Neural-network approach to modeling liquid crystals in complex confinement

T. Santos-Silva,[1] P. I. C. Teixeira,[2,3] C. Anquetil-Deck,[4,*] and D. J. Cleaver[4]

[1]*Faculdade de Engenharia, Universidade Católica Portuguesa, Estrada de Talaíde, P-2635-631 Rio de Mouro, Portugal*

[2]*Instituto Superior de Engenharia de Lisboa, Rua Conselheiro Emídio Navarro 1, P-1950-062 Lisbon, Portugal*

[3]*Centro de Física Teórica e Computacional, Faculdade de Ciências, Universidade de Lisboa, Avenida Professor Gama Pinto 2, P-1649-003 Lisbon, Portugal*

[4]*Materials and Engineering Research Institute, Sheffield Hallam University, Pond Street, Sheffield S1 1WB, United Kingdom*

(Received 10 November 2012; revised manuscript received 20 April 2014; published 28 May 2014)

Finding the structure of a confined liquid crystal is a difficult task since both the density and order parameter profiles are nonuniform. Starting from a microscopic model and density-functional theory, one has to either (i) solve a nonlinear, integral Euler-Lagrange equation, or (ii) perform a direct multidimensional free energy minimization. The traditional implementations of both approaches are computationally expensive and plagued with convergence problems. Here, as an alternative, we introduce an *unsupervised* variant of the multilayer perceptron (MLP) artificial neural network for minimizing the free energy of a fluid of hard nonspherical particles confined between planar substrates of variable penetrability. We then test our algorithm by comparing its results for the structure (density-orientation profiles) and equilibrium free energy with those obtained by standard iterative solution of the Euler-Lagrange equations and with Monte Carlo simulation results. Very good agreement is found and the MLP method proves competitively fast, flexible, and refinable. Furthermore, it can be readily generalized to the richer experimental patterned-substrate geometries that are now experimentally realizable but very problematic to conventional theoretical treatments.

DOI: 10.1103/PhysRevE.89.053316 PACS number(s): 02.70.−c, 68.08.−p, 64.70.mf, 61.30.Hn

I. INTRODUCTION

In a world suffused with images, displays are paramount. Of these, liquid crystal (LC) devices (LCDs) have a huge market share [1]. LCs are a state of matter intermediate between solid and liquid; they retain some of the order of a solid, but are free to flow as a liquid [2]. In particular, their constituent particles, which are typically elongated, all point, on average, in the same direction, termed the director; the extent of this alignment is given by the LC order parameter.

The director orientation is determined by effects external to the LC itself. All current LCDs are basically light valves that rely, for their operation, on the competing actions of bounding surfaces, known as *anchoring*, and applied fields on the director (see, e.g., [3,4]). Typically, a LC layer is sandwiched between suitably prepared substrates, which may favor the same (symmetric) or different (hybrid) alignments. An electric field is then used to deviate the orientational order profile from that induced by the substrates alone. Two examples are the conventional (and highly successful) twisted-nematic (TN) cell [4] found in most TV screens, and the more recent hybrid aligned nematic (HAN) cell of Bryan-Brown *et al.* [5,6]. The latter has led to a practical realization of a *bistable* device: unlike the TN cell, a bistable device has two optically distinct, stable states and an applied voltage is only needed when switching between them, with consequent substantial energy savings.

Applications of LCDs beyond displays include sensors. For example, it has been shown that a LC film deposited at the air-water interface can be switched in and out of the HAN state by varying the surfactant concentration in the water, thereby providing an easy-to-read surfactant detector [7]. LC confinement is also pertinent to some of the many fascinating LC colloid systems devised in recent years [8], where the LC matrix is squeezed to microscopic dimensions when the colloidal particles self-aggregate.

In order to predict the behavior of a confined LC, we need to find its structure. A theorist will start by selecting an appropriate model, which should be simple enough to be amenable to computation, and yet capture the relevant features of the real system. One such popular model is hard ellipsoids between hard walls. It is also necessary to write a suitable free energy (Helmholtz or Gibbs, as the case might be) for the system, on the basis of a statistical mechanical theory such as Onsager's, which we shall encounter below. The equilibrium state is, then, that which minimizes the free energy. Because the system is nonuniform, the free energy is a functional of the density-orientation profile, i.e., it depends on the density, director, and order parameter at each point in the region occupied by the LC, and minimization is difficult. Two major routes are then possible: (i) functionally to differentiate the free energy and thereby derive a nonlinear, integral Euler-Lagrange (EL) equation, which must then be solved; or (ii) to perform a direct multidimensional free energy minimization. Route (i), essentially in the form of Picard iteration [9], has been used by many workers including ourselves [10,11]; it is easier to implement, but may fail to converge, converge very slowly, or converge to a local minimum, especially if the density-orientation profile is strongly nonuniform and the initial guess is not carefully made. Route (ii) basically uses variants of the conjugate-gradient scheme [12,13]; it is more reliable, less dependent on the quality of the initial guess, and possibly somewhat faster, but harder to implement.

*Present address: Karlsruhe Institute of Technology, Institute for Meteorology and Climate Research, Atmospheric Aerosol Research Department (IMK-AAF), Hermann-von-Helmholtz-Platz 1, D-76344 Eggenstein-Leopoldshafen, Germany.

To our knowledge, only systems that exhibit spatial variability along one dimension have been investigated by microscopic theory. Whereas this comprises the most popular device geometries including the TN and hybrid HAN cells, it is now possible to pattern substrates along either one or two dimensions, and thereby create novel, more versatile aligning layers for LCs [14–17]. Various of these complex substrates have found application in bistable LCDs such as the zenithally bistable nematic [6], post-aligned bistable nematic [18], and square well array [19,20] devices. If one wishes to study these using the microscopic theorist's premier formalism, density-functional theory (DFT), one ends up needing to represent the density-orientation profile, which is a function of at least two angles, in addition to the spatial coordinates, on a very large grid of points. Moreover, the interactions between any two particles at any positions and with any orientations need to be specified by a potentially huge interaction kernel, computation of which requires very fast processors and/or very large RAM. Ideally, one would like to have a toolbox that would permit fast, accurate, and reliable calculation of the structure of a LC layer confined between substrates of many different patterns, in either symmetric or hybrid combinations. This would help guide researchers as to what configurations might be more promising for applications without actually having to manufacture them; the latter is a laborious and often expensive endeavor.

We therefore seek an alternative route to minimizing the free energy of a confined fluid of nonspherical particles. "Minimization" is, of course, one particular realization of the more general problem known as optimization. Of the many available optimization techniques, one that, to our knowledge, has not yet been exploited in the context of LC modeling is artificial neural networks (ANN). ANNs are a class of learn-by-example systems that have some features in common with biological neurones [21]. Biological neurones fire (or not) depending on the input (usually neurotransmitters) they receive (see Fig. 1). Some ANNs mimic this behavior by making artificial units compute an affine transformation for their input vector, followed by some monotonic activation function.

The main difficulty in solving a particular problem is representation, i.e., designing a neural network that is rich enough that its output is able to reproduce a function of the required complexity. In fact, in general the representation of the solution is itself a constraint. In this paper, we describe an approach to learning the density-orientation profile of a confined LC within the framework of an ANN. We focus on a particular type of ANN, the multilayer perceptron (MLP), which has been shown to be a universal approximator for all Borel-measurable functions [22]. This makes the MLP a good candidate for tackling the representation problem. Furthermore, the MLP is a well-studied learning system, for which many algorithms have been developed to speed up convergence. This paper does not concern the use of acceleration algorithms: rather, its main aim is to determine whether MLP networks are applicable to, and may offer an alternative way of addressing the problem of, calculating the structure of a confined LC. This requires modifying the MLP to perform *unsupervised* learning, as the value of the equilibrium free energy is not known *a priori*. We do not claim that the MLP we developed is the best method for any one particular application; our aim was just to add another tool to the theorist's toolbox.

