Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Physics Communications 171 (2005) 19–39
Computer Physics Communications
www.elsevier.com/locate/cpc

Deterministic event-based simulation of quantum phenomena

K. De Raedt [a], H. De Raedt [b,*], K. Michielsen [b]

[a] *Department of Computer Science, University of Groningen, Blauwborgje 3, NL-9747 AC Groningen, The Netherlands*
[b] *Department of Applied Physics, Materials Science Centre, University of Groningen, Nijenborgh 4, NL-9747 AG Groningen, The Netherlands*

Received 7 March 2005; accepted 15 April 2005

Available online 13 June 2005

Abstract

We propose and analyse simple deterministic algorithms that can be used to construct machines that have primitive learning capabilities. We demonstrate that locally connected networks of these machines can be used to perform blind classification on an event-by-event basis, without storing the information of the individual events. We also demonstrate that properly designed networks of these machines exhibit behavior that is usually only attributed to quantum systems. We present networks that simulate quantum interference on an event-by-event basis. In particular we show that by using simple geometry and the learning capabilities of the machines it is possible to simulate single-photon interference in a Mach–Zehnder interferometer. The interference pattern generated by the network of deterministic learning machines is in perfect agreement with the quantum theoretical result for the single-photon Mach–Zehnder interferometer. To illustrate that networks of these machines are indeed capable of simulating quantum interference we simulate, event-by-event, a setup involving two chained Mach–Zehnder interferometers, and demonstrate that also in this case the simulation results agree with quantum theory.

PACS: 02.70.-c; 03.65.-w

Keywords: Computer simulation; Machine learning; Quantum interference; Quantum theory

1. Introduction

Computer simulation is widely regarded as complementary to theory and experiment [1]. At present there are only a few physical phenomena that cannot be simulated on a computer. One such exception is the double-slit experiment with single electrons, as carried out by Tonomura and his co-workers [2]. This experiment is carried out in such a way that at any given time, only one electron travels from the source to the detector [3]. Only after a

* Corresponding author.
E-mail addresses: deraedt@cs.rug.nl (K. De Raedt), deraedt@phys.rug.nl (H. De Raedt), kristel@phys.rug.nl (K. Michielsen).
URL: http://www.compphys.org (H. De Raedt).

0010-4655/$ – see front matter
doi:10.1016/j.cpc.2005.04.012

substantial amount of electrons (approximately 50 000) have been detected an interference pattern emerges [2]. This interference pattern can be described by quantum theory. We use the term "quantum theory" for the mathematical formalism that gives us a set of algorithms to compute the probability for observing a particular event [4–6]. Of course, the quantum-mechanics textbook example [7,8] of a double-slit can be simulated on a computer by solving the time-dependent Schrödinger equation for a wave packet impinging on the double slit [9,10]. Alternatively, in order to obtain the observed interference pattern we could simply use random numbers to generate events according to the probability distribution that is obtained by solving the time-independent Schrödinger equation. However, that is not what we mean when we say that the physical phenomenon cannot be simulated on a computer. The point is that it is not known how to simulate, event-by-event, the experimental observation that the interference pattern appears only after a considerable number of events have been recorded on the detector. Quantum theory does not describe the individual events, e.g., the arrival of a single electron at a particular position on the detection screen [2,4,7,8]. Reconciling the mathematical formalism (that does not describe single events) with the experimental fact that each observation yields a definite outcome is often referred to as the quantum measurement paradox and is the central, most fundamental problem in the foundation of quantum theory [4,7,11].

If computer simulation is indeed a third methodology, it should be possible to simulate quantum phenomena on an event-by-event basis. In view of the fundamental problem alluded to above, there is little hope that we can find a simulation algorithm within the framework of quantum theory. However, if we think of quantum theory as a set of algorithms to compute probability distributions there is nothing that prevents us from stepping outside the framework that quantum theory provides. Therefore, we may formulate the physical processes in terms of events, messages, and algorithms that process these events and messages. In this paper, we demonstrate that simple deterministic, causal and classical processes that have a primitive form of learning capability can be used to simulate quantum systems, not by solving a wave equation but through event-by-event simulation. In other words, we show that fundamental quantum phenomena such as interference can be simulated by using algorithms that perform real-time recurrent learning [12]. In this paper, we also show that the same approach can be used for more conventional tasks that require some form of learning [12].

In Section 2 we introduce the basic concepts for constructing event-based, deterministic learning machines (DLMs). An essential property of these machines is that they process input event after input event and do not store information about individual events. A DLM can discover relations between input events (if there are any) and responds by sending its acquired knowledge in the form of another event (carrying a message) through one of its output channels. By connecting an output channel to the input channel of another DLM we can build networks of DLMs. As the input of a network receives an event, the corresponding message is routed through the network during which it is being processed. At any given time during the processing, there is only one input–output connection in the network that is actually carrying a message. The DLMs process the messages in a sequential manner and communicate with each other by message passing. There is no other form of communication between different DLMs. Although networks of DLMs can be viewed as networks that are capable of unsupervised learning, they have little in common with neural networks [12]. The first DLM described in Section 2 is equivalent to a standard linear adaptive filter [13] but the DLMs that we actually use for our applications do not fall into this class of algorithms.

In Section 3 we generalize the ideas of Section 2 and construct a DLM which groups K-dimensional data in two classes on an event-by-event basis, i.e. without using memory to store the whole data set. We demonstrate that this DLM is capable of detecting time-dependent trends in the data and of performing a blind classification [12]. Effectively, this DLM performs a principal-component analysis [14] on the fly, without explicitly diagonalizing the covariance matrix.

In Section 4 we show how to construct DLM-networks that generate output patterns that are usually thought of as being of quantum mechanical origin. We first build a DLM-network that simulates photons passing through a polarizer and show that quantum theory describes the output of this deterministic, event-based network. Then we describe a DLM-network that simulates a beam splitter and use this network to build a Mach–Zehnder interfer-

ometer and two chained Mach–Zehnder interferometers. We demonstrate that quantum theory also describes the behavior of these networks.

A summary and outlook is given in Section 5.

2. Deterministic learning machines

2.1. Learning points on the real axis

We consider a machine that has one input and two output channels labeled by ± 1 (see Fig. 1). The internal state of the machine after processing the nth input event ($n = 0, 1, \ldots$) is uniquely defined by the real variable x_n. At the next event $n+1$ the machine receives as input a real number y_{n+1}. For simplicity, but without loss of generality, we assume that $y_{n+1} \in [-1, 1]$. The machine responds by sending a message containing y_{n+1} through one of the two output channels $\Delta_{n+1} = \pm 1$. The machine selects the output channel $\Delta_{n+1} = +1$ or $\Delta_{n+1} = -1$ by minimizing the cost function $C(\Delta_{n+1})$ defined by

$$C(\Delta_{n+1}) = \big|y_{n+1} - x_n - (1-\alpha)\Delta_{n+1}|y_{n+1} - x_n|\big|, \tag{1}$$

updates its internal state according to the rule

$$x_{n+1} = x_n + (1-\alpha)\Delta_{n+1}|y_{n+1} - x_n|, \tag{2}$$

and sends a message with the input value y_{n+1} on the selected output channel Δ_{n+1}. The parameter $0 < \alpha < 1$ that enters Eqs. (1) and (2) controls the decision process. For simplicity we assume that α is fixed during the operation of the machine.

It is easy to see that $\Delta_{n+1} = +1$ if $x_n \leqslant y_{n+1}$ and $\Delta_{n+1} = -1$ if $x_n > y_{n+1}$. Thus, for this particular machine we have

$$\Delta_{n+1} = \frac{y_{n+1} - x_n}{|y_{n+1} - x_n|}. \tag{3}$$

Hence the update rule (2) can be written as the familiar recursion

$$x_{n+1} = \alpha x_n + (1-\alpha) y_{n+1}. \tag{4}$$

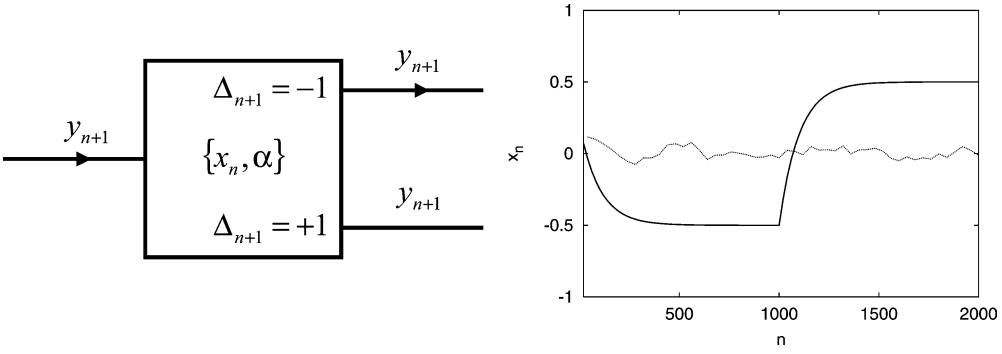

Fig. 1. *Left*: Schematic representation of the machine that responds to the input y_{n+1} by passing the input to one of the two output channels $\Delta_{n+1} = \pm 1$. The value of Δ_{n+1} depends on the current state of the machine, encoded in the variable x_n, the input y_{n+1}, and the update rule (2) in which α appears as a control parameter. *Right*: Evolution of the internal variable x_n as a function of the number of events n. Thick solid line: $y_{n+1} = -0.5$ for $n = 1, \ldots, 1000$ and $y_{n+1} = 0.5$ for $n = 1001, \ldots, 2000$; Thin solid line: Random sequence of $y_{n+1} = \pm 0.5$.

The solution of Eq. (4) reads

$$x_n = \alpha^n x_0 + (1-\alpha)\sum_{i=0}^{n-1} \alpha^{n-1-i} y_{i+1}, \tag{5}$$

where x_0 denotes the initial value of the internal variable.

As an illustration of how this machine learns, we consider the most simple example where $y_{n+1} = y$ for all $n \geqslant 0$. Then from Eq. (5) we find that

$$x_n = \alpha^n x_0 + (1-\alpha^n) y. \tag{6}$$

As $0 < \alpha < 1$, we conclude that $\lim_{n\to\infty} x_n = y$. Thus the machine "learns" the value of the input variable y. From Eq. (4) it follows that $x_n \leqslant y$ ($x_n \geqslant y$) implies $x_{n+1} \leqslant y$ ($x_{n+1} \geqslant y$). Hence x_n approaches y monotonically (and Δ_n is the same for all n). Therefore, if $y_n = y$, the machine always sends the value of y_n through the same output channel.

A distinct feature of this machine is its ability to adapt to changes in the input pattern. We illustrate this important property by two examples. Let $y_n = -0.5$ for $1 \leqslant n \leqslant 1000$ and $y_n = 0.5$ for $1000 < n \leqslant 2000$. During the first 1000 events the machine will learn -0.5. After 1000 events only 0.5 is being presented as input. Then, the machine "forgets" -0.5 and learns 0.5 as shown in the right panel of Fig. 1. In this simulation $\alpha = 0.99$. Alternatively, if y_n is a random sequence of ± 0.5 (each with the same probability) the machine has to learn -0.5 and 0.5 simultaneously. Because of this it cannot "forget" and it ends up oscillating around the mean of the input values (zero in this example) as illustrated in the right panel of Fig. 1. Let us now assume that our machine has reached this oscillating state. All input events $y_n = 0.5$ give $\Delta_n = +1$ and hence the machine sends 0.5 over the $+1$ channel. A second machine attached to this channel only receives 0.5 events and will learn 0.5. This suggests that a network of these machines can be used as an adaptive classifier.

