

Time-domain and frequency-domain modeling of nonlinear optical components at the circuit-level using a node-based approach

Martin Fiers,^{1,4,*} Thomas Van Vaerenbergh,^{1,4} Ken Caluwaerts,² Dries Vande Ginste,³
Benjamin Schrauwen,² Joni Dambre,² and Peter Bienstman^{1,4}

¹Photonics Research Group (INTEC), Ghent University—imec, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

²Electronics and Information Systems (ELIS), Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

³Electromagnetics Group (INTEC), Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

⁴Center for Nano- and Biophotonics (NB-Photonics), Ghent University,

Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

*Corresponding author: martin.fiers@intec.ugent.be

Received October 4, 2011; revised December 7, 2011; accepted December 19, 2011;
posted December 19, 2011 (Doc. ID 155249); published April 4, 2012

We present a tool that aids in the modeling of optical circuits, both in the frequency and in the time domain. The tool is based on the definition of a node, which can have both an instantaneous input-output relation and different state variables (e.g., temperature and carrier density) and differential equations for these states. Furthermore, each node has access to part of its input history, allowing the creation of delay lines or digital filters. Additionally, a node can contain subnodes, allowing the creation of hierarchical networks. This tool can be used in numerous applications such as frequency-domain analysis of optical ring filters, time-domain analysis of optical amplifiers, microdisks, and microcavities. Although we mainly use this tool to model optical circuits, it can also be used to model other classes of dynamical systems, such as electrical circuits and neural networks. © 2012 Optical Society of America

OCIS codes: 190.4390, 220.4830, 130.3120.

1. INTRODUCTION

There is a variety of tools available to simulate the behavior of optical components. Many of these tools are limited to small networks of only a handful of components. To model these components or small networks, different techniques are used, such as finite difference time domain (FDTD) (e.g., MEEP [1,2]), eigenmode expansion, time-domain traveling wave (TDTW) [3], split step methods (SSM) [4], and coupled mode theory (CMT). These tools mostly differ in complexity and in the level to which they contain physical details. FDTD, for example, is based on Maxwell's equations and therefore is a full-wave optical solver. The major drawback is that FDTD is computationally very expensive. On the other side, CMT is an approximate description, but extremely elegant and fast, only needing a few variables to describe a complex system. This is achieved by eliminating all spatial dependencies in the physical problem.

In this software landscape, there are tools to design complex optical circuits consisting of many components. For example, ASPIC [5] is used for calculating the steady-state response of optical circuits, and VPI [6] is mainly used to study the time-domain evolution. PicWAVE uses a TDTW optical model [7], and RSoft Optsim uses SSM [8]. There are also approaches that use modified node analysis (MNA), such as OptiSPICE [9,10], which allows simulation of mixed electronic and optical circuits. All of these new tools will become indispensable in the future when designing and optimizing large optical circuits.

In this paper, we present a different node-based approach. The advantage of our approach is that both time and frequency domain can be investigated in the same framework,

and that each component can be represented in a natural way using variables such as the optical field, the temperature, and the carrier density, without needing to be mapped on to voltage or current such as in the MNA approach. It uses only a small set of variables per component, similar to CMT, which means the simulations are extremely fast compared to other methods such as FDTD, TDTW, and SSM, with the drawback of losing accuracy. Also, we provide a mechanism to eliminate instantaneous components from the network, reducing the amount of components we need to simulate in time domain. Our tool, named CAPHE [11], can also be used to simulate novel computational systems such as photonic reservoirs [12]. It is written in C++ for optimal performance, with a Python front-end for ease of use and interfacing to a large collection of scientific libraries.

The rest of the paper is structured as follows: first we define a component with its basic properties such as the scatter matrix and the ordinary differential equations (ODEs). After that, we create a circuit consisting of several nodes, and we explain how we can derive the total scatter matrix from this. We then explain how scalable the software tool is, and how we can eliminate linear, instantaneous nodes from the circuit. After that we compare the accuracy of the simplified methods (used in this framework) with very accurate methods, and we conclude with an illustration of how eliminating nodes can speed up simulations considerably in the time domain.