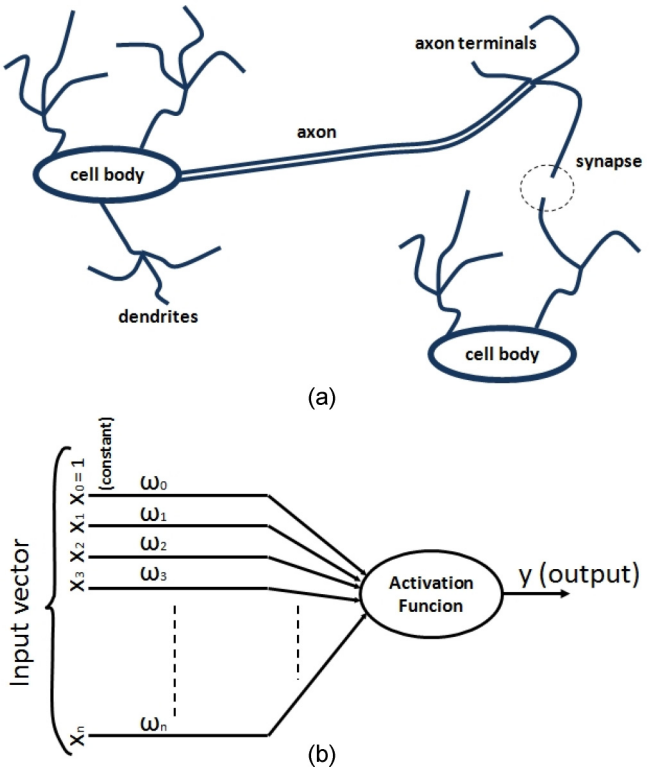

FIG. 1. (Color online) (a) A neurone fires (or not) on the basis of the stimuli it receives from other neurones. (b) An artificial neurone weights the inputs it receives and generates an output. It must be designed so as to produce the desired output.

The remainder of this paper is organized as follows: In Sec. II, we describe general features of ANNs and introduce their terminology. Then, in Sec. III, we recapitulate the model and density-functional theory (DFT) that we have used previously to study confined LCs [11,23,24], and show how our particular minimization problem can be solved using ANNs. In Sec. IV, we test our algorithm by contrasting its performance with that of the standard iterative solution of the EL equations, for a number of different boundary conditions. Validation of the two methods is then achieved through comparison with Monte Carlo (MC) simulation data. Finally, in Sec. V, we discuss the potential and limitations of our ANN approach, and outline some directions for future research.

II. NEURAL NETWORKS

A. Generalities

Artificial neural networks (ANNs) can learn and exhibit global behavior that is able to model real-world problems, although the learning is local to very simple units. These properties are shared by many other networks that use local interactions, such as Markov models [25] or Bayesian networks [26]. ANNs differ from these other approaches in their network architecture, and in how the units receive information, process it, send it to their neighbors, and learn

from the information they receive and from the consequences of their action. Neurones within ANNs (henceforth referred to as "units") use laws that are inspired by biological neurones, hence their name. Interactions between ANN units may, in general, be represented as a graph (cyclic or not). The influence of one unit on another is usually governed by some factor, usually called a weight. Weights code the interactions between units.

There are several types of ANN, and each type uses specific rules. Some of these rules are no longer biologically inspired, and have evolved to other, more efficient, forms. ANNs may be regarded as models of some unknown target function, and two major ANN categories may be considered: supervised and unsupervised. The former are applicable when there is knowledge of the solution of the target function, for some particular learning examples; supervised ANNs learn these examples and their solutions, and use the consequent knowledge to predict for unlearned examples [27]. The latter category, unsupervised learning, is applicable when there is no prior knowledge of the solutions for the learning examples; these networks learn directly from data, and try to capture features and/or some kind of organization within same. This learned knowledge is subsequently organized into self-generated categories and, after learning, the system is able to categorize any new examples it receives [28]. In addition to these major ANN categories, there are also some ANNs that are intended for storing and retrieving content-addressable information, i.e., information that is fully retrieved by providing the system with some small detail or tag.

The multilayer perceptron (MLP) is a type of supervised ANN that requires the user to define some energy function (or cost function) expressing how far the MLP is from learning the desired response for the learning examples. A typical energy function is the sum of the quadratic error over all learning examples [27]. The MLP represents unit interactions as weights, and assumes that the energy function may be expressed as a differentiable function of its weights. Learning is achieved, e.g., by using the gradient descent rule, although second-order methodologies (based on the Hessian matrix) may also be used. MLPs have been successfully employed to learn and model complex nonlinear systems [29]. Aside from the fact that MLPs can represent any Borel-measurable function (which includes all functions of practical interest), they have also been shown to be able to learn any function that they are able to represent [30].

The MLP may be seen as a directed graph, with several layers of units. The first layer of units receives the input; the last layer of units produces the output. Intermediate layers perform increasingly complex computations, such that the output of the units within a layer acts as input to the units in the next layer.

In order to use a MLP, we must sample some input data examples (the training set), and provide these samples to the real-world complex system to be modeled, thereby capturing the response vector of the real-world system to each element of those data (desired response vectors). As MLP is a learn-by-example algorithm, it learns by being fed each instance of the training set and iteratively tuning its weights according to the error between the MLP response vector (output layer response) and the desired response vector to that particular instance.

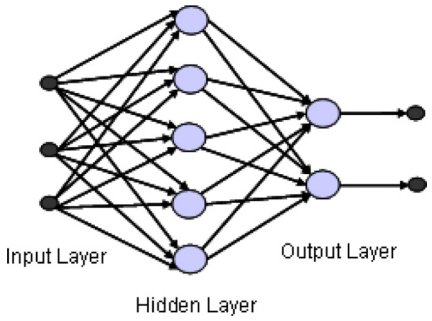

FIG. 2. (Color online) Example of MLP with three units in the input layer, five units in the hidden (intermediate) layer, and two units in the output layer.

B. Terminology

We use x_l as the input vector for layer l, w_{li} as the weight vector of unit i in layer l, g_{li} as the net input of unit i in layer l, given by $g_{li} = w_{li}x_l$, and a_{li} as the output of units i in layer l, given by $a_{li} = \varphi(g_{li})$, where φ is some activation function. Whenever labeling the layer is irrelevant, we omit the subscript l. In a MLP, each layer feeds the next layer (if there is one), so input vector x_l usually corresponds to output vector a_{l-1}. We also use y as the overall output vector of the neural network ($y = a_{l_{\max}}$), and s for some sample that feeds the first layer ($s = x_1$). We denote by d the desired response vector. We also use $y(s)$ and $d(s)$ when we explicitly want to express those vectors as functions of the sample.

Figure 2 shows the topology of a MLP with three units in the input layer, five units in the hidden (intermediate) layer, and two units in the output layer. Input layer units do not perform computations, they just provide input for the hidden layer. Each unit in the hidden and output layers computes $g_{li} = w_{li}x_l$, and then $a_{li} = \varphi(g_{li})$. A typical choice for the activation function is the logistic mapping: $\varphi(x) = 1/(1 + e^{-x})$. Other options include the hyperbolic tangent, the sine, or no activation function (linear unit).

The MLP uses gradient descent in order to update its weights. In general, $\Delta w_{lij} = -\eta \partial E/\partial w_{lij}$, where w_{lij} is the weight value between the jth unit in some layer $l-1$ and the ith unit in the next layer l, E is the cost function to be minimized, and η is a small learning factor. A typical choice for E is

$$E = \sum_{s\in\{\text{training set}\}} \sum_i [y_i(s) - d_i(s)]^2, \tag{1}$$

where i indexes the components of vectors $y(s)$ and $d(s)$. As samples are presented sequentially, the gradient descent rule is applied to the error of a given sample. Therefore, for a single iteration, the energy may be taken to be

$$E = \sum_i (y_i - d_i)^2. \tag{2}$$

The classic MLP learning method uses precomputation of the delta term $\delta_{li} = -\partial E/\partial g_{li}$ in order to make computation of $\partial E/\partial w_{lij}$ more efficient. $\partial E/\partial w_{lij}$ is easily computed from

δ_{li} as

$$\frac{\partial E}{\partial w_{lij}} = \frac{\partial E}{\partial g_{li}} \frac{\partial g_{li}}{\partial w_{lij}} = -\delta_i x_{lj} = -\delta_i a_{(l-1)j}, \tag{3}$$

where $a_{(l-1)j}$ stands for the output of unit j in the preceding layer [21]. After finding the delta terms in some layer $l+1$, those in the preceding layer l are easily computed from

$$\delta_{li} = \varphi'(g_{li}) \sum_{k\in\{l+1\}} \delta_{(l+1)k}\, w_{(l+1)ki}, \tag{4}$$

where k represents each unit in layer $l+1$. Note that the time taken to compute all of the delta terms increases only quadratically with the number of units. If we choose the logistic mapping as our activation function, $\varphi(x) = 1/(1+e^{-x})$, we may observe an interesting property: $\varphi'(g_i) = a_i(1-a_i)$. Therefore, after finding the delta terms in the output layer, their values can be very efficiently backpropagated from the output layer to the hidden layers. Delta terms in the output layer are also efficiently computed using

$$\delta_{oi} = [d_i(s) - y_i(s)]y_i(s)[1-y_i(s)], \tag{5}$$

where δ_{oi} represents the delta terms in the output layer.

