedure, whose inputs depend only on past signal values. These parts can be integrated autonomously and in parallel with the rest of the scheme because their outputs do not propagate back to the respective inputs instantaneously. Using words taken from the digital signal processing jargon, if an interconnection of blocks does not give rise to a *delay-free loop*, then causality is preserved in the corresponding computation.

We deal with systems that are conveniently represented in terms of network models, whose mathematical descriptions cannot preserve the modularity expressed by the network due to the presence of instantaneous loopbacks among blocks. In particular, we will devote our attention to networks whose blocks are represented by *filters*, each described by its own (either linear or nonlinear) transfer characteristic.

It is known that filter networks unveil a nontrivial computational problem when their topology includes loops along which signals propagate in negligible time. Graph topology-based methods exist which detect such loops [3]. In the linear case, the filters forming the delay-free loop can be gathered together into an equivalent, higher order filter [4]. In the nonlinear case the design of an exact (or approximated) lumped equivalent solving the noncomputability is less obvious.

Recently, a numerical method for the discrete-time computation of delay-free loop network topologies has been proposed limited to the case where all filters are linear: the particularity of this technique is the computation of every filter block even in presence of the delay-free loop [5]. Since the method works in discrete time, it assumes that every filter is converted into the digital domain prior to the application of the method itself. Due to the preservation of the filter blocks in the delay-free loop, ad hoc analog-to-digital transformations can be chosen depending on the complexity and particular properties of every single block [6], [7]. In this paper, we extend the aforementioned technique to networks containing multi-input/single-output nonlinear filters. We propose a method for the computation of these networks that is alternative to previously existing solutions working either in the Kirchhoff or wave domain [8]–[10]. The method works in the Kirchhoff domain and asks to solve a nonlinear implicit system at every discrete computation step. We provide an algorithm for the detection of delay-free loops in the network, and analyze the relationship existing between computability and causality. Two different approaches for the solution of the nonlinear implicit system are proposed. Moreover, it is shown how

Manuscript received July 18, 2007; revised April 29, 2008. First published July 9, 2008; current version published September 17, 2008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Patrice Abry.

F. Fontana is with the Dipartimento di Informatica, University of Verona, Verona 37134, Italy (e-mail: federico.fontana@univr.it).

F. Avanzini is with the Dipartimento di Ingegneria dell'Informazione, University of Padova, Padova 35131, Italy (e-mail: federico.avanzini@unipd.it).

Digital Object Identifier 10.1109/TSP.2008.928090

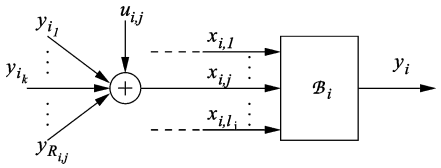


Fig. 1. Generic network block $\mathcal{B}_i$.

the technique can be scaled to the case of two or more networks that are connected to each other. Finally, two application examples are proposed: the first one deals with the computation in the discrete time of the Chua–Felderhoff *RLC* circuit [11], [12]; the second one shows the applicability of the method to a model of circadian cycles, whose oscillation emerges out of an intracellular signal transduction network for which a digital filter-based computation approach has not been attempted previously [13], [14].

This research initially stemmed from some specific problems encountered in the field of physically based sound synthesis, concerning the excitation of mechanical and fluid-dynamic systems [15]–[17]. By means of an earlier and simplified version of the proposed method, capable of handling single-input/single-output blocks, we have modeled the acoustics of a mechanical model of two nonlinear contacting resonators [18], and computed a digital version of the "Dolby B" audio tape noise reduction system [19].

The remainder of the paper is organized as follows. After introducing the needed notations and algebra in Sections II, we explore in Section III how the method can be used to detect delay-free loop networks, and how it applies to their computation. Conditions for the modularization of the computations are given in Section IV. Finally, the two application examples are discussed in Section V.

## II. Network Blocks and Topology

In this paper, we consider a digital network $\mathcal{N}$ composed of $m_N$ nonlinear and $m_L$ linear discrete-time multiple-input single-output (MISO) filters $\mathcal{B}_i$ $(i = 1,\dots,m_N + m_L)$. The generic $i$th filter has $l_i$ inputs $x_{i,1},\dots,x_{i,l_i}$, grouped into the column vector $\boldsymbol{x}_i$, and one output $y_i$ (see Fig. 1).

The formalism introduced in this section provides a compact matrix description of the network $\mathcal{N}$, which allows to detect memoryless interactions between blocks, and the consequent generation of delay-free computational loops between the output $y_i$ and the input $\boldsymbol{x}_i$ of the generic block $\mathcal{B}_i$.

### A. Blocks

For convenience the nonlinear and linear blocks are numbered from 1 to $m_N$ and from $m_N+1$ to $m_N+m_L$, respectively. The transfer characteristics are all causal. At the generic computation step $n$ they are written as follows [18]:

- nonlinear blocks

$$y_i[n] = f_i\big(\boldsymbol{x}_i[n], \boldsymbol{p}_i[n]\big), \quad i = 1,\dots,m_N \tag{1}$$

- linear blocks

$$y_i[n] = \boldsymbol{b}_i \boldsymbol{x}_i[n] + q_i[n], \quad i = m_N+1,\dots,m_N+m_L \tag{2}$$

in which $\boldsymbol{b}_i = |b_{i,1}\dots b_{i,l_i}|$.

In both (1) and (2), we have highlighted two contributions to the output $y_i$ at time step $n$: an *external* contribution due to the input $\boldsymbol{x}_i$, and an *internal* contribution due to the filter state. In (1) this separation is formal, and the internal state is represented by an argument $\boldsymbol{p}_i$ of the nonlinear function $f_i$ that does not depend on the input $\boldsymbol{x}_i$, and ensures that for every pair of steps $(n_1, n_2)$ such that $\boldsymbol{p}_i[n_1] = \boldsymbol{p}_i[n_2]$ if $\boldsymbol{x}_i[n_1] = \boldsymbol{x}_i[n_2]$ then $y_i[n_1] = y_i[n_2]$ (i.e., if the input is the same at both steps then so is the output). If $\boldsymbol{p}_i \equiv \boldsymbol{0}$ then $f_i$ reduces to an *algebraic* nonlinear element, otherwise $f_i$ is a *dynamic* nonlinear element [11].

In (2) this separation is explicit, since the output of a linear system can always be written as a superposition of the free and forced evolution. Consider a scalar $(l_i = 1)$ digital filter with $M$ zeros and $N$ poles, represented by the following transfer function in the Zeta-transformed domain [6]:

$$\frac{Y_i(z)}{X_i(z)} = \frac{b_i + \sum_{k=1}^{M} b_i^{(k)} z^{-k}}{1 - \sum_{k=1}^{N} a_i^{(k)} z^{-k}}. \tag{3}$$

Then $y_i[n]$ can be immediately expressed in the form (2), with

$$q_i[n] = \sum_{k=1}^{M} b_i^{(k)} x_i[n-k] + \sum_{k=1}^{N} a_i^{(k)} y_i[n-k]. \tag{4}$$

This result is straightforwardly generalized to the multiple input case $(l_i > 1)$, with

$$q_i[n] = \sum_{k=1}^{M} \boldsymbol{b}_i^{(k)} \boldsymbol{x}_i[n-k] + \sum_{k=1}^{N} a_i^{(k)} y_i[n-k]. \tag{5}$$

At every step $q_i[n]$ can be computed by disconnecting the $i$th filter input from the network or, equivalently, by feeding it with the input $\boldsymbol{x}_i[n] = \boldsymbol{0}$ [20]. The invariance of the response seen in the nonlinear case clearly holds also for the linear blocks. In this case $q_i[n_1] = q_i[n_2]$ implies $y_i[n_1] = y_i[n_2]$ when $\boldsymbol{x}_i[n_1] = \boldsymbol{x}_i[n_2]$.

Equations (1) and (2) can be rewritten in matrix form—we denote transposition by using the symbol $^T$:

$$\boldsymbol{y}_N[n] = \boldsymbol{f}(\boldsymbol{x}_N[n], \boldsymbol{p}[n]) \tag{6}$$

$$\boldsymbol{y}_L[n] = \boldsymbol{B}\boldsymbol{x}_L[n] + \boldsymbol{q}[n] \tag{7}$$

with

$$\boldsymbol{y}_N = |y_1 \dots y_{m_N}|^T, \quad \boldsymbol{y}_L = |y_{m_N+1} \dots y_{m_N+m_L}|^T,$$
$$\boldsymbol{x}_N = |\boldsymbol{x}_1^T \dots \boldsymbol{x}_{m_N}^T|^T, \quad \boldsymbol{x}_L = |\boldsymbol{x}_{m_N+1}^T \dots \boldsymbol{x}_{m_N+m_L}^T|^T,$$
$$\boldsymbol{p} = |\boldsymbol{p}_1^T \dots \boldsymbol{p}_{m_N}^T|^T, \quad \boldsymbol{q} = |q_{m_N+1} \dots q_{m_N+m_L}|^T,$$

and

$$\boldsymbol{f}(\boldsymbol{x}_N,\boldsymbol{p}) = \left|f_1(\boldsymbol{x}_1,\boldsymbol{p}_1)\ldots f_{m_N}(\boldsymbol{x}_{m_N},\boldsymbol{p}_{m_N})\right|^T,$$

$$\boldsymbol{B} = \begin{vmatrix} \boldsymbol{b}_{m_N+1} & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{b}_{m_N+2} & \ldots & \boldsymbol{0} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{0} & \ldots & \boldsymbol{0} & \boldsymbol{b}_{m_N+m_L} \end{vmatrix}.$$

### B. Topology

A digital nonlinear network $\mathcal{N}$ is formed by connecting the $m_N + m_L$ blocks $\mathcal{B}_i$ described in the previous section. In what follows the topology of the network is defined by a set of equations that specify how the outputs $y_i$ are connected to the inputs $\boldsymbol{x}_i$ of the network blocks.

The blocks $\mathcal{B}_i$ are connected to form the network $\mathcal{N}$ in such a way that the $j$th component $x_{i,j}$ of the $i$th input $\boldsymbol{x}_i$ is a linear combination of $R_{i,j}$ filter outputs, plus an external (possibly null) scalar signal $u_{i,j}$. For the generic $i$th block, this connectivity structure is illustrated in Fig. 1 and can be summarized as

$$x_{i,j}[n] = \sum_{k=1}^{R_{i,j}} c_{ik}\, y_{i_k}[n] + u_{i,j}[n], \quad j = 1,\ldots,l_i \qquad i = 1,\ldots,m_N + m_L \quad (8)$$

Equation (8) can be rewritten in matrix form

$$\boldsymbol{x}[n] = \boldsymbol{C}\boldsymbol{y}[n] + \boldsymbol{u}[n] \quad (9)$$

in which

$$\boldsymbol{x} = \begin{vmatrix} \boldsymbol{x}_N \\ \boldsymbol{x}_L \end{vmatrix}, \quad \boldsymbol{y} = \begin{vmatrix} \boldsymbol{y}_N \\ \boldsymbol{y}_L \end{vmatrix}, \quad \boldsymbol{u} = \begin{vmatrix} \boldsymbol{u}_N \\ \boldsymbol{u}_L \end{vmatrix},$$

$$\boldsymbol{u}_N = \left|\boldsymbol{u}_1^T \ldots \boldsymbol{u}_{m_N}^T\right|^T, \quad \boldsymbol{u}_L = \left|\boldsymbol{u}_{m_N+1}^T \ldots \boldsymbol{u}_{m_N+m_L}^T\right|^T$$

and where $\boldsymbol{u}_i = |u_{i,1}\ldots u_{i,l_i}|^T$.