2. MODEL

To design a complex circuit simulator, we first need to define the behavior of one component. A node consists of N ports;

see Fig. 1. A linear instantaneous transmission between port $s_{in,i}$ and $s_{out,j}$ is defined through the scatter matrix S_{ij} . Two optional time-domain descriptions can be added to enrich this component (see Fig. 1, bottom). First, one can add a *buffer* to store the inputs $s_{in,i}$ at previous timesteps. This can be used if one wishes to model a delayed waveguide or a digital filter. Second, we can add internal *states* to the node. This can be used to describe the rate equations of, e.g., a laser or the complex amplitude of a resonator. We use a set of ODEs to describe the component in terms of its internal variables. There is no restriction on the form of the equations, so highly nonlinear components can be easily modeled. Because of these two additions, the *output* $s_{out,i}$ is now a sum of the linear part and a term describing the nonlinear character of the component.

The main novelty of our framework is that the frequency-domain S-matrix for components that only have a linear instantaneous transmission can be used to significantly speed up time-domain simulations of networks with both instantaneous and noninstantaneous components. In the next two sections, we give more insight into the frequency and time-domain characteristics of these nodes.

A. Steady-state equations

As previously mentioned, a linear instantaneous component can be described by a single scatter matrix S , and a linear input-output relation, $s_{out,i}(t) = \sum_j S_{ij} s_{in,j}(t)$. Using this relationship, we can describe all linear phenomena in the frequency domain. For example, a ring resonator can be modeled by combining a directional coupler and a waveguide. Both have a very simple scatter matrix, but when combined, they create resonances at certain frequencies. As these nodes contain no memory, we call them memoryless (ML) nodes.

B. Time-domain equations

We represent time-domain signals as complex amplitudes $s(t)$, modulating a carrier frequency ω_c . The actual input at each port is then

$$E(t) = s(t)e^{j\omega_c t} + c.c. \quad (1)$$

Representing the signal by $s(t)$ is beneficial from a numerical point of view, as we can now integrate over $s(t)$, which varies much slower than $E(t)$. Obviously, as the bandwidth of the input signal increases, we will need more samples per time unit to correctly simulate the system.

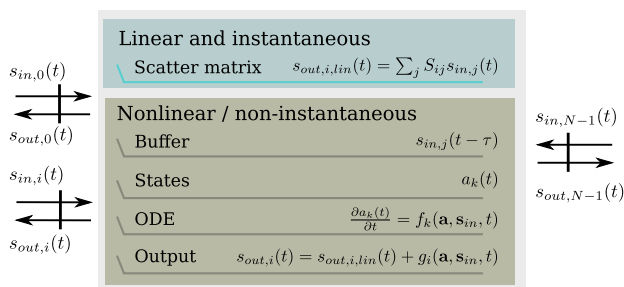


Fig. 1. (Color online) Structure of a node with N ports. A linear and instantaneous node is only described by a scatter matrix S . States (e.g., temperature and free carriers) can be added. In this case the node becomes nonlinear.

For each component, we can now optionally add time-domain equations, which leads in its most general form to an input-output relation of the following form:

$$s_{out,i}(t) = \sum_j S_{ij} s_{in,j}(t) + s_{ext,i}. \quad (2)$$

Here, $s_{ext,i}$ is a generalized source term. For example, for a continuous wave source, $s_{ext} = A$, where A is the complex amplitude of the source. For a waveguide with delay τ , the simple relation $s_{out,i}(t) = s_{ext,i} = B s_{in,1-i}(t - \tau)$, for $i \in [0, 1]$, holds. Here, B is the complex value that determines the loss and phase change of the waveguide. Note that for this waveguide, there is no longer an instantaneous behavior; i.e. S_{ij} is zero in Eq. (2).

Differential equations can be added to describe the evolution of some internal variables, e.g., temperature and free carriers in a laser, as a function of time and inputs.

As soon as there is a source term present in Eq. (2), the component is not instantaneous anymore. We call these nodes memory-containing (MC) nodes (Fig. 1, bottom), as opposed to the ML nodes. Depending on whether the delays in a waveguide are important for a simulation, one can model them with delay (which makes it MC), or without delay (as an ML component), the latter having the advantage that we can eliminate it from the total network. This is explained in the next section.