III. MLP CALCULATION OF THE EQUILIBRIUM DENSITY-ORIENTATION PROFILES OF A CONFINED NEMATIC LIQUID CRYSTAL

A. Model and theory

The model and theory that we use have been extensively described in our earlier publications [11,23,24], to which we refer readers for details. In short, following established practice in the field of generic LC simulation [31], we consider a purely steric microscopic model of uniaxial rod-shaped particles of length-to-breadth ratio $\kappa = \sigma_L/\sigma_0$, represented by the hard Gaussian overlap (HGO) potential [32]. For moderate κ, the HGO model is a good approximation to hard ellipsoids (HEs) [33,34]; furthermore, their virial coefficients (and thus their equations of state, at least at low to moderate densities) are very similar [35,36]. From a computational point of view, HGOs have the considerable advantage over HEs that the distance of closest approach between two particles is given in closed form [37]. Particle-substrate interactions have been modeled, as in [23,24,38], by a hard needle-wall (HNW) potential (Fig. 3): particles see each other as HGOs, but the substrates see a particle as a needle of length L (which need not be the same at both substrates, or in different regions of each substrate). Physically, $0 < L < \sigma_L$ corresponds to a system where the molecules are able to embed their side groups, but not the whole length of their cores, into the bounding walls. This affords us a degree of control over the anchoring properties: varying L between 0 and σ_L is equivalent to changing the degree of end-group penetrability into the confining substrates. In an experimental situation, this might be achieved by manipulating the density, orientation, or chemical affinity of an adsorbed surface layer. In what follows, we characterize the substrate condition using the parameter $L^* = L/\sigma_L$.

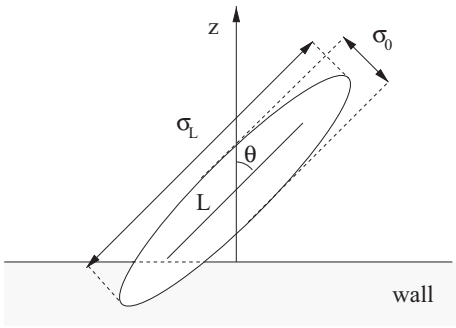

FIG. 3. The HNW potential: the molecules see each other (approximately) as uniaxial hard ellipsoids of axes $(\sigma_0,\sigma_0,\kappa\sigma_0)$, but the wall sees a molecule as a hard line of length L, which need not equal $\kappa\sigma_0$. Physically, this means that molecules are able to embed their side groups into the bounding walls. Varying L is therefore equivalent to changing the wall penetrability, which can be done independently at either wall.

We choose a reference frame such that the z axis is perpendicular to the substrates, and denote by $\omega_i = (\theta_i,\phi_i)$ the polar and azimuthal angles describing the orientation of the long axis of a particle. Because, for unpatterned substrates, the HNW interaction only depends on z and θ, it is reasonable to assume that there is no in-plane structure, so that all quantities are functions of z only. The grand-canonical functional [39] of an HGO film of bulk (i.e., overall) density ρ at temperature T then writes, in our usual approximations [11,23,24],

$$\frac{\beta\Omega[\rho(z,\omega)]}{S_{xy}} = \int \rho(z,\omega)[\ln\rho(z,\omega)-1]\,dz\,d\omega - \frac{\left(1-\frac{3}{4}\xi\right)\xi}{2(1-\xi)^2}\int \rho(z_1,\omega_1)\Xi(z_1,\omega_1,z_2,\omega_2)\rho(z_2,\omega_2)\,dz_1 d\omega_1 dz_2 d\omega_2$$
$$+\beta\int\left[\sum_{\alpha=1}^{2}\mathcal{V}^{\mathrm{HNW}}\left(\left|z-z_0^\alpha\right|,\theta\right)-\mu\right]\rho(z,\omega)\,dz\,d\omega, \tag{6}$$

where S_{xy} is the interfacial area, μ is the chemical potential, $\xi = \rho v_0 = (\pi/6)\kappa\rho\sigma_0^3$ is the bulk packing fraction, z_0^α $(\alpha = 1,2)$ are the positions of the two substrates, $\Xi(z_1,\omega_1,z_2,\omega_2)$ is now the area of a slice (cut parallel to the bounding plates) of the excluded volume of two HGO particles of orientations ω_1 and ω_2 and centers at z_1 and z_2 [40], for which an analytical expression has been derived [37]. $\rho(z,\omega)$ is the density-orientation profile in the presence of the external potential $\mathcal{V}^{\mathrm{HNW}}(z,\theta)$; it is normalized to the total number of particles N,

$$\int \rho(z,\omega)\,dz\,d\omega = \frac{N}{S_{xy}} \equiv M, \tag{7}$$

and is related to the probability that a particle positioned at z has orientation between ω and $\omega + d\omega$. This normalization is enforced through the chemical potential μ, which is essentially a Lagrange multiplier.

Three remarks are in order. First, note that each surface particle experiences an environment that has both polar *and* azimuthal anisotropy, as a consequence of the excluded-volume interactions between the particles in addition to the "bare" wall potential. Second, because we are dealing with hard-body interactions only, for which the temperature is an irrelevant variable, we can set $\beta = 1/k_B T = 1$ in all practical calculations (we retain it in the formulas for generality). Third, and finally, the prefactor multiplying the second integral in Eq. (6) is a simplified implementation of the Parsons-Lee density rescaling [41], which amounts to (approximately) summing the higher virial coefficients. In the spirit of [42], this prefactor is a function of the *bulk* density, and not of the *local* density, which should be valid provided the density does not exhibit sharp spatial variations. Equation (6) is, thus, the "corrected" Onsager approximation to the free energy of the confined HGO fluid, which we expect to perform better for particle elongations $\kappa \ll \infty$ inasmuch as structure is determined by location in the phase diagram. We do not expect, however, to see any new structure that is not captured by the Onsager approximation since what we are doing is simply rescaling density. More sophisticated approaches exist (see, e.g., [12,13]), but our purpose here, as stated above, has been to introduce a new calculational tool, so we apply it to the simplest possible microscopic treatment of anchoring phenomena that yields fairly good results [11,23,24].

Minimization of the grand-canonical functional can be performed either directly on Eq. (6) [route (ii) above] or, as in our earlier work, by first analytically deriving, and then numerically solving, the EL equation for the equilibrium density-orientation profile [route (i) above]:

$$\frac{\delta\Omega[\rho(z,\omega)]}{\delta\rho(z,\omega)} = 0 \Rightarrow \ln\rho(z,\omega) = \beta\mu - \frac{\left(1-\frac{3}{4}\xi\right)}{(1-\xi)^2} \times \int' \Xi(z,\omega,z',\omega')\rho(z',\omega')\,dz'd\omega', \quad (8)$$

where the effect of the wall potentials has been incorporated through restriction of the range of integration over θ:

$$\int' d\omega = \int_0^{2\pi} d\phi \int_{\pi-\theta_m}^{\theta_m} \sin\theta\, d\theta = \int_0^{2\pi} d\phi \int_{-\cos\theta_m}^{\cos\theta_m} dx, \quad (9)$$

with

$$\cos\theta_m = \begin{cases} 1 & \text{if} \quad \left|z - z_0^\alpha\right| \geqslant \frac{L}{2}, \\ \frac{|z-z_0|}{L/2} & \text{if} \quad \left|z - z_0^\alpha\right| < \frac{L}{2}, \end{cases} \quad (10)$$

z_0^α being, we recall, the position of substrate α. In either case, the solution is the density-orientation profile $\rho(z,\omega)$ that minimizes $\Omega[\rho(z,\omega)]$. In the next section, we propose a variant of a MLP ANN, which we denote minimization neural network (MNN), developed in order to follow route (ii).

B. MLP minimization

The MNN that we have designed to minimize the grand-canonical potential comprises, in common with most MLPs, an input layer, one or more hidden layers, and an output layer. Our MNN receives as input a position and an orientation, specified by (z,θ,ϕ), and outputs the expected value of the density-orientation profile at that point, $\rho(z,\theta,\phi)$.

First, note that, as in earlier work [11,23,24], integrations are performed by Gauss-Legendre quadrature by means of the algorithm due to Chrzanowska [43]; here, we have used 64 z points, 16 θ points, and 16 ϕ points. Therefore, it is enough that our MNN be able to estimate densities from a discrete set of (z,θ,ϕ) that are triplets of the chosen Gaussian abscissas. Input is coded by referencing (z,θ,ϕ) by their index within the quadrature. The MNN receives (z,θ,ϕ) coded in a 64-bit string: the first 32 bits code the z position, by setting to "1" the correct bit and to "0" the remaining bits; likewise, θ is coded by 16 bits and ϕ by the other 16 bits. This 64-bit input is fed into the input layer, along with an extra constant input (set to 1) that is required for any perceptron (in order to model its threshold) [44]. The input layer thus has 65 units.