Consider the network of three layers of machines shown in the left panel of Fig. 2. Each machine in the network learns the average of the numbers it receives at its input channel and sends the numbers which are smaller (larger or equal) than the number it learned to the -1 ($+1$) output channel. In our numerical experiments we set $\alpha = 0.99$. We start with 5000 events of random numbers $y_{n+1} \in \{-0.75, -0.25, 0.25, 0.75\}$, each occurring with equal probability. Machine 1 learns the average (zero in this example) and sends the negative (positive) y_{n+1} over the -1 ($+1$) channel to the input of machine 2 (3). Machine 2 (3) learns -0.50 (0.50), as shown in the top right panel of Fig. 2, and sends -0.75 (0.25) over its -1 output channel and -0.25 (0.75) over its $+1$ output channel. Machines 4 to 7 learn -0.75, -0.25, 0.25 and 0.75, respectively, as shown in the bottom right panel of Fig. 2. Each of these machines forwards the received input on its $+1$ (-1) output channel if the initial value of its internal variable is smaller (larger) than the received input value. Let us now assume that after 5000 events the input data set changes to $y_{n+1} \in \{-0.75, -0.25, 0.25, 0.50\}$. As can be seen from the right panel of Fig. 2, machines 1, 3 and 7 "forget" the number they learned and replace it by -0.0625, 0.375 and 0.50, respectively. All other machines are unaffected because they never get 0.50 as input. After another 5000 events we change the set of input values once more, this time to $y_{n+1} \in \{-0.60, -0.75, -0.25, 0.25, 0.50\}$, i.e. we add one element. Now, machine 1 learns -0.17, machine 2 learns -0.53 and the internal state of machine 3 remains unchanged. Machine 4 can now receive two numbers on its input channel, namely -0.75 and -0.60. As a consequence, machine 4 learns -0.675, i.e. the average of the two possible input numbers. Machine 4 puts -0.60 on its $+1$ output channel and -0.75 on its -1 output channel. In order for the network to learn all the numbers of the input set, we would have to attach one extra machine to each output channel of machine 4.

2.2. Learning points on a finite interval

For the machine defined by Eqs. (1) and (2), formulating the operation of the machine through the minimization of the difference between the input and internal variable may seem a little superfluous and indeed, for this particular

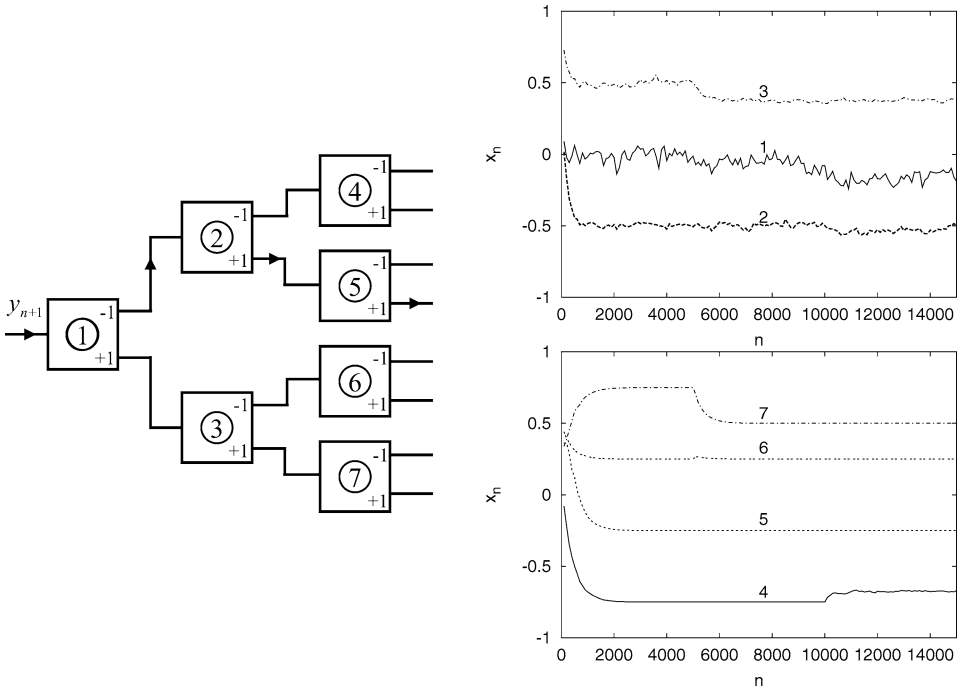

Fig. 2. *Left*: Diagram of the three-level machine that adaptively classifies the input data y_{n+1}. *Right*: Evolution of the internal variables x_n of the machines as a function of the number of events n. The machine number is used to label the corresponding line. *Top right*: First three machines; *Bottom right*: Third-level machines.

machine it is. However, this formulation is a convenient starting point for defining machines that can perform more intricate tasks. For instance, let us make an innocent looking change to the update rule (2) by writing

$$x_{n+1} = \alpha x_n + (1-\alpha)\Delta_{n+1}, \tag{7}$$

and replace the cost function (1) by the corresponding expression

$$C(\Delta_{n+1}) = \left|y_{n+1} - \alpha x_n - (1-\alpha)\Delta_{n+1}\right|. \tag{8}$$

For $\Delta_{n+1} = +1$ we have $x_{n+1} = 1-\alpha(1-x_n)$ and for $\Delta_{n+1} = -1$ we have $x_{n+1} = -1+\alpha(1+x_n)$. Therefore, if $0 < \alpha < 1$ and $|x_0| \leqslant 1$, the internal variable will always be in the range $[-1, 1]$. At each event the internal variable either increases by $(1-\alpha)(1-x_n)$ (if $\Delta_{n+1} = +1$) or decreases by $(1-\alpha)(1+x_n)$ (if $\Delta_{n+1} = -1$). In both cases this change is always nonzero, except if $x_n = \pm 1$ which can only occur if $y_{n+1} = \pm 1$. The ratio of the step sizes is $(1-x_n)/(1+x_n)$.

The machine defined by Eqs. (7) and (8) behaves differently from the machine defined by Eqs. (1) and (2). To see this, it is instructive to consider the case $0 \leqslant y_{n+1} = y < 1$ for all $n \geqslant 0$ (the case $-1 < y_{n+1} = y < 0$ can be treated in the same manner). For concreteness we assume that $-1 < x_0 < y$. At the first event, minimization of Eq. (8) yields $\Delta_1 = +1$ and $x_1 = 1+\alpha(x_0-1)$. In other words, the internal variable x moves towards y. As long as $x_n < y$, the machine selects $\Delta_{n+1} = +1$, always increasing its internal variable x_n. For some $n \geqslant 1$ we must have $x_n > y$. Then, making another move in the positive x-direction allows for two different decisions. If the error that results is larger than the error that is obtained by moving in the negative direction the machine decides to set $\Delta_{n+1} = -1$. Otherwise it makes another move in the positive x-direction ($\Delta_{n+1} = +1$). In any case, for some $n > 1$ the machine will select $\Delta_{n+1} = -1$. Note that when this happens, we must have $x_{n+1} < y$ and $\Delta_{n+2} = +1$.

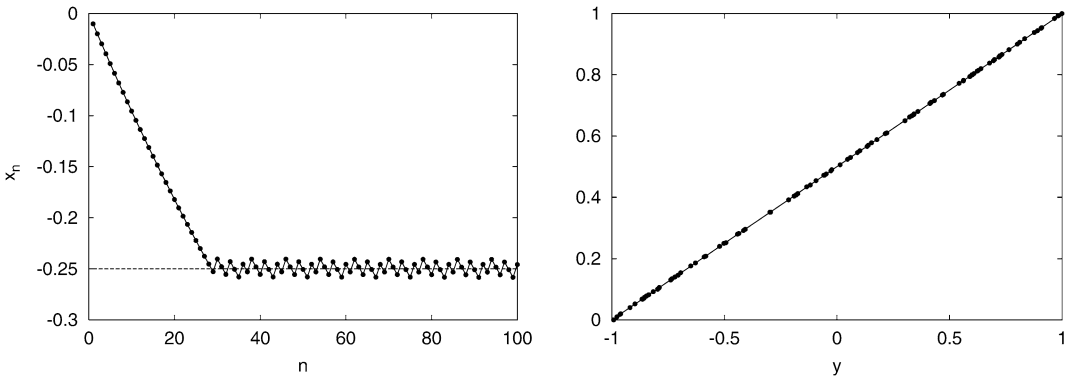

Fig. 3. *Left*: Time evolution of the internal variable x_n of the machine defined by Eqs. (7) and (8). The input events are $y = -0.25$, $\alpha = 0.99$, and the initial value $x_0 = 0$. For $n > 30$ the internal variable x_n oscillates about y. For $n > 500$ the sequence of increments ($\Delta_{n+1} = +1$) and decrements ($\Delta_{n+1} = -1$) of x_n repeats itself after 8 events (data not shown). Lines are guides to the eyes. *Right*: The number of increments of the internal variable ($\Delta_{n+1} = +1$) divided by the total number of events as a function of the value of the input variable y. Bullets: Each data point is obtained from a simulation of 1000 events with a fixed, randomly chosen value of $-1 < y < 1$, using the last 500 events to count the number of $\Delta_{n+1} = +1$ events. Solid line: $(1 + y)/2$.

This implies that after this nth event (that we denote by n_0) the internal variable will oscillate (forever) around the input value y. This process is illustrated in Fig. 3 (left).

For $m > n_0$ we have $|x_{m+1} - y| \leqslant (1 - \alpha)\max(1 - y, 1 + y)$. Thus, if $0 < 1 - \alpha \ll 1$, the amplitude of the oscillations is small. The machine "learns" the input value y and the ratio of the increments to decrements is $(1 + x_{m+1})/(1 - x_{m+1}) \approx (1 + y)/(1 - y)$. In this stationary regime of oscillating behavior, the number of times the machine activates the $+1$ (-1) channel is given by $(1 + y)/2$ $((1 - y)/2)$. The simulation results shown in Fig. 3 (right) confirm the correctness of this analysis. For a fixed (unknown) value of the input variable, the rate at which the machine defined by the rules (7) and (8) activates one of its output channels is determined by the value of its internal variable. Therefore, this rate reflects the value that the machine has learned by processing the input events. Depending on the application, the message that is sent through the active output channel can contain x_{n+1} or the input value y_{n+1} (there is nothing else that can be send). Obviously we can make the learning process more precise by increasing $\alpha < 1$. Of course, a larger value of α also results in slower learning: In general it will take more events for the internal variable to reach the value where it starts to oscillate.