3. CIRCUIT SCATTER MATRIX

We use the node from Fig. 1 as a basic building block to create an optical circuit. From this circuit, a total scatter matrix can be calculated in the frequency domain that describes the transmission to and from ports in the network. This matrix can become very big. Hence, in this section we also derive techniques to eliminate the ML nodes from the circuit. This is a crucial feature in our approach, which reduces the number of variables needed in the time domain, hence improving the simulation speed. To do this, we split the input/output vector $\mathbf{s}_{in/out}(t)$ into $\mathbf{s}_{in/out,MC}(t)$ and $\mathbf{s}_{in/out,ML}(t)$, the input and output, respectively, to MC and ML nodes. For simplicity we will omit the time dependency in the following equations. The size of this \mathbf{s} -vector equals the total number of ports in the circuit.

We can describe the way the different components are connected as follows:

$$\begin{pmatrix} \mathbf{s}_{in,MC} \\ \mathbf{s}_{in,ML} \end{pmatrix} = \mathbf{C}_{tot} \begin{pmatrix} \mathbf{s}_{out,MC} \\ \mathbf{s}_{out,ML} \end{pmatrix} = \begin{pmatrix} \mathbf{C}_{MC,MC} & \mathbf{C}_{MC,ML} \\ \mathbf{C}_{ML,MC} & \mathbf{C}_{ML,ML} \end{pmatrix} \begin{pmatrix} \mathbf{s}_{out,MC} \\ \mathbf{s}_{out,ML} \end{pmatrix}. \quad (3)$$

Here, $\mathbf{C}_{tot,ij}$ is a binary connection matrix that only contains a 1 if port i is connected to port j . As a consequence, \mathbf{C}_{tot} is symmetric and contains at most one element per row and at most one element per column, with zeros on the diagonal.

The behavior of each of the individual nodes can be described by the following equations:

$$\mathbf{s}_{out,ML} = \mathbf{S}_{ML,ML} \mathbf{s}_{in,ML}, \quad (4)$$

$$\mathbf{s}_{out,MC} = \mathbf{S}_{MC,MC} \mathbf{s}_{in,MC} + \mathbf{s}_{ext,MC}, \quad (5)$$

in which we define the scatter matrices $\mathbf{S}_{ML,ML}$ and $\mathbf{S}_{MC,MC}$. These are block diagonal matrices, with each block

representing the scatter matrix from an ML (resp. MC) node. The second term in Eq. (5), $s_{\text{ext},\text{MC}}$, is the generalized source term described earlier; see also Eq. (2).

With all equations above, we can derive the input at the active ports, given only $s_{\text{ext},\text{MC}}$. This is done as follows: replace $s_{\text{out},\text{ML}}$ in Eq. (3) using Eq. (4). Solve this system for $s_{\text{in},\text{MC}}$. This gives

$$\begin{aligned} s_{\text{in},\text{MC}} &= (\mathbf{C}_{\text{MC},\text{MC}} + \mathbf{C}_{\text{MC},\text{ML}} \mathbf{S}_{\text{ML},\text{ML}} (\mathbf{I} - \mathbf{C}_{\text{ML},\text{ML}} \mathbf{S}_{\text{ML},\text{ML}})^{-1} \mathbf{C}_{\text{ML},\text{MC}}) s_{\text{out},\text{MC}} \\ &= \mathbf{C} s_{\text{out},\text{MC}}. \end{aligned} \quad (6)$$

We then substitute (5) in the equation above. This yields

$$s_{\text{in},\text{MC}} = (\mathbf{I} - \mathbf{C} \mathbf{S}_{\text{MC},\text{MC}})^{-1} \mathbf{C} s_{\text{ext},\text{MC}} = \mathbf{S} s_{\text{ext},\text{MC}}. \quad (7)$$

We have now successfully eliminated the ML nodes and end up with a smaller scatter matrix \mathbf{S} of the network.