The output layer constructs a linear combination of hidden-layer outputs, followed by application of an activation function. The form of the activation function should mirror, as closely as possible, the distribution of the target values, densities in our case. We observe that, in this problem, the expected distribution for the logarithm of the density is more or less uniform. This justifies our choice of the exponential activation function for the output layer. Neural networks with an exponential activation function for the output layer have been applied previously in the context of information theory, mainly to estimate probability density functions [45,46].

Each unit inside a hidden layer receives input from all units in the preceding hidden layer, or (in the case of the first hidden layer) from all units in the input layer. Each hidden layer unit has its own combining weights. After combining inputs, each unit uses the logistic function as its activation function. The number of units inside each hidden layer is a parameter of the MNN. For the case of a single hidden layer, we explain in Sec. IV how we chose the number of units.

The objective is to make the MNN learn its weights such that the MNN output [the density-orientation profile $\rho(z,\theta,\phi)$ at the Gaussian abscissas triplets] minimizes the grand-canonical potential $\Omega[\rho(z,\theta,\phi)]$. To achieve this, we use backpropagation learning with momentum. The momentum is a parameter that holds memory of the previous learning steps, and provides step acceleration if the gradient direction does not change in the course of several iterations. Momentum must not be greater than 1 because this leads to divergence of the learning process (it would mean that past learning steps would exponentially gain more weight rather than being progressively "forgotten"). We denote the momentum parameter by α, and the learning step parameter by η.

We start by rewriting Eq. (6) in terms of discretized position and orientation variables as

$$\Omega = \frac{1}{2}\sum_{i,j}\left[g_i y_i \left(\frac{M}{\gamma}\right)^2 K_{ij} g_j y_j\right] + \sum_i \left\{g_i y_i \frac{M}{\gamma}\left[\ln\left(y_i \frac{M}{\gamma}\right) - 1\right]\right\}, \quad (11)$$

where indices i and j label points in the space of discretized variables (z,θ,ϕ), g_i and g_j are the scaling factors of Gaussian quadrature [47], $y(z,\theta,\phi) = \rho(z,\theta,\phi)/N$, K_{ij} is the interaction kernel:

$$K_{ij} = \frac{\left(1-\frac{3}{4}\xi\right)\xi}{(1-\xi)^2}\,\Xi(z_i,\theta_i,\phi_i,z_j,\theta_j,\phi_j), \tag{12}$$

and we have defined $\gamma = \sum_m g_m y_m$. For a particular input k,

$$\frac{\partial\Omega}{\partial y_k} = \frac{M g_k}{\gamma}\left[-\frac{M}{\gamma^2}\sum_{ij} g_i y_i K_{ij} g_j y_j + \frac{M}{\gamma}\sum_i g_i y_i K_{ik} - \frac{1}{\gamma}\sum_i g_i y_i \ln\left(y_i\frac{M}{\gamma}\right) + \ln\left(y_k\frac{M}{\gamma}\right)\right]. \tag{13}$$

Therefore, the gradient of Ω with respect to the weights $w_i^{(o)}$ connecting the ith unit in the hidden layer with the output layer, when the MNN is stimulated with input k, is given by

$$\frac{\partial\Omega}{\partial w_i^{(o)}} = \frac{\partial\Omega}{\partial y_k}\frac{\partial y_k}{\partial\mathcal{O}^{(o)}}\frac{\partial\mathcal{O}^{(o)}}{\partial w_i^{(o)}} = \frac{\partial\Omega}{\partial y_k} y_k \frac{\partial}{\partial w_i^{(o)}}\sum_j a_j w_j^{(o)} = \frac{\partial\Omega}{\partial y_k} y_k a_i, \tag{14}$$

where a_i denotes the output of the ith hidden-layer unit upon stimulation by input k, and $\mathcal{O}^{(o)}$, the net output (before application of the activation function) of the output layer, is a linear combination of its (hidden-layer) inputs. Note that, because the activation function is exponential, $\partial y_k/\partial\mathcal{O}^{(o)} = y_k$.

For the weights $w_{ij}^{(h)}$ connecting the ith input-layer unit and the jth hidden-layer unit, we have

$$\frac{\partial\Omega}{\partial w_{ij}^{(h)}} = \frac{\partial\Omega}{\partial y_k}\frac{\partial y_k}{\partial\mathcal{O}^{(o)}}\frac{\partial\mathcal{O}^{(o)}}{\partial a_j}\frac{\partial a_j}{\partial\mathcal{O}_j^{(h)}}\frac{\partial\mathcal{O}_j^{(h)}}{\partial w_{ij}^{(h)}} = \frac{\partial\Omega}{\partial y_k} y_k w_{ij}^{(h)} a_j(1-a_j)\frac{\partial}{\partial w_{ij}^{(h)}}\sum_{i',j'} x_{i'} w_{i'j'}^{(h)} = \frac{\partial\Omega}{\partial y_k} y_k w_{ij}^{(h)} a_j(1-a_j) x_i, \tag{15}$$

where $\mathcal{O}_j^{(h)}$ stands for the net output of the jth hidden-layer unit and we use the logistic activation function $a_j = 1/[1+\exp(-\mathcal{O}_j^{(h)})]$.

We define the momentum learning function as

$$m_{ij}(0) = 0, \tag{16}$$

$$m_{ij}(t+1) = \alpha m_{ij}(t) - \eta\frac{\partial\Omega}{\partial w_{ij}}, \tag{17}$$

where t is the iteration number ("time"). The MNN internal weights are initialized randomly in the range $(-1,1)$ (uniform distribution), and are updated according to $\Delta w_{ij} = m_{ij}(t)$.

We define *epoch* as the process of activating the MNN with every point of the input space, one at a time, and making the consequent updates of the weights. Note that the greatest computational expense in determining $\partial\Omega/\partial w_{ij}^{(h)}$ at input k is that incurred in performing the summation $\sum_{ij} g_i y_i K_{ij} g_j y_j$. Fortunately, this expression does not depend on k, and so it need only be computed once in each epoch, the result then being applied to every point of the input space.

IV. RESULTS

We tested our algorithm by considering a fluid of HGO particles of elongation $\kappa = 3$, sandwiched between two semipenetrable walls a distance $L_z = 4\kappa\sigma_0 = 12\sigma_0$ apart. The density-orientation profile $\rho(z,\omega)$ was computed for different values of the reduced bulk density $\rho^*_{\text{bulk}} \equiv \rho_{\text{bulk}}\sigma_0^3$ and of the reduced needle length $L^* = L/\sigma_L$.

The EL equation was solved by the Picard method, as described in [9–11], until the convergence errors, defined as (i) the sum of the absolute values of the differences between consecutive iterates of $\rho(\omega,z)$ at $64\times16\times16 = 16\,384$ points, and (ii) the difference between the surface tensions in two consecutive iterations were less than 10^{-3}. The mixing parameter was set at 0.9. The MNN was always started from a random intitial guess for $\rho(\omega,z)$, the density-orientation profile (corresponding to an isotropic distribution), so as not to bias the outcome. Most calculations were performed for a single hidden layer, which is guaranteed to be able to represent any Borel-measurable functions [22]; this turned out to be sufficient in most cases (but not all, see following).

In an attempt to ascertain a more comprehensive picture of the merits of our ANN approach, we have also implemented a conjugate gradient-based solver for our confined LC system. This is also a direct minimization method, a more conventional variant of route (ii). However, this alternative scheme proved unreliable in most situations, failing to find the equilibrium density-orientation profile except when provided with an initial guess that was very close to the optimal solution. Therefore, for this particular problem, the conjugate-gradient approach does not seem to offer a practical alternative to the EL and ANN routes considered in more depth.

A. Fine tuning the parameters

MNN uses three parameters: h, the number of units within its hidden layer (aside from the constant unit, set to 1); α, the momentum coefficient; and η, the learning step. α is required to be in range $[0,1)$. h is not known, but could be several tens to hundreds or thousands of units. η is usually smaller than 1, but there are no further requirements.

The first step was to fine tune these parameters. For this purpose, we ran two representative experiments with different mean densities $\rho^*_{\text{bulk}} = 0.28$ [corresponding to the bulk isotropic (I) phase] and $\rho^*_{\text{bulk}} = 0.35$ [corresponding to the bulk nematic (N) phase], performing MNN learning with every combination of $\alpha \in \{0.1,0.3,0.5,0.7,0.9\}$ and $\eta \in 0.1,0.4,0.7,1.0$; $h \in 10;25,50,100,150,200,250$. We let the network run for 5000 epochs (except when it diverged), and inspected the final results. We found that the time taken to process each learning epoch increased in proportion to the number of hidden units; even so, we decided to compare the results of different runs after a set number of epochs, rather than after a set learning time.