2.3. Learning points on a circle

In going from the first to the second example of Section 2 we changed the update rule such that the variable x_n is constrained to lie in the interval $[-1, 1]$. We now consider the two-dimensional analogue of the machines described in Section 2.2 for which the internal vector $(x_{1,n}, x_{2,n})$ and input vector $(y_{1,n+1}, y_{2,n+1})$ represent points on a circle. This machine receives as input a sequence of angles ϕ_{n+1} defined by

$$\cos\phi_{n+1} = \frac{y_{1,n+1}}{\sqrt{y_{1,n+1}^2 + y_{2,n+1}^2}}, \qquad \sin\phi_{n+1} = \frac{y_{2,n+1}}{\sqrt{y_{1,n+1}^2 + y_{2,n+1}^2}}, \tag{9}$$

and responds by activating one of the two output channels.

For all $n > 0$, the update rules are defined by

$$x_{1,n+1} = \alpha x_{1,n} + \beta\Theta_{n+1}, \qquad x_{2,n+1} = \alpha x_{2,n} + \beta(1 - \Theta_{n+1}), \tag{10}$$

where $\Theta_{n+1} = 0, 1$ and $0 < \alpha < 1$. In order that the internal vector $\mathbf{x}_{n+1} = (x_{1,n+1}, x_{2,n+1})$ stays on the unit circle we must have

$$\beta = -\alpha\big[x_{1,n}\Theta_{n+1} + x_{2,n}(1-\Theta_{n+1})\big] \pm \sqrt{1-\alpha^2+\alpha^2[x_{1,n}^2\Theta_{n+1} + x_{2,n}^2(1-\Theta_{n+1})]}. \tag{11}$$

Substitution of Eq. (11) in Eq. (10) gives us four different rules:

$$\begin{aligned} x_{2,n+1} &= s\sqrt{1+\alpha^2(x_{2,n}^2-1)}, \qquad x_{1,n+1} = \alpha x_{1,n} \qquad \text{if } \Theta_{n+1} = 0, \\ x_{1,n+1} &= s\sqrt{1+\alpha^2(x_{1,n}^2-1)}, \qquad x_{2,n+1} = \alpha x_{2,n} \qquad \text{if } \Theta_{n+1} = 1, \end{aligned} \tag{12}$$

where $s = \pm 1$ takes care of the fact that for each choice of Θ_{n+1}, the machine has to decide between two quadrants. The cost function is defined by

$$C = -(x_{1,n+1}y_{1,n+1} + x_{2,n+1}y_{2,n+1}). \tag{13}$$

Obviously, the cost function (13) is nothing but the inner product of the vectors $\mathbf{x}_{n+1}$ and $\mathbf{y}_{n+1}$. The new internal state itself is determined by calculating the cost equation (13) for each of the four candidate update rules listed in Eq. (12) and selecting the rule that yields the minimum cost. Note that the minimum of the cost function (13) does not depend on the length of the vector of input variables $(y_{1,n+1}, y_{2,n+1})$. From Eq. (12) it follows that if $\Theta_{n+1} = 0$ by the value of $x_{1,n+1}$ is obtained by rescaling of $x_{1,n}$ and $x_{2,n+1}$ is adjusted such that $x_{1,n+1}^2 + x_{2,n+1}^2 = 1$. For $\Theta_{n+1} = 1$ we interchange the role of the first and second element of $\mathbf{x}_{n+1}$.

In general the behavior of the machine defined by rules (12) and (13) is difficult to analyze without the use of a computer. However, for a fixed input vector $\mathbf{y}_{n+1} = \mathbf{y}$ it is clear what the machine will try to do: It will minimize the cost equation (13) by rotating its internal vector $\mathbf{x}_{n+1}$ to bring it as close as possible to $\mathbf{y}$. However, $\mathbf{x}_{n+1}$ will not converge to a limiting value but instead it will keep oscillating about the input value $\mathbf{y}$. An example of a simulation is given in Fig. 4 (left). For a fixed input vector $\mathbf{y}_{n+1} = \mathbf{y}$ the machine reaches a stationary state in which its internal vector oscillates about $\mathbf{y}$. In this stationary state the output signal consists of a finite sequence of ones and zeros. The machine repeats this sequence over and over again. Obviously, the whole process is deterministic. The details of the approach to the stationary state depend on the initial value of the internal vector $\mathbf{x}_0$, but the stationary state itself does not.

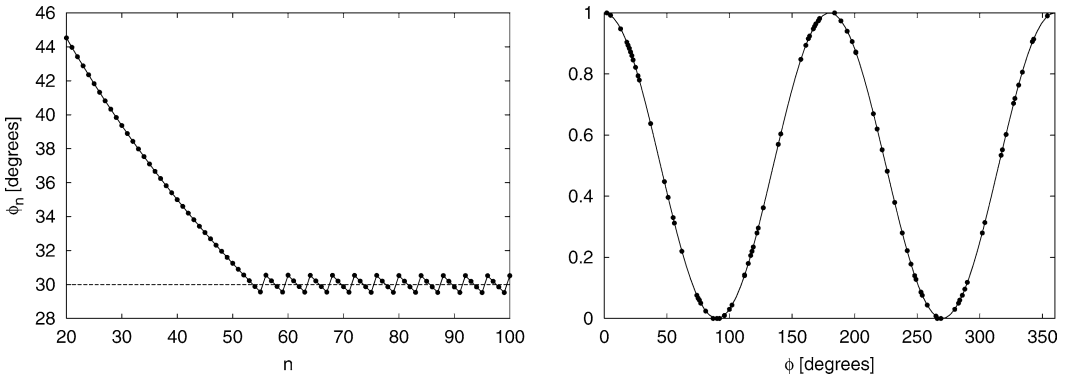

Fig. 4. *Left*: Time evolution of the angle representing the internal vector $\mathbf{x}_n$ of the machine defined by Eqs. (12) and (13). The input events are vectors $\mathbf{y}_{n+1} = (\cos 30^\circ, \sin 30^\circ)$. The direction of the initial vector $\mathbf{x}_0$ is chosen at random. In this simulation $\alpha = 0.99$. For $n > 60$ the ratio of the number of increments $(\Theta_{n+1} = 0)$ to decrements $(\Theta_{n+1} = 1)$ is $1/3$, which is $(\sin 30^\circ/\cos 30^\circ)^2$. Data for $n < 20$ has been omitted to show the oscillating behavior more clearly. Lines are guides to the eyes. *Right*: The number of $(\Theta_{n+1} = 1)$ events divided by the total number of events as a function of the value of the input variable ϕ. Bullets: Each data point is obtained from a simulation of 1000 events with a fixed, randomly chosen value of $0 \leqslant \phi < 360^\circ$, using the last 500 events to count the number of $(\Theta_{n+1} = 1)$ events. Solid line: $\cos^2\phi$.

These observations are of much more general nature than the example given in Fig. 4 (left) suggests. In fact, as the applications discussed below amply illustrate, the stationary-state analysis is a very useful tool to predict the behavior of the machines. Assuming that $0 < 1-\alpha \ll 1$ and that we have reached the stationary regime in which the internal vector performs small oscillations about $(\cos\phi, \sin\phi)$, a simple calculation shows that

$$\begin{aligned} \delta\phi_0 &= \phi_{1,n+1} - \phi_{1,n} \approx \frac{1-\alpha^2}{2}\frac{\cos\phi}{\sin\phi} \quad \text{if } \Theta_{n+1}=0, \\ \delta\phi_1 &= \phi_{1,n+1} - \phi_{1,n} \approx \frac{\alpha^2-1}{2}\frac{\sin\phi}{\cos\phi} \quad \text{if } \Theta_{n+1}=1. \end{aligned} \tag{14}$$

In the stationary regime, we have $N_0\delta\phi_0 \approx N_1\delta\phi_1$ where N_0 (N_1) is the number of $\Theta_{n+1}=0$ ($\Theta_{n+1}=1$) events. From Eq. (14) it then follows immediately that $N_0/(N_0+N_1) \approx \sin^2\phi$ and $N_1/(N_0+N_1) \approx \cos^2\phi$. The results of this analysis are in excellent agreement with the simulation results shown in Fig. 4 (right).

The conventional approach to regard the variables Θ_{n+1} as input is fundamentally different from the approach adopted in this paper. This can be seen by reformulating the update rules in terms of difference equations and to assume that the $\Theta_{n+1}=0,1$ are independent, uniform random variables with mean $\Theta = \langle\Theta_{n+1}\rangle$. The four rules (12) can be written as

$$\begin{aligned} x_{1,n+1}^2 &= \alpha^2 x_{1,n}^2 + (1-\alpha^2)\Theta_{n+1}, \\ x_{2,n+1}^2 &= \alpha^2 x_{2,n}^2 + (1-\alpha^2)(1-\Theta_{n+1}). \end{aligned} \tag{15}$$

Formally Eq. (15) has the same structure as Eq. (4). Averaging over many realizations of $\{\Theta_{n+1}=0,1\}$ and taking the limit $n\to\infty$ we obtain

$$\begin{aligned} \langle x_1^2\rangle &= \lim_{n\to\infty}\langle x_{1,n+1}^2\rangle = \Theta, \\ \langle x_2^2\rangle &= \lim_{n\to\infty}\langle x_{2,n+1}^2\rangle = 1-\Theta. \end{aligned} \tag{16}$$

In other words, a machine that operates according to the rules (12) and receives as input the random sequence Θ_{n+1} will (on average) approach a state in which the direction of its internal vector gives us an estimate of $\Theta = \langle\Theta_{n+1}=0,1\rangle$. In contrast, a machine that minimizes the cost equation (13) and updates its internal state according to Eq. (12) responds on either output channel $\Theta_{n+1}=0$ or output channel $\Theta_{n+1}=1$, with a frequency that is directly related to the difference between the current input angle and the angle defined by the internal vector.

2.4. Learning points on a K-dimensional hypersphere

Consider a sequence of events, characterized by vectors $\mathbf{y}_{n+1} = (y_{1,n+1}, y_{2,n+1}, \ldots, y_{K,n+1})$ for $n>0$. The vector $\mathbf{y}_{n+1}$ is the input for the machine. The internal state of the machine is described by a K-dimensional unit vector $\mathbf{x}_n = (x_{1,n}, x_{2,n}, \ldots, x_{K,n})$. We define the $2K$ candidate update rules $\{j=1,\ldots,K; s_j=\pm1\}$ by

$$\begin{aligned} x_{i,n+1} &= s_j\sqrt{1+\alpha^2(x_{i,n}^2-1)} \quad \text{if } i=j, \\ x_{i,n+1} &= \alpha x_{i,n} \quad \text{if } i\neq j. \end{aligned} \tag{17}$$

Note that $\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_n = 1$ implies $\mathbf{x}_{n+1}^{\mathrm{T}}\mathbf{x}_{n+1} = 1$ for each of the $2K$ update rules. The machine responds to the input $\mathbf{y}_{n+1}$ by selecting from the $2K$ possible rules in Eq. (17), the update rule that minimizes the cost

$$C = -\mathbf{x}_{n+1}^{\mathrm{T}}\mathbf{y}_{n+1}, \tag{18}$$

and by sending a message containing $\mathbf{y}_{n+1}$ (or, depending on the application, $\mathbf{x}_{n+1}$) on one of its output channels. Note that the minimum of the cost function (18) does not depend on the length of the vectors $\mathbf{x}_{n+1}$ or $\mathbf{y}_{n+1}$. Disregarding the variables s_j that merely serve to determine the sign of $x_{i,n+1}$ there are K rules. Hence there can be as many as K output channels. However, depending on the application, it may be expedient to reduce the number of output channels by arranging them in groups.