The matrix $\boldsymbol{C}$ completely specifies the topology of the network and has the following meaning: a nonzero element in the $i$th column of $\boldsymbol{C}$ indicates that the output of the $i$th block is connected to the input of some other block of the network. More precisely, $\boldsymbol{C}$ contains a nonzero element at position $j+m_{C,k-1}$, $i$ if and only if the output $y_i$ is connected to $x_{k,j}$, where the integer $m_{C,k}$ indicates the total number of scalar inputs up to the $k$th filter: $m_{C,k} = \sum_{i=1}^{k} l_i$. If we define the total number of scalar inputs as $m_C := m_{C,m_N+m_L}$, then the size of $\boldsymbol{C}$ is $m_C \times (m_N + m_L)$.

## III. Computation of the Network

### A. Detection of Delay-Free Loops

The formalism introduced in Section II applies to a network $\mathcal{N}$ having any sort of topology, including the case in which there are no delay-free loops. In general it is computationally convenient to identify the components in $\mathcal{N}$ that need the solution of delay-free loops, then to apply the method individually to these components while leaving the computation of the rest of the network to a sequential procedure.

This section describes a graph theoretic method that subdivides the network into a set of disjoint subnetworks and establishes an order of precedence in which subnetworks can be computed either sequentially or in parallel. As a byproduct, the proposed method detects delay-free loops in the network.

The method works in three main steps. First, a signal flow graph $\mathcal{G}$ is constructed based on the blocks and topology of the network $\mathcal{N}$. Second, the strongly connected components of the graph $\mathcal{G}$ are identified.[1] Third, an algorithm adapted from the graph-based framework of Crochiere and Oppenheim [21] is used for defining the order of precedence in the computation of the subnetworks. Each of these steps is detailed in the remainder of the section.

A directed graph $\mathcal{G}$ associated to $\mathcal{N}$ is constructed as follows: $\mathcal{G}$ has $m_N + m_L$ vertices that encode corresponding network blocks; an edge is drawn from vertex $p$ to vertex $q$ if $y_p$ has a delay-free effect on $y_q$, i.e., $\boldsymbol{C}_{r+m_{C,q-1},p} \neq 0$ and $b_{q,r} \neq 0$ for some $r$.

Strongly connected components of the directed graph $\mathcal{G}$ can be found using standard depth-first-search algorithms available in the literature [22]. Note that $\mathcal{G}$ has a nontrivial strongly connected component (i.e., one with two or more nodes) if and only if the network $\mathcal{N}$ contains a delay-free computational loop between the blocks of that component.

Strongly connected components are then labeled using the following algorithm:

1. Set $i = 1$.
2. **while** $\mathcal{G}$ is not empty **do**
3. Find $J_i$ components that do not have incoming edges from other components and label them as $\mathcal{G}_{i,j}$ $(j = 1,\ldots,J_i)$
4. Remove from $\mathcal{G}$ all the vertices that belong to $\mathcal{G}_{i,j}, \forall\, j$, together with their outgoing edges
5. Increment $i$
6. **end while**

A network component $\mathcal{N}_{i,j}$ is formed by the blocks and branches of $\mathcal{N}$ that were encoded respectively by the vertices and edges of $\mathcal{G}_{i,j}$. The labeling defined by the above algorithm provides the order of precedence in the computation of network components: by definition the components $\mathcal{N}_{1,j}$ can be computed directly from the inputs $\boldsymbol{u}$ and their internal state, since their inputs do not depend on the state of other subnetworks; the computation of $\mathcal{N}_{i,j}$, $i > 1$, can be carried out once the outputs from $\mathcal{N}_{i-1,j}$ are known. Note also that for a given $i$, the networks $\mathcal{N}_{i,j}$, $j = 1,\ldots,J_i$, can be computed in parallel.

Fig. 2 shows an example of a signal flow graph $\mathcal{G}$ associated to a network $\mathcal{N}$. Assuming that all the blocks in the network are single-input single-output systems (i.e., $l_i = 1\,\forall\, i$), the matrix $\boldsymbol{C}$ that specifies the network topology is in this case the transposed

[1] Recall that a strongly connected component of a directed graph is a maximal set of vertices that are all *reachable* from each other through a path of directed edges of $\mathcal{G}$.

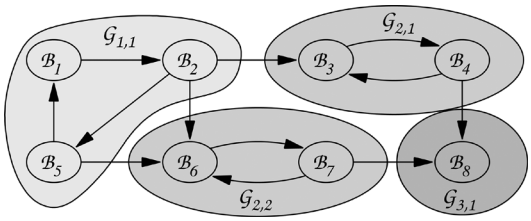


Fig. 2. Example of graph $\mathcal{G}$ associated to a network, and labeling of its four strongly connected components. The network component $\mathcal{N}_{1,1}$ is computed first, network components $\mathcal{N}_{2,1}$ and $\mathcal{N}_{2,2}$ are then computed in parallel, and the network component $\mathcal{N}_{3,1}$ is computed last.

of the adjacency matrix [22] of the graph $\mathcal{G}$.[2] All the components of this network, except for $\mathcal{N}_{3,1}$, contain delay-free loops.

Given the analysis presented in this section, a generic network $\mathcal{N}$ described through (6), (7), and (9) can be computed if its nontrivial (i.e., with two or more blocks) subnetworks $\mathcal{N}_{i,j}$ can be computed. Therefore, without loss of generality we restrict our attention to the computation of networks $\mathcal{N}$ whose signal flow graph $\mathcal{G}$ is strongly connected.

### B. Algebraic Rearrangement

Since $\boldsymbol{C}$ connects nonlinear and linear output to nonlinear and linear input blocks, it can be conveniently described as the juxtaposition of four submatrices, respectively, accounting for nonlinear-to-nonlinear ($\boldsymbol{C}_{\mathrm{NN}}$), linear-to-nonlinear ($\boldsymbol{C}_{\mathrm{NL}}$), nonlinear-to-linear ($\boldsymbol{C}_{\mathrm{LN}}$), and linear-to-linear ($\boldsymbol{C}_{\mathrm{LL}}$) connections [18]:

$$\begin{vmatrix} \boldsymbol{x}_N \\ \boldsymbol{x}_L \end{vmatrix} = \begin{vmatrix} \boldsymbol{C}_{\mathrm{NN}} & \boldsymbol{C}_{\mathrm{NL}} \\ \boldsymbol{C}_{\mathrm{LN}} & \boldsymbol{C}_{\mathrm{LL}} \end{vmatrix} \begin{vmatrix} \boldsymbol{y}_N \\ \boldsymbol{y}_L \end{vmatrix} + \begin{vmatrix} \boldsymbol{u}_N \\ \boldsymbol{u}_L \end{vmatrix}. \quad (10)$$

Using (6) and (7), $\boldsymbol{y}_N$ and $\boldsymbol{y}_L$ can be eliminated in (10) thus obtaining the two following equations:

$$\boldsymbol{x}_N = \boldsymbol{C}_{\mathrm{NN}}\boldsymbol{f}(\boldsymbol{x}_N, \boldsymbol{p}) + \boldsymbol{C}_{\mathrm{NL}}(\boldsymbol{B}\boldsymbol{x}_L + \boldsymbol{q}) + \boldsymbol{u}_N \quad (11)$$

$$\boldsymbol{x}_L = \boldsymbol{C}_{\mathrm{LN}}\boldsymbol{f}(\boldsymbol{x}_N, \boldsymbol{p}) + \boldsymbol{C}_{\mathrm{LL}}(\boldsymbol{B}\boldsymbol{x}_L + \boldsymbol{q}) + \boldsymbol{u}_L. \quad (12)$$

By moving $\boldsymbol{x}_L$ to the left-hand side of (12), one can write

$$(\boldsymbol{I} - \boldsymbol{C}_{\mathrm{LL}}\boldsymbol{B})\boldsymbol{x}_L = \boldsymbol{C}_{\mathrm{LN}}\boldsymbol{f}(\boldsymbol{x}_N, \boldsymbol{p}) + \boldsymbol{C}_{\mathrm{LL}}\boldsymbol{q} + \boldsymbol{u}_L \quad (13)$$

where $\boldsymbol{I}$ is the square identity matrix sized $m_C - m_{C,m_N}$. If the matrix $\boldsymbol{I} - \boldsymbol{C}_{\mathrm{LL}}\boldsymbol{B} \triangleq \boldsymbol{F}$ is invertible, then $\boldsymbol{x}_L$ can be isolated in (13):

$$\boldsymbol{x}_L = \boldsymbol{F}^{-1}\boldsymbol{C}_{\mathrm{LN}}\boldsymbol{f}(\boldsymbol{x}_N, \boldsymbol{p}) + \boldsymbol{F}^{-1}(\boldsymbol{C}_{\mathrm{LL}}\boldsymbol{q} + \boldsymbol{u}_L). \quad (14)$$

Finally, the right-hand side of (14) can be used to eliminate $\boldsymbol{x}_L$ in (11), in such a way that $\boldsymbol{x}_N$ remains the only unknown. In fact, after some algebraic manipulation (11) can be rewritten as

$$\boldsymbol{x}_N = \boldsymbol{W}_1\boldsymbol{f}(\boldsymbol{x}_N, \boldsymbol{p}) + \boldsymbol{W}_2\boldsymbol{q} + \boldsymbol{W}_3\boldsymbol{u}_L + \boldsymbol{u}_N \quad (15)$$

[2]Recall that the adjacency matrix of a graph has a nonzero element at the position $p$, $q$ if there exists an edge from vertex $p$ to vertex $q$. In the case of our matrix $\boldsymbol{C}$, the element $p$, $q$ is nonzero if there is an edge from $q$ to $p$, i.e., $\boldsymbol{C}$ is the transposed of the adjacency matrix.

with

$$\boldsymbol{W}_3 = \boldsymbol{C}_{\mathrm{NL}}\boldsymbol{B}\boldsymbol{F}^{-1}$$
$$\boldsymbol{W}_1 = \boldsymbol{W}_3\boldsymbol{C}_{\mathrm{LN}} + \boldsymbol{C}_{\mathrm{NN}}$$
$$\boldsymbol{W}_2 = \boldsymbol{W}_3\boldsymbol{C}_{\mathrm{LL}} + \boldsymbol{C}_{\mathrm{NL}}.$$

### C. Causality and Computability

The causality of a filter network $\mathcal{N}$ is a necessary condition for its computability. Conversely, the same condition is not necessary for the existence of solutions. An example can be provided of a simple linear delay-free loop network made by interconnecting *causal* blocks which provides *noncausal* signals: for this reason, it cannot be computed [5]. We want to stress here that the causality of all blocks $\mathcal{B}_i$, $i = 1, \ldots, m_N + m_L$, does not imply the causality of the resulting network $\mathcal{N}$.