We chose $L^* = 1$, which induces parallel anchoring and is typically more demanding numerically than homeotropic anchoring. Results are collected in Tables I–III. Tables I and II show, for two ρ^*_{bulk} values, the grand-canonical potential

TABLE I. Grand-canonical potential Ω after 5000 epochs, for $\kappa = 3$, $L^* = 1$, $\rho^*_{\text{bulk}} = 0.28$ (bulk I phase) and different choices of α, h, and η.

		α				
h	η	0.1	0.3	0.5	0.7	0.9
10	0.1	1.18672	1.144328	1.137906	1.138144	1.155396
10	0.4	1.143171	1.137942	1.138922	1.148959	1.138407
10	0.7	1.137403	1.135984	1.146504	1.139451	1.135816
10	1.0	1.136813	1.138326	1.135614	1.135183	1.134833
25	0.1	1.155936	1.146062	1.139696	1.137812	1.151736
25	0.4	1.142638	1.138965	1.137261	1.149138	1.140928
25	0.7	1.137169	1.136048	1.147395	1.137414	1.136469
25	1.0	1.135599	1.139655	1.135423	1.134769	1.134763
50	0.1	1.161899	1.147519	1.139771	1.138908	1.150827
50	0.4	1.143001	1.1393	1.137638	1.14981	1.145919
50	0.7	1.13813	1.136625	1.14542	1.138345	1.136331
50	1.0	1.135791	1.139874	1.13528	1.135041	1.134815
100	0.1	1.153975	1.143765	1.141101	1.138393	1.150638
100	0.4	1.142178	1.139652	1.137911	1.149062	1.141389
100	0.7	1.138173	1.136224	1.146931	1.138316	1.135955
100	1.0	1.135891	1.13979	1.135491	1.16345	1.134821
150	0.1	1.151959	1.143478	1.141839	1.139804	1.148809
150	0.4	1.141686	1.140083	1.138062	1.147389	1.139929
150	0.7	1.138489	Diverged	1.144011	1.138239	1.136154
150	1.0	Diverged	1.139816	1.135752	1.135015	1.135209
200	0.1	1.150801	1.143571	Diverged	1.137666	1.149625
200	0.4	1.141804	1.139046	1.13702	1.147081	1.140075
200	0.7	Diverged	Diverged	1.144176	1.138129	Diverged
200	1.0	Diverged	1.139729	1.135639	Diverged	1.135132
250	0.1	1.153768	1.142571	1.140022	Diverged	1.148498
250	0.4	1.140632	Diverged	Diverged	1.147263	Diverged
250	0.7	Diverged	Diverged	1.143332	Diverged	Diverged
250	1.0	Diverged	1.138431	1.13553	Diverged	Diverged

Ω obtained for the full set of h, η, and α considered. In Table III, summary data are presented showing the 1000- and 2000-epoch Ω values from batches of 10 equivalent runs with $h = 10$ and a range of η and α values. From these we conclude the following:

(1) The probability of divergence increases with h, η, α, and ρ^*_{bulk}. We also observe that for small h (less than 25) learning is fairly robust: there are no instances of repeated divergence, although in batches of 10 simulations (see Table III), there were occasional simulations that diverged for $\eta = 1.0$, $\alpha = 0.9$.

(2) Learning speed increases with η and α (as expected). Provided that $h \leqslant 25$ we may, if convergence speed is important and occasional divergence can be accommodated, set $\eta = 1.0$ and $\alpha = 0.9$. If, alternatively, divergence must be avoided, we should set $\eta = 0.7$ and $\alpha = 0.9$.

(3) Given that they yield very similar average results, either $h = 10$ or 25 could be chosen. Our results suggest that $h = 25$ may perform slightly the better of the two, but the reduced computation time per epoch (and, hence, shorter overall time) could be an argument for choosing $h = 10$. That said, it is known that using a larger number of units leads to higher representation power. We therefore decided to select $h = 25$ in our subsequent calculations using the MNN for determine the density profile for alternative boundary condition problems.

TABLE II. Same as Table I, but for $\rho^*_{\text{bulk}} = 0.35$ (bulk N phase).

		α				
h	η	0.1	0.3	0.5	0.7	0.9
10	0.1	9.101475	8.971296	8.955008	8.963734	9.023824
10	0.4	8.966737	8.962024	8.962016	8.999954	8.955046
10	0.7	8.952314	8.954215	8.966831	8.951803	8.968915
10	1.0	8.959308	8.965214	8.94864	8.967841	8.948766
25	0.1	9.032404	8.977229	8.954232	8.965194	9.030154
25	0.4	8.974211	8.953414	8.971946	8.99511	8.962507
25	0.7	8.970363	8.959918	8.979666	8.950231	8.959656
25	1.0	8.950319	8.961905	8.957792	8.958059	8.957027
50	0.1	9.09479	8.959672	8.964864	8.962899	9.015685
50	0.4	8.976157	8.953012	8.952501	8.99655	8.96172
50	0.7	8.961037	8.961134	8.973959	8.969405	8.958759
50	1.0	8.949888	8.972432	8.958441	8.947867	Diverged
100	0.1	9.091432	8.97678	8.96335	Diverged	9.031278
100	0.4	8.972937	8.962469	8.966482	9.013989	8.972309
100	0.7	8.969883	Diverged	8.985739	8.960194	Diverged
100	1.0	Diverged	8.952872	8.948058	Diverged	9.149149
150	0.1	9.065501	8.957815	Diverged	8.963453	9.03997
150	0.4	8.985513	Diverged	Diverged	8.994269	Diverged
150	0.7	Diverged	Diverged	8.985764	Diverged	Diverged
150	1.0	Diverged	8.962265	Diverged	Diverged	Diverged
200	0.1	9.077518	Diverged	Diverged	8.963536	9.034062
200	0.4	8.961021	Diverged	Diverged	9.014442	Diverged
200	0.7	Diverged	Diverged	8.981122	Diverged	Diverged
200	1.0	Diverged	8.9528	Diverged	Diverged	Diverged
250	0.1	9.057811	8.990148	Diverged	Diverged	9.008942
250	0.4	Diverged	Diverged	Diverged	8.995841	Diverged
250	0.7	Diverged	Diverged	8.981818	Diverged	Diverged
250	1.0	Diverged	8.952725	Diverged	Diverged	Diverged

In summary, in what follows we employ (except where otherwise indicated) $h = 25$, $\eta = 1.0$, and $\alpha = 0.9$. Convergence of the MNN was deemed to have been achieved if the difference (defined as in [11]) between two solutions 1000 iterations apart was less than 10^{-4}.

B. MNN versus iterative solution

We next assessed our new MNN method by comparing its results for the structure of a symmetric film with those obtained by standard iterative solution of the EL equation (8), and with computer simulations (NVT Monte Carlo for $N = 1000$ particles), where available. Details of the simulations are given in [23,24]. Once $\rho(\omega,z)$ has been found, we can integrate out the angular dependence to get the density profile

$$\rho(z) = \int \rho(z,\omega)\, d\omega, \tag{18}$$

and use this result to define the orientational distribution function (ODF) $\hat{f}(z,\omega) = \rho(z,\omega)/\rho(z)$, from which we can calculate the orientational order parameters in the laboratory-fixed frame [10]. These are the five independent components of the nematic order parameter tensor $Q_{\alpha\beta} = \langle \frac{1}{2}(3\hat{\omega}_\alpha\hat{\omega}_\beta - \delta_{\alpha\beta})\rangle$.

TABLE III. Grand-canonical potential Ω values after 1000 and 2000 MNN epochs, for $h = 10$ and different choices of momentum and learning step parameters. The HGO model parameters are $\kappa = 3$, $\rho^*_{\rm bulk} = 0.35$ (bulk N phase), and $L^* = 1$ (impenetrable walls). Statistics are compiled from batches of 10 runs; * means that one run diverged and was not included in the calculations.