2.5. Communication between events

The machines analyzed in the previous subsections have one input channel that receives input and two output channels, only one of which sends out data (a message) at a particular event. An obvious generalization is to construct machines that accept, at a given instance, input from one out of two different sources. This is absolutely necessary if we want to build machines in which events can communicate or, in physical terms, interact with each other. We now demonstrate that the machines that we introduced above already have the capability to let events interact with each other. Therefore we do not need to add a new feature or rule to the machines.

Consider a machine that has two input channels 0 and 1 and an internal vector $\mathbf{x}_n$ with $K = 4$ components. At the $(n+1)$th event, either input channel 0 receives the two-component vector $\mathbf{y}_{n+1} = (y_{1,n+1}, y_{2,n+1})$ or input channel 1 receives the two-component vector $\mathbf{y}_{n+1} = (y_{3,n+1}, y_{4,n+1})$.

In the former case the machine transforms this input into the input vector $\hat{\mathbf{y}}_{n+1} = (y_{1,n+1}, y_{2,n+1}, x_{3,n}, x_{4,n})$ of four elements by using the current internal vector as a source for the missing elements. Similarly, in the latter case the input vector becomes $\hat{\mathbf{y}}_{n+1} = (x_{1,n}, x_{2,n}, y_{3,n+1}, y_{4,n+1})$. Then the machine uses $\hat{\mathbf{y}}_{n+1}$ to determine the cost and selects the update rule according to the procedure described in Section 2.4 (with $\hat{\mathbf{y}}_{n+1}$ replacing $\mathbf{y}_{n+1}$). This machine learns the two-dimensional vectors $\mathbf{y}_{n+1} = (y_{1,n+1}, y_{2,n+1})$ and $\mathbf{y}_{n+1} = (y_{3,n+1}, y_{4,n+1})$ separately, as if it consists of two separate, independent two-dimensional machines, with the additional crucial feature that the internal vector represents a point on a 4-dimensional unit sphere.

It is not difficult to imagine what this machine does in the case that it receives events on only one of the two input channels (say 0). Irrespective of the initial value of the internal vector $\mathbf{x}_0$, the machine will always select the update rule with $j = 1, 2$ (see Eq. (17)) and the two components $x_{3,n}$ and $x_{4,n}$ will vanish exponentially fast with increasing n (recall that $0 < \alpha < 1$). Thus, after a few events the internal state of the machine indicates that the machine receives events on only one channel.

If the machine receives input on both channels (but never simultaneously), Eq. (17) implies that the machine only scales the two components of the internal state that it uses to provide the missing elements for building the input $\hat{\mathbf{y}}_{n+1}$. Therefore, in the stationary regime, the length of the two-dimensional vector $(x_{1,n}, x_{2,n})$ $((x_{3,n}, x_{4,n}))$ is proportional to the number of events on input channel 0 (1). Furthermore the number of $j = 1, 2$ $(j = 3, 4)$ events is approximately equal to the number of events on input channel 0 (1). Although this may seem a very elementary form of communication, it is sufficient to construct machines that perform very complicated tasks.

2.6. Summary

The machines described above are simple deterministic machines that make decisions. The machine responds to the input event by choosing from all possible alternatives, the internal state that minimizes the error between the input and the internal state itself. Then the machine sends a message through one of its output channels. The message contains information about the decision the machine took while updating its internal state and, depending on the application, also contains other data that the machine can provide. By updating its internal state, the machine "learns" about the input it receives and by sending messages through one of its two output channels, it tells its environment about what it has learned. In the sequel we will call such a machine a *deterministic learning machine* (DLM). For a particular choice of the update rule (see Section 2.1), the machine performs linear estimation but as the other examples of this section amply demonstrate, minor modifications to this rule and/or cost function yield machines that may behave in a substantially different manner.

3. Application to blind classification

The DLM of Section 2.1 learns about the input data by moving a point on a line. Obviously, this point separates two parts of the line. The generalization to K-dimensional space is a $(K-1)$-dimensional hyperplane that divides

the space into two parts. Thus, to interpret two-dimensional data the DLM should learn a line instead of a point. We represent the line by a segment L_n defined by its mid-point $\mathbf{x}_n$ and its direction $\mathbf{d}_n$. As the DLM receives an event $\mathbf{y}_{n+1}$, i.e. a point in a two-dimensional plane, the DLM updates its internal line segment L_n and sends the information describing L_n through the -1 $(+1)$ channel, depending on whether it lies on the left (right) side of the line. The update procedure consists of two steps. First we define two support points $\mathbf{v}_1$ and $\mathbf{v}_2$ on either side of $\mathbf{x}_n$ along the direction $\mathbf{d}_n$ by

$$\mathbf{v}_1 = \mathbf{x}_n - \mathbf{d}_n/2, \qquad \mathbf{v}_2 = \mathbf{x}_n + \mathbf{d}_n/2, \tag{19}$$

and we update the two support points according to

$$\begin{aligned} \hat{\mathbf{v}}_1 &= \mathbf{v}_1 + (1-\alpha)(\mathbf{y}_{n+1} - \mathbf{v}_1)\|\mathbf{y}_{n+1} - \mathbf{v}_1\|, \\ \hat{\mathbf{v}}_2 &= \mathbf{v}_2 + (1-\alpha)(\mathbf{y}_{n+1} - \mathbf{v}_2)\|\mathbf{y}_{n+1} - \mathbf{v}_2\|, \end{aligned} \tag{20}$$

where $0 < \alpha < 1$ controls the learning process. Then we compute the new mid-point and direction of the line segment:

$$\mathbf{x}_{n+1} = (\hat{\mathbf{v}}_1 + \hat{\mathbf{v}}_2)/2, \qquad \mathbf{d}_{n+1} = (\hat{\mathbf{v}}_1 - \hat{\mathbf{v}}_2)/\|\hat{\mathbf{v}}_1 - \hat{\mathbf{v}}_2\|. \tag{21}$$

From Eq. (20) it follows that the support point farthest away from $\mathbf{y}_{n+1}$ makes the largest move. Therefore, as new input data is received by the DLM, both the mid-point and the direction of the line segment change. Note that the update rule (20) is non-linear in the difference between internal and input vector. Although a linear update rule also works, our numerical experiments (results not shown) indicate that the non-linear rule (20) performs much better.

In general $\mathbf{x}_n$ will converge to the mean of the input vectors and $\mathbf{v}_1$ and $\mathbf{v}_2$ will be pulled most strongly in the direction of largest variance. Therefore L_n will be (approximately) perpendicular to the largest principal component of the covariance matrix of the input data. In other words, the DLM defined above can find the eigenvector that corresponds to the largest eigenvalue of the covariance matrix by processing data points in a sequential manner, i.e. without actually having to compute the elements of the covariance matrix.

As an illustration of the capabilities of the DLM introduced in this section, let us consider a classification task in which we want to blindly group events into two categories. The input data $\mathbf{y}_{n+1} = (y_{1,n+1}, y_{2,n+1})$ are generated through a Gaussian random process described by:

$$y_{1,n} = \cos(\gamma n + s)\pi + r_1, \qquad y_{2,n} = \sin(\gamma n + s)\pi + r_2, \tag{22}$$

where s is a uniform random bit. The random numbers r_1 and r_2 are drawn from the normal distribution $N(0, 1/2)$. In our numerical example we take $\gamma = 1/5000$ and $\alpha = 0.99$. From Eq. (22) it is clear that the input events consist of points in a plane that are drawn from one of two $(s = 0, 1)$ Gaussian distributions, the centers of which rotate with a period of 10 000 events. The mean of all input data is $(0, 0)$ and there is no preferred direction of largest variance. The reason of course is that the center of the Gaussian distributions slowly moves on the unit circle. Clearly, this kind of classification task can only be performed by permanently updating the estimate of the direction and that is exactly what the DLM does. In Fig. 5 we present results of a blind classification experiment that illustrates the operation of the DLM defined by the rules (19)–(21). The DLM processes event-by-event, each time updating its estimate for the separatrix. For comparison we also show the result obtained by the principal component analysis [14] using as input the group of 100 most recent data points processed by the DLM. The differences between both classifiers are rather small so that it is clear that the DLM-based classifier performs very well.

The two-dimensional DLM described above can easily be extended to a DLM that processes K-dimensional input data. Instead of a line segment the DLM has to learn a segment of a $(K-1)$-dimensional hyperplane. This can be done by extending the procedure used in the two-dimensional case. The hyperplane segment is described by a mid-point $\mathbf{x}_n$ and $K-1$ orthonormal directions $\mathbf{d}_k$ for $k = 1, \ldots, K-1$. We choose K points $\{\mathbf{v}_k\}$ on the hyperplane defined by $\{\mathbf{d}_k\}$ and $\mathbf{x}_n$ such that the distance between each pair of points is one. As new input data

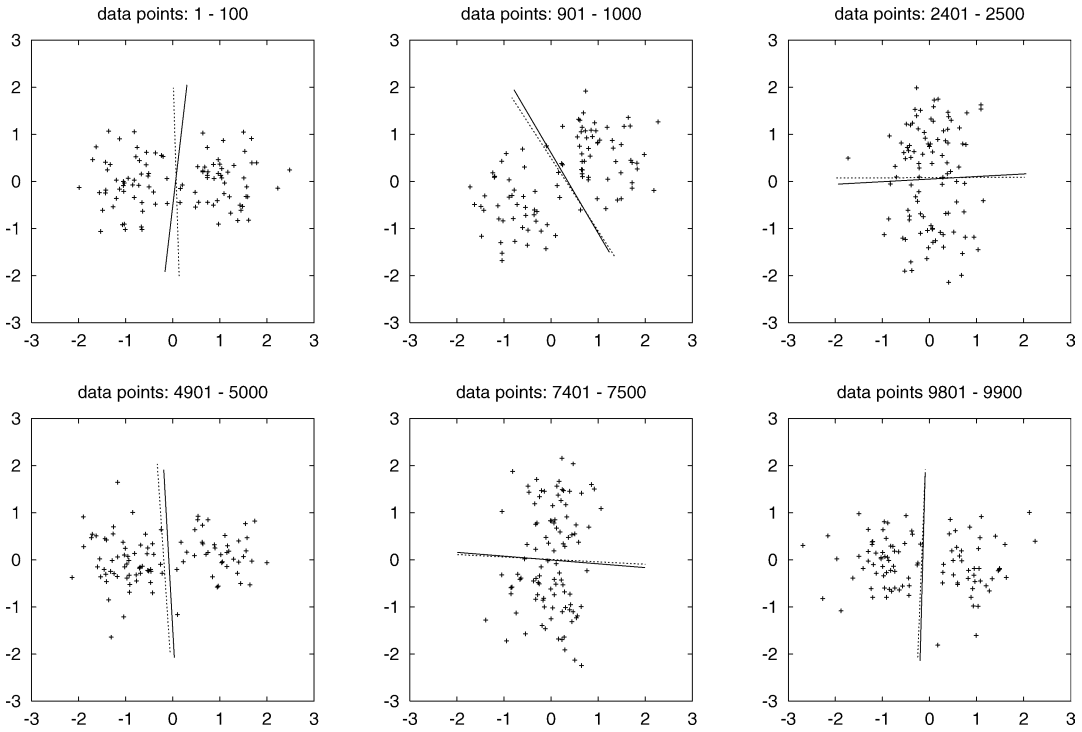

Fig. 5. Snapshots of the input data and results of a DLM-based classifier defined by Eqs. (19)–(21) (solid line) and a conventional principal-component-based classifier (dashed line) [14]. The data points are random deviates with a normal distribution with variance $1/2$ and means $\pm(\cos(2\pi n/10000), \sin(2\pi n/10000))$. Each panel shows the output of the DLM-based classifier after it has processed, point-by-point, the 100 data points shown. The classifier smoothly follows the rotation of the means. In contrast to the event-by-event processing of the DLM-based classifier, the principal-component-based classifier processes the whole set of 100 data points simultaneously.