More specifically, we may wonder if a causal delay-free loop network is also computable, or, conversely, if causal delay-free loop networks exist which cannot be computed using the algebraic rearrangement given in Section III-B. In the linear case, the answer to this question is that causality is equivalent to computability. In fact, it can be demonstrated that if the linear part of the network, i.e., the subnetwork $\mathcal{N}_L$ obtained by removing all nonlinear blocks along with their input and output branches, is causal, then $\boldsymbol{F}^{-1}$ exists [5]. Key passage in that demonstration is to rearrange the equations in the unknown variable $\boldsymbol{y}_L$. In our case, this rearrangement leads to the following equation:

$$(\boldsymbol{I} - \boldsymbol{B}\boldsymbol{C}_{\mathrm{LL}})\boldsymbol{y}_L[n] = \boldsymbol{B}\boldsymbol{u}_L[n] + \boldsymbol{q}[n] + \boldsymbol{C}_{\mathrm{LN}}\boldsymbol{y}_N[n]. \quad (16)$$

It can be proved that the invertibility of the matrix $\boldsymbol{I} - \boldsymbol{B}\boldsymbol{C}_{\mathrm{LL}}$ is a necessary and sufficient condition for the causality of $\mathcal{N}_L$. It is easily verified [5] that this matrix and the matrix $\boldsymbol{F}$ defined in Section III-B have the same determinant, $\det(\boldsymbol{F})$. Therefore, this value completely characterizes the causality as well as computability of $\mathcal{N}_L$. Note that this result says nothing about the stability of the network.

The above rearrangement can be extended to the nonlinear case. By substituting (15) in (6), we obtain

$$\boldsymbol{y}_N[n] = \boldsymbol{f}(\boldsymbol{W}_1\boldsymbol{y}_N[n] + \tilde{\boldsymbol{x}}_N[n], \boldsymbol{p}) \quad (17)$$

in which $\tilde{\boldsymbol{x}}_N[n] = \boldsymbol{W}_2\boldsymbol{q}[n] + \boldsymbol{W}_3\boldsymbol{u}_L[n] + \boldsymbol{u}_N[n]$ collects the contributions of the internal state components $\boldsymbol{q}[n]$ and the external inputs. The only unknown in (17) is $\boldsymbol{y}_N[n]$.

Analogously to the linear case, a sufficient condition can be given which ensures that the output $\boldsymbol{y}_N[n]$ exists and is uniquely determined by the input $\boldsymbol{u}[n]$ and the internal states $\boldsymbol{p}[n]$ and $\boldsymbol{q}[n]$. In fact, the implicit function theorem states that given the function

$$\boldsymbol{g}(\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N) \triangleq \boldsymbol{y}_N - \boldsymbol{f}(\boldsymbol{W}_1\boldsymbol{y}_N + \tilde{\boldsymbol{x}}_N, \boldsymbol{p}) \quad (18)$$

if there is a point $(\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N)$ for which $g(\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N) = 0$ and the Jacobian $[\partial \boldsymbol{g}/\partial \boldsymbol{y}_N]$ is not singular, then a function $\boldsymbol{y}_N(\tilde{\boldsymbol{x}}_N)$ is defined implicitly in an open neighborhood of that point, which

satisfies $\boldsymbol{g}\big(\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N(\tilde{\boldsymbol{x}}_N)\big) = 0$ for any $\big(\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N\big)$ in that neighborhood. In our case the Jacobian can be written from (18) as

$$\left[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}_N}\right](\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N) \triangleq \boldsymbol{I} - \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_N}\right](\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N) \cdot \boldsymbol{W}_1. \tag{19}$$

The structural similarity between the matrix $\boldsymbol{I} - \boldsymbol{B}\boldsymbol{C}_{\mathrm{LL}}$, introduced in (16) to isolate the output in $\mathcal{N}_L$, and the Jacobian (19), which proves the existence the output $\boldsymbol{y}_N$ in the nonlinear part of the network, is evident. Notice, however, that $\boldsymbol{W}_1$ integrates structural knowledge of the overall network. Interestingly, $\boldsymbol{W}_1 = \boldsymbol{C}_{\mathrm{NN}}$ in the particular case of a purely nonlinear network and, hence, there is substantial formal analogy between (19) and its linear counterpart, i.e., $\boldsymbol{I} - \boldsymbol{B}\boldsymbol{C}_{\mathrm{LL}}$. In general $\boldsymbol{W}_1$ inherits the structure of $\boldsymbol{W}_3$, which is typically a full matrix.

If the Jacobian (19) is globally nonsingular, then the output $\boldsymbol{y}_N$ can always be uniquely determined from $\tilde{\boldsymbol{x}}_N$. If the network is globally computable, then it is also unconditionally causal. As opposed to the linear case, here we are not able to infer the opposite, i.e., that causality implies computability. In conclusion, the nonsingularity of $\boldsymbol{F}$ completely characterizes causality as well as computability in $\mathcal{N}_L$, while the nonsingularity of $[\partial \boldsymbol{g}/\partial \boldsymbol{y}_N](\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N)$ implies the computability (and, hence, causality) of $\mathcal{N}$ in the neighborhood of $(\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_N)$.

### D. Solution of the Network

The computation of the network $\mathcal{N}$ at time $n$ can be decomposed into the following steps (refer also to Fig. 3).

1: Compute $\boldsymbol{y}_N[n]$ and $\boldsymbol{x}_N[n]$ from (17) and (15) using inputs $\boldsymbol{u}[n]$ and states $\boldsymbol{p}[n]$, $\boldsymbol{q}[n]$.
2: Compute $\boldsymbol{x}_L[n]$ and $\boldsymbol{y}_L[n]$ from (14) and (7), respectively.
3: Update states into $\boldsymbol{p}[n+1]$ and $\boldsymbol{q}[n+1]$.

In particular $\boldsymbol{q}[n+1]$ can be computed in the last step by feeding each filter with a null signal [20]. Note that no computation is needed for $\boldsymbol{q}[n+1]$ if the filters are realized in transposed direct form [4], [5]. The critical step in this procedure is the first one, i.e., the computation of the nonlinear blocks. More precisely the computation of $\boldsymbol{y}_N$ from (17) requires to solve an implicit system of nonlinear equations. In [23] a strategy was proposed which amounts to applying the Newton–Raphson (NR) method to find $\boldsymbol{y}_N[n]$ iteratively. This strategy has some similarities to what Borin *et al.* have proposed in a Kirchhoff-based nonlinear system solver [10]. Note however that only algebraic nonlinearities are considered in [10], while the nonlinearity in (17) is dynamic due to the presence of the internal state $\boldsymbol{p}$.

The NR algorithm [24] searches for a local zero of the function $\boldsymbol{g}$ defined in (18). A pseudo-code description of the algorithm is the following:

1. Set $k = 0$ and $\boldsymbol{y}_{Nk} = \boldsymbol{y}_N[n-1]$
2. **repeat**
3. Compute $\boldsymbol{g}(\boldsymbol{y}_{Nk})$ from (18)
4. Compute $\boldsymbol{J}_k^{-1} = [\partial \boldsymbol{g}/\partial \boldsymbol{y}_N](\tilde{\boldsymbol{x}}_N, \boldsymbol{y}_{Nk})$
5. Compute $\boldsymbol{y}_{N(k+1)} = \boldsymbol{y}_{Nk} - \boldsymbol{J}_k^{-1} \cdot \boldsymbol{g}(\boldsymbol{y}_{Nk})$
6. Compute $\mathrm{err} = \mathrm{abs}(\boldsymbol{y}_{N(k+1)} - \boldsymbol{y}_{Nk})$
7. Increment $k$
8. **until** err becomes small enough
9. Set $\boldsymbol{y}_N[n] = \boldsymbol{y}_{Nk}$

An alternative and less computationally intensive approach is based on fixed-point (FP) iteration [24]. If we redefine the function $\boldsymbol{g}$ as

$$\boldsymbol{g}(\boldsymbol{y}_N) = \boldsymbol{f}\big(\boldsymbol{W}_1 \boldsymbol{y}_N + \tilde{\boldsymbol{x}}_N, \boldsymbol{p}\big) \tag{20}$$

then FP iteration computes the network outputs $\boldsymbol{y}_N[n]$ as follows:

1. Set $k = 0$ and $\boldsymbol{y}_{Nk} = \boldsymbol{y}_N[n-1]$
2. **repeat**
3. Compute $\boldsymbol{y}_{N(k+1)} = \boldsymbol{g}(\boldsymbol{y}_{Nk})$ from (20)
4. Compute $\mathrm{err} = \mathrm{abs}(\boldsymbol{y}_{N(k+1)} - \boldsymbol{y}_{Nk})$
5. Increment $k$
6. **until** err becomes small enough
7. Set $\boldsymbol{y}_N[n] = \boldsymbol{y}_{Nk}$

However, in order for FP iteration to converge, one must ensure that the nonlinear function $\boldsymbol{g}$ satisfies more restrictive hypotheses. Namely, $\boldsymbol{g}$ must possess a "small" Lipschitz constant:

$$\|\boldsymbol{g}(\boldsymbol{y}) - \boldsymbol{g}(\boldsymbol{y}^*)\| \leq M_p \|\boldsymbol{y} - \boldsymbol{y}^*\| \tag{21}$$

with $0 \leq M_p < 1$. If $\boldsymbol{f}(\cdot, \boldsymbol{p})$ has a Lipschitz constant $\bar{M}_p$, then an estimate of $M_p$ can be given:

$$\begin{aligned} \|\boldsymbol{g}(\boldsymbol{y}) - \boldsymbol{g}(\boldsymbol{y}*)\| &= \|\boldsymbol{f}(\boldsymbol{W}_1\boldsymbol{y} + \tilde{\boldsymbol{x}}_N, \boldsymbol{p}) - \boldsymbol{f}(\boldsymbol{W}_1\boldsymbol{y}^* + \tilde{\boldsymbol{x}}_N, \boldsymbol{p})\| \\ &\leq \bar{M}_p \|\boldsymbol{W}_1(\boldsymbol{y} - \boldsymbol{y}^*)\| \\ &\leq \bar{M}_p \|\boldsymbol{W}_1\| \cdot \|\boldsymbol{y} - \boldsymbol{y}^*\|. \end{aligned} \tag{22}$$

Therefore, $\boldsymbol{g}$ has a Lipschitz constant $M_p = \bar{M}_p \|\boldsymbol{W}_1\|$. Moreover, assuming that $\boldsymbol{C}_{\mathrm{NN}} = \boldsymbol{0}$ one can write[3]

$$M_p \leq \bar{M}_p \|\boldsymbol{C}_{\mathrm{NL}}\| \cdot \|\boldsymbol{B}\| \cdot \left\|\boldsymbol{F}^{-1}\right\| \cdot \|\boldsymbol{C}_{\mathrm{LN}}\|. \tag{23}$$

Such an estimate proves useful when making preliminary considerations on the use of FP iteration [19].

## IV. Modularity

Each time a network $\mathcal{N}$ is exposed to a structural change, for instance due to the removal, the rewiring or the insertion of one or more blocks, the $\boldsymbol{B}$ and/or $\boldsymbol{C}$ matrices must be reformulated. The removal of one block simply requires to cut the corresponding column in $\boldsymbol{C}$, as well as the row and column in $\boldsymbol{B}$ describing its instantaneous behavior, if the block is linear. Rewiring implies a rearrangement of the nonzero elements in $\boldsymbol{C}$. Conversely, the insertion of one or more blocks requires to

[3]This assumption is not restrictive: if $\boldsymbol{C}_{\mathrm{NN}} \neq \boldsymbol{0}$ one can find an equivalent network with $\boldsymbol{C}_{\mathrm{NN}} = \boldsymbol{0}$, by inserting "dummy" linear elements $y_i = x_i$ between nonlinear-to-nonlinear connections.

