A Universal Concept Based on Cellular Neural Networks for Ultrafast and Flexible Solving of Differential Equations

Jean Chamberlain Chedjou and Kyandoghere Kyamakya

***Abstract*— This paper develops and validates a comprehensive and universally applicable computational concept for solving nonlinear differential equations (NDEs) through a neurocomputing concept based on cellular neural networks (CNNs). High-precision, stability, convergence, and lowest-possible memory requirements are ensured by the CNN processor architecture. A significant challenge solved in this paper is that all these cited computing features are ensured in all system-states (regular or chaotic ones) and in all bifurcation conditions that may be experienced by NDEs. One particular quintessence of this paper is to develop and demonstrate a solver concept that shows and ensures that CNN processors (realized either in hardware or in software) are universal solvers of NDE models. The solving logic or algorithm of given NDEs (possible examples are: Duffing, Mathieu, Van der Pol, Jerk, Chua, Rössler, Lorenz, Burgers, and the transport equations) through a CNN processor system is provided by a set of templates that are computed by our comprehensive templates calculation technique that we call nonlinear adaptive optimization. This paper is therefore a significant contribution and represents a cutting-edge real-time computational engineering approach, especially while considering the various scientific and engineering applications of this ultrafast, energy-and-memory-efficient, and high-precise NDE solver concept. For illustration purposes, three NDE models are demonstratively solved, and related CNN templates are derived and used: the periodically excited Duffing equation, the Mathieu equation, and the transport equation.**

***Index Terms*— Cellular neural networks (CNNs)-based neurocomputing, CNN-based ultrafast solving of nonlinear differential equations (NDEs), CNN processor concept as a universal differential equation model solver, nonlinear adaptive optimization (NAOP).**

NOMENCLATURE

$\vec{x}$	Vector of decision variables.
$\vec{\lambda}$	Vector of multiplier variables.
$f(\vec{x})$	Objective function.
$g_a(\vec{x})$	Constraints functions.
$L(\vec{x}, \vec{\lambda})$	Lagrange function.
t	Time.
$\dot{x}_i$	First derivative of x_i with respect to t.
$\ddot{x}_i$	Second derivative of x_i (inertia).
∂_x	Partial derivative with respect to x.
α, β	Learning rate parameters.
D_{ij}	Damped mass matrix.
$\vec{F}$	Force producing the potential energy.
E_P	Potential energy.
E_C	Kinetic energy.
E_T	Total energy.
u_j	Input of the CNN-cell with index j.
A_{ij}	Elements of the state controlled template A.
B_{ij}	Elements of the feedback template B.
C_{ij}	Elements of feedforward template C.
$\vec{I}(I_1, \ldots, I_n)$	Vector of the thresholds of CNN-cells.
$\vec{Y}(y_1^*, \ldots, y_n^*)$	Vector of the sigmoid functions (SFs).
$\vec{U}(u_1, \ldots, u_n)$	Vector of the inputs of CNN-cells.
α_j^*	Parameters for monitoring the shape of SF.
$\vec{\Theta}(x_1, \ldots, x_n)$	Vector flow of the CNN-model.
$\vec{\Phi}(y_1, \ldots, y_n)$	Vector flow of the equation to be solved by CNN.
$F_i(\vec{\Phi}, t)$	Nonlinear functions of Φ and t.
$\varepsilon(t)$	Step-function.
$\delta(t)$	Dirac function.
$x_i(0), y_i(0)$	Initial conditions/states.
$\nabla\chi_j(t)$	Relative error of the learning process.
$\nabla\chi_{\text{opt}}^*$	Minimum of the relative error.
t^*	Time at which the relative error is minimized.
$\|x\|$	Absolute value of x.
$\vec{\lambda}$	Vector of multiplier variables
$\lambda_{\max}$	Lyapunov exponent (LE).
i	Index of both rows and decision variables.
j	Index of columns.
a	Index of multiplier variables.
M	Number of constraints.
n	Order of the equation to be solved by CNN.

Manuscript received April 23, 2013; revised January 22, 2014, April 14, 2014, and April 24, 2014; accepted April 30, 2014. Date of publication June 6, 2014; date of current version March 16, 2015.

The authors are with the Transportation Informatics Group, Institute of Smart Systems Technologies, University of Klagenfurt, Klagenfurt 9020, Austria (e-mail: kyandoghere.kyamakya@aau.at; jean.chedjou@aau.at).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2323218

I. INTRODUCTION

IN VARIOUS areas of science and engineering, solving nonlinear differential equation (NDE) system models is a very crucial task in related system modeling, simulations. and/or optimizations endeavors [1].

Traditional requirements while solving NDE models are evidently high precision, sure, and stable convergence, the highest

possible computing speed, and the lowest-possible memory requirement [2]. All these features do result and contribute, in the best-case, to a cost- and energy-efficient solving of underlying NDE system models. Such requirements can only be fulfilled if a given solver concept or paradigm does fully master the related computational complexity while preserving a consistent complexity-related scalability. Both practice and literature clearly show and underscore that the traditional numerical methods do not fully satisfy these relatively hard and challenging requirements [3].

A core hypothesis of this paper is that cellular neural network (CNN) processor concepts and models that may be realized either in hardware (hard-core/hard-coded version or digitally emulated version) or in software are the neurocomputing-based solver platform of predilection that offers the full potential of scalability, flexibility, and high-speed while ensuring sure convergence and high-precision.

This paper is organized as follows. Section II presents an overview of the classical methods for solving complex ODEs. Some pros and cons of the classical methods are discussed. Section III does focus on the complexity analysis of CNN processor systems. We provide some seminal references addressing this issue, which is of necessary importance when dealing with the simulation of complex differential equation models through CNN processors. Section IV describes the general methodology for solving differential equation models through CNN processors as a universal solver machine. We provide an in-depth description of all steps involved in the application of the nonlinear adaptive optimization (NAOP) concept (this is the new concept developed in this paper based on NAOP) for CNN templates calculation and solution of complex ODEs and PDEs. Sections V–VII do then focus on the proof of concept through two selected examples of nonlinear ODEs and one example of PDE that are solved using the novel concept developed in this paper: the periodically excited Duffing oscillator (Section V), the forced Mathieu equation with time varying coefficients (Section VI), and the transport equation (Section VII). For each of these equations, corresponding precise templates are calculated through NAOP. Several concluding remarks are presented in Section VIII along with the presentation of some interesting open research questions (as an outlook) that are under investigation in some of our on-going works.

II. Related Works and Applications Background

During the last couple of decades both scientific computing and the so-called computational engineering have experienced a very strong boom that has been particularly also sustained and motivated by the ever increasing performance of computing systems. Thus, a huge research has been producing since then various contributions related to the development of both analytical and numerical methods and computing concepts/platforms (either digital or analog) for solving nonlinear and even stochastic ordinary differential equations (ODEs) system models [ODEs, partial differential equations (PDEs), stochastic differential equations, and stochastic PDEs (SPDEs)]. The interest devoted to solving these complex system models (either solving in any case and/or speeding-up the solving) is explained by their various scientific, engineering and industrial applications. Examples of applications can be found in various areas such as intelligent transportation systems [4], image processing, wireless communications [5], mechatronics [6], control systems [7], medical image processing [8], clinical diagnosis, computer graphics, aeroacoustics, aerodynamics, electromagnetics [9], and quantum mechanics [10], just to name a few.

However, despite the interesting and extensive involvement of complex differential equation models in different areas of science and engineering the traditional methods (algorithms and computing schemes) to solve or simulate them fail to cope with a series of issues related to the high complexity. The relevant literature clearly suggests that the traditional solving and computing approaches (for stiff/stochastic ODEs and PDEs) are very slow [11], generally less precise [12], and not always robust enough [13]. Furthermore, those traditional paradigms are computationally expensive and not capable of realizing real-time computation at an acceptable and realistic performance/cost ratio. Of course, not all applications do require a real-time (ultrafast) processing. Nevertheless, it makes a big difference if a simulation takes two weeks or rather 2 min. Beyond that, many applications are really emerging and for which a really real-time (ultrafast) computation is needed with deadlines of a couple of seconds, whereby the current state-of-the-art just enables a best-case solving within many hours or days. Such a problem formulation calls for speed-ups of many orders of magnitudes, and this is knowingly not trivially reachable by the current instruments of the related state-of-the-art (while maintaining the same level of precision) [14]. Even the most recent advances in parallel computing high performance computing (HPC) do not succeed in breaking the so-called Amdahl law and thus still suffer from this clear and crucial limitation [15].

Thus, developing a reliable methodical instrumentation that enables a real-time (ultrafast) and cost-effective solving of such complex system models problem formulations does constitute a real and interesting breakthrough that has a tremendous scientific and market potential.

In practice, some parameters of NDE models of real systems may be stochastic [16], time-varying [17], space varying [18], or even spatio-temporally varying [19]. Deriving exact analytical solutions of these equations is very tough and practically impossible [20]. Numerical solutions are possible, but not for all levels of nonlinearity [21]. Furthermore, some forms of nonlinear (integro-) differential equations (systems) are not yet numerically solvable [22]. However, even for the cases for which numerical solutions are available many of the methods are still exposed to transient phenomena [23], stiffness [24], overflows, and round-off errors during computations [25]. They are also often very time-consuming [26] due to a generally nonpolynomial complexity. Furthermore, in some cases there is no *a priori* convergence guarantee. It is therefore clear that numerical solutions in this context are computationally expensive (very slow) and the convergence of the solutions is often very difficult/questionable, for example, under complex Dirichlet boundary conditions [27].

Overall, the classical numerical methods fail to cope with the high complexity. Thus, traditional numerical approaches and processing schemes are not capable of realizing accurate and ultrafast/real-time solving of stiff ODEs and/or PDEs at an acceptable and realistic cost.

A series of works has explored the better promising (compared with numerical approaches) paradigm of solving differential equations through neural networks, and this in various architectures [28], [29]. The major limitation of all those concepts is the lack of a systematic approach for training the neural network for any given underlying differential equation model (ODE or PDE) corresponding to a given problem. Furthermore, there are clear limitations related to the integration interval. Neither comprehensive concepts nor clear details have been provided so far by the relevant literature regarding these issues.