$\mathbf{y}_{n+1}$ is received by the DLM these points are updated according to (the generalization of) Eq. (20). As in the two-dimensional case, from the updated points we can calculate the new mid-point and the new directions. However, unlike in the two-dimensional case, these directions do not need to be orthonormal. The orthonormality is then restored by using the (modified) Gramm–Schmidt procedure [15].

4. Application to deterministic simulation of quantum interference

4.1. Photon polarization

We demonstrate that the DLM defined by Eqs. (12) and (13) and a passive element that performs a plane rotation are sufficient to perform a deterministic simulation of the quantum theory [6] of photon polarization.

We start by recalling some elementary facts about photon polarization [16,17]. Some optically active materials like calcite split an incoming beam of light into two spatially separated beams [16,18]. The light intensity of these beams is related to the angle of polarization ψ of the electromagnetic wave, relative to the orientation ϕ of the material [18]. We disregard all imperfections of real experiments and assume that the experimental data are in exact agreement with the wave mechanical theory. Then the intensities I_0 of beam 0 and I_1 of beam 1 are given by [16,17]

$$I_0 = \cos^2(\psi - \phi), \qquad I_1 = \sin^2(\psi - \phi), \tag{23}$$

respectively. If the incident beam has a random polarization, averaging of Eq. (23) over all ψ shows that half of the light intensity will go to beam 0 and the other half to beam 1.

If the conventional light source is replaced by a source that emits one photon at a time, the photon leaves the material either in the direction of beam 0 or beam 1, never in both [16]. Collecting photons over a sufficiently long period shows that Eq. (23) still gives the number of photons detected in the direction of beam 0 (1), divided by the total amount of detected photons [16]. Quantum theory [6] describes the polarization in terms of a two-dimensional (complex-valued) vector and the action of the material is to rotate this vector by an angle ϕ (set by the experimentalist) [17]. The probability to observe photons in beam 0 (1) is given by the square of the 0th (1st) element of the vector [17]. In addition, as the photon leaves the material in beam 0 (1), its polarization is ϕ ($\phi+\pi/2$) [17]. Thus the piece of material can be used to prepare and also determine the polarization of the photons and is called a "polarizer" [18].

According to quantum theory [6], the polarizer rotates the vector of polarization amplitudes in the following manner [17]:

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}. \tag{24}$$

Still according to quantum theory [6], the intensity in beam 0 (1) is given by $|b_0|^2$ ($|b_1|^2$). An incident beam with an angle of polarization ψ is described by the vector $(a_0, a_1) = (\cos\psi, \sin\psi)$. From Eq. (24) we obtain $(b_0, b_1) = (\cos(\psi-\phi), \sin(\psi-\phi))$ and hence $I_0 = |b_0|^2 = \cos^2(\psi-\phi)$ and $I_1 = |b_1|^2 = \sin^2(\psi-\phi)$, in agreement with Eq. (23).

We now construct a simple deterministic machine that generates events of which the distribution agrees with the probability distributions predicted by quantum theory [6]. The layout of this "polarizer" is shown in Fig. 6. The incoming event (photon) carries an (unknown) angle ψ_{n+1}. The purpose of the passive element $R(\phi)$ is to perform a rotation

$$R(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}, \tag{25}$$

of the input vector $\mathbf{y}_{n+1} = (\cos\psi_{n+1}, \sin\psi_{n+1})$ by the angle ϕ. The resulting vector $\mathbf{z}_{n+1} = (\cos(\psi_{n+1} - \phi), \sin(\psi_{n+1} - \phi))$ is sent to the input of a DLM that operates according to Eqs. (12) and (13). If $\Theta_{n+1} = 0$, the DLM responds by sending the vector $\mathbf{z}'_{n+1} = (\cos\phi, \sin\phi)$ through the output channel 0. If $\Theta_{n+1} = 1$, the DLM responds by sending the vector $\mathbf{z}'_{n+1} = (\cos(\phi+\pi/2), \sin(\phi+\pi/2))$ through the output channel 1. Clearly this procedure is strictly deterministic. We emphasize that the DLM processes information event by event and does not store the data contained in each event.

In Fig. 6 (right) we show simulation results for the machine depicted in Fig. 6 (left). Each data point represents the intensity in beam 0 (1), i.e. the number of $\Theta = 0$ (1) events divided by the total amount of events. The machine is initialized once by choosing a random direction of the vector $\mathbf{x}_0$. The angle of rotation ϕ is kept fixed for 1000 events, then a uniform random number is used to select another direction, and this procedure is repeated 100 times. In all these numerical experiments we set $\alpha = 0.99$. Fig. 6 shows the results for two different numerical experiments: In the first set of 100 runs, the direction of polarization ψ of the incoming photons is also determined by means of uniform random numbers. In the second set of 100 runs, the direction of polarization of the incoming photons is fixed ($\psi = 25°$). From Fig. 6 (right) it is clear that quantum theory [6] provides a very good description of the input–output behavior of the DLM shown in Fig. 6 (left).

As a second illustration we use the same DLM to simulate an experiment with three polarizers described by Feynman [16]. The diagram of this experiment is shown in Fig. 7 (left). A randomly polarized beam of photons passes through the first polarizer (without loss of generality we set its angle ϕ_1 equal to zero). Each output channel is used as input to another polarizer. Both these polarizers are tilted by the same angle $\phi_2 = \phi_3 = \phi$. According

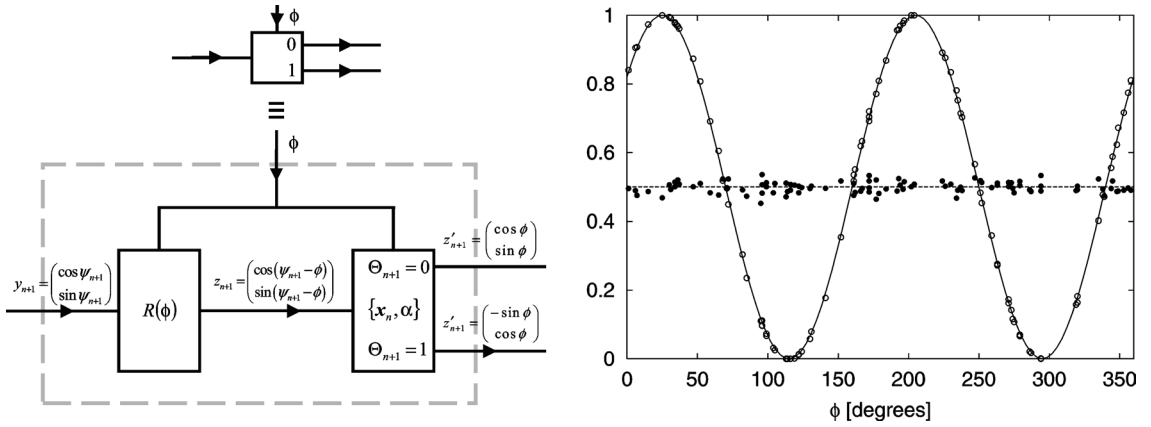

Fig. 6. *Left*: Diagram of the DLM network that simulates a polarizer on a deterministic, event-by-event basis. *Right*: Simulation results for the DLM network shown on the left. Each data point represents the number of events in an output channel accumulated after 1000 input events. After each set of 1000 events, the orientation ϕ of the polarizer is changed randomly. Open circles: Normalized intensity in output channel 0 for incoming photons with a polarization angle $\psi = 25°$; Solid line: Result ($\cos^2(\psi - \phi)$) obtained from quantum theory [6] for incoming photons with a polarization angle $\psi = 25°$; Bullets: Normalized intensity in output channel 1 for incoming photons with a random polarization angle ψ; Dashed line: Result of quantum theory [6] for incoming photons with a random polarization angle ψ.

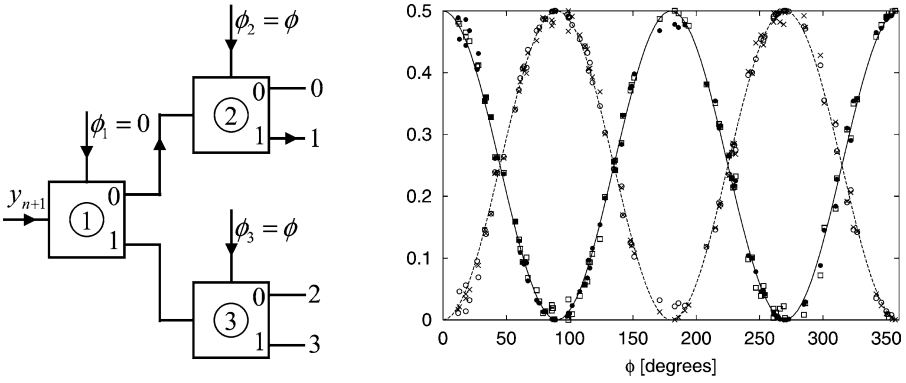

Fig. 7. *Left*: Schematic representation of an experiment with three polarizers [16]. *Right*: Simulation results for the network of DLMs shown on the left. Each data point represents the normalized intensity accumulated over 1000 events. After each set of 1000 events, the orientation ϕ of the polarizers 2 and 3 is changed randomly. Bullets: Output channel 0; Crosses: Output channel 1; Open circles: Output channel 2; Open squares: Output channel 3. Lines represent the results of quantum theory [6].

to quantum theory [6], the intensity at the output of these four channels is (from top to bottom, see Fig. 7 (left)) $2^{-1}\cos^2\phi$, $2^{-1}\sin^2\phi$, $2^{-1}\sin^2\phi$, and $2^{-1}\cos^2\phi$. The results of our numerical experiments are shown in Fig. 7 (right). The simulation procedure is the same as the one used to generate the data of Fig. 6. Also in these numerical experiments we set $\alpha = 0.99$. We emphasize once more that the randomness in these discrete-event simulations only enters through the characterization of the photon source and through our procedure of selecting the direction of the polarizer for each set of 1000 events. Actually, the latter only serves to counter the possible objection that the apparent quantum mechanical behavior would be caused by monotonically changing the direction of the polarizers. As in the previous example, it is clear that quantum theory [6] describes the input–output behavior of the three-DLM network very well.

