Generic method for solving partial differential equations through the design of problem-specific continuous automata

Nicolas Fressengeas, Hervé Frezza-Buet

Generic method for solving partial differential equations through the design of problem-specific continuous automata

N.Fressengeas
Laboratoire Matériaux Optiques, Photonique et Systèmes, 57070 Metz, France
Unité de Recherche Commune à l'Université de Metz et Supélec - CNRS UMR 7132

H.Frezza-Buet
Information, Multimodality and Signal, Supélec, 2, rue Edouard Belin, 57070 Metz, France

This paper presents an original and generic numerical method for solving partial differential equations. A new mathematical and systematic method stemming from the local very nature of any differential problem is proposed: a custom tailored continuous automaton is purposely derived from any given differential problem so that its steady state yields the solution in a quantitatively correct way. The combined use of formal computing and continuous automata thus offers the unique possibility to completely automate the process from formal problem specification to its numerical solution.

PACS numbers: 02.30.Jr, 02.60.Cb, 02.60.Jh, 02.60.Lj

I. AUTOMATA AND PARTIAL DIFFERENTIAL EQUATIONS

Ever since von Neumann[1], the question of modelling continuous physics with a discrete set of cellular automata has been raised, whether they handle discrete or continuous values. Many answers have been brought forth through, for instance, the work of Stephen Wolfram[2] summarized in a recent book[3]. This problem has been mostly tackled by rightfully considering that modelling physics through Newton and Leibniz calculus is fundamentally different from a discrete modelization as implied by automata.

Indeed, the former implies that physics is considered continuous either because materials and fields are considered continuous in classical physics or because quantum physics wave functions are themselves continuous. On the contrary, modelling physics through automata implies modelling on a discrete basis, in which a unit element called *a cell*, interacts with its surroundings according to a given law derived from local physics considerations. Such discretized automaton based models have been successfully applied to various applications ranging from reaction-diffusion systems[4] to forest fires[5], through probably one of the most impressive achievements: the Lattice Gas Automata[6], where atoms or molecules are considered individually. In this frame, simple point mechanics interaction rules lead to complex behaviors such as phase transition and turbulence. This peculiar feature of automata, making complex group behavior emerge from fairly simple individual rules aroused the interest around them for the past decades.

However, to the best of our knowledge, beside particular solutions[4, 7], the predictions of calculus based, continuous models and those of discrete, automata based ones, are seldom quantitatively identical, though qualitative similarity is often obtained. This is mostly explained by the fact that the two drastically different approaches are applied to their own class of problems. Attempts at a quantitative link have however been made by showing connexions between an automaton and a particular differential problem[8] or by designing methods for describing automata by differential equations[9, 10, 11] allowing in the way to *assess* the performance of two different *implementation* of the same problem, which are in fact basically two different automata for the description of the same physics.

That is the reason why this paper is devoted to the introduction of a new and systematic method allowing to derive a continuous automaton from any given differential problem whose boundary conditions are of the Dirichlet type. The process of derivation stems from the idea that since differential problems are expressed in a purely local manner, their solution can be computed in an equally local way. However, as will be shown, the locality of the computation is not an *a priori* hypothesis but rather a consequence of the method. Nonetheless, the probably most interesting aspect of it is the possibility to completely automate the way from the formal expression of the differential problem down to its solution, thanks to formal computing and to a cellular automata based environment, analogous to previously reported ones[12, 13].

II. SOLVING PDE'S THANKS TO CONTINUOUS AUTOMATA

The very characteristic of continuous physics is its intensive use of *fields* which are, mathematically speaking, mappings of a given vectorial physical quantity — belonging to $\mathbb{R}^n$— over a given physical space $\mathbb{R}^m$, for $(m, n) \in \mathbb{N}^2$. Therefore, a particular local differential problem $\mathbb{P}$ stemming from local relationships, can be expressed in terms of a functional equation $\Phi^{(\xi)} = 0$, where Φ, represents the known differential relationships and where the field ξ is the unknown. Φ is defined as follows, where $p \in \mathbb{N}$ can be thought of as the problem

dimensionality —or the number of independent scalar relationships— and where $(A)^B$ stands for the set of functions from A to B:

$$\begin{array}{rccc} \Phi & : & (\mathbb{R}^m)^{\mathbb{R}^n} & \mapsto & (\mathbb{R}^m)^{\mathbb{R}^p} \\ & & \xi & \to & \Phi^{(\xi)} \end{array} .$$