Concerning CNN processors-based solvers of differential equations, they do (among neuronal network architectures) offer the biggest potential due to the following interesting properties: 1) they are dynamical systems in form of coupled nonlinear oscillators [30]; 2) their related easy and straightforward implementability and/or emulation either in software or in digital hardware [31]; 3) their further property of being a universal machine platform [32]; and 4) their proven property of being a supercomputer on chip [33].

III. On the Complexity Analysis of a Cellular Neural Networks' Processors System as Universal Solver Machine

One basic assumption of this paper is that both computational and computing complexity of a problem solving through a CNN processor system is one of the actually best ones. This assumption is supported by the extensive related works in [34]–[36].

IV. General Methodology for Solving Differential Equation Models Through CNNs Processors as Universal Solver Machine

In essence, this section does address all key steps (see Section IV-A–E) involved in the overall process of solving ODEs and PDEs by CNNs.

A. Remodeling Procedure

The key issue is to explain how to perform a transformation of a given model available in form of (ordinary or partial) differential equations to prepare it for a solving by a CNN processors system. This is an analytical process that can eventually be supported by symbolic mathematical instruments for large problem sizes; for small problem sizes, a manual transformation is straight-forward.

1) Regarding ODE: the solutions of this equation are dependent variables [e.g., $y(t)$]. These solutions are generally expressed as functions of a single independent variable (e.g., t). The total derivative of the dependent variables is introduced with respect to the single independent variable. Furthermore, the ODE is originally provided in the general form of an eventually n^{th}– order equation, which should therefore be rewritten (or transformed) in the simplified form of a set of coupled first-order ODEs

$$\begin{cases} y_1^{(1)} = F_1(\vec{\Phi}, t) \\ \quad \cdots \\ y_1^{(n)} = F_n(\vec{\Phi}, t) \end{cases} \tag{1}$$

$\vec{\Phi}(y_1, \ldots, y_n) = \vec{\Phi}\left(y_1, \ldots, y_1^{n-1}\right)$ is the vector flow representing the state variables of the nth– order ODE in (1).

2) Regarding PDE: the solutions of this equation are dependent variables [e.g., $y(x, t)$]. These solutions are generally expressed as functions of several independent variables (e.g., x, t). The partial derivatives of the dependent variables are introduced with respect to all independent variables, which are generally spatiotemporal variables (i.e., variation in both space and time). Thus, all partial derivatives with respect to the space-dimensions are approximated (see application example in Section VII) using the finite difference method (FDM). This approximation leads to a set of coupled first-order ODEs similar to (1).

B. Model-Mapping Procedure

This section explains the procedure of mapping the transformed system model in (1) to the one expressing the dynamics of a corresponding CNN processors system. The result is, to name it in an abstract way, the CNN processors architecture. The mapping is expressed in the form of a nonlinear optimization problem, which does in essence express the fact that two dynamical system models should display the same behavior under all contextual conditions. This is also an analytical process that can eventually be supported by symbolic mathematical instruments for large problem sizes. The outcome of the model-mapping procedure is the derivation of the Lagrange function, which represents the total energy of the system. The overall procedure is organized around the following steps.

Step 1: This step provides a brief explanation of the general theory of optimization and presents the general form of the Lagrange function. The Lagrange function is expressed as a combination of both the objective function and the related constraints. The objective function is expressed by (2). Thereby, $\vec{x} = [x_1, x_2, \ldots, x_i]^T \in \Re$ denotes the vector of decision variables (also called decision neurons) of the optimization process and $f(\vec{x})$ is a function of these variables. The constraints are expressed by (3) in which $g_a(\vec{x})$ is a function of the decision variables. Furthermore, the Lagrange function (4) is expressed as a combination of the objective function and the related constraints. Thereby, $\vec{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_a]^T \in \Re$ is the vector of multiplier variables (also called multiplier neurons) of the process under optimization. The parameters i and a (4) are, respectively, the indexes of decision variables and multiplier variables. M is the number of constraints

$$\text{Min} f(\vec{x}) \tag{2}$$

$$g_a(\vec{x}) = 0 \tag{3}$$

$$L(\vec{x}, \vec{\lambda}) = f(\vec{x}) + \sum_{a}^{M} [\lambda_a g_a(\vec{x})]. \tag{4}$$

Step 2: This step presents the mathematical expression of the CNNs model (used in the mapping procedure) and provides an explicit definition of all parameter settings of the CNN-model. This model is represented mathematically by the expression of the state-controlled CNN processor architecture as given by

$$\frac{dx_i}{dt} = -x_i + \sum_{j=1}^{n} \left[A_{ij} x_j + B_{ij} y_j^* + C_{ij} u_j \right] + I_i \quad (5a)$$

where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, n$. The function y_j^* is an approximation of the basic and well-known piecewise linear (PWL) sigmoid function (SF) proposed in [37]. The advantages of using y_j^* (5b) instead of the PWL SF in [37] are threefold. The first advantage is the possibility of performing fast computing of the first derivative of the SF. Indeed, the basic differential multiplier method (BDMM) algorithm (Section IV-C) requires computation of the first derivative of the SF. However, the computation of the first derivative of the PWL SF in [37] is very time- and memory-consuming due to its PWL form. Therefore, the function y_j^* is used as an alternative solution to perform fast computations of the first derivative of the SF with high approximation accuracy and very low memory requirements. The second advantage is the smoothness of y_j^* as this feature is not a characteristic to the PWL function in [37]. This smoothness leads to decreasing accumulation of round-off errors during computation and consequently improves both speed and accuracy of computations. The third advantage is the offered possibility of using the coefficient α_j^* as a decision variable of the optimization process to determine the suitable shape of y_j^* according to a given ODE or PDE under investigation. This underscores the flexibility of the optimization concept developed in this paper. This flexibility leads to increasing accuracy in computation

$$y_j^* = \frac{e^{\alpha_j^* x_j} - 1}{e^{\alpha_j^* x_j} + 1}. \quad (5b)$$

Equation (5a) and (5b) can be rewritten into the following equivalent vector form:

$$\frac{d\vec{\Theta}}{dt} = -\vec{\Theta} + A^*\vec{\Theta} + B^*\vec{Y} + C^*\vec{U} + \vec{I}. \quad (5c)$$

The function $\vec{\Theta}(x_1, \ldots, x_n) = \vec{\Theta}(x_1, \ldots, x_1^{n-1})$ is the vector flow, which components x_i represent the states of all elementary cells involved in the full CNN-processor model. Therefore, according to (5c) the full CNN-processor model is a network of coupled first-order ODEs. In (5c), the parameters A, B, and C are the matrices (called CNN-templates) that are expressed in terms of their respective elements A_{ij}, B_{ij}, and C_{ij}.

Step 3: An in-depth explanation of the mapping-procedure of (1) into (5c) is provided and, the resulting Lagrange function is derived as the mathematical expression of the mapping-procedure. The overall mapping-procedure is conducted as follows. The ODE under investigation (1) is represented in the state space (or phase space) in terms of the components of Φ. Similarly, the dynamics of CNN is represented in the state space in terms of the components of Θ. The objective function is derived based on the condition that a best-possible mapping must be achieved between the mathematical ODE model under investigation (1) and the CNN model (5c). Equivalently to this statement, the two vector flows Φ and Θ must evolve always on a common trajectory in the phase space representation. Therefore, the objective function $f(\vec{x}, \vec{y}, t)$ can be expressed by

$$f(\vec{x}, \vec{y}, t) = \text{Min}\left[\sum_{j=1}^{n} (x_j(t) - y_i(t))^2 \right]. \quad (6)$$

Equation (6) is not a combinatoric optimization, rather a functional optimization over time. This equation expresses the minimization of the global distance between the vector flows $\vec{\Phi}$ and $\vec{\Theta}$. According to (6), the two inputs ($\vec{\Phi}$ and $\vec{\Theta}$) are compared and should be as equal as possible for all t and for all state variable dimensions j. The general modeling principle can be summarized (using the physical principles) by the following two key requirements.

1) Minimization of the global distance between the vector flows $\vec{\Phi}$ and $\vec{\Theta}$ (6).
2) The minimum of the global distance between $\vec{\Phi}$ and $\vec{\Theta}$ must be equal to zero (for all t and for all state variable dimensions j) to insure that $\vec{\Phi}$ and $\vec{\Theta}$ are always identical (same trajectory).

Regarding the first requirement, the minimum [defined in (6)] is not necessarily equal to zero because we are dealing with the optimization of complex functions undergoing nonlinear and chaotic dynamics over time. The second requirement (to be achieved) imposes a pairwise equality between the individual components of $\vec{\Phi}$ and $\vec{\Theta}$. This is an important condition for $\vec{\Phi}$ and $\vec{\Theta}$ to evolve on a common trajectory. The pairwise equality observed (for all t and for all state variable dimensions j) between the components of $\vec{\Phi}$ and $\vec{\Theta}$ is modeled by the constraints formulated in

$$(x_j(t) - y_j(t)) = 0. \quad (7a)$$

According to [38] and [39], the improvement of the robustness of the convergence properties of the BDMM [to tackle the problem formulated in (6) and (7a)] is insured by the augmented Lagrange method. This method is derived from (6) and (7a) by introducing an additive penalty force expressed in the form of a quadratic energy. This justifies the constraints formulated in

$$(x_j(t) - y_j(t))^2 = 0. \quad (7b)$$

Combining the objective function with the related constraints leads to the augmented Lagrange expression formulated in (8). The expression of $L(\vec{\Phi}, \vec{\Theta}, \lambda_j, \gamma_j)$ depends implicitly on A_{ij}, B_{ij}, C_{ij} and α_j^* (5)

$$\begin{aligned} &L(x_j, y_j, A_{ij}, B_{ij}, C_{ij}, \alpha_j, \lambda_j, \gamma_j) \\ &\quad = \sum_{j=1}^{n} (x_j - y_j)^2 + \sum_{j=1}^{n} \lambda_j (x_j - y_j) + \sum_{j=1}^{n} \gamma_j (x_j - y_j)^2 \end{aligned} \quad (8)$$

The quantities x_j, y_j, A_{ij}, B_{ij}, C_{ij} and α_j^* are the decision variables/neurons and, λ_j and γ_j are multiplier variables.