Note that the matrix inversion in Eq. (6) is of a special type: $\mathbf{C}_{\text{ML},\text{ML}}$ only permutes the elements of $\mathbf{S}_{\text{ML},\text{ML}}$, and $\mathbf{S}_{\text{ML},\text{ML}}$ is a block diagonal matrix. The resulting matrix $\mathbf{I} - \mathbf{C}_{\text{ML},\text{ML}} \mathbf{S}_{\text{ML},\text{ML}}$ is therefore sparse. However, the resulting matrix after inversion is not always sparse. This depends on the topology of the original network and on the individual scatter matrices of the ML nodes.

The final ingredient in our model is the internal states, which are stored in the total variable vector $\mathbf{a}(t)$.

$$\frac{d\mathbf{a}(t)}{dt} = \mathbf{f}(\mathbf{a}, s_{\text{in}}, t). \quad (8)$$

This ODE can now be solved easily as we can evaluate \mathbf{f} as follows: at each timestep, we loop over all MC to calculate s_{ext} . Then we can calculate s_{in} from Eq. (7). With this, we can evaluate $f_k(\mathbf{a}, s_{\text{in}}, t) = \frac{da_k(t)}{dt}$.

4. OPTIMIZATIONS IN THE FREQUENCY DOMAIN

In Eqs. (6) and (7), we need to solve a system of equations. For example, in Eq. (7), we need to solve

$$(\mathbf{I} - \mathbf{C} \mathbf{S}_{\text{MC},\text{MC}}) \mathbf{X} = \mathbf{C} \quad (9)$$

for \mathbf{X} . This can be done by first doing an LU factorization, followed by forward and backward substitution to find \mathbf{X} . A similar reasoning is done for the inversion in Eq. (6). Solving a system is almost always preferred above matrix inversion in terms of speed and stability. Optionally, since these matrices are sparse, we can use Clark Kent LU (KLU) to solve this system very efficiently [13–15].

To benchmark the speed-up, we consider the use case of a coupled resonator optical waveguide (CROW). A CROW is a sequence of optical rings; see Fig. 2. Each section is made of a directional coupler (with coupling values κ_i) and two waveguides, which then couples to the next section. This structure is used for creating optical filters.

The directional coupler and the waveguide are ML components with four (resp. two) ports. This means there are eight ports per CROW section. By increasing the number of sections, we find out what network size our framework can handle in the frequency domain. This is shown in Fig. 3, where

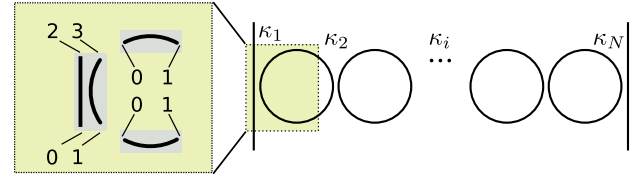


Fig. 2. (Color online) A coupled resonator optical waveguide (CROW). Each section is subdivided in a directional coupler and two waveguides. Port numbers are shown in the left.

we compare the time spent by different matrix strategies as a function of the ML ports. As can be seen in the figure, a large number of CROW sections can easily be handled. This proves the technique is useful for analyzing very complex systems in the steady-state regime.

5. ROBUSTNESS AND ACCURACY IN THE TIME DOMAIN

This section describes two algorithms for integrating the ODE and compares the results to rigorous FDTD simulations.

The first integration scheme is a simple forward Euler with fixed timestep dt , and the second one is an advanced stepping routine based on Bulirsch–Stoer [16]. The latter uses an adaptive stepsize to guarantee accuracy and stability. As example network, we use a system of two coupled photonic crystal cavities. The model equations for this system are

$$\frac{da_j}{dt} = \left[i(\omega_0 + \delta\omega_j) - \frac{1}{\tau} \right] a_j + ds_{j;0,+} + ds_{j;1,+}, \quad (10)$$