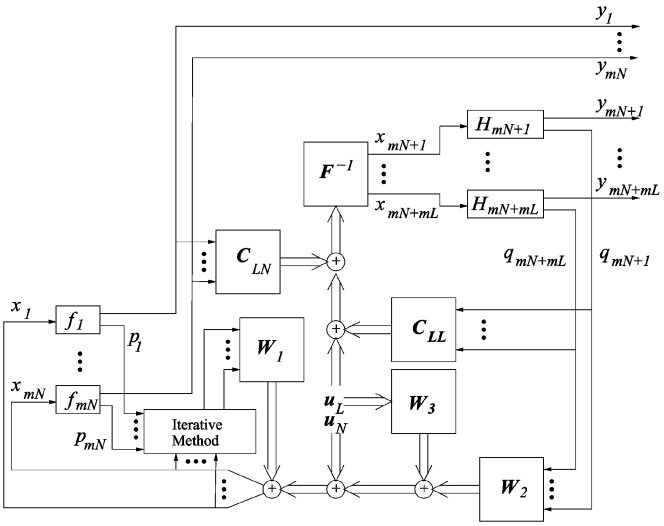


Fig. 3. Schematic of the proposed method.

resize $\boldsymbol{C}$ and possibly $\boldsymbol{B}$, with the inclusion of nonzero elements in these two matrices.

In this section we show that the insertion operation can be realized by means of a block-sized rearrangement of $\boldsymbol{B}$ and $\boldsymbol{C}$, holding the nonrestrictive hypothesis that the structure of the resulting network still admits a description as the one given in Section II. The possibility to restructure the network by rearranging blocks of matrices is advantageous when encoding the method into a software program that enables interactive access to the network structure.

Two filter networks $\mathcal{N}'$ and $\mathcal{N}''$, that are described by (1)–(8), can be assembled together into one super-network $\mathcal{N}$ having the same description provided that the signals exchanged between the two networks superimpose linearly to their respective inputs in the way expressed by (9). Holding this assumption, the cross-connections between $\mathcal{N}'$ and $\mathcal{N}''$ affect only the connectivity (8), in which an additive component must be included to account for the signals coming from the mutual network. In this way (6)–(9) are rewritten for $\mathcal{N}'$ and $\mathcal{N}''$ respectively as (see Fig. 4)

$$\begin{aligned}\boldsymbol{y}'_N[n] &= \boldsymbol{f}'(\boldsymbol{x}'_N[n], \boldsymbol{p}'[n]),\\ \boldsymbol{y}'_L[n] &= \boldsymbol{B}'\boldsymbol{x}'_L[n] + \boldsymbol{q}'[n],\\ \boldsymbol{x}'[n] &= \boldsymbol{C}'\boldsymbol{y}'[n] + \boldsymbol{D}'\boldsymbol{y}''[n] + \boldsymbol{u}'[n]\end{aligned} \tag{24}$$

and

$$\begin{aligned}\boldsymbol{y}''_N[n] &= \boldsymbol{f}''(\boldsymbol{x}''_N[n], \boldsymbol{p}''[n]),\\ \boldsymbol{y}''_L[n] &= \boldsymbol{B}''\boldsymbol{x}''_L[n] + \boldsymbol{q}''[n],\\ \boldsymbol{x}''[n] &= \boldsymbol{C}''\boldsymbol{y}''[n] + \boldsymbol{D}''\boldsymbol{y}'[n] + \boldsymbol{u}''[n].\end{aligned} \tag{25}$$

In $\mathcal{N}'$, the additive component appears in (24) in the form of a matrix $\boldsymbol{D}'$ sized $m'_C \times (m''_N + m''_L)$, which contains a nonzero element at position $j + m'_{C,k-1}$, $i$ if the output of the $i$th filter in $\mathcal{N}''$ sums to the $j$th input of the $k$th filter in $\mathcal{N}'$, otherwise the same element is null. Symmetrical considerations can be made for the meaning of $\boldsymbol{D}''$ in the connectivity equation of $\mathcal{N}''$ in (25).

The equation sets (24) and (25) can be grouped together to form the following system:

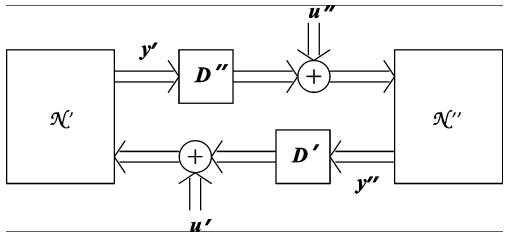


Fig. 4. Interconnection of two filter networks $\mathcal{N}'$ and $\mathcal{N}''$.

$$\begin{aligned}\boldsymbol{y}_N[n] &= \boldsymbol{f}(\boldsymbol{x}_N[n], \boldsymbol{p}[n])\\ \boldsymbol{y}_L[n] &= \boldsymbol{B}\boldsymbol{x}_L[n] + \boldsymbol{q}[n]\\ \hat{\boldsymbol{x}}[n] &= \boldsymbol{D}\hat{\boldsymbol{y}}[n] + \hat{\boldsymbol{u}}[n]\end{aligned} \tag{26}$$

in which the first vector equation is obtained by putting the first equation in (24) on top of the first equation in (25), i.e.,

$$\boldsymbol{y}_N = \begin{vmatrix} \boldsymbol{y}'_N \\ \boldsymbol{y}''_N \end{vmatrix}, \quad \boldsymbol{f} = \begin{vmatrix} \boldsymbol{f}' \\ \boldsymbol{f}'' \end{vmatrix},$$
$$\boldsymbol{x}_N = \begin{vmatrix} \boldsymbol{x}'_N \\ \boldsymbol{x}''_N \end{vmatrix}, \quad \boldsymbol{p} = \begin{vmatrix} \boldsymbol{p}' \\ \boldsymbol{p}'' \end{vmatrix}.$$

The second equation in (26) is constructed in the same way, i.e.,

$$\boldsymbol{y}_L = \begin{vmatrix} \boldsymbol{y}'_L \\ \boldsymbol{y}''_L \end{vmatrix}, \quad \boldsymbol{B} = \begin{vmatrix} \boldsymbol{B}' & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}'' \end{vmatrix},$$
$$\boldsymbol{x}_L = \begin{vmatrix} \boldsymbol{x}'_L \\ \boldsymbol{x}''_L \end{vmatrix}, \quad \boldsymbol{q} = \begin{vmatrix} \boldsymbol{q}' \\ \boldsymbol{q}'' \end{vmatrix}.$$

By following the same construction technique, the third equation in (26) is obtained by defining,

$$\hat{\boldsymbol{x}} = \begin{vmatrix} \boldsymbol{x}' \\ \boldsymbol{x}'' \end{vmatrix}, \quad \boldsymbol{D} = \begin{vmatrix} \boldsymbol{C}' & \boldsymbol{D}' \\ \boldsymbol{D}'' & \boldsymbol{C}'' \end{vmatrix},$$
$$\hat{\boldsymbol{y}} = \begin{vmatrix} \boldsymbol{y}' \\ \boldsymbol{y}'' \end{vmatrix}, \quad \hat{\boldsymbol{u}} = \begin{vmatrix} \boldsymbol{u}' \\ \boldsymbol{u}'' \end{vmatrix}.$$

The first and second equations in (26) have a structure which is identical to that of (6) and (7), respectively. Yet, the third equation does not have the same structure as (9) since $\boldsymbol{D}$ cannot be partitioned into four submatrices having the same meaning as $\boldsymbol{C}_{\mathrm{NN}}$, $\boldsymbol{C}_{\mathrm{NL}}$, $\boldsymbol{C}_{\mathrm{LN}}$, and $\boldsymbol{C}_{\mathrm{LL}}$ in (10).

The structure of $\boldsymbol{D}$ can be observed if we explode its four components $\boldsymbol{C}'$, $\boldsymbol{D}'$, $\boldsymbol{D}''$, and $\boldsymbol{C}''$, which do possess that structure by their own definition,

$$\boldsymbol{D} = \begin{vmatrix} \boldsymbol{C}' & \boldsymbol{D}' \\ \boldsymbol{D}'' & \boldsymbol{C}'' \end{vmatrix} = \begin{vmatrix} \boldsymbol{C}'_{\mathrm{NN}} & \boldsymbol{C}'_{\mathrm{NL}} & \boldsymbol{D}'_{\mathrm{NN}} & \boldsymbol{D}'_{\mathrm{NL}} \\ \boldsymbol{C}'_{\mathrm{LN}} & \boldsymbol{C}'_{\mathrm{LL}} & \boldsymbol{D}'_{\mathrm{LN}} & \boldsymbol{D}'_{\mathrm{LL}} \\ \boldsymbol{D}''_{\mathrm{NN}} & \boldsymbol{D}''_{\mathrm{NL}} & \boldsymbol{C}''_{\mathrm{NN}} & \boldsymbol{C}''_{\mathrm{NL}} \\ \boldsymbol{D}''_{\mathrm{LN}} & \boldsymbol{D}''_{\mathrm{LL}} & \boldsymbol{C}''_{\mathrm{LN}} & \boldsymbol{C}''_{\mathrm{LL}} \end{vmatrix}. \tag{27}$$

By rotating $\boldsymbol{D}$ so as to swap the second and third row as well as the second and third column, the matrix takes the form required by the connectivity equation. We denote the resulting matrix again with $\boldsymbol{C}$,

$$C = \begin{vmatrix} \boldsymbol{C}'_{\mathrm{NN}} & \boldsymbol{D}'_{\mathrm{NN}} & \boldsymbol{C}'_{\mathrm{NL}} & \boldsymbol{D}'_{\mathrm{NL}} \\ \boldsymbol{D}''_{\mathrm{NN}} & \boldsymbol{C}''_{\mathrm{NN}} & \boldsymbol{D}''_{\mathrm{NL}} & \boldsymbol{C}''_{\mathrm{NL}} \\ \boldsymbol{C}'_{\mathrm{LN}} & \boldsymbol{D}'_{\mathrm{LN}} & \boldsymbol{C}'_{\mathrm{LL}} & \boldsymbol{D}'_{\mathrm{LL}} \\ \boldsymbol{D}''_{\mathrm{LN}} & \boldsymbol{C}''_{\mathrm{LN}} & \boldsymbol{D}''_{\mathrm{LL}} & \boldsymbol{C}''_{\mathrm{LL}} \end{vmatrix}. \tag{28}$$