C. Neuroprocessor Training Procedure and Derivation of the Corresponding Mathematical Model

This section provides a brief explanation of the neuro-processor training procedure and discusses the related challenges encountered by the traditional neuro-computing methods for solving ODEs and/or PDEs. We end this section by deriving the general mathematical expression of the resulting neuro-processor model.

In essence, the key issue is how to efficiently solve the model-mapping problem, which has been expressed in form of a complex nonlinear optimization problem (in Section IV-B). The result is, to name it in an abstract way, the CNN processors configuration provided by a series of matrices called templates. In this phase, the issue is to solve the model-mapping problem. This problem can be solved either numerically (NAOP presented in this paper is an example of a numerical solution) or using alternative selected instruments from soft-computing that are particle swarm optimization, answer-set programming, or using digitally emulated analog computer(s).

The neurocomputing paradigm (e.g., neural networks) has been already used for solving differential equations. However, already published works are mainly just focusing on proof-of-concept examples and do not solve a series of fundamental issues related among others to a systematic methodological framework for efficiently solving any given complex problem setting. An open challenge is related to suggesting and validating a successful and systematic concept for controlling and predicting the six performance key features that are: precision, speed of computation, memory requirements, energy consumption (of the computing process), stability, and robustness. The theoretical analysis of both stability and bifurcation scenarios help to depict, control and predict the various states (regular and irregular states) of the novel CNN-based computing concept. The possibility of controlling these states is a key innovation. This possibility is not offered by the traditional related processing/computing paradigms for solving NDEs. Furthermore, to the best of our knowledge, no comprehensive and extensive benchmarking has been presented so far in the literature.

We now explain the general theory leading to the derivation of the mathematical model of the neuroprocessor. This theory does fully exploit the neuron-dynamics process as an optimization strategy that maps the optimization problem into the energy expression of a neural network (i.e., a Hopfield network) to find the optimal solution. In this context, this energy is expressed into the Lagrange form (8) and the minimization of the Lagrange function leads to a stable state. The stability (see stability or convergence analysis in subsection E) in this context expresses the robustness of the optimization process as such. This robustness is characterized by a straightforward (monotone) convergence to the optimal solution. This convergence is achieved (in this paper) by involving the so-called BDMM [40], [41], which is a combination of two gradient-techniques: 1) the first technique is based on gradient decent. Here, the state variables of the network slide downhill, in opposition to the gradient, to find the minimum of the function and 2) the second technique is applying the gradient ascent. In this case, the maximum function is obtained by evolving in the positive direction of the gradient. We apply BDMM to the Lagrange function in (8) to obtain (9). Specifically, the gradient descent is applied on decision neurons and the gradient ascent is applied on multiplier neurons

$$\begin{aligned} \dot{x}_i &= -\alpha\, \partial_{x_i}(L) \\ \dot{\lambda}_a &= +\beta\, \partial_{\lambda_a}(L). \end{aligned} \tag{9}$$

The relation/expression (9) is the characteristic model of BDMM. This model (from which the CNN-templates for a given ODE or PDE are calculated) implicitly reveals how far the dynamics of decision neurons (x_i) is coupled to the dynamics of multiplier-neurons (λ_a). The parameters α and β are the step sizes for updating the decision- and multiplier-neurons, respectively.

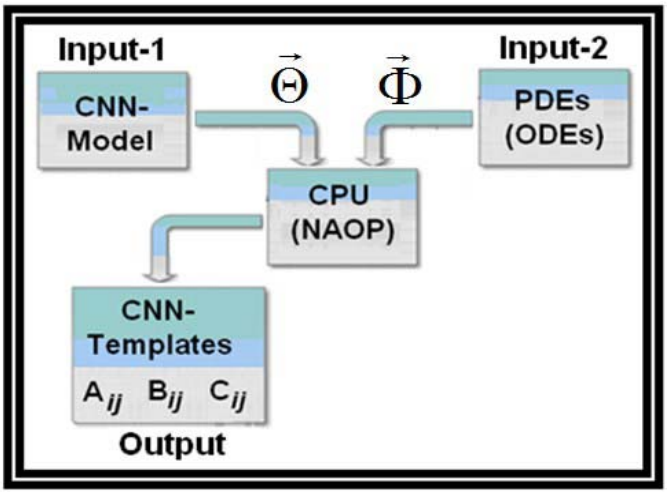
D. Novel Concept NAOP for CNN Templates Calculation

The template calculation process is in the core based on the NAOP concept. The strengths of this concept are its robustness and flexibility, as well as its efficiency of deriving the CNN templates in all states (regular and chaotic states) of the nonlinear ODEs or PDEs under investigation. The analytical conditions for controlling and predicting the robustness of the NAOP concept are derived in Section IV-E.

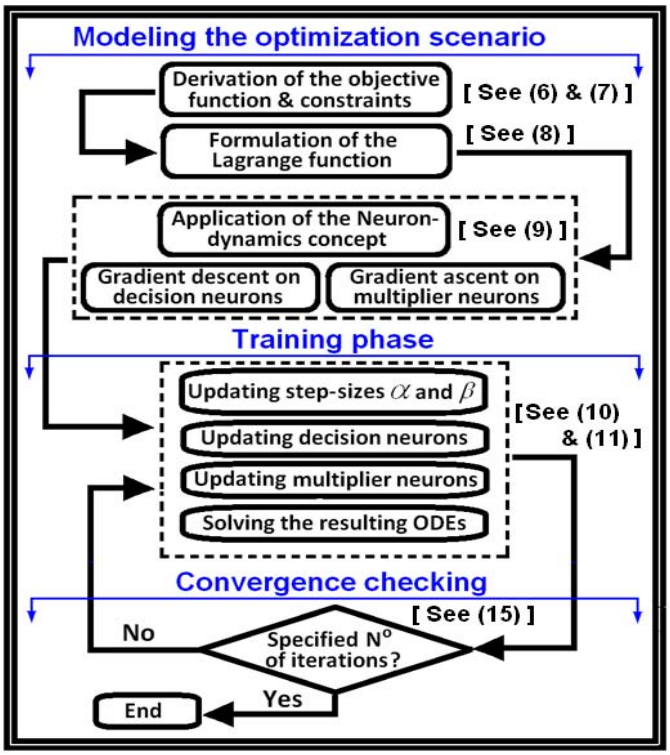
We now provide an in-depth description of the NAOP concept in the frame of a CNN-based solving of NDEs. The complete flow diagram of this approach is schematically presented in Fig. 1(a) and (b). NAOP is performed by a complex computing procedure that does work on two basic inputs vectors ($\vec{\Theta}$ and $\vec{\Phi}$), as shown in Fig. 1(a). Using the defined inputs vectors, the output of the NAOP procedures does provide, after extensive iterative computations or training steps [see the details of these steps in Fig. 1(b)] the appropriate CNN templates to solve the corresponding Input-2 of Fig. 1(a) ODEs/PDEs expressed in the form of (1) as soon as the convergence of the training process is achieved. In NAOP, the convergence to local and/or global minima is the key purpose governing the template calculation process, the very global aim being the tracking and finding of the global minimum independently of basins of attraction.

Using the bifurcation theory, various basins of attraction are investigated sequentially in NAOP and corresponding CNN templates are determined. The basin of attraction expresses the sensitivity of a system/model to changes in the initial states/conditions. If (for a given basin of attraction) some local attractors diverge from the expected global minimum, new sets of initial conditions are automatically generated to annihilate the divergence leading to a possible convergence to the expected global minimum. This global minimum is the (unique) solution of the mapping-based training process which generates CNN templates corresponding to the real solution of the ODEs or PDEs under investigation, Input-2 of Fig. 1(a) [see also (1)].

In NAOP, a large number of randomly generated attractors [either regular or chaotic, as shown in Fig. 3(a) and (b)] are obtained through various iteration steps, whereby each



(a)



(b)

Fig. 1. (a) Illustration of the mapping concept of two inputs leading to CNN templates calculation by the novel optimization technique NAOP. (b) Synoptic representation of the key steps involved in the complete NAOP process leading to the derivation of CNN templates for solving a given ODE.

attractor corresponds to a specific set of CNN templates. An attempt to map these attractors to those generated by the model (i.e., ODE or PDE under investigation) is performed (through the temporal mapping in the nth– dimensional phase space of (5c) and (1) by comparing the respective dynamics of both the CNN model (5c) and the ODE/PDE under investigation (1). This mapping is performed in a sequential process [as shown in Fig. 1(b)] leading to the convergence to the expected global minimum when the mapping is achieved successfully. However, it is worth mentioning that during the training process the various numerical trials have revealed that it is always possible and straightforward to find the optimal solution (i.e., the expected global minimum) using NAOP. This is a strong point of the concept developed in this paper as it is well known that many optimization concepts presented in the relevant literature are subjected to a key weakness (i.e., the difficulty to achieve convergence) due to the well-known inherent local minima problem of the Hopfield neural networks [42].

NAOP has been demonstrated to be capable of mapping several forms (or types) of nonlinear ODEs/PDEs into CNN models by deriving the corresponding appropriate templates. The general mathematical model of the NAOP processor is obtained by substituting the Lagrange function (expressed in terms of both decision and multiplier variables/neurons) denoted by $L(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j)$ into (9). This substitution leads to the expressions in (10). The NAOP concept is thus supported by the set of coupled nonlinear equations in (10) from which the CNN-templates are calculated. Thus, the full training phase is supported by

$$\dot{x}_j = G_j(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j) \quad (10a)$$

$$\dot{y}_j = P_j(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j) \quad (10b)$$

$$\dot{A}_{ij} = Q_l(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j) \quad (10c)$$

$$\dot{B}_{ij} = R_l(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j) \quad (10d)$$

$$\dot{C}_{ij} = S_l(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j) \quad (10e)$$

$$\dot{\alpha}_j^* = T_j(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j) \quad (10f)$$

$$\dot{\lambda}_j = V_j(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j) \quad (10g)$$

$$\dot{\gamma}_j = W_j(x_j,\ y_j,\ A_{ij},\ B_{ij},\ C_{ij},\ \alpha_j^*,\ \lambda_j,\ \gamma_j) \quad (10h)$$

In equations (10a)–(10h), the quantities G_j, P_j, Q_l, R_l, S_l, T_j, V_j, and W_j are the nonlinear functions. The indexes are defined as follows: $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, and $l = 1, 2, \ldots, n^2$. These indexes are chosen according to the order of a given ODE under investigation by CNN. An example for illustration: if we are considering a second-order ODE to be solved by CNN, the appropriate index is $n = 2$. Thus, (10) is made up of at least 22 nonlinear and coupled first-order ODEs since some additional constraints (e.g., over-dimensioning, etc.) can be considered (if necessary) to improve the robustness of the optimization process and facilitate the convergence to the global minimum as well as to improve the accuracy (or precision) of results.