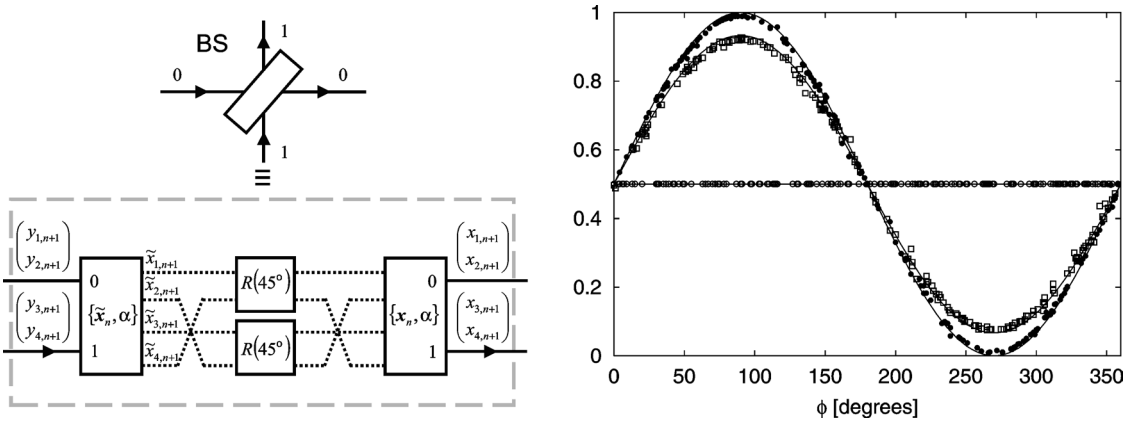

Fig. 8. *Left*: Diagram of the network of two DLMs that performs a deterministic simulation of a single-photon beam splitter (BS) on an event-by-event basis [19]. The solid lines represent the input and output channels of the BS. Dashed lines indicate the flow of data within the BS. *Right*: Simulation results for the beam splitter shown on the left. Input channel 0 receives $(y_{1,n+1}, y_{2,n+1}) = (\cos\psi_0, \sin\psi_0)$ with probability p_0. Input channel 1 receives $(y_{3,n+1}, y_{4,n+1}) = (\cos\psi_1, \sin\psi_1)$ with probability $p_1 = 1 - p_0$. Each data point represents 10 000 events. After each set of 10 000 events, a uniform random number in the range $[0, 360]$ is used to choose the angles ψ_0 and ψ_1. Markers give the simulation results for the normalized intensity in output channel 0 as a function of $\phi = \psi_0 - \psi_1$. Open circles: $p_0 = 1$; Bullets: $p_0 = 0.5$; Open squares: $p_0 = 0.25$. Lines represent the results of quantum theory [6].

4.2. Beam splitter

We now show that two $K = 4$ DLMs and two passive devices that perform a plane rotation by 45° are sufficient to build a network that behaves as if it where a single-photon beam splitter. First we describe the network and then we demonstrate that it acts as a beam splitter.

The network shown in Fig. 8 has two input channels (0 and 1) and two output channels (0 and 1). The network receives events at one of the two input channels. Each input event carries information in the form of a two-dimensional unit vector. Either input channel 0 receives $(y_{1,n+1}, y_{2,n+1})$ or input channel 1 receives $(y_{3,n+1}, y_{4,n+1})$. The input is fed into the device described in Section 2.5. The purpose of this front-end DLM is to transform the information contained in two-dimensional input vectors (of which only one is present for any given input event), into a four-dimensional unit vector. The four-dimensional internal vector of this device is split into two groups of two-dimensional vectors $(\widetilde{x}_{1,n+1}, \widetilde{x}_{4,n+1})$ and $(\widetilde{x}_{3,n+1}, \widetilde{x}_{2,n+1})$ and each of these two-dimensional vectors is rotated by 45°. Put differently, the four-dimensional vector is rotated once in the $(1, 4)$-plane about 45° and once in the $(3, 2)$-plane about 45°. The order of the rotations is irrelevant. The resulting four-dimensional vector is then sent to the input of a second $K = 4$ DLM. This back-end DLM sends $(x_{1,n+1}, x_{2,n+1})/\sqrt{x_{1,n+1}^2 + x_{2,n+1}^2}$ through output channel 0 if it used rule $j = 1, 2$ (see Eq. (17)) to update its internal state. Otherwise it sends $(x_{3,n+1}, x_{4,n+1})/\sqrt{x_{3,n+1}^2 + x_{4,n+1}^2}$ through output channel 1.

The operation of the network depicted in Fig. 8 can be analyzed analytically if we disregard transient effects and assume that the information carried by events on channel 0 (1) is given by $\mathbf{y}_{n+1} = \mathbf{y} = (y_1, y_2)$ ($\mathbf{y}'_{n+1} = \mathbf{y}' = (y_3, y_4)$). We denote by p the number of events on input channel 0 divided by the total number of events. Then, the number of events on input channel 1 is given by $1 - p$.

In the stationary regime, the internal state $(\widetilde{x}_{1,n+1}, \widetilde{x}_{2,n+1}, \widetilde{x}_{3,n+1}, \widetilde{x}_{4,n+1})$ of the front-end DLM (see Fig. 8) learns $(w_1, w_2, w_3, w_4) = (y_1\sqrt{p}, y_2\sqrt{p}, y_3\sqrt{1-p}, y_4\sqrt{1-p})$. Carrying out the two plane rotations of 45° we see that the back-end DLM receives as input the four-dimensional vector $(w_1 - w_4, w_3 + w_2, w_3 - w_2, w_1 + w_4)/\sqrt{2}$. In the stationary regime, the internal vector $(x_{1,n+1}, x_{2,n+1}, x_{3,n+1}, x_{4,n+1})$ of the back-end DLM oscillates about $(w_1 - w_4, w_3 + w_2, w_3 - w_2, w_1 + w_4)/\sqrt{2}$. Therefore, in the stationary regime and for fixed two-dimensional vectors on input channels 0 and 1, the input–output relation of the BS network of Fig. 8 can be

written as

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} \xrightarrow{\text{BS}} \frac{1}{\sqrt{2}} \begin{pmatrix} w_1 - w_4 \\ w_3 + w_2 \\ w_3 - w_2 \\ w_1 + w_4 \end{pmatrix}. \tag{26}$$

Using two complex numbers instead of four real numbers Eq. (26) can also be written as

$$\begin{pmatrix} w_1 + i w_2 \\ w_3 + i w_4 \end{pmatrix} \xrightarrow{\text{BS}} \frac{1}{\sqrt{2}} \begin{pmatrix} w_1 - w_4 + i(w_3 + w_2) \\ w_3 - w_2 + i(w_1 + w_4) \end{pmatrix}. \tag{27}$$

In quantum theory [6] the presence of photons in the input modes 0 or 1 is represented by the probability amplitudes (a_0, a_1) [17,20–22]. According to quantum theory [6], the probability amplitudes (b_0, b_1) of the photons in the output modes 0 and 1 of a beam splitter are given by [17,20–22]

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \begin{pmatrix} a_0 + i a_1 \\ a_1 + i a_0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}. \tag{28}$$

Identifying a_0 with $w_1 + i w_2 = (y_1 + i y_2)p$ and a_1 with $w_3 + i w_4 = (y_3 + i y_4)(1 - p)$ it is clear that by construction, the DLM network in Fig. 8 will allow us to simulate a beam splitter, not by calculating the amplitudes (28) but by a deterministic event-by-event simulation.

In Fig. 8 (right) we present results of discrete-event simulations using the DLM network depicted in Fig. 8 (left). Before the simulation starts, the internal vectors of the DLMs are given a random value (on the unit sphere). Each data point represents 10 000 events. All these simulations were carried out with $\alpha = 0.99$. For each set of 10 000 events, a uniform random number in the range $[0, 360]$ generates two angles ψ_0 and ψ_1. Input channel 0 receives $(y_{1,n+1}, y_{2,n+1}) = (\cos\psi_0, \sin\psi_0)$ with probability p_0. Input channel 1 receives $(y_{3,n+1}, y_{4,n+1}) = (\cos\psi_1, \sin\psi_1)$ with probability $p_1 = 1 - p_0$. Random processes only enter in the procedure to generate the input data. The DLM network processes the events sequentially and deterministically. From Fig. 8 it is clear that the output of the deterministic DLM-based beam splitter reproduces the probability distributions as obtained from quantum theory [6].

4.3. Mach–Zehnder interferometer

In quantum physics [6], single-photon experiments with one beam splitter provide direct evidence for the particle-like behavior of photons [4,20]. The wave mechanical character appears when one performs single-particle interference experiments. In this subsection we construct a DLM network that displays the same interference patterns as those observed in single-photon Mach–Zehnder interferometer experiments [20].

The schematic layout of the DLM network is shown in Fig. 9. Not surprisingly, it is exactly the same as that of a real Mach–Zehnder interferometer. The BS network described in the previous subsection is used for the beam splitters. The phase shift is taken care of by a passive device that performs a plane rotation. Clearly there is a one-to-one mapping from each relevant component in the interferometer to a processing unit in the DLM network. Recall that the processing units in the DLM network only communicate with each other through the message (photon) that propagates through the network.

According to quantum theory [6], the probability amplitudes (b_0, b_1) of the photons in the output modes 0 (N_2) and 1 (N_3) of the Mach–Zehnder interferometer are given by [17,20–22]

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \begin{pmatrix} e^{i\phi_0} & 0 \\ 0 & e^{i\phi_1} \end{pmatrix} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}. \tag{29}$$

Note that in a quantum mechanical setting it is impossible to simultaneously measure $(N_0/(N_0 + N_1), N_1/(N_0 + N_1))$ and $(N_2/(N_0 + N_1), N_3/(N_0 + N_1))$: Photon detectors operate by absorbing photons. However, in our deterministic, event-based simulation there is no such problem.