Following and using the finite difference scheme, the same problem $\mathbb{P}$ can be expressed in a discretized way on a finite mesh $\Omega \subset \mathbb{R}^m$ as $\tilde{\Phi}^{(\tilde{\xi})} = 0$ where

$$\begin{array}{rccc} \tilde{\Phi} & : & (\Omega)^{\mathbb{R}^n} & \mapsto & (\Omega)^{\mathbb{R}^p} \\ & & \tilde{\xi} & \to & \tilde{\Phi}^{(\tilde{\xi})} \end{array} \tag{1}$$

In this context, solving the discretized differential problem $\tilde{\mathbb{P}}$ means finding a field $\tilde{\xi}$ for which $\tilde{\Phi}^{(\tilde{\xi})}$ is as close to 0 as possible over $(\Omega)^{\mathbb{R}^p}$, given a distance on $(\Omega)^{\mathbb{R}^p}$. This, in turn, is equivalent to zeroing —in the same sense as previously— $\tilde{\Phi}^{(\tilde{\xi})}(\omega)$ for *all* $\omega \in \Omega$, given a distance on $\mathbb{R}^p$. Finally, this can be equivalently done by similarly zeroing

$$\mathcal{E}^{(\tilde{\xi})} = \sum_{\omega \in \Omega} \left| \tilde{\Phi}^{(\tilde{\xi})}(\omega) \right| \tag{2}$$

where $|\quad|$ is any given norm on $\mathbb{R}^p$. Let us note that $\mathcal{E}^{(\tilde{\xi})}$ can here be understood as a measure of the violation of the physical relationships $\tilde{\mathbb{P}}$ by a given field $\tilde{\xi}$.

The straightforward method for numerically evaluating the solution $\tilde{\xi}^\star$ to $\tilde{\mathbb{P}}$, *i.e.* the value of $\tilde{\xi}$ that best zeroes —*i.e.* that *minimizes*— $\mathcal{E}^{(\tilde{\xi})}$, is the standard Newton method applied to a multidimensional optimization problem. Let us note here however that this minimization process does not ensure the zeroing of $\mathcal{E}^{(\tilde{\xi})}$, which is to be verified *a posteriori* by evaluating $\mathcal{E}^{(\tilde{\xi}^\star)}$.

To undertake this optimization task, we previously need to define a canonical basis of $(\Omega)^{\mathbb{R}^n}$ with respect to which the *gradient* and *Hessian* will be taken. If δ is the Kronecker symbol and $\{r\}_i$ is the canonical basis of $\mathbb{R}^n$, let us define $\{e\}_{(\omega,i)}$, the canonical basis of $(\Omega)^{\mathbb{R}^n}$ as the set of functions $e_{(\omega,i)}$, for all $\omega \in \Omega$ and all $1 \leq (i \in \mathbb{N}) \leq n$:

$$\begin{array}{rccc} e_{(\omega,i)} & : & \Omega & \mapsto & \mathbb{R}^n \\ & & \omega' & \to & \delta_{\omega\omega'} r_i \end{array} \tag{3}$$

Therefore and using these definitions, the Newton method consists in building a series $\left\{\tilde{\xi}_i\right\}_i$ defined as follows, the limit of which should be the sought solution $\tilde{\xi}^\star$ to $\tilde{\mathbb{P}}$:

$$\begin{array}{rcl} \tilde{\xi}_{t+1} & = & \tilde{\xi}_t - \mu^{(\mathcal{E})}_{|\{e\}_{(\omega,i)}}\left(\tilde{\xi}_t\right) \\ \text{with } \mu^{(\mathcal{E})}_{|\{e\}_{(\omega,i)}}\left(\tilde{\xi}_t\right) & = & \bar{\mathrm{H}}^{(\mathcal{E})}_{|\{e\}_{(\omega,i)}}\left(\tilde{\xi}_t\right).\mathrm{grad}^{(\mathcal{E})}_{|\{e\}_{(\omega,i)}}\left(\tilde{\xi}_t\right) \\ \text{where } \bar{\mathrm{H}} \text{ is the inverse of the Hessian matrix.} & & \end{array} \tag{4}$$

The above expression requires some derivability conditions on $\mathcal{E}$, and thus on both $\tilde{\Phi}$ and the chosen norm on $\mathbb{R}^n$. The former is assumed, since it stems from the differential problem $\mathbb{P}$ itself. The latter is ensured by the appropriate choice of the used norm. As another precaution to be taken on that choice, the used norm must ensure that no component of the gradient – and thus of Hessian inverse – neither supersedes the others nor is superseded by them, for this is known to create stability problems in the iteration defined by (4). The conventional $|\quad|_2$ norm, or its square, is for instance a good choice, provided $\mathbb{P}$ is conveniently normalized.