E. Training and Convergence of the NAOP Concept

The key important issue in this context is how to control the training, robustness and convergence of the NAOP concept. The training phase and convergence checking are clearly shown in Fig. 1(b).

Regarding the training process, this phase is conducted by updating the step sizes of decision- and multiplier-neurons. The accuracy of the training based on NAOP is obtained by comparing $\Theta(x_1, \ldots, x_n)$ with $\Phi(y_1, \ldots, y_n)$. The metric of

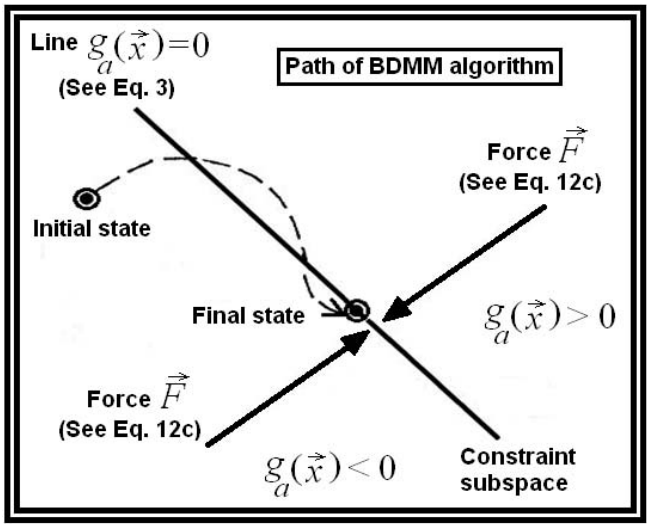


Fig. 2. Sign flip concept is illustrated as a key step toward the convergence of the BDMM algorithm. Damped oscillations are exhibited around the constraint subspace leading to constraints minimization (system's final state).

the comparison is the relative error ($\nabla_{\chi_j}(t)$) expressed in

$$\nabla\chi_j(t) = \sum_{j=1}^{n} \left| \frac{x_j(t) - y_j(t)}{y_j(t)} \right|. \tag{11}$$

The relative error $\nabla_{\chi_j}(t)$ expresses the rate of divergence between the two trajectories described by the vector flows. A computation of the relative error has revealed the existence of several minima of $\nabla_{\chi_j}(t)$ each of which corresponds to a specific set of CNN-templates. Therefore the optimal (or plausible) set of CNN-templates is obtained at a specific time t^* corresponding to the global minimum of $\nabla_{\chi_j}(t)$ denoted $\nabla^*_{\chi_{\text{opt}}}$. Equation (11) is used to sort the suitable CNN-templates among the huge amount of solutions provided at each time/iteration t by (10).

Regarding the convergence, the stability (or convergence) of the optimization process is ensured by the sign flip concept. Fig. 2 shows the core idea explaining the sign flip concept. This concept is summarized as follows. As the parameters α and β increase, the system in (9) exhibits damped oscillations around the constraints subspace $g_a(\vec{x}) = 0$. These oscillations are characterized by the alternation (in time-domain) of the sign of the constraint function (i.e., $g_a(\vec{x}) < 0$, $g_a(\vec{x}) > 0$ and vice-versa) during the optimization process to converge to a point located on the constraint subspace defined by $g_a(\vec{x}) = 0$ (Fig. 2). The damped mass matrix D_{ij} is used to provide an insight of the above-mentioned convergence. Indeed, the damped mass matrix controls the energy dissipation within the system. Thus, according to (15) a positive value of the damped mass matrix is an insight that the total energy E_T within the system decreases with time and finally, the system settles down into the state where the energy is minimized. This characterizes an attraction of the system state to the constraint subspace (stability). At the attraction point, or convergence state (see Fig. 2), all constraints are fulfilled. However, when the damped mass matrix is negative the total energy within the system increases with time. A negative damped mass matrix is an insight that all constraints are not fulfilled (instability). To tackle this problem, the learning rate parameters (α and β) and the system's parameters can be tuned/monitored to bring the system into its stable state. A brief summary of the key analytical steps involved in the stability/convergence analysis is as follows. Equations (4) and (9) are used to establish the second-order nonlinear ODE in

$$\ddot{x}_i + \sum_j D_{ij}\dot{x}_j + \vec{F} = 0 \tag{12a}$$

$$D_{ij} = \alpha \left[\frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_a \left(\lambda_a \frac{\partial^2 g_a(\vec{x})}{\partial x_i \partial x_j} \right) \right] \tag{12b}$$

$$\vec{F} = (2\alpha\beta) \sum_a \left(g_a(\vec{x}) \frac{\partial g_a(\vec{x})}{\partial x_i} \right) \tag{12c}$$

where $\ddot{x}_i$ is an inertial force, D_{ij} is a damping matrix, and $\vec{F}$ is an internal force needed to fulfill the constraints (Fig. 2). This force produces the potential energy into the system. The total energy E_T of the system is expressed as the sum of both kinetic- and potential-energies

$$E_T = E_C + E_P \tag{13a}$$

Considering the state variable x_i in (12), the energies E_C and E_P are expressed as follows:

$$E_P(x_i) = \int F\, \partial x_i, \tag{13b}$$

$$E_C(x_i) = \sum_i \left[\frac{1}{2} (\dot{x}_i)^2 \right] \tag{13c}$$

Thus, substituting (13b) and (13c) into (13a) leads to the expression of the total energy in

$$E_T(x_i) = \sum_i \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \int F\, \partial x_i. \tag{13d}$$

Combining (12c) and (13d) leads to

$$E_T(x_i) = \sum_i \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \int \left(2\alpha\beta \sum_a \left[g_a \frac{\partial g_a}{\partial x_i} \right] \right) \partial x_i. \tag{13e}$$

Equation (13e) can be written into the following simplified form:

$$E_T(x_i) = \sum_i \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \alpha\beta \sum_a \left[\int \left(\frac{\partial g_a^2}{\partial x_i} \right) \partial x_i \right] \tag{13f}$$

Finally, an evaluation of the integral part/term in (13f) leads to the following expression of the total energy:

$$E_T(x_i) = \sum_i \left[\frac{1}{2} (\dot{x}_i)^2 \right] + \alpha\beta \sum_a \left[g_a^2 \right]. \tag{14}$$

The expression (14) is used to derive (15). Equation (15) is a key expression of the time derivative of the total energy in the system. This expression is the characteristic formula used to illustrate and underscore the significance of the sign flip

$$\dot{E}_T(x_i) = -\sum_i \sum_j \dot{x}_i D_{ij} \dot{x}_j. \tag{15}$$

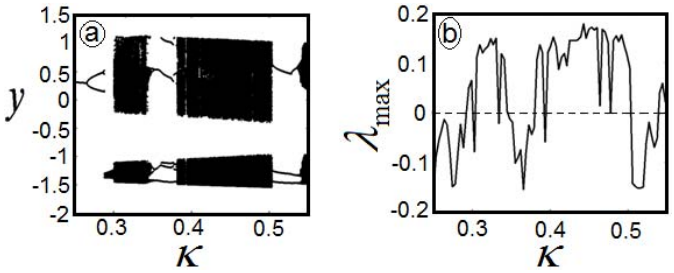


Fig. 3. (a) Bifurcation diagram. (b) Lyapunov Exponent (LE). (a) and (b) States of the system in (16) when monitoring the parameter κ between [0.25 and 0.55]. Windows of regular states alternate with windows of chaotic states. Our work in [45] provides the full detail.

Thus, if the damped mass matrix D_{ij} is positive, then the time derivative of the total energy is negative and the system converges to fulfill both constraints and objective function. The essence of this principle is being exploited and realized by the NAOP concept under design in this paper. NAOP is used to compute the CNN templates corresponding to the system-mapping for a set of given NDEs under consideration for solving through CNN processors.

In the following sections, the NAOP concept is used to solve three differential equation models (ODEs and PDEs) for illustration. Furthermore, a benchmarking is considered as proof of concepts to validate the concept developed in this paper.

V. Applying NAOP to Solving the ODE Model of the Periodically Excited Duffing Oscillator With a Double-Well Potential

A. Model Development and Parameter Settings

The ODE model of a periodically excited Duffing oscillator with double-well potential is considered for solving by CNN

$$\ddot{y} + \mu_1 \dot{y} - \omega_1^2 y + c_0 y^3 = \kappa \sin(\Omega t + \varphi). \tag{16}$$

The model in (16) describes real physical systems such as a Bunckle beam, mechanical vibrations in physical structures, or the well-known Sommerfeld effect [43], [44], just to name a few. μ_1 stands for the dissipative coefficient, ω_1 denotes the natural frequency of the forced oscillator, c_0 is the cubic nonlinearity, κ represents the amplitude of excitation, Ω is devoted to the frequency of external excitation, and φ stands for the initial phase of excitation. For the sake of generalization of the NAOP concept, (16) is transformed into the following form:

$$\begin{aligned} \dot{y}_1 &= y_2 + [\delta(t) * y_1(0)] * \varepsilon(t) \\ \dot{y}_2 &= -\mu_1 \dot{y}_1 + \omega_1^2 y_1 - c_0 y_1^3 \\ &\quad + [\delta(t) * y_2(0) \quad + \kappa \sin(\Omega t + \varphi)] * \varepsilon(t). \end{aligned} \tag{17}$$

Equations (5) and (17) are further used to establish the mathematical model of the corresponding Lagrange function as expressed in (8). Thus, substituting this function into (9) leads to the derivation of a set of coupled first-order nonlinear ODEs [as expressed in (10)] from which the CNN templates are calculated through NAOP.