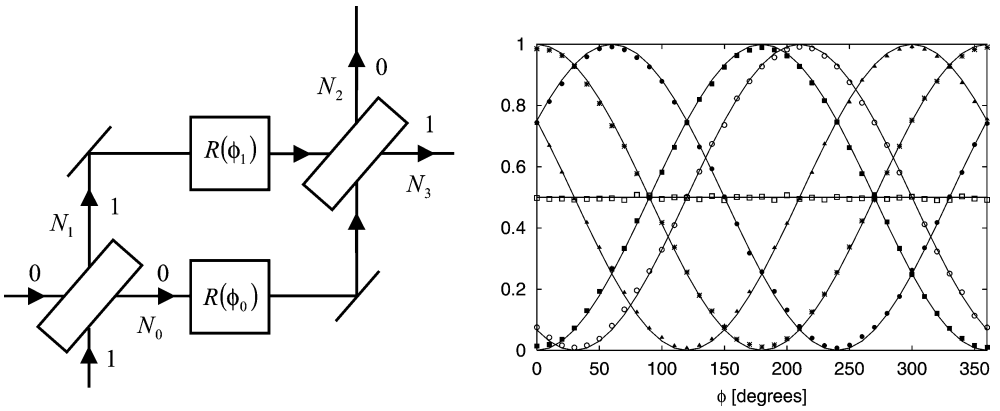

Fig. 9. *Left*: Diagram of a DLM network that simulates a single-photon Mach–Zehnder interferometer on an event-by-event basis [19]. The DLM network consists of two BS devices (see Fig. 8 (left)) and two passive devices ($R(\phi_0)$ and $R(\phi_1)$) that perform plane rotations by ϕ_0 and ϕ_1, respectively. There is a one-to-one correspondence between the elements of a physical Mach–Zehnder interferometer [18,20] and the units in the DLM network. The number of events N_i in channel $i = 0, \ldots, 3$ corresponds to the probability for finding a photon on the corresponding arm of the interferometer. *Right*: Simulation results for the DLM-network shown on the left. Input channel 0 receives $(y_{1,n+1}, y_{2,n+1}) = (\cos\psi_0, \sin\psi_0)$ with probability one. A uniform random number in the range $[0, 360]$ is used to choose the angle ψ_0. Input channel 1 receives no events. Each data point represents 10 000 events ($N_0 + N_1 = N_2 + N_3 = 10\,000$). Initially the rotation angle $\phi_0 = 0$ and after each set of 10 000 events, ϕ_0 is increased by 10°. Markers give the simulation results for the normalized intensities as a function of $\phi = \phi_0 - \phi_1$. Open squares: $N_0/(N_0 + N_1)$; Solid squares: $N_2/(N_2 + N_3)$ for $\phi_1 = 0$; Open circles: $N_2/(N_2 + N_3)$ for $\phi_1 = 30^\circ$; Bullets: $N_2/(N_2 + N_3)$ for $\phi_1 = 240^\circ$; Asterisks: $N_3/(N_2 + N_3)$ for $\phi_1 = 0$; Solid triangles: $N_3/(N_2 + N_3)$ for $\phi_1 = 300^\circ$. Lines represent the results of quantum theory [6].

In Fig. 9 we present a small selection of simulation results for the Mach–Zehnder interferometer built from DLMs. We assume that input channel 0 receives $(y_{1,n+1}, y_{2,n+1}) = (\cos\psi_0, \sin\psi_0)$ with probability one and that input channel 1 receives no events. This corresponds to $(a_0, a_1) = (\cos\psi_0 + i\sin\psi_0, 0)$. We use uniform random numbers to determine ψ_0. In all these simulations $\alpha = 0.99$. The data points are the simulation results for the normalized intensity $N_i/(N_0 + N_1)$ for $i = 0, 2, 3$ as a function of $\phi = \phi_0 - \phi_1$. Lines represent the corresponding results of quantum theory [6]. From Fig. 9 it is clear that quantum theory provides an excellent description of the deterministic, event-based processing by the DLM network.

The examples presented in Fig. 9 do not rule out that there may be settings for the angles ψ_0, ϕ_0 and ϕ_1 for which quantum theory fails to give a good description of the behavior of the DLM network. However extensive series of simulations show that this is not the case. Instead of presenting the results of these simulations we will demonstrate that quantum theory [6] also describes the stationary-state input–output behavior of more extended DLM networks.

As an example we consider the DLM network depicted in Fig. 10. Obviously this network maps exactly onto two chained Mach–Zehnder interferometers [19]. Now there are seven parameters p_0, ψ_0, ψ_1, ϕ_0, ϕ_1, ϕ_2, and ϕ_3 that may be varied, so simply plotting selected cases is not the proper procedure to establish that quantum theory describes the stationary-state behavior of the DLM network. Therefore we adopt the following strategy. For each set of 10 000 events, we use seven random numbers to fix the parameters p_0, ψ_0, ψ_1, ϕ_0, ϕ_1, ϕ_2, and ϕ_3. Then we collect the data for these 10 000 events and compare the intensity in output channel 0 (N_4) and 1 (N_5) with the corresponding results of quantum theory [6]. The latter is given by

$$\begin{pmatrix} b_0 \\ b_1 \end{pmatrix} = \frac{1}{2\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \begin{pmatrix} e^{i\phi_2} & 0 \\ 0 & e^{i\phi_3} \end{pmatrix} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \begin{pmatrix} e^{i\phi_0} & 0 \\ 0 & e^{i\phi_1} \end{pmatrix} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}. \tag{30}$$

For each choice of $\{p_0, \psi_0, \psi_1, \phi_0, \phi_1, \phi_2, \phi_3\}$ we compute the differences $||b_0|^2 - N_4/(N_4 + N_5)|$ and $||b_1|^2 - N_5/(N_4 + N_5)|$. N_4 (N_5) is the number of events in output channel 0 (1) of the third beam splitter. $N_0 + N_1 =$

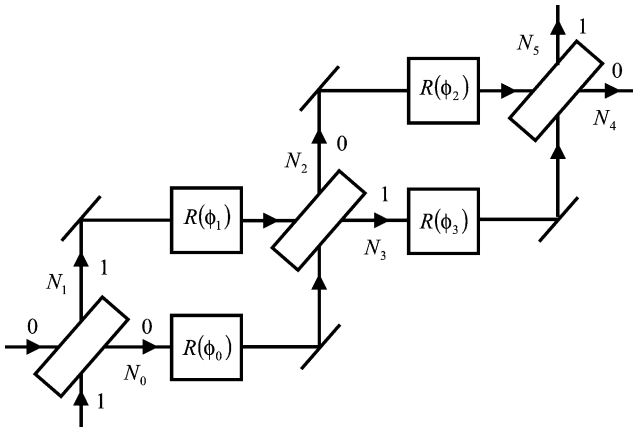

Fig. 10. Diagram of a DLM network that simulates single-photon propagation through two chained Mach–Zehnder interferometers on an event-by-event basis.

$N_2 + N_3 = N_4 + N_5$ is the total number of events (10 000 in this case). In Fig. 11 we show $||b_0|^2 - N_4/(N_4 + N_5)|$ as a function of p_0, $\psi_0 - \psi_1$, $\phi_0 - \phi_1$, and $\phi_2 - \phi_3$. In all these simulations $\alpha = 0.99$. Once again it is clear that quantum theory [6] provides a very good description of a DLM-based simulation of two chained Mach–Zehnder interferometers.

4.4. Technical note

All simulations that we presented in this section have been performed for $\alpha = 0.99$. From the description of the learning process it is clear that α controls the rate of learning or, equivalently, the rate at which learned information can be forgotten. Furthermore it is evident that the difference between a constant input to a DLM and the learned value of its internal variable cannot be smaller than $1 - \alpha$. In other words, α also limits the precision with which the internal variable can represent a sequence of constant input values. On the other hand, the number of events has to balance the rate at which the DLM can forget a learned input value. The smaller $1 - \alpha$ is, the larger the number of events has to be for the DLM to adapt to changes in the input data.

We use the last example of Section 4.3 to illustrate the effect of changing α and the total number of events N. In Fig. 12 we show the results of repeating the procedure used to obtain the data shown in Fig. 11 but instead of $\alpha = 0.99$ and $N = 10\,000$ events per data point, we used $\alpha = 0.9999$ and $N = 1\,000\,000$ event per data point. As expected, the difference between the simulation data and the results of quantum theory decreases if $1 - \alpha$ decreases and N increases accordingly. Comparing Fig. 11 with Fig. 12 it is clear that the decrease of this difference is roughly proportional to the inverse of the square root of the number of events. Note that each data point in Fig. 11 is generated without the use of random processes.

4.5. Stochastic learning machines

In the stationary regime, the sequence of messages that a DLM (network) generates is strictly deterministic. For some applications, e.g., for quantum physics [6], it may be desirable to randomize these sequences. A marginal modification turns a DLM into a stochastic learning machine (SLM). Here the term *stochastic* does not refer to the learning process but to the method that is used to select the output channel that will carry the outgoing message.

In the stationary regime the components of the internal vector represent the probability amplitudes. Comparing the (sums of) squares of these amplitudes with a uniform random number $0 < r < 1$ gives the probability for sending the message over the corresponding output channel. For instance, in the case of the beam splitter BS

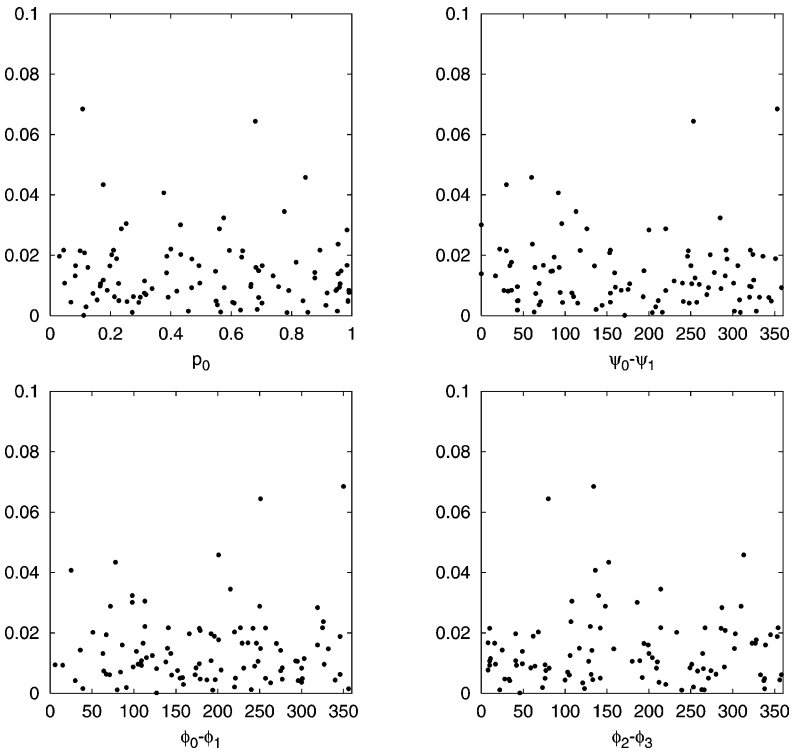

Fig. 11. Absolute value of the difference between the normalized intensity $N_4/(N_4 + N_5)$ in output channel 0 of the event-based DLM simulation and the result of quantum theory [6] for the system of two chained Mach–Zehnder interferometers shown in Fig. 10 [19]. Input channel 0 receives $(y_{1,n+1}, y_{2,n+1}) = (\cos\psi_0, \sin\psi_0)$ with probability p_0. Input channel 1 receives $(y_{3,n+1}, y_{4,n+1}) = (\cos\psi_1, \sin\psi_1)$ with probability $1 - p_0$. For each event a uniform random number in the range $[0, 360]$ determines ψ_0 or ψ_1. Each data point represents a simulation of 10 000 events ($N_0 + N_1 = N_2 + N_3 = N_4 + N_5 = 10\,000$). *Top-left*: Difference as a function of p_0; *Top-right*: Difference as a function of $\psi_0 - \psi_1$; *Bottom-left*: Difference as a function of $\phi_0 - \phi_1$; *Bottom-right*: Difference as a function of $\phi_2 - \phi_3$.

(see Fig. 8) we replace the back-end DLM by a SLM. This SLM will send a message over output channel 0 if $x^2_{1,n+1} + x^2_{2,n+1} \leqslant r$. Otherwise it will activate output channel 1. Although the learning process of this modified BS network is still deterministic, in the stationary regime the output messages are randomly distributed over the two output channels. Of course, the distribution of output messages is the same as that of the original DLM-network.