	$\eta = 1.0, \alpha = 0.9$		$\eta = 1.0, \alpha = 0.7$		$\eta = 0.7, \alpha = 0.9$		$\eta = 0.7, \alpha = 0.7$	
	1000	2000	1000	2000	1000	2000	1000	2000
Mean	8.964565*	8.957609*	8.968601	8.965057	8.961541	8.959473	8.970223	8.962241
Median	8.960294	8.958088	8.969784	8.965568	8.960423	8.958575	8.970974	8.962251
Maximum	Diverged	Diverged	8.975090	8.971631	8.970380	8.968318	8.977538	8.972639
Minimum	8.949732	8.948179	8.955962	8.952717	8.951463	8.949201	8.957841	8.951727
St. deviation	0.016548*	0.008898*	0.006345	0.006298	0.006714	0.006718	0.006962	0.006562

In the case under study there is no twist, i.e., the director is confined to a plane that we can take as the xz plane and $Q_{yy} = Q_{yz} = 0$. The three remaining order parameters Q_{xy}, Q_{zz}, and Q_{xz} [because $Q_{\alpha\beta}$ is traceless, $Q_{xx} = -(Q_{xx} + Q_{zz})$] are in general all nonzero owing to surface-induced biaxiality (see our earlier work for $L^* = 1$ [11]). This effect has not been neglected in the present treatment, but in what follows we show results for Q_{zz} only, as it (i) allows one readily to distinguish between homeotropic and planar states; and (ii) is usually the largest order parameter (in absolute value). It is given by

$$Q_{zz}(z) = \int P_2(\cos\theta)\,\hat{f}(z,\omega)\,d\omega. \tag{19}$$

Figures 4–11 show the density and order parameter profiles from EL and MNN minimization of the free energy, compared with MC simulation data. There is very good agreement between EL and MNN at the lower (isotropic) density, for all values of the reduced needle length L^*. At the higher (nematic) density, however, there is again perfect agreement for $L^* = 0$, $\frac{1}{3}$, and 1; for $L^* = \frac{2}{3}$, EL minimization predicts alignment of the LC parallel to the walls, as seen in simulations [23], whereas MNN with a single hidden layer predicts homeotropic alignment, which is metastable for this value of L^*. Indeed, we expect a crossover from homeotropic to planar equilibrium alignment at $L^* \approx 0.5$ [23], so it is not altogether surprising that convergence to the absolute free energy minimum should be harder in this range, where bistability is often observed in simulations. Use of a MNN with *two* hidden layers, however, allowed us to reach the correct equilibrium state at little extra computational cost, in 3 out of 10 attempts, all starting from different random initial guesses for $\rho(\omega,z)$. In Fig. 9 we plot both the metastable and true equilibrium profiles, for which the free energy (in reduced units) is, respectively, 8.937 and 8.783. In the latter case, there is again perfect agreement between the EL and MNN results. Note that the metastable state is also a solution of the EL equation, for a different choice of initial guess.

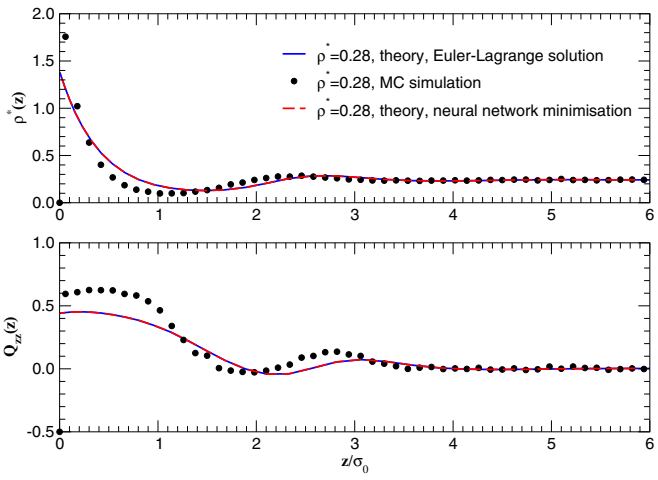

FIG. 4. (Color online) Density $\rho^*(z)$ (top) and order parameter $Q_{zz}(z)$ (bottom) profiles for a symmetric film of HGO particles of elongation $\kappa = 3$, for $\rho^*_{\rm bulk} = 0.28$. The needle length is $L^* = 0$ on both walls, inducing homeotropic anchoring (only one half of system is shown). Lines are from theory using the standard solution of the EL equation (solid) and our MNN (dashed), symbols are from simulation. This density lies in the isotropic phase.

C. Assessment of computational costs

In order to contrast the performances of MNN minimization versus iterative solution of the EL equation, we ran three different codes on a laptop computer with a CPU of 2.20 GHz and 2 GB RAM, under the MS Windows Vista operating system. The confined HGO fluid parameters are $\kappa = 3$, $L^* = 1$, $\rho_{\rm bulk} = 0.35$, corresponding to the most demanding case of a bulk N phase between fully impenetrable walls. The EL code was run until the error, defined as the sum of the

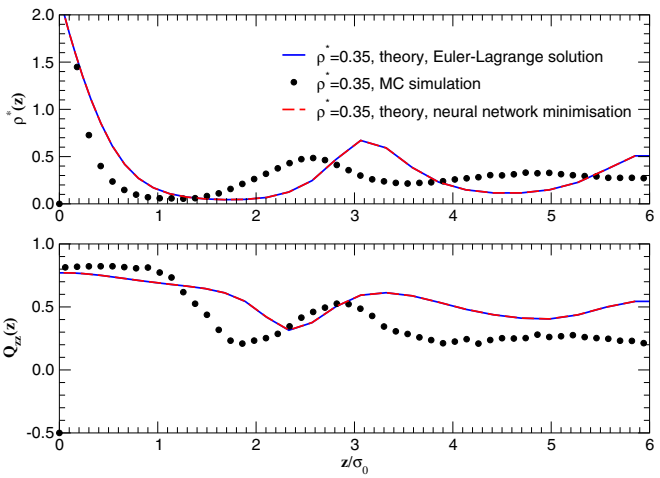

FIG. 5. (Color online) Same as Fig. 4, but for $\rho^*_{\rm bulk} = 0.35$. This density lies in the nematic phase.

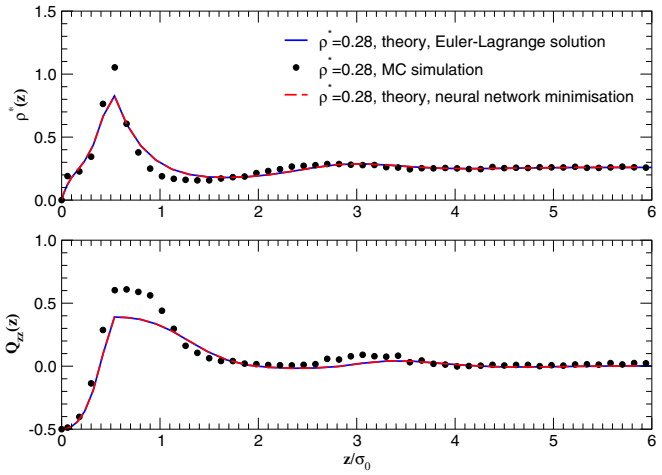

FIG. 6. (Color online) Density $\rho^*(z)$ (top) and order parameter $Q_{zz}(z)$ (bottom) profiles for a symmetric film of HGO particles of elongation $\kappa = 3$, for $\rho^*_{\text{bulk}} = 0.28$. The needle length is $L^* = \frac{1}{3}$ on both walls, inducing homeotropic anchoring (only one half of system is shown). Lines are from theory using the standard solution of the EL equation (solid) and our MNN (dashed), symbols are from simulation. This density lies in the isotropic phase.

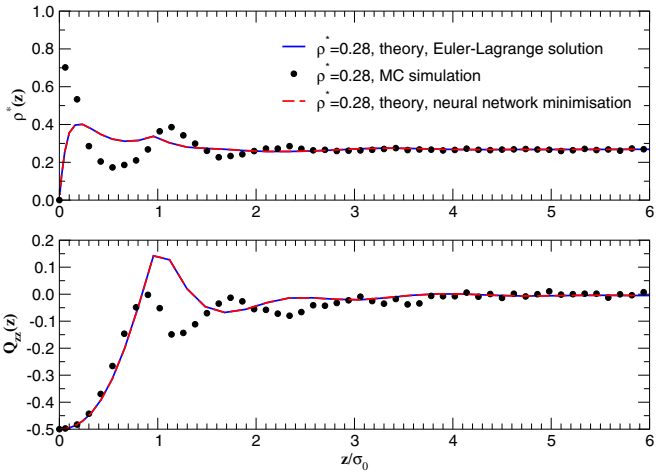

FIG. 8. (Color online) Density $\rho^*(z)$ (top) and order parameter $Q_{zz}(z)$ (bottom) profiles for a symmetric film of HGO particles of elongation $\kappa = 3$, for $\rho^*_{\text{bulk}} = 0.28$. The needle length is $L^* = \frac{2}{3}$ on both walls, inducing parallel anchoring (only one half of system is shown). Lines are from theory using the standard solution of the EL equation (solid) and our MNN (dashed), symbols are from simulation. This density lies in the isotropic phase.

absolute values of the differences between consecutive iterates at $64\times16\times16 = 16\,384$ points, was less than 10^{-2}; the MNN was run for 1000 or 2000 iterations.

(A) Standard iterative solution of the EL equation coded in C: runtime for 711 iterations is 436 minutes; final free energy is 8.965674.