$$s_{j;1,-} = \exp(i\phi) s_{j;0,+} + da_j, \quad (11)$$

$$s_{j;1,-} = \exp(i\phi) s_{j;0,+} + da_j, \quad (12)$$

for $j = 1, 2$. Here $d = i \exp(i\phi/2) / \sqrt{\tau}$, where τ is the lifetime of the cavity and ϕ represents the phase that depends on the waveguide length and the resonator mirror reflection properties. The nonlinear frequency shift is $\delta\omega_j = -|a_j|^2 / (P_0 \tau^2)$, with P_0 the “characteristic nonlinear power” of the cavity. In these equations $|a_j|^2$ is the energy in the cavity mode. $|s_{j;k,+}|^2$ (resp. $|s_{j;k,-}|^2$) represents the power flowing in (resp. out) port k (for $k = 0, 1$) of cavity j (for $j = 1, 2$). Port 1 of cavity 1 is connected with port 0 of cavity 2. Thus, $|s_{1;0,+}|^2 \equiv P_{\text{in}}$ is the input power, and $|s_{2;1,-}|^2 \equiv P_{\text{out}}$ is the transmitted power. We

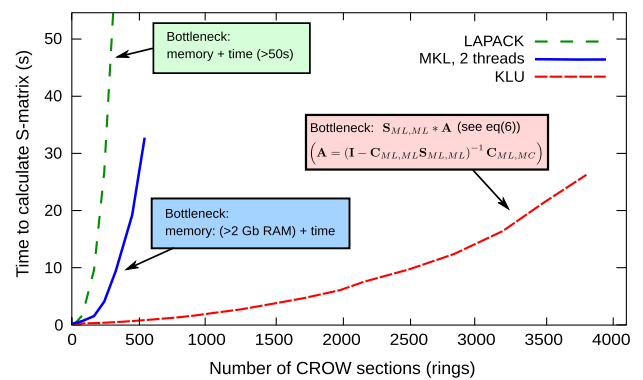


Fig. 3. (Color online) Calculating the frequency response of a passive network. Using KLU, a sparse matrix solver suited for circuit matrices, we can easily calculate scatter matrices of very large networks.

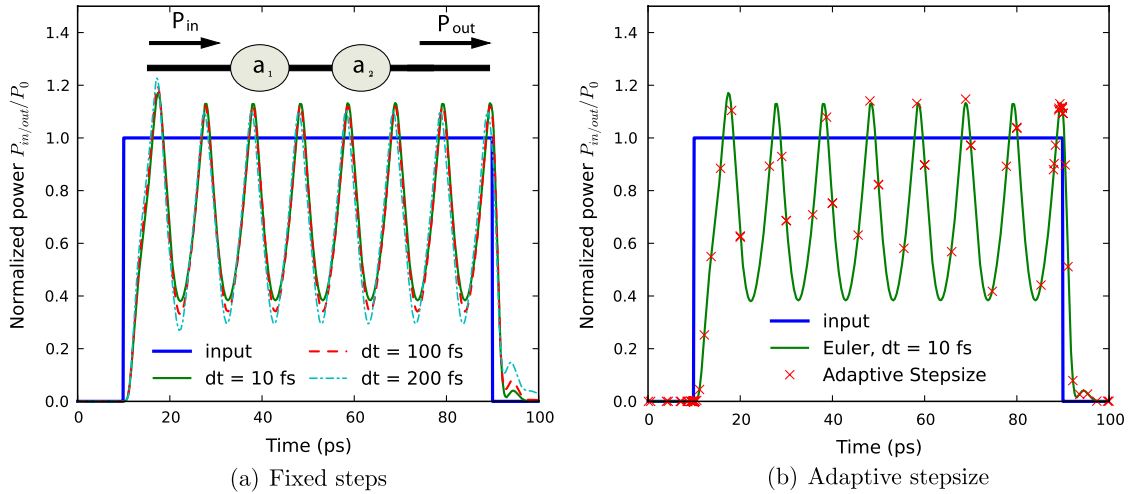


Fig. 4. (Color online) Two integration routines. (a) Forward Euler integration. With larger timesteps, the accuracy decreases. (b) using adaptive stepsize and an advanced stepper routine, the solver automatically uses the optimal dt in order to maintain a desired accuracy (e.g., during switch on, the dt is reduced). Same parameters used as in [17].

assume no input from the right, $s_{2,1,+} = 0$. This system will exhibit self-pulsation under certain assumptions, as studied in [17].