This rotation implies that in the third equation in (26) the second and third rows of

$$\hat{\boldsymbol{x}} = \begin{vmatrix} \boldsymbol{x}'_N \\ \boldsymbol{x}'_L \\ \boldsymbol{x}''_N \\ \boldsymbol{x}''_L \end{vmatrix}, \quad \hat{\boldsymbol{y}} = \begin{vmatrix} \boldsymbol{y}'_N \\ \boldsymbol{y}'_L \\ \boldsymbol{y}''_N \\ \boldsymbol{y}''_L \end{vmatrix}, \quad \hat{\boldsymbol{u}} = \begin{vmatrix} \boldsymbol{u}'_N \\ \boldsymbol{u}'_L \\ \boldsymbol{u}''_N \\ \boldsymbol{u}''_L \end{vmatrix},$$

must be correspondingly swapped. By respectively denoting the rotated versions of $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{u}}$ again with $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{u}$, then we obtain an equation that matches (10), with $\boldsymbol{x}_N, \boldsymbol{x}_L, \boldsymbol{y}_N, \boldsymbol{y}_L, \boldsymbol{u}_N$, and $\boldsymbol{u}_L$ given by the previously defined vectors and $\boldsymbol{C}$ as in (28). The decomposition of this matrix now gives

$$\boldsymbol{C}_{\mathrm{NN}} = \begin{vmatrix} \boldsymbol{C}'_{\mathrm{NN}} & \boldsymbol{D}'_{\mathrm{NN}} \\ \boldsymbol{D}''_{\mathrm{NN}} & \boldsymbol{C}''_{\mathrm{NN}} \end{vmatrix}, \quad \boldsymbol{C}_{\mathrm{NL}} = \begin{vmatrix} \boldsymbol{C}'_{\mathrm{NL}} & \boldsymbol{D}'_{\mathrm{NL}} \\ \boldsymbol{D}''_{\mathrm{NL}} & \boldsymbol{C}''_{\mathrm{NL}} \end{vmatrix},$$
$$\boldsymbol{C}_{\mathrm{LN}} = \begin{vmatrix} \boldsymbol{C}'_{\mathrm{LN}} & \boldsymbol{D}'_{\mathrm{LN}} \\ \boldsymbol{D}''_{\mathrm{LN}} & \boldsymbol{C}''_{\mathrm{LN}} \end{vmatrix}, \quad \boldsymbol{C}_{\mathrm{LL}} = \begin{vmatrix} \boldsymbol{C}'_{\mathrm{LL}} & \boldsymbol{D}'_{\mathrm{LL}} \\ \boldsymbol{D}''_{\mathrm{LL}} & \boldsymbol{C}''_{\mathrm{LL}} \end{vmatrix}.$$

In conclusion, the interconnection of $\mathcal{N}'$ and $\mathcal{N}''$ via $\boldsymbol{D}'$ and $\boldsymbol{D}''$ is realized by diagonally joining $\boldsymbol{B}'$ and $\boldsymbol{B}''$ into a new matrix $\boldsymbol{B}$, and by composing $\boldsymbol{C}$ according to the structure given by (28).

Using the matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ of $\mathcal{N}$ in the definition of $\boldsymbol{F}$ leads to the following result:

$$\boldsymbol{F} = \begin{vmatrix} \boldsymbol{F}' & -\boldsymbol{D}'_{\mathrm{LL}}\boldsymbol{B}'' \\ -\boldsymbol{D}''_{\mathrm{LL}}\boldsymbol{B}' & \boldsymbol{F}'' \end{vmatrix}. \tag{29}$$

As a particular case, if $\mathcal{N}''$ does not contain linear blocks then it is $\boldsymbol{F} = \boldsymbol{F}'$. Hence, the inverse $\boldsymbol{F}^{-1}$ needs not to be recalculated. Furthermore if $\boldsymbol{D}'_{\mathrm{LL}} = \boldsymbol{D}''_{\mathrm{LL}} = \boldsymbol{0}$ (no cross-connections exist between the linear blocks of $\mathcal{N}'$ and $\mathcal{N}''$), then $\boldsymbol{F}^{-1}$ is obtained by diagonally joining $\boldsymbol{F}'^{-1}$ and $\boldsymbol{F}''^{-1}$. In the general case, i.e., when there are branches connecting linear blocks from one network to the other, one can still exploit previous knowledge of $\boldsymbol{F}'^{-1}$ and $\boldsymbol{F}''^{-1}$ and compute $\boldsymbol{F}^{-1}$ through simple block matrix inversion [25]

$$\boldsymbol{F}^{-1} = \begin{vmatrix} \boldsymbol{S}^{-1}_{\boldsymbol{F}''} & \boldsymbol{F}'^{-1}\boldsymbol{D}'_{\mathrm{LL}}\boldsymbol{B}''\boldsymbol{S}^{-1}_{\boldsymbol{F}'} \\ \boldsymbol{F}''^{-1}\boldsymbol{D}''_{\mathrm{LL}}\boldsymbol{B}'\boldsymbol{S}^{-1}_{\boldsymbol{F}''} & \boldsymbol{S}^{-1}_{\boldsymbol{F}'} \end{vmatrix}, \tag{30}$$

where $\boldsymbol{S}_{\boldsymbol{F}'} = \boldsymbol{F}'' - \boldsymbol{D}''_{\mathrm{LL}}\boldsymbol{B}'\boldsymbol{F}'^{-1}\boldsymbol{D}'_{\mathrm{LL}}\boldsymbol{B}''$ and $\boldsymbol{S}_{\boldsymbol{F}''} = \boldsymbol{F}' - \boldsymbol{D}'_{\mathrm{LL}}\boldsymbol{B}''\boldsymbol{F}''^{-1}\boldsymbol{D}''_{\mathrm{LL}}\boldsymbol{B}'$ are the Schur components of $\boldsymbol{F}'$ and $\boldsymbol{F}''$, respectively.

It is straightforward to extend the block-based insertion procedure to the case where $R$ networks $\mathcal{N}^{(0)}, \ldots, \mathcal{N}^{(R-1)}$ are connected together. In fact, it is sufficient to initially define $\mathcal{N} = \mathcal{N}^{(0)}$, then iterate the following operation $R-1$ times starting with $i = 1$: at step $i$ the network $\mathcal{N}^{(i)}$ is joined to the super-network $\mathcal{N}$ obtained at the previous step, and $i$ is incremented. After the last iteration, $\mathcal{N}$ is the interconnection of $\mathcal{N}^{(0)}, \ldots, \mathcal{N}^{(R-1)}$.

## V. Application and Examples

The proposed method allows to find the (supposed to exist) solution of a causal discrete-time filter network, provided the availability of a numerical strategy solving the implicit system (15) or, equivalently, (17). Inaccuracies in the simulation of a continuous system reside in the approximations that are introduced by the analog-to-digital maps transforming the continuous-time blocks into the discrete-time domain. Such domain transformations must be performed before computation.

Since the structure of the filter blocks is preserved by the method, specific analog-to-digital maps can be selected depending on the features and the complexity of each block. This freedom of choice allows to control the accuracy with which every filter block is modeled in the discrete-time domain. The local control of the accuracy may help add insight in systems for which the exploration of aspects such as the local sensitivity to parameter changes and the robustness against local perturbations is crucial as much as the accurate representation of their dynamics. This is the case, for instance, of some biological regulation networks [26], [27]. Later in this section we will provide an example of an intracellular transduction network whose numerical solution is sensitive to the value of the temporal step, and where the exact computation of the delay-free loops contained in it leads to a more robust solution.

In the next paragraphs we compute a chaotic circuit that has been previously studied in several other works by the audio signal processing community, to prove the reliability of the proposed approach. Besides this test, the methods proposed here have already been successfully employed in simulations of nonlinear acoustic systems, to which the reader is referred [18], [19].

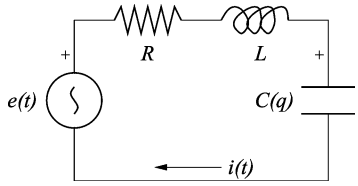


Fig. 5. Chua–Felderhoff nonlinear *RLC* circuit.

### A. Realization of Chua–Felderhoff Circuits

Because of their sensitivity to the integration scheme, Chua–Felderhoff circuits have been used in the past to test the robustness of methods for the computation of nonlinear digital filter networks [9], [10].

Fig. 5 shows the electrical equivalent of the Chua–Felderhoff *RLC* circuit. Particular to this network is the charge-dependent capacitance $C(q)$ which establishes the following relation between the voltage $v_C$ at the capacitor and its charge $q$ [10]:

$$v_C(q) = \frac{1}{2v_0C_0^2}\left(q^2 + q\sqrt{q^2 + 4v_0^2C_0^2}\right) \tag{31}$$

in which $v_0$ and $C_0$ are characteristic parameters of the capacitor.

By summing the voltage at all passive components using the orientations given in Fig. 5, i.e., $e = v_R + v_L + v_C$, and by recalling that $v_R = R\dot{q}$ and $v_L = L\ddot{q}$ respectively for the voltage at the resistor and the inductor, then it is immediate to derive the following equation:

$$e - v_C(q) = R\dot{q} + L\ddot{q}. \tag{32}$$

The right-hand side of (32) states that the charge $q$ is filtered out of a linear system modeling the *RL* series. The left-hand side of

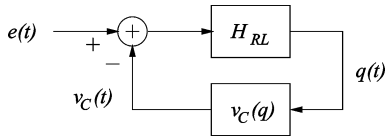


Fig. 6. Filter model of the Chua–Felderhoff circuit.

(32) states that $q$ is instantaneously needed to compute the signal $e - v_C$, that is, the voltage measured at the *RL* series feeding the linear filter. The meaning of this equation becomes evident if we move to the Laplace domain and rewrite (32) as

$$\frac{Q(s)}{E(s) - V_C(s)} = \frac{1}{Rs + Ls^2} = H_{RL}(s) \tag{33}$$

in which $H_{RL}(s)$ is the voltage-to-charge transfer function accounted for by the *RL* series.

The filter network modeling the Chua–Felderhoff circuit is shown in Fig. 6. It is straightforward to see that this network contains one linear and one nonlinear block, and captures the relation (32) along with the nonlinear function (31).

The analog-to-digital transformation of $H_{RL}(s)$ into $H_{RL}(z)$ can be obtained by mapping the Laplace into the Zeta domain by means of the *bilinear* transform $s \leftrightarrow (2/\Delta_t)(z-1)/(z+1)$:

$$H_{RL}(z) = \frac{\Delta_t^2}{4L + 2R\Delta_t} \frac{1 + 2z^{-1} + z^{-2}}{1 - \frac{4L}{2L+R\Delta_t} z^{-1} + \frac{4L-2R\Delta_t}{4L+2R\Delta_t} z^{-2}}. \tag{34}$$

From (6)–(9), it descends $y_1 = v_C$, $y_2 = q$, $u_2 = e$, furthermore

$$\boldsymbol{B} = b_2 = \frac{\Delta_t^2}{4L + 2R\Delta_t}, \quad \boldsymbol{C} = \begin{vmatrix} 0 & 1 \\ -1 & 0 \end{vmatrix}. \tag{35}$$

By setting $L = 100\ \mu$H, $R = 180\ \Omega$, $C_0 = 80$ pF, $v_0 = 0.6$ V, and $e(t) = e_0 \sin(2\pi f_0 t)$ with $e_0 = 3.57$ V and $f_0 = 1/(2\pi\sqrt{LC_0})$ Hz, we obtain the charge/current phase portrait in Fig. 7, for which we have chosen a sampling frequency $1/\Delta_t = 32 f_0 \approx 57$ MHz, and computed the current at every temporal step by using first-order differences of the charge: $i[n] = (q[n] - q[n-1])/\Delta_t$. Note that the discrete-time version of the excitation signal is obtained by posing $e[n] = e(n\Delta_t) = e_0 \sin(\pi n/16)$.