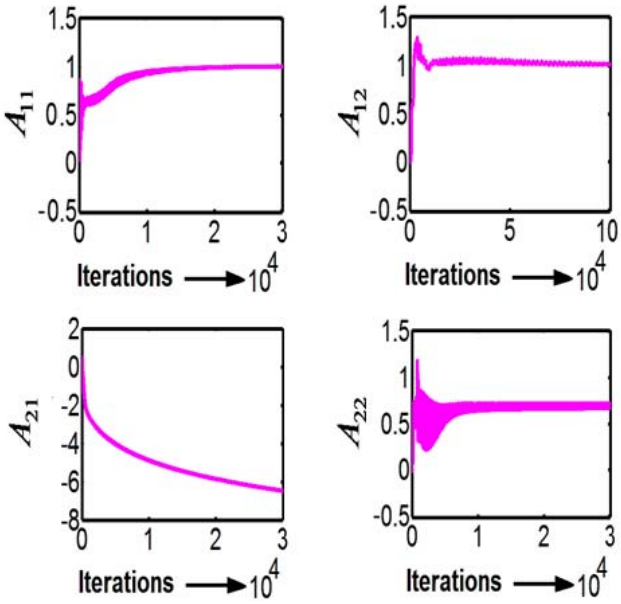


Fig. 4. Evolution and convergence of the control CNN-templates A_{ij} for (16).

To derive the appropriate CNN templates corresponding to (16), we now consider for a first calculation exercise the following parameter settings: $\mu_1 = 0.3$, $\omega_1 = 1$, $c_0 = 1$, $\Omega = 1.2$, and $\varphi = \pi/2$. The coefficient κ is the control parameter of the bifurcation analysis. The results of this analysis are shown in Fig. 3. This figure reveals the possible states of the system modeled by (16). The existence of regular and chaotic dynamics is confirmed by the bifurcation diagram in Fig. 3(a) and the corresponding $\lambda_{\max}$ in Fig. 3(b). Chaotic states are observed for positive values of $\lambda_{\max}$ while the negative values of $\lambda_{\max}$ correspond to regular states. Some values of $\lambda_{\max}$ are shown in Table I together with the corresponding states. According to the results in Table I, the optimization procedure is carried out in both regular and chaotic states. Using the parameter settings considered for the first calculation exercise, the NAOP is exploited to calculate the corresponding CNN templates as well as the corresponding values of the coefficients α_j^*, which are considered as decision variables during the optimization process. The template calculation process has revealed the dependence of these templates with regards to time/iterations (Fig. 4). According to Fig. 4, each iteration results/leads to the calculation of a new combination of templates. Thus the sorting process is considered to determine the appropriate setting of templates corresponding to a given problem. The appropriate templates that are obtained after convergence of the training process represent the CNN signature of the autonomous part of the model in (16). The appropriate CNN-templates are obtained at a global minimum point $(t^*, \nabla\chi^*_{\text{opt}})$. Thus, for the first calculation exercise, it has been obtained that the global minimum point corresponds to $(t^*, \nabla\chi^*_{\text{opt}}) = (312\,388,\ 13\ 10^{-5})$. The corresponding CNN-templates are: $A_{11} = 0.9995$, $A_{12} = 0.9997$, $A_{21} = -6.5552$, $A_{22} = 0.7081$, $B_{11} = 0$, $B_{12} = 0$, $B_{21} = 9.9085$, $B_{22} = 0$, $C_{11} = -0.0005$, $C_{12} = 0$,

TABLE I
BIFURCATION STATES IN RELATION TO κ AND RESPECTIVE VALUES OF THE CORRESPONDING LE (λ_{max}) DEDUCED FROM FIG. 3

Figures	Ctrl. Parameter.	Respective LE	Bifurcation types
Fig. 5(a)	κ =0.250	λ_{max} = -0.1545	Regular
Fig. 5(b)	κ =0.300	λ_{max} = +0.0650	Chaos
Fig. 6(a)	κ =0.450	λ_{max} = +0.1685	Chaos
Fig. 6(b)	κ =0.510	λ_{max} = -0.1483	Regular

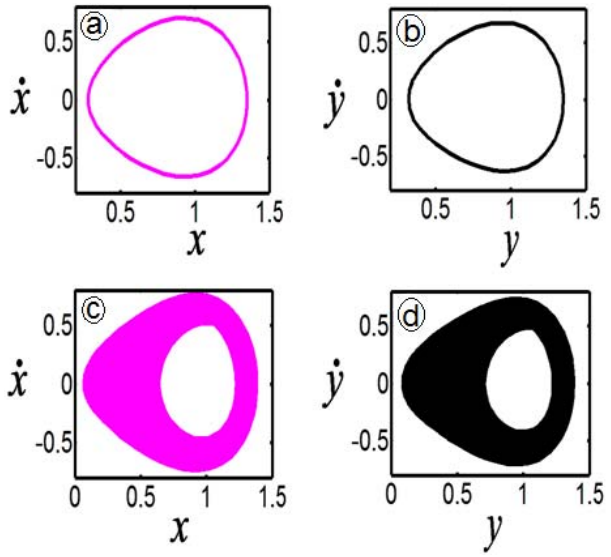


Fig. 5. (a) and (c) Phase portraits obtained as numerical solutions of (16) using the CNN-model in (5a). (b) and (d) Corresponding phase portraits obtained as direct numerical solutions of (16). (a) and (b) $\kappa = 0.250$. (c) and (d) $\kappa = 0.300$. The other parameters are defined in the text (see Section V).

$C_{21} = 0.9957$, $C_{22} = 0$, $I_1 = 0$, $I_2 = 0$, $\alpha_1^* = 1.5675$, and $\alpha_2^* = 0$. These CNN-templates are, for verification and validation, further inserted in the CNN-model in (5a).

The phase portraits shown in Fig. 5(a), (c), (e), and (g) are obtained as solutions of (16) using the CNN-model in (5a). The control parameter κ is monitored/varied in the window $[0.25, 0.55]$. Figs. 5 and 6 confirm the extreme sensitivity of the model in (16) to tiny variations of the amplitudes κ of the external excitation as it is clearly shown in Fig. 3. Indeed, several bifurcation scenarios are depicted such as periodic bifurcation ($\kappa = 0.25$), quasi-periodic bifurcations ($\kappa = 0.275$, $\kappa = 0.29$, $\kappa = 0.366$, $\kappa = 0.510$) and chaotic bifurcations ($\kappa = 0.30$, $\kappa = 0.45$, $\kappa = 0.55$). The phase portraits in Figs. 5(b), 5(d), 6(f), and 6(h) are obtained from the direct numerical simulation of (16). The possible states of the system are defined in Table I in terms of the control parameter κ. Both regular states (i.e., periodic, quasi-periodic, and torus) and chaotic states are observed.

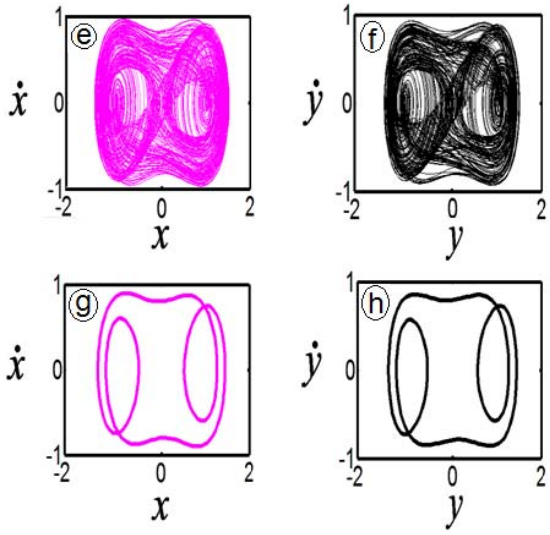


Fig. 6. (e) and (g) Phase portraits obtained as numerical solutions of (16) using the CNN-model in (5a). (f) and (h) Corresponding phase portraits obtained as direct numerical solutions of (16). (e) and (f) κ=0.450. (g) and (h) $\kappa = 0.510$. The other parameters are defined in Fig. 5.

It is worth mentioning that we have derived a unique/fixed set of CNN-templates from which all the bifurcation sequences in Figs. 5 and 6 are obtained. This is another strong point (i.e., a significant contribution) of the NAOP concept presented in this paper.

Overall, the strong point of NAOP (this is one of the significant contributions of this paper) is the straightforward, rapid, and easy possibility of detecting the global minimum (i.e., the unique/optimal solution) among the huge amount of local minima depicted during the optimization process. Furthermore, the results shown in Figs. 5 and 6 demonstrate, confirm and validate the great and straightforward possibility of synthesizing nonlinear regular and chaotic oscillators using the novel NAOP-concept involving CNN-based solving of differential equations as developed in this paper.

B. Benchmarking of the NAOP-Concept

Now, a validation exercise of the NAOP concept is conducted through a comparison of the results provided by the CNN-model in (5a) defined by templates obtained by NAOP (method-1) with the results of a direct numerical solving by MATLAB of (16) (method-2). Using the same parameter-setting values, the corresponding phase portraits obtained from the first method are shown in Figs. 5(a), 5(c), 6(e), and 6(g) while the results from the second method are shown in Figs. 5(b), 5(d), 6(f), and 6(h). According to Figs. 5 and 6, a straightforward qualitative judgment of the results provided by the two methods reveals that both concepts (CNN versus direct MATLAB solving) do provide the same bifurcation (i.e., same qualitative changes in the dynamics of a system). There is clearly a good qualitative agreement between the two methods. Furthermore, the quantitative analysis of the results provided by the two methods does confirm the good agreement between them.