In [17] we also showed that the waveforms from the simulation are almost identical to the waveforms from a full-wave FDTD simulation modeling the same system. The difference in simulation time motivates the use of this simulator: it takes 10 hours to simulate this oscillation in two-dimensional (2D) FDTD, versus a few milliseconds with CMT, with only a slight sacrifice in accuracy. The equations governing this system are highly nonlinear, yet the numerics remain very stable. Furthermore, using adaptive stepsize, one is assured that the most important details of the simulation are taken into consideration, with an automatic choice of discretization steps. Whereas for a fixed stepsize algorithm, like forward Euler, accuracy over the whole simulation domain can only be obtained by choosing a very small stepsize [dt = 10fs; see Fig. 4(b)], this is not the case for the adaptive stepsize algorithm: during switch on and switch off, there are a lot of discretization steps, while in be-

tween the adaptive stepsize solution can follow the reference solution with fewer discretization steps [Fig. 4(b)].

6. IMPROVING THE SIMULATION SPEED IN THE TIME DOMAIN

If a network contains both ML and MC nodes, one can eliminate the ML nodes prior to starting the time-domain simulation. The speed of the time-domain simulation depends on the size of the scatter matrix after eliminating the ML nodes. We demonstrate this by modeling a large network of interconnected nodes. The topology is a regular 2D grid of semiconductor optical amplifiers (SOAs). Each SOA is connected to its nearest neighbors, in a structure called a swirl topology [18]; see Fig. 5(a). To connect the SOAs, we used a combination of splitters and waveguides. We compare two systems. In the first system the ML nodes behave as MC nodes, and in the second system we first eliminate all ML nodes. In the first case, we need to calculate the light propagation in each splitter and waveguide separately, which means the simulation will take

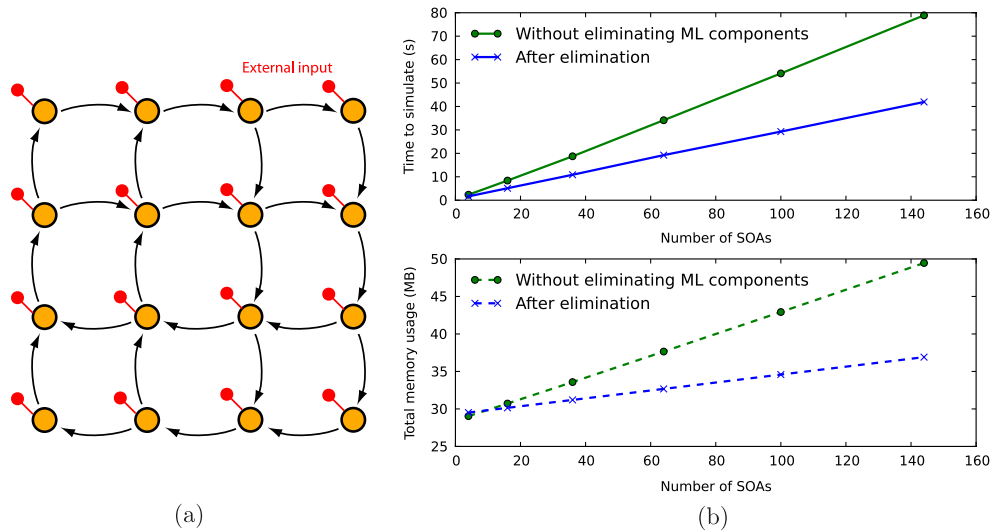


Fig. 5. (Color online) (a) Topology used to simulate a complex system with ML and MC nodes. Each circle represents an SOA. Splitters are not shown. (b) The simulation time and memory usage increases linearly with the number of SOAs. Clearly there is an advantage by eliminating the ML nodes, both in terms of speed and memory usage.

longer, and consume more memory, than in the second case. This is illustrated in Fig. 5(b).

The total simulation time is mainly determined by the evaluation of the ODEs of the individual SOAs and the matrix multiplication from Eq. (7). There is a clear benefit of eliminating the ML nodes: the simulation speed is approximately halved, as shown in Fig. 5(a) (top). The calculation time for evaluating the ODEs is the same for both systems. The memory usage is shown in the bottom graph of Fig. 5(a); it is a sum of the memory allocated in C++ and in Python. The offset is due to initialization overhead in Python. For all simulations, we used a buffer that stores 500 timesteps. Because we eliminated the ML nodes, the memory requirements are greatly reduced. Since we use sparse matrices, calculation time and memory requirements scale linearly as a function of the network size.