However, the effective computation of such a series implies to compute, for each step t, the gradient and inverse Hessian with respect to $\{e\}_{(\omega,i)}$, which implies getting access to the whole Ω, in contradiction with our initial goal which was to design a *local-only* computationnal method. Therefore, we will now present a method inspired from the stochastic gradient descent method [14], the locality of which will be established in the next section.

The stochastic gradient method consists in updating $\tilde{\xi}$ by considering only a few of its components at a time. We choose to consider a single ω in Ω at each step, thus modifying only $\tilde{\xi}^\omega$, the ω-related components of $\tilde{\xi}$. Therefore, the gradient and Hessian appearing in (4) are taken not with respect to $\{e\}_{(\omega,i)}$ but rather with a subset $\{e\}_\omega$ of it, defined as the set of the $e_{(\omega,i)}$'s restricted to ω. One such step is therefore defined as follows:

$$\tilde{\xi}^\omega_{t+1} = \tilde{\xi}^\omega_t - \mu^{(\mathcal{E})}_{|\{e\}_\omega}\left(\tilde{\xi}_t\right) \tag{5}$$

The above relationship describes a series for a given point ω of Ω. For the series (4) to be completely approximated by the stochastic method, the relationship (5) is of course to be iterated with a random choice of $\omega \in \Omega$ at each step. Thus, provided $\mu^{(\mathcal{E})}_{|\{e\}_\omega}$ is somehow local, an issue that will be addressed in the next section, the above considerations allow to consider (5), at ω, as the definition of a *continuous* automaton, which is an extension of classical cellular automata for which the cell states are allowed to take their values in $\mathbb{R}^n$.

Furthermore, the automata described by (5) for each ω also departs from the strict definition of a cellular automaton by the fact that the update rule for all cells ω are only the same for a vast majority of them, but not strictly all. Indeed, because of the existing boundary conditions, the H and grad operators will not give the same result for all points, since the boundary conditions are considered as constants. Hence, a Dirichlet boundary is described in the automaton by a constant cell, the value of which is given by the automaton initial state.

III. LOCALIZATION OF EACH CELL NEIGHBORHOOD

The definition of the *neighborhood* $\mathcal{V}$ of a given function $\zeta \in \left((\Omega)^{\mathbb{R}^n}\right)^{\mathbb{R}}$ is to be understood as being the set of all the points ω needed in the evaluation of ζ:

$$\begin{aligned}\mathcal{V} \; : \; \left((\Omega)^{\mathbb{R}^n}\right)^{\mathbb{R}} &\mapsto \mathcal{P}(\Omega) \\ \zeta &\to \left\{\omega \in \Omega \; : \; \exists\, \tilde{\xi} \; : \; \mathrm{grad}^{(\zeta)}_{|\{e\}_\omega}\left(\tilde{\xi}\right) \neq 0\right\}\end{aligned}$$

The automaton described in the previous section is thus practically usable if the calculations needed to evaluate it are *local*, *i.e.* expression (5) can be evaluated without requiring access to Ω as a whole. This can be formally stated as $\mathcal{V}\left(\left|\mu^{(\mathcal{E})}_{|\{e\}_\omega}\right|\right) \neq \Omega$.This can happen only if some kind of locality condition on $\mathbb{P}$ is assumed.

To show that this is indeed the case, let us first define the *reciprocal neighborhood* $\bar{\mathcal{V}}$ for a given $\omega \in \Omega$ and a given $\tilde{\Phi}$ as in (1): $\bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right) = \left\{\omega' \; : \; \omega \in \mathcal{V}\left(\left|\tilde{\Phi}(\omega')\right|\right)\right\}$. Given the definition (4) of $\mu^{(\mathcal{E})}_{|\{e\}_\omega}$, the gradient can be linearly distributed over the additive components of $\mathcal{E}$:

$$\mu^{(\mathcal{E})}_{|\{e\}_\omega} = \sum_{\omega' \in \bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)} \bar{\mathrm{H}}^{(|\mathcal{E}|)}_{|\{e\}_\omega} \mathrm{grad}^{\left(|\tilde{\Phi}(\omega')|\right)}_{|\{e\}_\omega} \quad (6)$$

$$= \left(\sum_{\omega' \in \bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)} \mathrm{H}^{\left(|\tilde{\Phi}(\omega')|\right)}_{|\{e\}_\omega}\right)^{-1} \sum_{\omega' \in \bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)} \mathrm{grad}^{\left(|\tilde{\Phi}(\omega')|\right)}_{|\{e\}_\omega} \quad (7)$$