(B) MNN coded in C, using $h = 25$, the K_{ij} matrix is computed on the fly: runtime for 1000 iterations is 150 minutes. Using a prestored K would be faster, but by far not as fast as MATLAB because MATLAB uses machine code optimized for matrix calculation.

(C) MNN coded in MATLAB, using $h = 25$ and prestored K_{ij} matrix: runtime for 1000 iterations is 4 min. The K_{ij} matrix, which is reusable for any simulation with the same model parameters, takes an additional 38 min to generate.

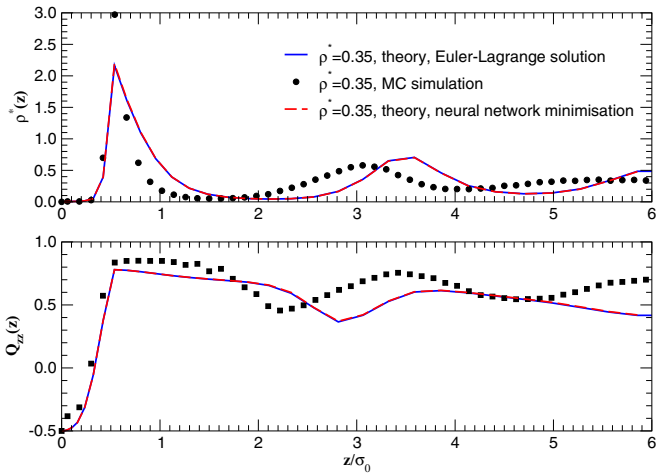

FIG. 7. (Color online) Same as Fig. 6, but for $\rho^*_{\text{bulk}} = 0.35$. This density lies in the nematic phase.

Storing the K_{ij} matrix has obvious advantages in processing time, but we must remember that K_{ij} is a huge matrix that uses 512 MB storage for this problem, making this approach very difficult to scale up to higher dimensions; the size of K_{ij} increases quadratically with the number of Gaussian quadrature points included in any extra dimension.

Simulations of types B and C are algorithmically equivalent, the only difference is whether or not matrix K_{ij} is prestored, so the final free energies are the same. Results depend on the MNN parametrization (see Table III); for $\eta = 0.7, \alpha = 0.9$, the mean and median final free energy are, respectively, 8.959 473 and 8.958 575. This is within less than 0.1% of the EL result.

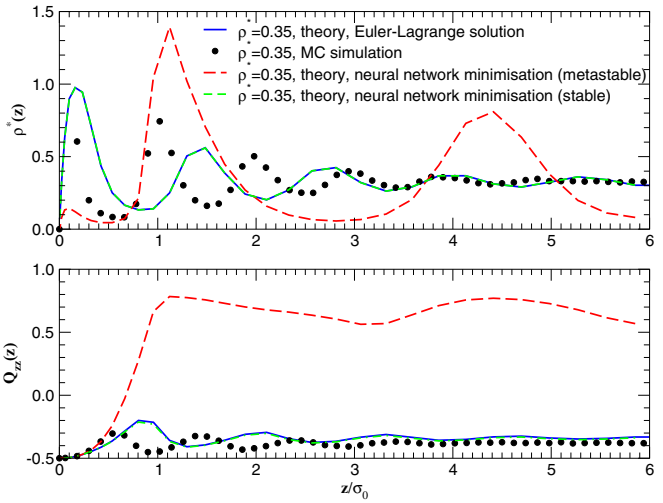

FIG. 9. (Color online) Same as Fig. 8, but for $\rho^*_{\text{bulk}} = 0.35$. This density lies in the nematic phase. MNN results were calculated using two two hidden layers, $\eta = 1.0$ and $\alpha = 0.97$. The metastable profiles are for homeotropic alignment, the true equilibrium profiles are for planar alignment.

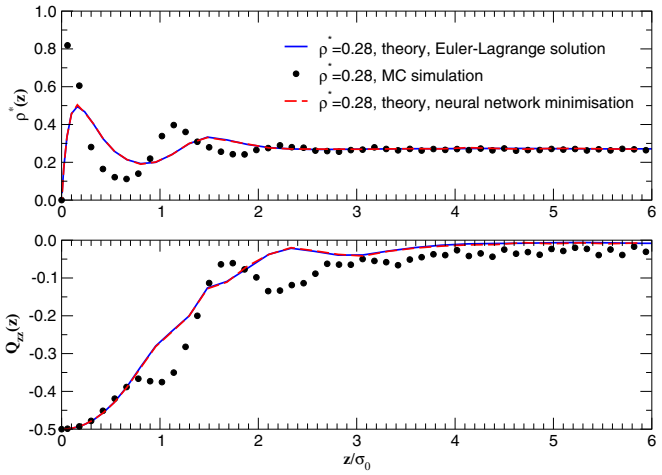

FIG. 10. (Color online) Density $\rho^*(z)$ (top) and order parameter $Q_{zz}(z)$ (bottom) profiles for a symmetric film of HGO particles of elongation $\kappa = 3$, for $\rho^*_{\text{bulk}} = 0.28$. The needle length is $L^* = 1$ on both walls, inducing parallel anchoring (only one half of system is shown). Lines are from theory using the standard solution of the EL equation (solid) and our MNN (dashed), symbols are from simulation. This density lies in the isotropic phase.

V. DISCUSSION AND CONCLUSIONS

We have developed and implemented a MNN for the grand-canonical functional of confined hard nonspherical particles. This has been tested for the HGO fluid treated at the level of the simple Onsager approximation with a "bulk" Parsons-Lee scaling. Results were found to be in very good agreement with those from iterative solution of the EL equation, provided we use two hidden layers. Our MNN appears, however, to be substantially faster, which fact, coupled with its reliability, makes it a strong candidate for solving the structure of confined fluids. Speed and economy of memory storage are of particular importance if one wishes to consider systems where there is spatial variation in more than one dimension, or where the particles are biaxial, in which cases the EL-based method becomes at worst inapplicable, at best extremely expensive on computer resources. To see why this is so, consider a stripe-patterned substrate: now each particle needs to be specified by one additional spatial coordinate, say x, along the plane of the substrate. Hence, the interaction kernel K_{ij} [Eq. (12)] would depend on two additional variables x_i and x_j; if we choose the number of integration points along x to be, e.g., $n_x = 32$, then K_{ij} would grow by a factor of $n_x^2 = 1024$. The same comment applies if the particles are biaxial, where now the additional coordinate is the third Euler angle χ. By contrast, our MNN is sufficiently fast that K_{ij} can be computed "on the fly," with substantial RAM savings.

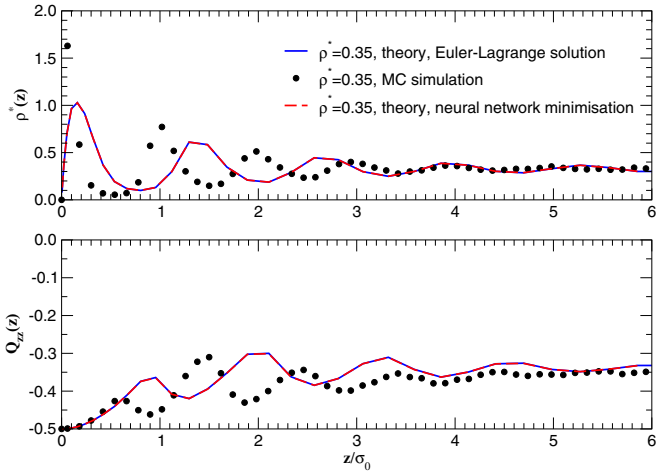

FIG. 11. (Color online) Same as Fig. 10, but for $\rho^*_{\text{bulk}} = 0.35$. This density lies in the nematic phase.

The MNN method as presented is general and can be applied to any functional of the density-orientation profile; we have tested it for the simplest possible, Onsager-type approximation to the free energy of a fluid of hard rods. More sophisticated theoretical approaches are of course available, such as a weighted-density [42] or fundamental-measure [48,49] approximation, which would very likely be more accurate, i.e., reproduce simulations more closely, but our purpose, as stated above, was just to develop a new solution algorithm and assess its performance.

Our method can be fine tuned in a number of ways, which might yield further performance and reliability gains. In particular, more work needs to be done to ensure that we always converge to the true absolute minimum of the free energy, even when there are competing metastable states.

(1) *Symmetries*. For systems with symmetric substrate conditions, we may reduce the storage requirements by using an interaction matrix of size $(n_z/2) \times n_z$ (where n_α is the number of discrete values of coordinate α); indeed, equivalent efficiencies can be achieved where there are any other planes of symmetry. Thus, if we assume that there are planes of symmetry perpendicular to the x, z, and ϕ axes, the interaction matrix will scale as $\mathcal{N} = (n_x^2/2) \times (n_z^2/2) \times n_\theta^2 \times (n_\phi^2/2)$. Its size will then be $\mathcal{N}$ times the 8 bytes that are needed for representing high-precision floating-point numbers. For $n_x = n_z = 32$, $n_\theta = n_\phi = 16$, this yields a total RAM requirement of 64 GB, which *is* achievable with high-end computational hardware and/or parallelization.