Similarities with previous results obtained using methods working in the wave and Kirchhoff domain using the same simulation parameters are evident [9], [10].

### *B. Signal Transduction in Circadian Cycles*

The dynamics of intracellular signals is often inspected using deterministic or stochastic models, that compute the concentrations along time of the chemical species involved in a biological process [28]–[30]. Both approaches move from the information contained in a signal transduction network, that describes how the chemical reactions are coupled together to form this process. Each reaction transforms reactants into products and it is associated to a *rate equation*, that is usually expressed in the form of a nonlinear differential equation providing the dynamical description, or *kinetics* of a reaction. The rate equations associated to a signal transduction network must be completed with a set of kinetic parameters, that are obtained from *in vivo* or *in vitro*

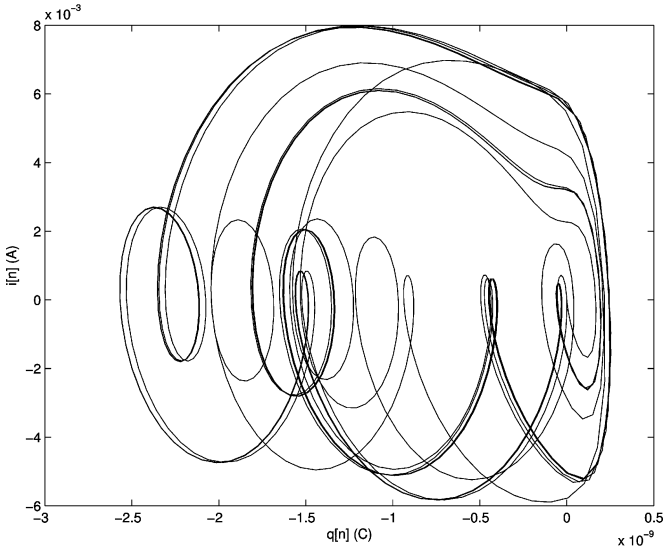


Fig. 7. Phase portrait of the Chua–Felderhoff circuit model.

measurements or from qualitative estimations of temporal variations of concentration. More recent approaches to biological dynamics rely on algorithms that are rooted in formal language theory [31].

Most deterministic models, hence, derive a nonlinear ODE system from a signal transduction network. This system often contains tens of rate equations in the unknown concentration variables. Almost all biologically relevant signal transduction networks contain feedback loops: from a biochemical point of view, loops are interesting because they account for coupled reactions providing a cyclic transformation of the species involved in the loop. This implies that every chemical reaction forming the loop is indirectly fed by products of the reaction itself. Since the concentration of reactants determines the kinetics, these reactions are ultimately subjected to a form of feedback control. The reversibility of biochemical species through transduction loopbacks is a key aspect in the regulation and renovation cycles of the living cell.

If we assume that the biological information propagates slowly enough over a transduction network, i.e., that the transformation of the matter across the reactions is not instantaneous, then these ODE systems can be solved using explicit integration methods that step over adequate time slots. Cases exist in which slow propagation cannot be invoked due to the presence of fast coupled reactions in the transduction network [32]. In these cases, the application of explicit integration methods can lead to inaccurate or unstable solutions.

The aspect of propagation is usually neglected during the solution of deterministic ODE systems modeling biochemical signal transduction, furthermore rarely emphasis is given to how rate equations are integrated. In the following we illustrate an example which demonstrates that the mere application of an integration method can lead to inaccuracies in the results, even in cases where the numerical solution is inherently robust.

We analyze a model of circadian cycles (or *rhythms*) in *Drosophila melanogaster* [14]. Existing in almost every living organism, circadian rhythms are evoked by temporal variations in the expression level of specific genes. The periodicity of such

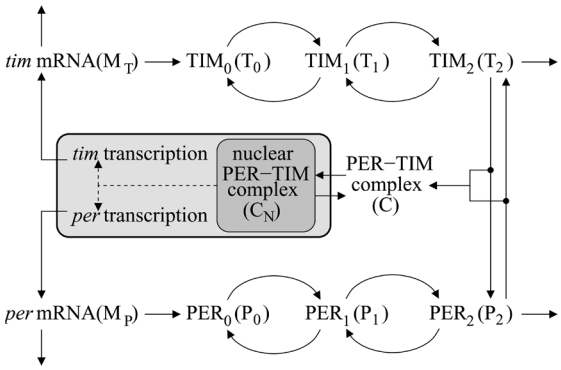


Fig. 8. Signal transduction network model of circadian rhythms in *Drosophila melanogaster* [33].

variations gives rise to a surprisingly robust biological clock, synchronized with daylight and performing a complete cycle about every 24 hours. The model we analyze has a particularly low sensitivity to the variations of its kinetic parameters. This property is explained by assuming that it incorporates robustness in the transduction network.

Fig. 8 shows this network. In this figure, every arrow corresponds to an intracellular chemical reaction involving DNA, messenger RNA (mRNA), and proteins. Intranuclear reactions are depicted inside the rectangle. Arrows which are not targeted to a product account for reactions of degradation.

According to Fig. 8, the genes that produce the Timeless (TIM) and Period (PER) proteins via the synthesis of the respective *tim* and *per* mRNA are inhibited by the presence, inside the nucleus, of the PER–TIM protein complex. This complex grows outside of the nucleus, as a result of two parallel series of reactions. As soon as it migrates inside the nucleus, it acts as a suppressor of the *tim* and *per* transcription, thus providing a negative feedback. The action of the PER–TIM complex over the synthesis of mRNA is visualized in Fig. 8 using a couple of arrows in dashed line. It is interesting to recall that the formation of the PER-TIM complex is regulated also by the degradation induced on mature TIM by light, whose role is not taken into account in our analysis.

*1) Rate Equation Model:* The derivation of the rate equations from the reactions shown in Fig. 8 is not explained in this paper [13], [34]. The coupled reaction

$$\text{per mRNA} \;\rightarrow\; \text{PER}_0 \leftrightarrow\; \text{PER}_1 \leftrightarrow\; \text{PER}_2 \tag{36}$$

forming the lower extra-nuclear pathway of the transduction network, results in the following rate equations:

$$\begin{aligned}
\dot{M}_P &= v_{sP}\frac{K_{IP}^4}{K_{IP}^4+C_N^4} - v_{mP}\frac{M_P}{K_{mP}+M_P} - k_d M_P\\
\dot{P}_0 &= k_{sP}M_P - V_{1P}\frac{P_0}{K_{1P}+P_0} + V_{2P}\frac{P_1}{K_{2P}+P_1} - k_d P_0\\
\dot{P}_1 &= V_{1P}\frac{P_0}{K_{1P}+P_0} - V_{2P}\frac{P_1}{K_{2P}+P_1}\\
&\quad - V_{3P}\frac{P_1}{K_{3P}+P_1} + V_{4P}\frac{P_2}{K_{4P}+P_2} - k_d P_1\\
\dot{P}_2 &= V_{3P}\frac{P_1}{K_{3P}+P_1} - V_{4P}\frac{P_2}{K_{4P}+P_2}\\
&\quad - k_3 P_2 T_2 + k_4 C - v_{dP}\frac{P_2}{K_{dP}+P_2} - k_d P_2
\end{aligned} \tag{37}$$

in which $M_P$, $P_0$, $P_1$, and $P_2$ are molecule concentrations of the biochemical species that are involved in the *per* pathway. Symmetrical functional relationships hold for the species $M_T$, $T_0$, $T_1$, $T_2$ (upper pathway), and give rise to structurally identical rate equations just by substituting the $P$ symbol in every equation with a $T$ [13]. Finally, the coupled reaction (nuclear PER-TIM) $\leftrightarrow$ PER-TIM forming the middle pathway results in the following two rate equations:

$$\begin{aligned}
\dot{C} &= k_3 P_2 T_2 - k_4 C - k_1 C + k_2 C_N - k_{dC} C\\
\dot{C}_N &= k_1 C - k_2 C_N - k_{dN} C_N
\end{aligned} \tag{38}$$

in which $C$ and $C_N$ are molecule concentrations of the PER-TIM and nuclear PER-TIM complex, respectively.

*2) Filter Network Model:* Similarly to what happened to the nonlinear *RLC* voltage (32), every ODE in (37) and (38) is the result of superposing input components and algebraic nonlinearities to linear terms accounting for degradation. For instance, the second equation in (37) has the form

$$\dot{P}_0 = k_{sP}M_P + f_2(P_0, P_1) - k_d P_0 \tag{39}$$

in which we recognize the three additive terms on the right-hand side to be respectively the input, the nonlinear component, and the degradation term.

By repeating the reasoning that we made to obtain (33) from (32), we can model $P_0$ in (39) as the output of a linear filter, having transfer function $H(s) = 1/(s + k_d)$ and being fed with $k_{sP}M_P + f_2(P_0, P_1)$. Note that $f_2$ is a function of the filter output $P_0$.

Structurally identical models hold for all ODEs in (37), so that we can build the filter network of Fig. 9. Let the reader note the structural similarity existing between this filter network and the transduction network in Fig. 8.

In Fig. 9, it is

$$\begin{aligned}
f_1(C_N, M_P) &= v_{sP}\frac{K_{IP}^4}{K_{IP}^4+C_N^4} - v_{mP}\frac{M_P}{K_{mP}+M_P}\\
f_2(P_0, P_1) &= V_{1P}\frac{P_0}{K_{1P}+P_0} - V_{2P}\frac{P_1}{K_{2P}+P_1}\\
f_3(P_1, P_2) &= V_{3P}\frac{P_1}{K_{3P}+P_1} - V_{4P}\frac{P_2}{K_{4P}+P_2}\\
f_4(P_2) &= v_{dP}\frac{P_2}{K_{dP}+P_2}.
\end{aligned} \tag{40}$$

As before, the nonlinearities $f_5$, $f_6$, $f_7$, and $f_8$, are, respectively, obtained from $f_1$, $f_2$, $f_3$, and $f_4$ by substituting $P$ with $T$ symbols. Furthermore

$$\begin{aligned}
f_9(P_2, T_2) &= k_3 P_2 T_2\\
H_i(s) &= H(s), \qquad i = 10, \ldots, 17\\
H_{18}(s) &= \frac{1}{s + k_1 + k_4 + k_{dC}}\\
H_{19}(s) &= \frac{1}{s + k_2 + k_{dN}}.
\end{aligned} \tag{41}$$

Rewriting (40) and (41) in the forms (1) and (2) is a straightforward but tedious exercise, that is omitted here. The indexes of the input/output signals to/from every block are directly derived from the corresponding block indices: for instance, it is $y_{12} = P_1$ as well as $\boldsymbol{x}_3 = |\, P_1 \quad P_2\,|^T$.