The leftmost summation term appearing in the above equation has been restricted to those ω' in Ω for which the gradient does not vanish. The rightmost summations product is obtained by similarly distributing the Hessian. The neighborhood of a product being included in the union of its operands neighborhoods, the neighborhood of $\mu^{(\mathcal{E})}_{|\{e\}_\omega}$ can be limited to

$$\mathcal{V}\left(\left|\mu^{(\mathcal{E})}_{|\{e\}_\omega}\right|\right) \subset \mathcal{V}\left|\left(\sum_{\omega' \in \bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)} \mathrm{H}^{\left(|\tilde{\Phi}(\omega')|\right)}_{|\{e\}_\omega}\right)^{-1}\right| \quad (8)$$

$$\bigcup \mathcal{V}\left|\sum_{\omega' \in \bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)} \mathrm{grad}^{\left(|\tilde{\Phi}(\omega')|\right)}_{|\{e\}_\omega}\right| \quad (9)$$

For obvious reasons, the neighborhood of a derivative, or a gradient, is included in the neighborhood of its operand. The same holds for the neighborhood of a Hessian since any line or column of the Hessian is a derivative of the gradient. Therefore, the right-hand term of the above union is included in $\sum_{\omega' \in \bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)} \mathcal{V}\left(\left|\tilde{\Phi}(\omega')\right|\right)$.

Furthermore, the neighborhood of a matrix norm $|M|$ is obviously included in the union of the neighborhoods of all its components. The same holds for the inverse matrix $\left|(M)^{-1}\right|$ since each of its components can be obtained by a combination of the components of M. We can therefore conclude that the left-hand term of the union in (9) is also a subset of $\sum_{\omega' \in \bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)} \mathcal{V}\left(\left|\tilde{\Phi}(\omega')\right|\right)$.

Therefore, provided we can assume that both $\bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)$ and $\mathcal{V}\left(\left|\tilde{\Phi}(\omega)\right|\right)$ are small enough for all $\omega \in \Omega$ —which is ensured if the differential problem $\mathbb{P}$ is defined locally—, the calculations to be undertaken to evaluate $\mu^{(\mathcal{E})}_{|\{e\}_\omega}$ for each cell ω are local to some extended neighborhood of that cell:

$$\mathcal{V}\left(\left|\mu^{(\mathcal{E})}_{|\{e\}_\omega}\right|\right) \subset \bigcup_{\omega' \in \bar{\mathcal{V}}\left(\omega, \tilde{\Phi}\right)} \mathcal{V}\left(\left|\tilde{\Phi}(\omega')\right|\right)$$

IV. APPLICATION EXAMPLES

As hinted before, we have implemented the continuous automaton described in the previous sections with the help of an off-the-shelf formal computing software and a cellular automata environment analogous to those reported in [12, 13]. We have thus automated the computation from the specification of the discrete differential problem $\tilde{\mathbb{P}}$ to the design of the adequate continuous automaton.

To better understand this process, we will now provide a simple academic example of the solving of Poisson Equation for V in the three dimensions of space: $\triangle V(x, y, z) = \rho(x, y, z)$ for any given ρ, the boundary conditions being set on the sides of the computing cube window. The corresponding discrete problem is straightforward and is obtained through finite difference centered second derivatives on each dimensions of space, for the same space step δ.

The automaton obtained has 28 different update rules, to account for the boundary conditions. The update rule obtained for ω such as the boundaries conditions are not in $\mathcal{V}\left(\left|\mu^{(\mathcal{E})}_{|\{e\}_\omega}\right|\right)$ concerns the vast majority of the mesh nodes ω and is shown below. It is a centro-symmetric three dimensional convolution kernel, a two-dimensional slice of which is as follows:

$$V \longleftarrow \frac{1}{42} * \left[V * \begin{array}{ccccc} & & -1 & & \\ & -2 & 12 & -2 & \\ -1 & 12 & 0 & 12 & -1 \\ & -2 & 12 & -2 & \\ & & -1 & & \end{array} + \delta^2 * \rho * \begin{array}{ccc} & 1 & \\ 1 & -6 & 1 \\ & 1 & \end{array}\right]$$

When launched, the system converges to a fixed point, corresponding to the result that, in that case, can also be