(2) *Network topology*. So far, we have studied the impact on learning speed of varying the number of units within each layer, the learning rate, and momentum factor, but not of increasing the number of hidden layers.

(3) *Training algorithm*. Several methods can be used to improve the training speed of a standard MLP. Amongst these, the most common and most successful are the conjugate-gradient algorithms and quasi-Newton algorithms. The network employed in our ANN is not a standard MLP, and in order to benefit from these algorithms it would be necessary to adapt them to an unsupervised network. Quasi-Newton algorithms include Broyden-Fletcher-Goldfarb-Shanno (BFGS), one-step secant, and Levenberg-Marquardt algorithms. These base their weight update functions on the Hessian matrix, rather than the gradient of the energy. In a very large data set, finding the Hessian matrix is extremely time and memory consuming, which renders the quasi-Newton approach unfeasible on a standard computer. We have adapted the conjugate-gradient algorithm to compute the error gradient of the free energy for our unsupervised network (instead of the quadratic error of a standard MLP). We then used the Fletcher-Reeves update to compare the learning speed with that obtained with our

backpropagation algorithm. This algorithm, like Polák-Ribiere and Powell-Beale, requires a line search to determine the minimum energy in a particular direction. This line search is itself extremely time consuming, and our experiments with Fletcher-Reeves have shown an overall learning speed slower than backpropagation. Moreover, convergence seemed more likely to become trapped in local minima. One possible solution would be to implement a scaled-conjugate-gradient algorithm [50]; these have been designed with the aim of avoiding the time-consuming line search of standard conjugate-gradient algorithms.

Given the above, the MLP approach would appear to offer a significant opportunity in the context of complex LC alignment calculations. There is no prospect of conventional iterative approaches being able to deal with cases with in-plane substrate variation, so this alternative is very welcome.

ACKNOWLEDGMENTS

We thank D. de las Heras, L. Harnau, M. Schmidt, and N. M. Silvestre for discussions. This work was funded by the British Council under Treaty of Windsor Grant No. B-54/07; by the Portuguese Foundation for Science and Technology (FCT), through Contracts No. PTDC/FIS/098254/2008, Projecto Estratégico "Centro de Física Teórica e Computacional 2011-2011" No. PEst-OE/FIS/UI0618/2011, and No. EXCL/FIS-NAN/0083/2012; and by the UK Engineering and Physical Research Council, Grant No. GR/S59833/01.

[1] http://www.displaybank.com/eng/.
[2] P. G. de Gennes and J. Prost, *The Physics of Liquid Crystals*, 2nd ed. (Oxford University Press, Oxford, 1992).
[3] B. Jérôme, Rep. Prog. Phys. **54**, 391 (1991).
[4] T. J. Sluckin, Contemp. Phys. **41**, 37 (2000).
[5] G. P. Bryan-Brown, E. L. Wood, and I. C. Sage, Nature (London) **399**, 338 (1999).
[6] C. V. Brown, M. J. Towler, V. C. Hui, and G. P. Bryan-Brown, Liq. Cryst. **27**, 233 (2000).
[7] N. A. Lockwood and N. L. Abbott, Current Opinion Coll. Interf. Sci. **10**, 111 (2005).
[8] P. Poulin, H. Stark, T. C. Lubensky, and D. A. Weitz, Science **275**, 1770 (1997).
[9] J. P. Hansen and I. R. McDonald, *Theory of Simple Liquids*, 2nd ed. (Academic, London, 1986).
[10] M. M. Telo da Gama, Mol. Phys. **52**, 611 (1984).
[11] A. Chrzanowska, P. I. C. Teixeira, H. Eherentraut, and D. J. Cleaver, J. Phys.: Condens. Matter **13**, 4715 (2001).
[12] D. de las Heras, L. Mederos, and E. Velasco, Phys. Rev. E **68**, 031709 (2003).
[13] D. de las Heras, E. Velasco, and L. Mederos, J. Chem. Phys. **120**, 4949 (2004).
[14] J. P. Bramble, S. D. Evans, J. R. Henderson, C. Anquetil, D. J. Cleaver, and N. J. Smith, Liq. Cryst. **34**, 1059 (2007).
[15] Y. Yi, V. Khire, C. Bowman, J. Maclennan, and N. Clark, J. Appl. Phys. **103**, 093518 (2008).
[16] C. Anquetil-Deck and D. J. Cleaver, Phys. Rev. E **82**, 031709 (2010).
[17] C. Anquetil-Deck, D. J. Cleaver, and T. J. Atherton, Phys. Rev. E **86**, 041707 (2012).
[18] S. Kitson and A. Giesow, Appl. Phys. Lett. **80**, 3635 (2002).
[19] C. Tsoktas, A. J. Davidson, C. V. Brown, and N. J. Mottram, Appl. Phys. Lett. **90**, 111913 (2007).
[20] G. G. Wells and C. V. Brown, Appl. Phys. Lett. **91**, 223506 (2007).
[21] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. (Prentice-Hall, New Jersey, 1998).
[22] K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**, 359 (1989).
[23] P. I. C. Teixeira, F. Barmes, and D. J. Cleaver, J. Phys.: Condens. Matter **16**, S1969 (2004).
[24] P. I. C. Teixeira, F. Barmes, C. Anquetil-Deck, and D. J. Cleaver, Phys. Rev. E **79**, 011709 (2009).
[25] H. Bourlard, N. Morgan, and S. Renals, Speech Commun. **11**, 237 (1992).
[26] J. Gutiérrez and Ali S. Hadi, *Expert Systems and Probabilistic Network Models* (Springer, Berlin, 1997).
[27] P. J. Cheng and S. C. Lin, Int. J. Machine Tools Manufact. **40**, 1185 (2000).
[28] N. Intrator, Neural Comput. **4**, 98 (1992).
[29] M. Gardner and S. Dorling, Atmos. Environ. **34**, 21 (2000).
[30] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms* (Spartan, Washington DC, 1962).
[31] C. M. Care and D. J. Cleaver, Rep. Prog. Phys. **68**, 2665 (2005).
[32] M. Rigby, Mol. Phys. **68**, 687 (1989).
[33] J. W. Perram and M. S. Wertheim, J. Comput. Phys. **58**, 409 (1985); J. W. Perram, J. Rasmussen, E. Praestgaard, and J. L. Lebowitz, Phys. Rev. E **54**, 6565 (1996).
[34] M. P. Allen, G. T. Evans, D. Frenkel, and B. M. Mulder, in *Advances in Chemical Physics*, edited by I. Prigogine and S. A. Rice (John Wiley and Sons, Hoboken, NJ, USA, 1993), Vol. 86, pp. 1–166.
[35] V. R. Bhethanabotla and W. Steele, Mol. Phys. **60**, 249 (1987).
[36] S. L. Huang and V. R. Bhethanabotla, Int. J. Mod. Phys. C **10**, 361 (1999).
[37] E. Velasco and L. Mederos, J. Chem. Phys. **109**, 2361 (1998).
[38] D. J. Cleaver and P. I. C. Teixeira, Chem. Phys. Lett. **338**, 1 (2001).
[39] R. Evans, Adv. Phys. **28**, 143 (1979).
[40] A. Poniewierski, Phys. Rev. E **47**, 3396 (1993).
[41] J. D. Parsons, Phys. Rev. A **19**, 1225 (1979); S. D. Lee, J. Chem. Phys. **78**, 4972 (1987).
[42] A. M. Somoza and P. Tarazona, J. Chem. Phys. **91**, 517 (1989); E. Velasco, L. Mederos, and D. E. Sullivan, Phys. Rev. E **62**, 3708 (2000).
[43] A. Chrzanowska, J. Comput. Phys. **191**, 265 (2003).

[44] W. McCulloch and W. Pitts, Bull. Math. Biophys. **7**, 115 (1943).
[45] D. F. Specht, Neural Networks **3**, 109 (1990).
[46] D. S. Modha and Y. Fainman, IEEE Trans. Neural Networks **5**, 519 (1994).
[47] See, e.g., W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed. (Cambridge University Press, Cambridge, 1993).
[48] H. Hansen-Goos and K. Mecke, Phys. Rev. Lett. **102**, 018302 (2009).
[49] R. Wittmann and K. Mecke, J. Chem. Phys. **140**, 104703 (2014).
[50] M. F. Møller, Neural Networks **6**, 525 (1993).