Replacing DLMs by SLMs in a DLM-network changes the order in which messages are being processed by the network but leaves the content of the messages intact. Therefore, in the stationary regime, the distribution of messages over the outputs of the SLM-network is essentially the same as that of the original DLM network.

As an illustration of the use of SLMs, we replace the two back-end DLMs in the Mach–Zehnder interferometer network (see Fig. 9 (left)) by their "randomized" version and repeat the procedure that generates the data of Fig. 9 (right). The results of these simulations are shown in Fig. 13. Not unexpectedly, the randomness in the output channel selection is reflected by a (small) increase of the scatter on the data points. In this simulation, the output channels 0 and 1 of each beam splitter are activated in a random manner and the functional dependence of $N_0/(N_0 + N_1)$, $N_1/(N_0 + N_1)$, $N_2/(N_2 + N_3) = N_2/(N_0 + N_1)$ and $N_3/(N_2 + N_3)$ on ϕ is still in full agreement with quantum theory [6]. In other words, this SLM-network performs a genuine, event-by-event simulation of the ideal (perfect detectors, etc.) version of both the single-photon beam splitter and Mach–Zehnder interferometer experiments by Grangier et al. [20].

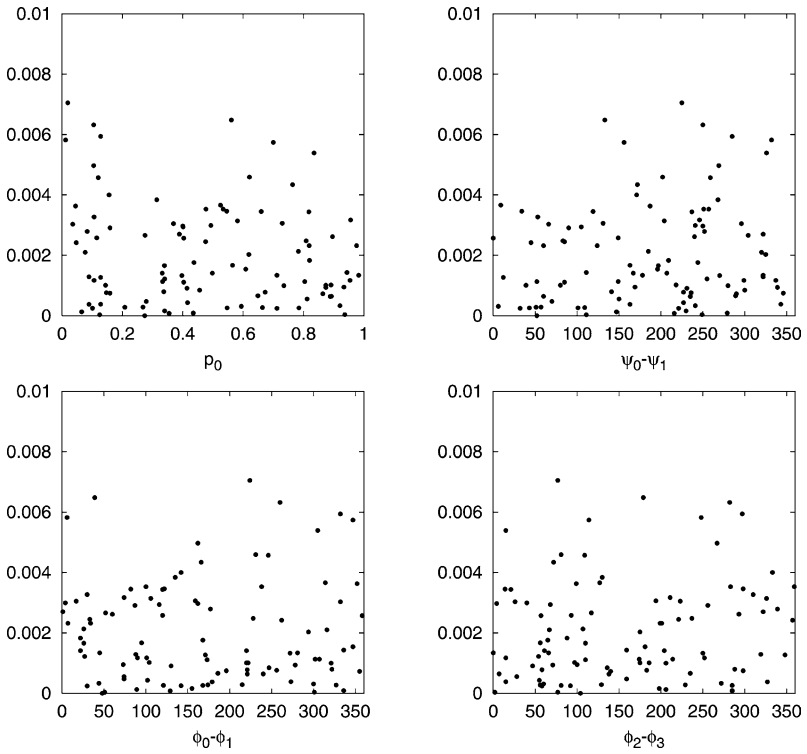

Fig. 12. Same as Fig. 11 except that $\alpha = 0.9999$ (instead of $\alpha = 0.99$) and that the 1 000 000 events (instead of 10 000) per data point were processed by the DLM network depicted in Fig. 10.

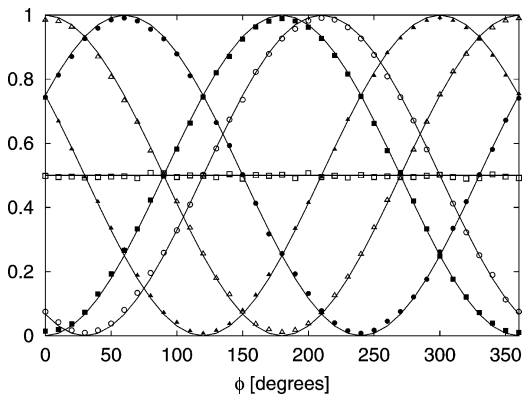

Fig. 13. Simulation results for a Mach–Zehnder interferometer built from SLMs instead of DLMs. Each beam splitter sends messages over its output channels 0 and 1 in a random manner. The simulation procedure and annotations are exactly the same as in Fig. 9.

5. Discussion

We have proposed a new procedure to construct deterministic algorithms that have primitive learning capabilities. We have used these algorithms to build deterministic learning machines (DLMs). A DLM learns by processing event after event but does not store the data contained in an individual event. Connecting the input of a DLM to the

output of another DLM yields a locally connected network of DLMs. A DLM within the network locally processes the information contained in an event and responds by sending a message that may be used as input for another DLM. A distinct feature of a DLM network is that at any given time, only one event (message) is propagating through the network. The DLMs process messages in a sequential manner and only communicate with each other by message passing.

We have demonstrated that DLM networks can discover relationships between successive events (see Section 3) and that certain classes of DLM networks exhibit behavior that is usually only attributed to quantum systems. In Section 4, we have presented DLM networks that simulate quantum interference on an event-by-event basis. More specifically, we map each physical part of the real Mach–Zehnder interferometer onto a DLM and the messages (phase shifts in this case) are carried by photons. No ingredient other than simple geometry is used to specify the update rules of the DLMs.

As the network processes event after event, the network generates output events that build an interference pattern that is described by the quantum theory [6] of the single-photon beam splitter and Mach–Zehnder interferometer. To illustrate that DLM networks are indeed capable of simulating quantum interference on an event-by-event basis we also simulate an experiment involving three beam splitters (i.e. two chained Mach–Zehnder interferometers) and demonstrate that quantum theory [6] also describes the behavior of this network.

We emphasize that our approach is not a proposal for another interpretation of quantum mechanics. Our approach is not an extension of quantum theory in any sense, on the contrary. The probability distributions of quantum theory appear as the result of a deterministic, causal learning process, and not vice versa (see Section 4) [11]. Our results suggest that quantum mechanical behavior may originate from an underlying deterministic process [23,24]. Indeed, it is somewhat ironic that in order to mimic the apparent randomness with which events are observed in experiments, we have to explicitly randomize the output of the DLMs to mask the underlying deterministic processes (see Section 4.5).

At this point it may be worthwhile to recall what a DLM actually does. In a simple physical picture, a DLM is a device (e.g., beam splitter, polarizer) that exchanges information with the particles that pass through it. The DLM tries to do this in an effective manner. It learns by comparing the message carried by an event with predictions based on the knowledge acquired by the DLM during the processing of previous events. Effectively this comparison amounts to a minimization of the squared error (see Section 2). Schrödinger used exactly the same principle to derive his famous equation [25] but called this approach “unverständlich” in a subsequent publication [26].

The results presented in Section 4 suggest that we may have discovered a systematic procedure to construct algorithms that simulate quantum phenomena using deterministic, local, and event-by-event-based processes, without using concepts such as wave fields or particle-wave duality. In fact, elsewhere we show that the approach introduced in this paper can be employed to perform event-based simulations of a universal quantum computer [27,28]. As it has been shown that the time evolution of the wave function of a quantum system can be simulated on a quantum computer [22,29], it should be possible, at least in principle, to compute the real-time dynamics of these systems through event-by-event simulation using DLM networks.

Acknowledgements

We are grateful to Professors M. Imada, S. Miyashita, and M. Suzuki for many useful comments on the principles of the simulation method described in this paper.

References

[1] D.P. Landau, K. Binder, A Guide to Monte Carlo Simulation in Statistical Physics, Cambridge University Press, Cambridge, 2000.
[2] A. Tonomura, The Quantum World Unveiled by Electron Waves, World Scientific, Singapore, 1998.

[3] In this paper we disregard limitations of real experiments such as detector efficiency, imperfection of the source, biprism, etc.
[4] D. Home, Conceptual Foundations of Quantum Physics, Plenum Press, New York, 1997.
[5] N.G. Van Kampen, Physica A 153 (1988) 97.
[6] We make a distinction between quantum theory and quantum physics. We use the term *quantum theory* when we refer to the mathematical formalism, i.e. the postulates of quantum mechanics (with or without the wave function collapse postulate) [8] and the rules (algorithms) to compute the wave function. The term *quantum physics* is used for microscopic, experimentally observable phenomena that do not find an explanation within the mathematical framework of classical mechanics.
[7] R.P. Feynman, R.B. Leighton, M. Sands, The Feynman Lectures on Physics, vol. 3, Addison-Wesley, Reading MA, 1996.
[8] L.E. Ballentine, Quantum Mechanics: A Modern Development, World Scientific, Singapore, 2003.
[9] H. De Raedt, Computer simulation of quantum phenomena in nano-scale devices, in: D. Stauffer (Ed.), Annual Reviews of Computational Physics IV, World Scientific, 1996, p. 107.
[10] A large collection of video's of such simulations can be found at http://www.compphys.org/quantummechanics.
[11] R. Penrose, The Emperor's New Mind, Oxford University Press, Oxford, 1990.
[12] S. Haykin, Neutral Networks, Prentice-Hall, New Jersey, 1999.
[13] S. Haykin, Adaptive Filter Theory, Prentice-Hall, New Jersey, 1986.
[14] K.V. Mardia, J.T. Kent, J.M. Bibby, Multivariate Analysis, Academic Press, London, 1982.
[15] G.H. Golub, C.F. Van Loan, Matrix Computations, John Hopkins University Press, Baltimore, MD, 1996.
[16] R.P. Feynman, Int. J. Theor. Phys. 21 (1982) 467.
[17] G. Baym, Lectures on Quantum Mechanics, W.A. Benjamin, Reading, MA, 1974.
[18] M. Born, E. Wolf, Principles of Optics, Pergamon, Oxford, 1964.
[19] Sample Fortran and Java programs and interactive programs that perform event-based simulations of a beam splitter, one Mach–Zehnder interferometer, and two chained Mach–Zehnder interferometers can be found at http://www.compphys.net/dlm.
[20] P. Grangier, R. Roger, A. Aspect, Europhys. Lett. 1 (1986) 173.
[21] J.G. Rarity, P.R. Tapster, Phil. Trans. Roy. Soc. London A 355 (1997) 2267.
[22] M. Nielsen, I. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, 2000.
[23] G. 't Hooft, Determinism beneath quantum mechanics, quant-ph/0212095.
[24] G. 't Hooft, Quantum mechanics and determinism, hep-th/0105105.
[25] E. Schrödinger, Ann. Phys. 79 (1926) 361.
[26] E. Schrödinger, Ann. Phys. 79 (1926) 491.
[27] H. De Raedt, K. De Raedt, and K. Michielsen, New method to simulate quantum interference using deterministic processes and application to event-based simulation of quantum computation, J. Phys. Soc. Jpn., in press.
[28] K. Michielsen, K. De Raedt, and H. De Raedt, Simulation of Quantum Computation: A deterministic event-based approach, J. Comp. Theor. Nanoscience, in press.
[29] C. Zalka, Proc. Roy. Soc. London A 454 (1998) 313.