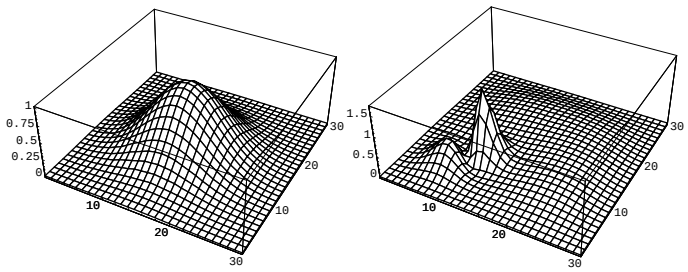

FIG. 1: Left : laser beam Gaussian profile to be coupled in the waveguide (A.U.). Right : beam profile after a 3mm propagation. The window size is 30μm and the beam wavelength is 250μm. The highest peak on the right evidences the light which is coupled into the waveguide.

obtained with more conventional methods. For a $20\times20\times20$ mesh and for a value of ρ varying from 1 to 0 from one side of the cube to the other, the *a posteriori* computed remaining error $\mathcal{E}$ (after normalization) is 7×10^{-3} for $\delta=1$ and decreases with it.

However, as we will show now, a better assessment of the performance of this new method will be established by solving a physically realistic problem, which is not so easy to solve by other conventional methods. Indeed, we now aim to compute the coupling of a Gaussian laser beam of width W into a Gaussian shaped waveguide of width $\frac{W}{2}$ and modulation depth 10^{-2}. The centers of both beam and waveguide are set to a distance of $\frac{W}{2}$, both being aligned in the same direction. In the computation process, we will *not* make the standard simplifying paraxial approximation, which makes our problem difficult to solve by conventional methods.

The non-paraxial propagation equation to be solved is thus the following, where A is the wave electric field to be found, z is the propagation direction, k is the wave vector, n and δn are the given refraction index and a small variation of it:

$$\frac{\partial A}{\partial z}-\frac{i}{2k}\triangle A=\frac{ik}{n}\delta nA. \tag{10}$$

The problem is solved by deriving two real equations from (10), discretizing them with finite difference centered derivatives except along z where left-handed derivatives are needed because of the impossibility to give a boundary condition on both sides of the propagation axis. The adequate continuous automaton (it also has 28 update rules but is too complicated to show here) is then computed from the discretized problem. When launched on a $30\times30\times30$ network, it stabilizes to a steady state shown on figure 1. The *a posteriori* remaining error is now computed to be less than 10^{-10}.

V. CONCLUSION

We have described and successfully assessed what we believe to be the first method allowing to tailor a Continuous Automaton so that its steady state *quantitatively* solves a given differential problem. We believe that this method can be applied to most continuous differential problems. The accompanying automation thus offers the unique possibility to reduce differential problem solving to the mere specification of the problem with an adequate formal language and its feeding to a specifically designed Continuous Automaton. Future work will have to assess the possibilities of the here proposed method against differential problems which solutions lead to chaotic behaviors, a possible difficulty for our multidimensional Newton method.

[1] A. W. Burks, *Von Neumann's Self-reproducing Automata* (University of Michigan, 1969).
[2] S. Wolfram, Rev. Mod. Phys. **55**, 61 (1983).
[3] S. Wolfram, *A new kind of science* (Wolfram Media, Champaign, 2002).
[4] J. R. Weimar and J. P. Boon, Phys.Rev.E **49**, 1749 (1994).
[5] B. Drossel and F. Schwabl, Phys. Rev. Lett. **69**, 1629 (1992).
[6] D. H. Rothman and S. Zaleski, Rev. Mod. Phys. **66**, 1417 (1994).
[7] N. R. S. Simons, G. E. Bridges, and M. Cuhaci, J. Comput. Phys. **151**, 816 (1999), ISSN 0021-9991.
[8] T. Tokihiro, D. Takahashi, J. Matsukidaira, and J. Satsuma, Phys Rev. Lett. **76**, 3247 (1996).
[9] A. Doeschl, M. Davison, H. Rasmussen, and G. Reid, Math. Comp. Mod. **40**, 977 (2004).
[10] W. Kunishima, A. Nishiyama, H. Tanaka, and T. Tokihiro, Journ. Phys Soc. Japan **73**, 2033 (2004).
[11] S.Omohundro, Physica D **10D**, 128 (1984).
[12] M. Cannataro, S. D. Gregorio, R. Rongo, W. Spataro, G. Spezzano, and D. Talia, Parallel Computing **21**, 803 (1995).
[13] G. Spezzano and D. Talia, in *Virtual shared memory for distributed architectures* (Nova Science Publishers, Inc., Commack, NY, USA, 2001), pp. 51–68.
[14] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control* (Wiley-Interscience, 2003).