VI. Applying NAOP to Solving a Nonlinear and Time-Varying ODE: Illustration of the Ultrafast Potentialities of the CNN-Processor

A. Model Development and Parameter Settings

This section considers a nonlinear oscillator described by the second-order nonlinear and time-varying ODE model in

$$\ddot{y} + \kappa_1 \left[\frac{(e^{c_1 y} - 1)}{(e^{c_1 y} + 1)}\right] + \left[\omega_1^2 + \kappa_0 \sin(\omega_0 t)\right] y - [\kappa_2 \sin(\omega_2 t)]\dot{y} - \kappa_3 \sin(\omega_3 t + \varphi) = 0 \quad (18)$$

where κ_1 and c_1 are coefficients of the nonlinear term, κ_0 and κ_2 stand for the amplitude of the time-varying coefficients, and κ_3 denotes the amplitude of external excitation. ω_1 stands for the natural frequency of the oscillator, ω_0 and ω_2 denote the frequencies of the time-varying coefficients, and ω_3 represents the frequency of external excitation.

We now use the technique that has been intensively developed in Section V to derive the appropriate CNN templates corresponding to (18). Using the parameter values $\kappa_1 = -2$, $c_1 = 2$, $\omega_1 = 1$, $\kappa_2 = 0.01$, $\omega_2 = 1$, $\kappa_3 = 0.5$, $\omega_3 = 1$, $\varphi = 2n\pi$, $\kappa_0 = 0.01$, and $\omega_0 = 15$, the NAOP is exploited to calculate the corresponding CNN templates. A full explanation of the training process based on the NAOP concept has been provided in the preceding Section V (see first application example). Considering (18), this process has been exploited to derive the following corresponding CNN templates obtained at the global minimum point $(t^*,\ \nabla\chi^*_{\text{opt}}) = (78\,052,\ 3.047410^{-7})$: $A_{11} = 1.0015$, $A_{12} = 0.9391$, $A_{21} = -0.9636$, $A_{22} = 0.9974$, $B_{11} = 6.571710^{-4}$, $B_{12} = 0.2380$, $B_{21} = 1.8677$, $B_{22} = 0.0386$, $C_{11} = 0.0048$, $C_{12} = 0.0048$, $C_{21} = 0.5017$, $C_{22} = 0.5017$, $\alpha_1^* = 2.0995$, $\alpha_2^* = 0.5798$. These CNN templates are, for verification and validation, further inserted in the CNN model in (5a) and the phase portraits are considered. The phase portraits shown in Fig. 7 are obtained using both the CNN model in (5a) [Fig. 7(a) and (c)] and the direct numerical simulation of (18) using MATLAB [Fig. 7(b) and (d)]. The depiction of two different phase portraits under the same parameter settings is a proof that the system undergoes a long transient phase. This transient phase is characterized by the growing of the amplitude of oscillations in time-domain and consequently the depiction of different phase portraits under/for the same parameter settings and same initial conditions (i.e., $x(0) = y(0) = 1$, and $\dot{x}(0) = \dot{y}(0) = 1$).

B. Benchmarking of the NAOP-Concept

The purpose of the benchmarking is to compare simulation duration (in seconds) taken by each approach until one reaches (or can see) a complete cycle of the phase portrait. We should notice that in general there is a transient phase that can be long for some approaches until one reaches the target phase portrait. The duration of the transient phase is therefore a key feature of a given solving approach that determines the needed simulation duration until the target is reached.

Fig. 7(a) and (c) [respectively, Fig. 7(b) and (d)] shows the phase portraits obtained using two different methods under the same parameter settings. As it appears in Fig. 7, the two methods provide the same phase portraits. However, the difference between the two methods is obtained with regards to the computing speed.

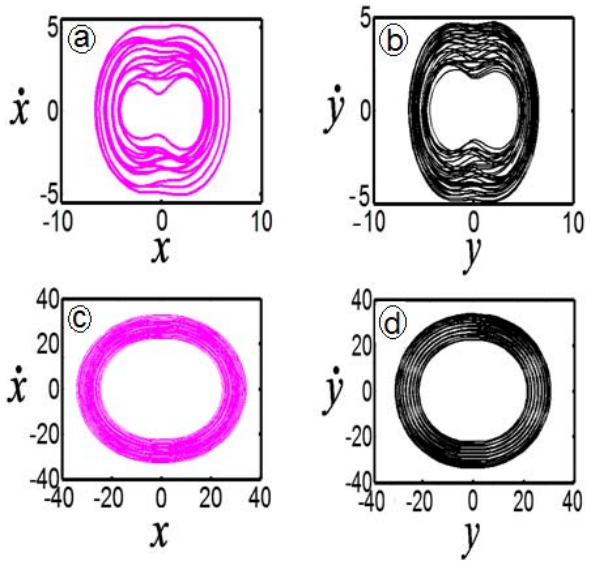


Fig. 7. (a) and (c) Phase portraits obtained as numerical solution of (18) using the CNN-model in (5a). (b) and (d) Corresponding phase portraits obtained as direct solution of (18). The parameter settings are defined in the text (Section VI).

Regarding the phase portraits in Fig. 7(a) and (b), the corresponding simulation durations are $T_{\text{Sim}}(\text{CNN}) = 0.232237s$ for the simulation of (5a) using the CNN processor [Fig. 7(a)] and $T_{\text{Sim}}(\text{Direct}) = 5.597485s$ for the direct numerical simulation of (18) using MATLAB [Fig. 7(b)].

It is worth mentioning that the simulation durations above have been measured on the same computer-environment using the MATLAB commands tic and toc.

The MATLAB solver used is the ODE23t and the simulation time-intervals (domains of iterations) within which the phase portraits in Fig. 7(a) and (b) are obtained correspond to $[B_{\min}, B_{\max}] = [0, 120]$ and $[B_{\min}, B_{\max}] = [0, 2830]$ for Fig. 7(a) and (b), respectively. Let us mention that (in the case of direct numerical simulation) it is not possible to obtain the graph in Fig. 7(b) when simulating in the interval $[B_{\min}, B_{\max}] = [0, 2800]$. Therefore, while using the direct numerical simulation to obtain the target phase portrait the lower boundary of the simulation interval must be chosen such that $B_{\min} > 2800$. This observation is due to the presence of the transient phase. The duration of this transient phase is longer when using the direct numerical simulation (method-2) while it is very short when performing the simulation using CNN (method-1). Thus, according to the observed simulation durations measured above, the CNN processor for solving (18) is approximately 24 times faster than the direct numerical simulation in MATLAB, $T_{\text{Sim}}(\text{Direct}) \approx 24 * T_{\text{Sim}}(\text{CNN})$. Such a difference in computation speed (or speed-up) is very remarkable, especially while considering that traditional (HPC) or other related schemes cannot reach such a performance easily.

Also, the phase portraits in Fig. 7(c) and (d) are obtained using the same parameter settings values and the same initial conditions as in Fig. 7(a) and (b). In the case of Fig. 7(c),

the simulation time interval is $[B_{\min}, B_{\max}] = [0, 1800]$ while in the case of Fig. 7(d) the simulation time interval is $[B_{\min}, B_{\max}] = [0, 13\,830]$. Mention that it is not possible to obtain the graph in Fig. 7(d) when simulating in the interval $[B_{\min}, B_{\max}] = [0, 13\,800]$. Therefore, the lower boundary of the simulation interval must be chosen such that $B_{\min} > 13\,800$.

Overall, using the two different methods (CNN versus direct-simulation) on the same computing environment, and the same programming language, it has been demonstrated that the method based on the CNN paradigm is faster than its counterpart. Furthermore, it has been demonstrated that identical phase portraits are obtained. This confirms the effectiveness of applying the novel method (i.e., NAOP) developed in this paper as a universal solver of nonlinear and chaotic ODEs.

The following section does present a demonstration of the applicability as well as the effectiveness of the NAOP concept for solving PDEs.

VII. Applying NAOP to Solving PDEs

A. Model Development and Parameter Settings

This section demonstrates the applicability of the NAOP scheme for solving PDEs. This scheme is an excellent numerical solver of PDEs as it can be designed to share a common property with a given PDE. This property emphasizes the strong dependence of the dynamical behavior of both NAOP and PDEs to only their spatial local interactions. This statement expresses the possibility of approximating PDEs on a finite spatial grid by NAOP using cloning templates (i.e., space invariant templates) [32]–[35]. Thus, a given PDE (under investigation) is discretized in space leading to a set of coupled ODEs. The NAOP concept is further applied to derive the CNN-templates corresponding to the set of coupled ODEs. The process/principle/procedure leading to the respective CNN templates calculation is then the same, as in Sections V and VI.

We now consider for illustration the transport equation (19). This is a well-known prototype of PDE which leads to various potential applications in the fields of transportation, chemistry, and semiconductor physics, just to name a few [46], [47]

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2} - v\frac{\partial u(x,t)}{\partial x}. \tag{19}$$

Equation (19) is defined for $t > 0$ and $0 < x < L$. $u(x,t)$ is the concentration (or density), D is the diffusion coefficient, v is the velocity, and x is the coordinate in the direction of flow. The initial condition is $u(x,0) = u_i$ and the boundary conditions are $u(0,t) = u_0$ and $u(L,t) = u_L$.

The partial derivatives of $u(x,t)$ with respect to x can be approximated using the FDM

$$\left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} \tag{20a}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+1} - 2u_i - u_{i-1}}{(\Delta x)^2} \tag{20b}$$

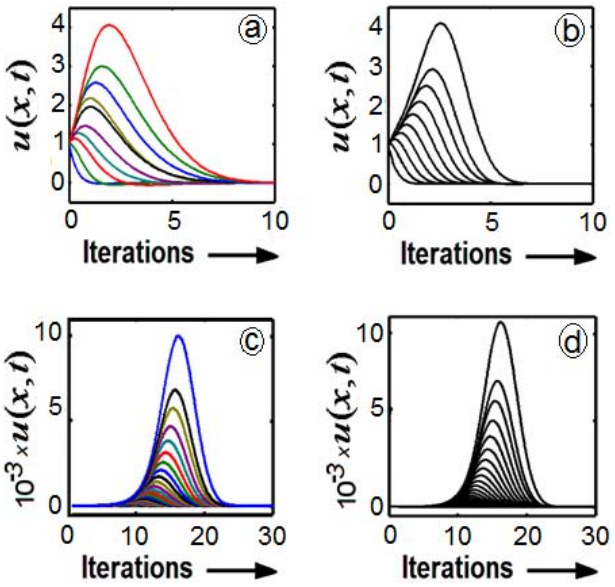


Fig. 8. (a) and (c) Numerical solutions of (21) using the CNN-model in (5a). (b) and (d) Corresponding numerical solutions obtained as a direct numerical solution of (18). The parameter settings are defined in the text (see Section VII). (a) and (b) Obtained using 10 grid-points. (c) and (d) Obtained using 50 grid-points.