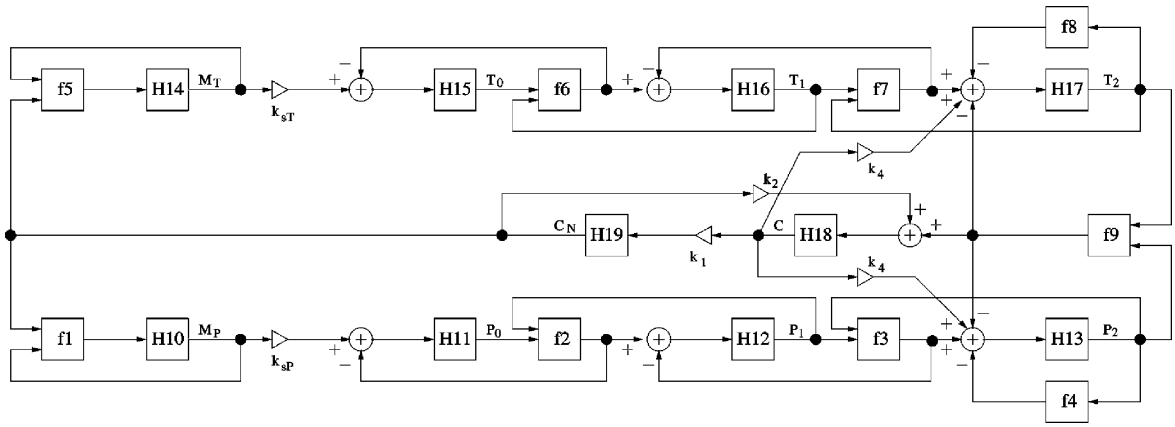


Fig. 9. Filter network for circadian rhythms.

TABLE I
PARAMETERS USED IN THE CIRCADIAN CYCLE MODEL

| $v_{sP}$ | $K_{IP}$ | $v_{mP}$ | $K_{mP}$ | $k_{sP}$ | $V_{1P}$ | $K_{1P}$ | $V_{2P}$ | $K_{2P}$ | $V_{3P}$ | $K_{3P}$ | $V_{4P}$ | $K_{4P}$ | $v_{dP}$ | $K_{dP}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $v_{sT}$ | $K_{IT}$ | $v_{mT}$ | $K_{mT}$ | $k_{sT}$ | $V_{1T}$ | $K_{1T}$ | $V_{2T}$ | $K_{2T}$ | $V_{3T}$ | $K_{3T}$ | $V_{4T}$ | $K_{4T}$ | $v_{dT}$ | $K_{dT}$ |
| 1 | 1 | 0 | 0.2 | 0.9 | 8 | 2 | 1 | 2 | 8 | 2 | 1 | 2 | 2 | 0.2 |
| nmol/h | nmol | nmol/h | nmol | nmol/h | nmol/h | nmol | nmol/h | nmol | nmol/h | nmol | nmol/h | nmol | nmol/h | nmol |

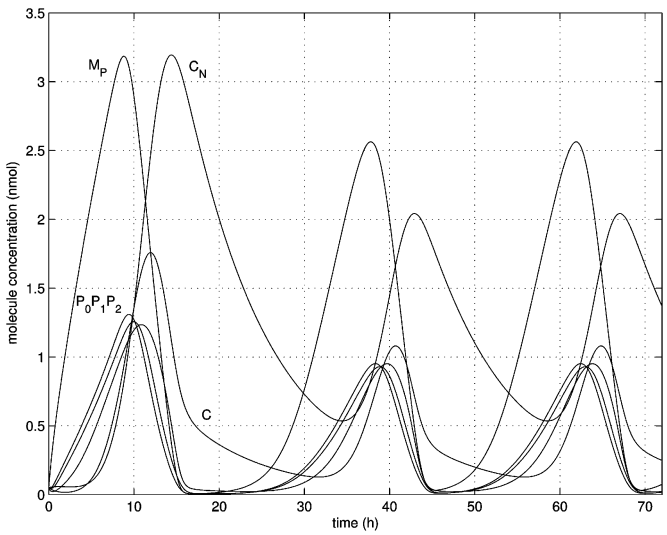


Fig. 10. Evolution of the molecule concentrations $M_P$, $P_0$, $P_1$, $P_2$, $C$, and $C_N$ in the circadian cycle filter model.

Every linear filter models a pure exponential decay. We discretize the lowpass transfer function $H(s)$ over a temporal step $\Delta_t$ by means of a simple backward Euler analog-to-digital map: $s \leftrightarrow (1 - z^{-1})/\Delta_t$, thus obtaining the discrete-time transfer function

$$H(z) = \frac{\Delta_t}{1 + \Delta_t k_d} \frac{1}{1 - \frac{1}{1+\Delta_t k_d} z^{-1}}. \tag{42}$$

(In general less trivial transformations are needed for filters having more sophisticate transfer functions.) Discrete-time versions of $H_{18}$ and $H_{19}$ are obtained by changing, in (42), the constant $k_d$ with $k_1 + k_4 + k_{dC}$ and $k_2 + k_{dN}$, respectively.

In this way, we form the system of vector equations (6)–(9). Branches containing pure multipliers are embedded directly in the matrix $\boldsymbol{C}$. Hence, we will have $\boldsymbol{C}(11, 10) = k_{sP}$, $\boldsymbol{C}(15, 14) = k_{sT}$, $\boldsymbol{C}(13, 18) = \boldsymbol{C}(17, 18) = k_4$, $\boldsymbol{C}(19, 18) = k_1$, and $\boldsymbol{C}(18, 19) = k_2$. Since there are nine nonlinear blocks, the decomposition (10) results in a submatrix $\boldsymbol{C}_{\mathrm{NN}}$ sized $9 \times 9$. The submatrices $\boldsymbol{C}_{\mathrm{NL}}$, $\boldsymbol{C}_{\mathrm{LN}}$, and $\boldsymbol{C}_{\mathrm{LL}}$ come out as a consequence.

*3) Simulation and Results:* A simulation of the filter network of Fig. 9 using the parameters listed in Table I [13], along with $k_1 = 0.6$ nmol/h, $k_2 = 0.2$ nmol/h, $k_3 = 1.2$ nmol/h, $k_4 = 0.6$ nmol/h, and $k_d = k_{dC} = k_{dN} = 0.01$ nmol/h, is shown in Fig. 10.

In this simulation every state variable has been set to 0.05 nmol, furthermore we have chosen a temporal step $\Delta_t = 7.5$ s, which allows a quite fine granularity of observation of the biological phenomenon. Using this step, a 72 h simulation as that shown in Fig. 10 required 60 s computation time on a 1.7 GHz Linux laptop running Matlab.

It can be observed that all molecule concentrations reach a stationary 24 hour oscillation period after an initial transient lasting about 20 hours. The ability of the signals $P_0$, $P_1$ and $P_2$ to propagate information is evident from the causal evolution of their concentrations along time. Identical considerations and plots can be drawn for $T_0$, $T_1$ and $T_2$, transducing signals across the upper pathway in Fig. 8.

It is interesting to compare the accuracy of the proposed model with that of a similar, explicitly computable filter network. Explicit computation can be obtained by turning all delay-free loops into corresponding computable structures. The most obvious choice is to insert a delay unit in series with every linear filter, hence changing every transfer function $H_i(z)$, $i = 10, \ldots, 19$, into $\tilde{H}_i(z) = H_i(z)/z$. Such a substitution

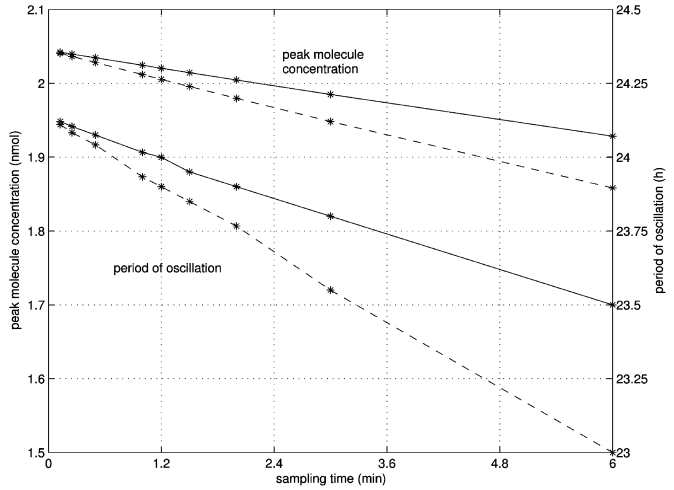


Fig. 11. Dependence on sampling time of the peak molecule concentrations (upper plots) and periods of oscillation (lower plots) for $C_N$. Solid line: Delay-free loop network. Dashed line: explicitly computable network. Asterisks denote the available data, from which lines have been interpolated.

can be avoided for $H_{18}$, since leaving this transfer function undelayed does not harm explicit computability. Of course, the explicit computation requires less simulation time compared to the delay-free loop computation, mainly because the solution of the implicit system (17) is no longer needed.

Fig. 11 shows peak molecule concentration values and periods of oscillation for $C_N$ during the stationary regime, obtained by simulating the model respectively using the proposed method and its explicit counterpart. The data in Fig. 11 result from simulations in which $\Delta_t$ was set, respectively, to 7.5, 15, 30, 60, 72, 90, 120, 180, and 360 s.

The results obtained from the delay-free loop filter network are more robust against the granularity of the computation step, in terms of accuracy of both peak concentration values and oscillation periods. Similar results hold for all signals traveling along the network. For small steps (approximately $\Delta_t < 10$ s) the accuracy of the explicit scheme becomes comparable to that of the implicit computation.

Interpolation of these data results in almost linear functions for both schemes, independently of the choice of peak molecule concentrations or periods of oscillation to benchmark accuracy. Solid lines denote interpolations of the results from the implicit scheme, while dashed lines interpolate explicit computations. These lines exhibit different slopes depending on the computational scheme: those provided by the explicit scheme are steeper, indicating faster accuracy decay with increasing computation step.

Differences in the results emerge also when modeling the chaotic oscillations of $M_P$, $M_T$, and $C_N$ using the delay-free filter network instead of the explicitly computable scheme. Chaos arises when $v_{mT}$ and $v_{dT}$ are respectively set to 0.35 and 5.3 nmol/h, while leaving all other values untouched [33]. Fig. 12 shows phase portraits of these molecule concentrations taken during a 236 day temporal window, after removing the first 3 days to let the trajectories move into the basin of the strange attractor. The data produced by the delay-free and explicitly computable filter network are shown in the upper and lower plot, respectively. In both cases we have set $\Delta_t = 30$ s,

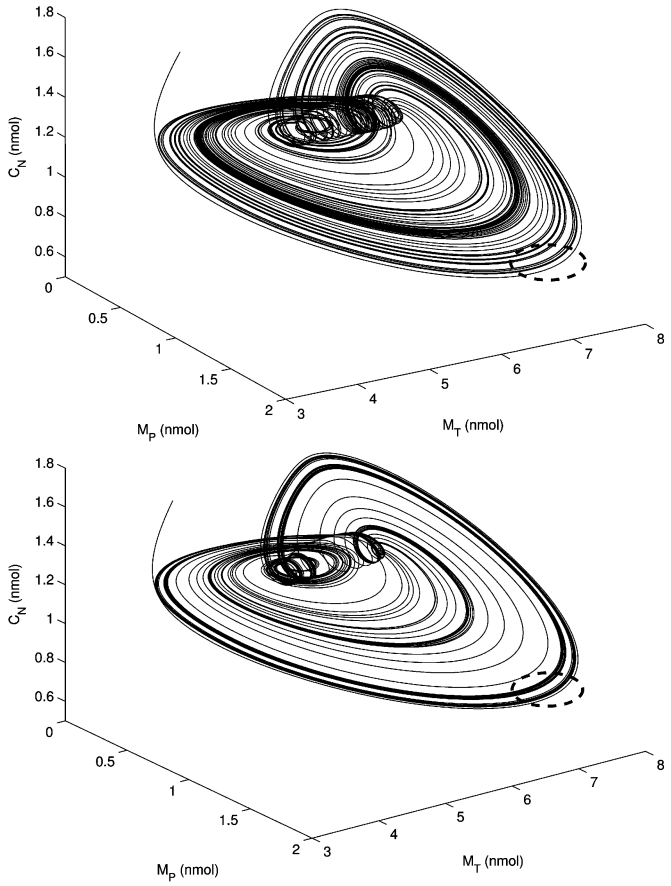


Fig. 12. Chaotic oscillations in the circadian cycle model. Upper plot: delay-free filter network. Lower plot: explicit scheme. Ellipses in dashed line highlight a plane section across which trajectories move.

along with starting with the same initial conditions as those used to obtain the plots in Fig. 10.