7. CONCLUSION

In this paper, we demonstrated a framework that enables modeling of optical circuits both in the time and in the frequency domain. It is suited for calculating the steady-state characteristics of very large networks, and to model highly nonlinear systems in the time domain after eliminating linear instantaneous components. By eliminating these components, we reduce the effective size of the network, and the time-domain simulation speeds up. The tool is very general, and the internal variables can be expressed naturally depending on the application domain, which makes it attractive for other dynamical systems, such as electrical circuits and neural networks.

ACKNOWLEDGMENTS

This work is supported by the Interuniversity Attraction Pole (IAP), Photonics@be of the Belgian Science Policy Office, and the European Research Council NaResCo Starting grant. M. Fiers acknowledges the Special Research Fund of Ghent University. T. Van Vaerenbergh and K. Caluwaerts are supported by the Flemish Research Foundation (FWO-Vlaanderen) for a Ph.D. grant.

REFERENCES

1. A. F. Oskooi, D. Roundy, M. Ibanescu, P. Bermel, J. D. Joannopoulos, and S. G. Johnson, "MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method," *Comput. Phys. Commun.* **181**, 687–702 (2010).
2. E. Lambert, M. Fiers, S. Nizamov, M. Tassaert, and W. Bogaerts, "Python bindings for the open source electromagnetic simulator Meep," *Comput. Sci. Eng.* **13**, 53–65 (2011).
3. L. M. Zhang, S. F. Yu, M. C. Nowell, D. D. Marcenac, J. E. Carroll, and R. G. S. Plumb, "Dynamic analysis of radiation and side-mode suppression in a second-order DFB laser using time-domain large-signal traveling wave model," *IEEE J. Quantum Electron.* **30**, 1389–1395 (1994).
4. G. Agrawal, *Nonlinear Fiber Optics*, 3rd ed. (Academic, 2007).
5. <http://www.aspicdesign.com/>
6. http://www.vpiphotonics.com/optical_systems.php
7. <http://www.photond.com/products/picwave.htm>
8. <http://www.rsoftdesign.com/products.php?sub=System+and+Network&itm=OptSim>
9. P. Gunupudi, T. Smy, J. Klein, and Z. Jakubczyk, "Self-consistent simulation of opto-electronic circuits using a modified nodal analysis formulation," *IEEE Trans. Adv. Packaging* **33**, 979–993 (2010).
10. T. Smy, P. Gunupudi, S. McGarry, and W. N. Ye, "Circuit-level transient simulation of configurable ring resonators using physical models," *J. Opt. Soc. Am. B* **28**, 1534–1543 (2011).
11. <http://www.intec.ugent.be/caphe/>
12. K. Vandoorne, W. Dierckx, B. Schrauwen, D. Verstraeten, R. Baets, P. Bienstman, and J. Van Campenhout, "Toward optical signal processing using photonic reservoir computing," *Optics Express* **16**, 11182–11192 (2008).
13. T. A. Davis and E. P. Natarajan, "Algorithm 8xx: Klu, a direct sparse solver for circuit simulation problems," *ACM Trans. Math. Softw.* **5**, 1–17.
14. T. A. Davis, *Direct Methods for Sparse Linear Systems* (SIAM, 2006).
15. K. Stanley, "Klu: a Clark Kent sparse LU factorization algorithm for circuit matrices," *SIAM Conference on Parallel Processing for Scientific Computing (PP04)* (2004).
16. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. (Cambridge University, 2007).
17. B. Maes, M. Fiers, and P. Bienstman, "Self-pulsing and chaos in short chains of coupled nonlinear microcavities," *Phys. Rev. A* **80**, 033805 (2009).
18. K. Vandoorne, J. Dambre, D. Verstraeten, B. Schrauwen, and P. Bienstman, "Parallel reservoir computing using optical amplifiers," *IEEE Trans. Neural Netw.* **22**, 1469–1481 (2011).