Substituting (20) into (19) leads to the set of coupled ODEs in (21). i is the index of grid-points

$$\frac{du_i}{dt} = \xi_1 u_{i+1} + \xi_2 u_i + \xi_3 u_{i-1}. \tag{21a}$$

The parameters ξ_1, ξ_2, and ξ_3 are defined as follows:

$$\xi_1 = \left[\frac{D}{(\Delta x)^2} - \frac{v}{(2\Delta x)}\right] \tag{21b}$$

$$\xi_2 = \left[-\frac{2D}{(\Delta x)^2}\right] \tag{21c}$$

$$\xi_3 = \left[\frac{D}{(\Delta x)^2} + \frac{v}{(2\Delta x)}\right]. \tag{21d}$$

We now use the NAOP technique (see Section V) to derive the appropriate CNN templates corresponding to (21). The parameter values used are $\xi_1 = -0.35$, $\xi_2 = -2$, $\xi_3 = 3$, and $u(x,0) = 1$. Two levels of discretization (or resolution) are considered along the x-axis (while keeping the same length). The first level uses 10 grid points and the second level uses 50 grid points. Each grid point is represented/modeled by a first-order ODE. For each level of discretization the NAOP has been exploited to calculate the CNN templates corresponding to (21).

While considering the discretization into 10 grid points, the corresponding CNN-templates A (state controlled template) and B (feedback template) have been obtained and are shown at the top of the next page. The feedforward template C (or input matrix) is zero (i.e., $C_{ij} = 0$) and $\alpha_i^* = 2$.

These CNN templates are, for verification and validation, further inserted in the CNN model in (5a) and the solutions of (21) are considered (method 1). The solutions displayed in Fig. 8(a) and (c) are obtained using the CNN-model in (5a) while Fig. 8(b) and (d) stand for the corresponding solutions

$$A = \begin{bmatrix} -1.254 & -0.140 & -0.048 & -0.010 & 0.0002 & 0.0016 & 0.0010 & 0.0001 & -0.001 & -0.001 \\ 2.1862 & -0.517 & -0.251 & -0.077 & 0.1003 & 0.0160 & 0.0132 & 0.0056 & -0.001 & -0.007 \\ 1.4255 & 0.5873 & 0.0004 & -0.138 & 0.0107 & -0.049 & -0.010 & 0.0072 & 0.0111 & 0.0098 \\ 1.0498 & 0.5873 & 0.1407 & 0.0551 & 0.0832 & -0.094 & -0.058 & -0.020 & 0.0053 & 0.0254 \\ 0.1600 & 0.2583 & 0.8653 & 0.1330 & 0.0779 & -0.046 & -0.073 & -0.053 & -0.019 & 0.0232 \\ 0.3093 & 0.6351 & 0.6943 & 0.5271 & 0.2859 & 0.0875 & -0.024 & -0.060 & -0.049 & -0.003 \\ 0.1448 & 0.4314 & 0.6059 & 0.5968 & -0.145 & 0.2529 & 0.0821 & -0.025 & -0.072 & -0.063 \\ 0.0594 & 0.2573 & 0.7537 & 0.5564 & 0.5293 & 0.3944 & -0.216 & 0.0503 & -0.071 & -0.016 \\ 0.0188 & 0.1589 & 0.5330 & 0.1265 & 0.5254 & 0.4792 & 0.3319 & 0.1377 & -0.061 & -0.013 \\ 0.0009 & 0.0912 & 0.1227 & 0.1386 & 0.5130 & 0.5621 & 0.6509 & 0.3290 & 0.0804 & 0.1062 \end{bmatrix}_{A_{ij}}$$

$$B = \begin{bmatrix} -0.139 & -0.031 & -0.041 & -0.023 & -0.010 & -0.002 & 0.0005 & 0.0009 & 0.0006 & 0.0002 \\ 0.1510 & -0.178 & -0.109 & -0.093 & -0.059 & -0.025 & -0.004 & 0.0040 & 0.0052 & 0.0040 \\ 0.1824 & 0.1589 & 0.1363 & 0.0683 & -0.002 & -0.035 & -0.038 & -0.026 & -0.012 & -0.002 \\ 0.2199 & 0.1882 & 0.1828 & 0.1536 & 0.0963 & 0.0334 & -0.009 & -0.024 & -0.024 & -0.014 \\ 0.2102 & 0.1823 & 0.1767 & 0.1670 & 0.1409 & 0.0983 & 0.8509 & 0.0139 & -0.007 & -0.013 \\ 0.1653 & 0.181 & 0.1589 & 0.1543 & 0.1444 & 0.1244 & 0.0947 & 0.0608 & 0.0293 & 0.0401 \\ 0.0993 & 0.1504 & 0.1420 & 0.1356 & 0.1305 & 0.1221 & 0.1076 & 0.0882 & 0.0619 & 0.0401 \\ 0.0428 & 0.1284 & 0.1279 & 0.1196 & 0.1127 & 0.1063 & 0.0983 & 0.0874 & 0.0700 & 0.0561 \\ 0.0073 & 0.1015 & 0.1139 & 0.1079 & 0.0972 & 0.0847 & 0.0714 & 0.0590 & 0.0456 & 0.0483 \\ -0.011 & 0.0759 & 0.1063 & 0.1124 & 0.1059 & 0.0863 & 0.0540 & -0.016 & -0.038 & -0.183 \end{bmatrix}_{B_{ij}}$$

obtained by the direct numerical simulation of (21) using MATLAB (method-2).

Overall, the solutions in Fig. 8 reveal a very good qualitative and quantitative agreement between the results provided by the two methods. Furthermore, Fig. 8 reveals the apparition of new modes (i.e., new solutions) when increasing the number of grid-points. These modes show resonance dynamics whereby the values of top points increase with the increasing index of grids. It should be mentioned that similar results (i.e., numerical solutions of the transport equation) were reported in [46] and [47].

VIII. Conclusion

The novel concept presented in this paper is particularly challenging. It does demonstrate a systematic and straightforward way to solve nonlinear ODEs using the CNN paradigm. The key challenge has been the possibility and then the appropriate way and algorithmic path for mapping the system-model of a given nonlinear ODEs unto the system model expressed by a CNN-processor system.

Furthermore, the approach developed in this paper is very flexible as it is applicable for solving various types and forms of nonlinear ODEs and/or PDEs. The NAOP concept has been shown to be central as it enables a systematic straightforward computing of necessary templates for a given ODE system-model. Two illustrative ODE and one PDE examples have been provided in this paper.

Using NAOP, the calculation of respective CNN templates for solving some of the classical and well-known ODE models (i.e., Rayleigh, Lorenz, and Rössler equations, etc.) is straightforward. Of particular interest and worth a mentioning is that the same templates provide the model solutions for all possible bifurcation states, regular, or chaotic.

This paper has presented and validated a comprehensive theoretical and practical concept based on the CNN paradigm for enabling an ultrafast, potentially low-cost, energy-efficient, high-precision, and flexible solving of nonlinear ODEs and PDEs. Since all ODEs can be solved by the CNN processors-based solver independently of both their form and related nonlinearity type, the quintessence of this paper does evidently represent a significant scientific achievement.

All CNN related features related to ultrafasting computing are the foundation for enabling a really real-time (ultrafast) computational engineering. A referencing of seminal works on the complexity analysis related to CNN as a universal machine has been provided in this paper as well.

The benefits of a CNN-based differential equations solver under all possible bifurcation contexts are already visible and significant, and this independently of the effective CNN implementation in hardware: n-CPU, DSP, field programmable gate array, Graphics Processing Unit (GPU), n-CPU+GPU, or on a very large scale integration analog CNN-Chip.

References

[1] W. Wiechert, "The role of modeling in computational science education," *Future Generat. Comput. Syst.*, vol. 19, no. 8, pp. 1363–1374, Nov. 2003.

[2] M. Thongmoon and R. McKibbin, "A comparison of some numerical methods for the advection-diffusion equation," *Res. Lett. Inf. Math. Sci.*, vol. 10, no. 1, pp. 49–62, Mar. 2006.

[3] D. Shieh, Y. Chang, and G. R. Carmichael, "The evaluation of numerical techniques for solution of stiff ordinary differential equations arising from chemical kinetic problems," *Environ. Softw.*, vol. 3, no. 1, pp. 28–38, Mar. 1988.

[4] D. Helbing, A. Hennecke, V. Shvetsov, and M. Treiber, "Micro- and macro–simulation of freeway traffic," *Math. Comput. Model.*, vol. 35, nos. 5–6, pp. 517–547, Mar. 2002.

[5] Y. Chen, B. Yang, Q. Meng, Y. Zhao, and A. Abraham, "Time-series forecasting using a system of ordinary differential equations," *Inf. Sci.*, vol. 181, no. 1, pp. 106–114, Jan. 2011.

[6] J. Bocko, V. Nohajová, and T. Harčarik, "Symmetries of differential equations describing beams and plates on elastic foundations," *Proc. Eng.*, vol. 48, pp. 40–45, Nov. 2012.

[7] W. Heins and S. K. Mitter, "Conjugate convex functions, duality, and optimal control problems I: Systems governed by ordinary differential equations," *Inf. Sci.*, vol. 2, no. 2, pp. 211–243, Apr. 1970.