Such differences can be appreciated by observing the number and points of intersection of the trajectories across a plane section, as that depicted in the same figure in dashed line. Even a qualitative inspection of such intersections shows that both these figures noticeably change depending on the type of scheme adopted for the integration. An analysis aimed at comparing the trajectories resulting by the implicit and explicit scheme would require a detailed treatment of the system behavior during chaotic regimes, and is left to further research.

## VI. Conclusion

We have proposed a method for the solution of nonlinear digital filter networks containing delay-free loops, whose features turn useful when the network topology and the structure of the individual processing blocks must be preserved during the computation. Examples show that the method provides accurate solutions of dynamic models described by nonlinear ODE systems, which are not obvious to be integrated holding the constraint of instantaneous propagation in the discrete computational model.

Concerning the field of audio signal processing, the method may find convenient application especially in the simulation of some analog systems of the past decades, whose peculiar

sonic characteristics were largely due to the presence of feedback loops in their circuitry. Such systems, ranging from nonlinear filters to tube amplifiers, are now object of an intense research and development activity for the market of "virtual analog," aiming at accurately reproducing their features through real-time software running on general purpose computer architectures. In parallel to these applications, there is a growing interest for the virtual reproduction of sounds produced by dynamic contacts of simulated materials, and their applications to the field of human–computer interaction [35].

The solution of a nonlinear implicit system at every step seems to be unavoidable. This aspect prevents the method from proving really useful for the computation of stochastic signals. Especially in systems biology applications, where a stochastic characterization of the signals is often needed, the reader should rather survey algorithms that are explicitly capable of dealing with random processes [2]. On the other hand, if a delay-free loop network contains only linear blocks, then the application of standard mathematical tools for the linear filtering of stochastic signals may enable to derive specific results for such networks once they are algebraically rearranged using the proposed method.

While deriving the equations, we have shown that the computability of the solution is in relationship with the network structure. Moreover, in Section V-B we have seen that the same network has a structural correspondence with the signal transduction network it originates from: this means that such a filter network might enable to conduct investigations on specific aspects of a biological model *directly on the computational scheme*, i.e., using this scheme not as a mere integrator, but, rather, as a complete *in silico* version of the model. Taken together, these two facts suggest that the proposed method may help disclose so-called *emergent* network properties, that are today intensively researched in computational biology for their connective role between structural characters and functional meaning of biological networks.

## ACKNOWLEDGMENT

The authors gratefully acknowledge M. Echenim for revising the English of the original manuscript.

## REFERENCES

[1] H. Kitano, "Computational systems biology," *Nature*, vol. 420, pp. 206–210, Nov. 2002.
[2] H. D. Jong, "Modeling and simulation of genetic regulatory systems: A literature review," *J. Comput. Biol.*, vol. 1, pp. 67–103, 2002.
[3] J. Szczupak and S. K. Mitra, "Detection, location, and removal of delay-free loops in digital filter configurations," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-23, no. 6, pp. 558–562, 1975.
[4] S. K. Mitra, *Digital Signal Processing. A computer-Based Approach*. New York: McGraw-Hill, 1998.
[5] F. Fontana, "Computation of linear filter networks containing delay-free loops, with an application to the waveguide mesh," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 6, pp. 774–782, Nov. 2003.
[6] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[7] C. Wan and A. M. Schneider, "Further improvements in digitizing continuous-time filters," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 533–542, Mar. 1997.
[8] A. Fettweis, "Wave digital filters: Theory and practice," *Proc. IEEE*, vol. 74, pp. 270–327, Feb. 1986.
[9] A. Sarti and G. D. Poli, "Toward nonlinear wave digital filters," *IEEE Trans. Signal Process.*, vol. 47, no. 6, pp. 1654–1668, Jun. 1999.
[10] G. Borin, G. D. Poli, and D. Rocchesso, "Elimination of delay-free loops in discrete-time models of nonlinear acoustic systems," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 5, pp. 597–605, 2000.
[11] L. O. Chua, "Nonlinear circuits," *IEEE Trans. Circuits Syst.*, vol. CAS-31, no. 1, pp. 69–87, Jan. 1984.
[12] T. Felderhoff, "Simulation of nonlinear circuits with period doubling and chaotic behavior by wave digital filter principles," *IEEE Trans. Circuits Syst.*, vol. 41, no. 7, pp. 485–491, Jul. 1994.
[13] J.-C. Leloup and A. Goldbeter, "A model for circadian rhythms in Drosophila incorporating the formation of a complex between the PER and TIM proteins," *J. Biologic. Rhythms*, vol. 13, no. 1, pp. 70–87, Feb. 1998.
[14] A. Goldbeter, "Computational approaches to cellular rhythms," *Nature*, vol. 420, pp. 238–244, Nov. 2002.
[15] G. Borin and G. D. Poli, "A hysteretic hammer-string interaction model for physical model synthesis," in *Proc. Nordic Acoustical Meeting (NAM)*, Helsinki, Finland, Jun. 1996, pp. 399–406.
[16] F. Avanzini and D. Rocchesso, "Efficiency, accuracy, and stability issues in discrete time simulations of single reed instruments," *J. Acoust. Soc. Amer.*, vol. 111, no. 5, pp. 2293–2301, May 2002.
[17] F. Avanzini, S. Serafin, and D. Rocchesso, "Interactive simulation of rigid body interaction with friction-induced sound generation," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 5, pt. 2, pp. 1073–1081, 2005.
[18] F. Fontana, F. Avanzini, and D. Rocchesso, "Computation of nonlinear filter networks containing delay-free paths," in *Proc. Conf. Digital Audio Effects (DAFx)*, Naples, Italy, Oct. 2004, pp. 113–118.
[19] F. Avanzini and F. Fontana, "Exact discrete-time realization of a Dolby B encoding/decoding architecture," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Montreal, Quebec, Canada, Sep. 18–20, 2006, pp. 297–302.
[20] A. Härmä, "Implementation of frequency-warped recursive filters," *EURASIP Signal Process.*, vol. 80, no. 3, pp. 543–548, Mar. 2000.
[21] R. Crochiere and A. Oppenheim, "Analysis of linear digital networks," *Proc. IEEE (Special Issue on Digital Signal Processing)*, vol. CAS-23, pp. 581–595, Apr. 1975.
[22] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. New York: McGraw-Hill, 1990.
[23] F. Avanzini, F. Fontana, and D. Rocchesso, "Efficient computation of nonlinear filter networks with delay-free loops and applications to physically-based sound models," in *Proc. 4th Int. Workshop Multidimensional Systems (NDS)*, Wuppertal, Germany, Jul. 2005, pp. 110–115.
[24] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems*. New York: Wiley, 1991.
[25] T. Kailath, A. H. Sayed, and B. Hassabi, *Linear estimation*. Upper Saddle River, NJ: Prentice-Hall, 2000.
[26] N. Barkal and S. Leibler, "Robustness in simple biochemical networks," *Nature*, vol. 387, pp. 913–917, Jun. 1997.
[27] T. Wilhelm, J. Behre, and S. Shuster, "Analysis of structural robustness of metabolic networks," *IEE Syst. Biol.*, vol. 1, no. 1, pp. 114–120, 2004.
[28] P. Atkins and J. de Paula, *Physical Chemistry*, 7th ed. Oxford, U.K.: Oxford Univ. Press, 2002.
[29] O. Wolkenhauer, M. Ullah, W. Kolch, and K.-H. Cho, "Modelling and simulation of intracellular dynamics: Choosing an appropriate framework," *IEEE Trans. Nano-Bioscience*, vol. 3, no. 3, pp. 200–207, Sep. 2004.
[30] T. E. Turner, S. Schnell, and K. Burrage, "Stochastic approaches for modelling *in vivo* reactions," *Comput. Biol. Chem.*, vol. 2004, no. 28, pp. 165–178, 2004.
[31] F. Fontana and V. Manca, "Discrete solution of differential equations by metabolic P systems," *Theor. Comput. Sci.*, vol. 372, no. 2–3, pp. 165–182, Mar. 2007.
[32] R. Zou and A. Ghosh, "Automated sensitivity analysis of stiff biochemical systems using a fourth-order adaptive step size Rosenbrock integration method," *IEE Syst. Biol.*, vol. 153, pp. 78–90, 2006.
[33] D. Gonze, J. Halloy, J.-C. Leloup, and A. Goldbeter, "Stochastic models for circadian rhythms: Effect of molecular noise on periodic and chaotic behaviour," *Comptes Rendus Biologies*, no. 326, pp. 189–203, 2003.
[34] A. Goldbeter, *Biochemical Oscillations and Cellular Rhythms. The Molecular Bases of Periodic and Chaotic Behaviour*. New York: Cambridge Univ. Press, 2004.
[35] D. Rocchesso, R. Bresin, and M. Fërnstrom, "Sounding objects," *IEEE Multimedia*, vol. 10, no. 2, pp. 42–52, Apr.–Jun. 2003.

**Federico Fontana** (M'03) received the Laurea degree in electronic engineering from the University of Padova, Italy, in 1996 and the Ph.D. degree in computer science from the University of Verona, Italy, in 2003.

From 1998 to 2000, he was collaborating with the Department of Information Engineering at the University of Padova, working on sound synthesis and spatialization by physical modeling. During the same period, he was consulting for national and international companies and research centers, in the design and realization of real-time music and audio processing algorithms. In 2001, he was visiting the Laboratory of Acoustics and Audio Signal Processing at the Helsinki University of Technology, Espoo, Finland. Since 2005, he has been an Assistant Professor in the Department of Computer Science, University of Verona. He has been involved in several national and international research projects. His interests concern algorithms for sound synthesis and processing, sonic interaction design, and computational models of biological signal transduction networks.

**Federico Avanzini** received the Laurea degree in physics from the University of Milano, Milan, Italy, in 1997 and the Ph.D. degree in computer science from the University of Padova, Italy, in 2001 with a research project on sound and voice synthesis by physical modeling.

Within his doctoral activities, he has worked as a visiting researcher at the Laboratory of Acoustics and Audio Signal Processing, Helsinki University of Technology. From 2002 to 2004, he was a Postdoctoral Researcher in the Department of Information Engineering of the University of Padova, where he has been an Assistant Professor since 2005, teaching courses in computer science and sound and music computing. His main research interests concern algorithms for sound synthesis and processing, nonspeech sound in multimodal interfaces, voice production, and articulatory modeling. He has been involved in numerous national and European projects (FP5, FP6), as well as industry-funded projects. His homepage is at http://www.dei.unipd.it/~avanzini