[8] J. Weickert, "Efficient image segmentation using partial differential equations and morphology," *Pattern Recognit.*, vol. 34, no. 9, pp. 1813–1824, Sep. 2001.
[9] J. L. Volakis, T. Özdemir, and J. Gong, "Hybrid finite-element methodologies for antennas and scattering," *IEEE Trans. Antennas Propag.*, vol. 45, no. 3, pp. 493–507, Mar. 1997.
[10] C. Chainais-Hillairet, M. Gisclon, and A. Jüngel, "A finite-volume scheme for the multidimensional quantum drift-diffusion model for semiconductors," *Numer. Methods Partial Differ. Equ.*, vol. 27, no. 6, pp. 1483–1510, Nov. 2011.
[11] A. Sóbester, P. B. Nair, and A. J. Keane, "Genetic programming approaches for solving elliptic partial differential equations," *IEEE Trans. Evol. Comput.*, vol. 12, no. 4, pp. 469–478, Aug. 2008.
[12] D. Göddeke, R. Strzodka, and S. Turek, "Performance and accuracy of hardware-oriented native-, emulated- and mixed-precision solvers in FEM simulations," *Int. J. Parallel Emergent Distrib. Syst.*, vol. 22, no. 4, pp. 221–256, Aug. 2007.
[13] M. A. Martinello, D. J. Muñoz, and S. A. Giner, "Mathematical modelling of low temperature drying of maize: Comparison of numerical methods for solving the differential equations," *Biosyst. Eng.*, vol. 114, no. 2, pp. 187–194, Feb. 2013.
[14] Y. Shi, W. H. Green, H.-W. Wong, and O. O. Oluwole, "Accelerating multi-dimensional combustion simulations using GPU and hybrid explicit/implicit ODE integration," *Combustion Flame*, vol. 159, no. 7, pp. 2388–2397, Jul. 2012.
[15] X.-H. Sun and Y. Chen, "Reevaluating Amdahl's law in the multicore era," *J. Parallel Distrib. Comput.*, vol. 70, no. 2, pp. 183–188, Feb. 2010.
[16] L. Li and J. Cao, "Cluster synchronization in an array of coupled stochastic delayed neural networks via pinning control," *Neurocomputing*, vol. 74, no. 5, pp. 846–856, Feb. 2011.
[17] F. D. O. Souza and R. M. Palhares, "Interval time-varying delay stability for neural networks," *Neurocomputing*, vol. 73, nos. 13–15, pp. 2789–2792, Aug. 2010.
[18] I. E. Lagaris, C. L. Likas, and D. G. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1041–1049, Sep. 2000.
[19] P. T. Vesanen *et al.*, "The spatial and temporal distortion of magnetic fields applied inside a magnetically shielded room," *IEEE Trans. Magn.*, vol. 48, no. 1, pp. 53–61, Jan. 2012.
[20] M. Pototschnig, J. Niegemann, L. Tkeshelashvili, and K. Busch, "Time-domain simulations of the nonlinear Maxwell equations using operator-exponential methods," *IEEE Trans. Antennas Propag.*, vol. 57, no. 2, pp. 475–483, Feb. 2009.
[21] I. Pollak, A. S. Willsky, and Y. Huang, "Nonlinear evolution equations as fast and exact solvers of estimation problems," *IEEE Trans. Signal Process.*, vol. 53, no. 2, pp. 484–498, Feb. 2005.
[22] P. G. Saffman, "Dynamics of vorticity," *J. Fluid Mech.*, vol. 106, pp. 49–58, May 1981.
[23] R. E. Ewing and H. Wang, "A summary of numerical methods for time-dependent advection-dominated partial differential equations," *J. Comput. Appl. Math.*, vol. 128, nos. 1–2, pp. 423–445, Mar. 2001.
[24] G. Naldi and L. Pareschi, "Numerical schemes for hyperbolic systems of conservation laws with stiff diffusive relaxation," *SIAM J. Numer. Anal.*, vol. 37, no. 4, pp. 1246–1270, Apr. 2000.
[25] K. Kalliojarvi and J. Astola, "Roundoff errors in block-floating-point systems," *IEEE Trans. Signal Process.*, vol. 44, no. 4, pp. 783–790, Apr. 1996.
[26] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.
[27] M. Kumar and N. Yadav, "Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey," *Comput. Math. Appl.*, vol. 62, no. 10, pp. 3796–3811, Nov. 2011.
[28] I. G. Tsoulos, D. Gavrilis, and E. Glavas, "Solving differential equations with constructed neural networks," *Neurocomputing*, vol. 72, nos. 10–12, pp. 2385–2391, Jun. 2009.
[29] J.-C. Ban, C.-H. Chang, and S.-S. Lin, "On the structure of multi-layer cellular neural networks," *J. Differ. Equ.*, vol. 252, no. 8, pp. 4563–4597, Apr. 2012.
[30] L. Pivka, C. W. Wu, and A. Huang, "Chua's oscillator: A compendium of chaotic phenomena," *J. Franklin Inst.*, vol. 331, no. 6, pp. 705–741, Nov. 1994.
[31] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, nos. 1–3, pp. 239–255, Dec. 2010.
[32] T. Roska and L. O. Chua, "The CNN universal machine: 10 years later," *J. Circuits, Syst. Comput.*, vol. 12, no. 4, pp. 377–388, Aug. 2003.
[33] L. O. Chua and T. Roska, "Reprogrammable CNN and supercomputer," U.S. Patent 5 355 528, Oct. 11, 1994.
[34] T. Roska, "Computational and computer complexity of analogic cellular wave computers," *J. Circuits, Syst. Comput.*, vol. 12, no. 4, pp. 539–562, Aug. 2003.
[35] L. O. Chua, "CNN: A vision of complexity," *Int. J. Bifurcation Chaos*, vol. 7, no. 10, pp. 2219–2425, Oct. 1997.
[36] P. Arena, M. Bucolo, S. Fazzino, L. Fortuna, and M. Frasca, "The CNN paradigm: Shapes and complexity," *Int. J. Bifurcation Chaos*, vol. 15, no. 7, pp. 2063–2090, Jul. 2005.
[37] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 35, no. 10, pp. 1257–1272, Oct. 1988.
[38] S. Z. Li, "Improving convergence and solution quality of Hopfield-type neural networks with augmented Lagrange multipliers," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1507–1516, Nov. 1996.
[39] F. M. Ham and I. Kostanic, *Principles of Neurocomputing for Science & Engineering*. New York, NY, USA: McGraw-Hill, 2001.
[40] J. C. Platt and A. H. Barr, "Constrained differential optimization for neural networks," Dept. Comput. Sci., California Inst. Technol., Pasadena, CA, USA, Tech. Rep. TR-88-17, Apr. 1988.
[41] Y.-F. Wang, J. B. Cruz, Jr., and J. H. Mulligan, Jr., "Multiple training concept for back-propagation neural networks for use in associative memories," *Neural Netw.*, vol. 6, no. 8, pp. 1169–1175, Jul. 1993.
[42] N. Smaoui and S. Al-Enezi, "Modelling the dynamics of nonlinear partial differential equations using neural networks," *J. Comput. Appl. Math.*, vol. 170, no. 1, pp. 27–58, Sep. 2004.
[43] K. S. McFall and J. R. Mahan, "Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions," *IEEE Trans. Neural Netw.*, vol. 20, no. 8, pp. 1221–1233, Aug. 2009.
[44] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 987–1000, Sep. 1998.
[45] J. C. Chedjou, H. B. Fotsin, P. Woafo, and S. Domngang, "Analog simulation of the dynamics of a van der Pol oscillator coupled to a Duffing oscillator," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 48, no. 6, pp. 748–757, Jun. 2001.
[46] S. T. Smith, R. O. Fox, and V. Raman, "A quadrature closure for the reaction-source term in conditional-moment closure," *Proc. Combustion Inst.*, vol. 31, no. 1, pp. 1675–1682, Jan. 2007.
[47] T.-W. Tang and H. Gan, "Two formulations of semiconductor transport equations based on spherical harmonic expansion of the Boltzmann transport equation," *IEEE Trans. Electron Devices*, vol. 47, no. 9, pp. 1726–1732, Sep. 2000.

Jean Chamberlain Chedjou received the B.S. degree in physics and the M.S. and Ph.D. degrees in electronics from the University of Yaoundé I, Yaoundé, Cameroon, in 1990, 1992, and 1999, respectively, and the Ph.D. degree in electrical engineering and information technology from the Leibniz University of Hanover, Hanover, Germany, in 2004. He is currently pursuing the Habilitation degree with the Institute for Smart Systems Technologies, Alpen-Adria University of Klagenfurt, Klagenfurt, Austria.

He was a part-time Assistant Lecturer with the Department of Physics, University of Yaoundé I, from 1992 to 1995, Assistant Lecturer (1996-1999) and Senior Lecturer (1999–2003 & 2006–2007) with the Laboratory of Electronics, University of Dschang, Cameroon. He was a Deutscher Akademischer Austauschdienst Dienst Scholar and a Research Fellow with the Institute of Communications Engineering, Leibniz University of Hanover, in 2003 and from 2003 to 2004, respectively. In 2004 and from 2005 to 2006, he was a Junior Researcher with the Abdus Salam International Centre for Theoretical Physics, Trieste, Italy. From 2005 to 2006, he was an Agence Universitaire de la Francophonie Research Fellow. He has authored and co-authored five books and more than 50 journals and conference papers. His current research interests include nonlinear dynamics in intelligent transportation systems (ITS), cellular neural networks and applications in ITS, electronics circuits engineering, chaos theory, analog systems simulation, synchronization and related applications in engineering.

Kyandoghere Kyamakya received the Ir. Civil and Ph.D. degrees in electrical engineering from the University of Kinshasa, Kinshasa, Congo, and the University of Hagen, Hagen, Germany, in 1990 and 1999, respectively.

He was a Post-Doctoral Researcher of Mobility Management in Wireless Networks with the Leibniz University of Hanover, Hanover, Germany, for three years, where he was also a Junior Professor of Positioning Location-Based Services from 2002 to 2005. Since 2005, he has been a Full Professor of Transportation Informatics and the Director of the Institute for Smart Systems Technologies at the University of Klagenfurt, Klagenfurt, Austria.