Marlics: A finite difference liquid crystal simulation package

R. F. de Souza[a,], E. K. Omori[a], R. S. Zola[a,b]

[a]Departamento de Fsica, Universidade Estadual de Maring, Avenida Colombo 5790, 87020 - 900 Maring PR, Brazil
[b]Universidade Tecnologica Federal do Paran, Rua Marcilio Dias 635, 86812-460 Apucarana, Paran, Brazil

Abstract

In this paper we present Marlics to the world. Marlics is a software written in C++ to solve the Berris-Edwards equation of nematodynamics without flow for both nematic and cholesteric liquid crystals. The program takes as input an descriptive file giving the simulations parameters and initial conditions generating a series of different snapshots. The code is organized in class modules which can be modified by the user base to attend their further needs. Review the abstract after the paper is finished

Keywords: Liquid crystals, Landau-de Gennes, finite differences.

PROGRAM SUMMARY

Program Title: MarLicS

Licensing provisions: GNU General Public License v3.0 (GPL)

Programming language:C++

Supplementary material: Complete instructions about program usage can be found in the user-guide.

Nature of problem: Marlics was developed to simulate liquid crystal devices via solution of the Berris-Edwards system of differential equations equations without flow.

Solution method: The system of equations is solved using finite differences in both time and space. The time integration is performed using an explicit integrator with or without variable time-step.

External routines: The code needs the GSL (Gnu scientific library), an implementation of the CBLas library and an implementation of the OpenMp library(optional).

Running time: From minutes to hours depending on the problem size.

Computer: Single or multi-core processor with shared memory.

RAM: From hundreds of megabytes to gigabytes depending on the problem size.

Restrictions: The code is parallelized using OpenMp, consequently it can only be run in parallel with shared memory processors.

Additional comments: The source code comes with a Mathematica notebook with can be used to aid the user to implement additional interactions, boundary conditions or other situations situation not covered by the current software.

*Corresponding author.
E-mail address: EuOuZola@somewhere.edu

1. Introduction

Liquid crystals are excellent materials to perform these functions, since it can selectively reflect or transmit the incoming light depending on its state, which can be controlled by an external perturbation. In displays, one of the most common application, it used of electric fields the control the liquid crystals state []. The difference among the many types of displays is due to its boundary conditions, with strongly interferes in the device operation [].

The design of new devices requires a great amount of experimentation and empirical knowledge, which would require a unpreactical number of experiments to be performed. As an alternative, the researcher can turn to modeling softwares to aid in the discovery process. Before prototyping a device it can be simulated in a package in many forms before being prototyped, saving time and resources.

There are available some commercial softwares available for simulation of liquid crystals devices, for instance [] and []. These softwares provides out of the box capabilities for simulation and visualization, however it lacks the possibility for extension by the user, also, if the project is discontinued, there is no way to implement new features. Recently, an open source software was released to search for the minimum energy of liquid crystal[**?**], however the code is focused in the equilibrium state of the sample.

In this paper we present *marlics*, it is a open source code designed to simulate the dynamics of liquid crystal order parameters. The code The code is written in C++ using an independent system of classes, which provide the building blocks for the most common system, but also, it is easy to extend for cases not covered by the actual program.

2. Theoretical background

The scalar order parameters $\{S, P\}$ and the director and codirector $\{\vec{n}, \vec{l}\}$ can be combined in unique order parameter Q_{ij} , which is a second rank tensor whose elements are given by:

$$Q_{ij} = \frac{1}{2}S\,(3n_in_j - 1) + P(l_il_j - m_im_j) \tag{1}$$

where $i = 1, 2, 3$ and $j = 1, 2, 3$. This order parameter is symmetric and traceless, therefore only 5 independent elements, for example $\{Q_{11}, Q_{12}, Q_{13}, Q_{22}, Q_{23}\}$ are necessary to fully determine it.

Deviations from the equilibrium value of the scalar order paramter, or spatial variations of the director has associate and energy density given by Landau- de Gennes energy density ($f_{LDG}(\mathbf{Q})$:

$$\begin{aligned} f_{LDG}(\mathbf{Q}) = {} & \frac{a}{2}(T - T^*)\mathrm{Tr}(\mathbf{Q}^2) + \frac{B}{3}\mathrm{Tr}(\mathbf{Q}^3) + \frac{C}{4}\mathrm{Tr}(\mathbf{Q}^2)^2 \\ & + \frac{1}{2}L_1\left(\partial_iQ_{jk}\right)\left(\partial_iQ_{jk}\right) + \frac{1}{2}L_2\left(\partial_iQ_{ji}\right)\left(\partial_kQ_{jk}\right) \\ & + \frac{1}{2}L_3Q_{ij}\left(\partial_iQ_{kl}\right)\left(\partial_jQ_{kl}\right) + \frac{4\pi}{P_0}L_q\epsilon_{ijk}Q_{ij}\left(\partial_jQ_{ik}\right) \end{aligned} \tag{2}$$

where $Q_{ij,k} = \partial Q_{ij}/\partial x_k$, ϵ_{ijk} is the Levi-Civita tensor, $\{L_1, L_2, L_3, L_q, L_s\}$ are the elastic constants, $\{a, B, C\}$ are thermodynamic constants related to the nematic isotropic transition and T and T^* are the system temperature and the virtual nematic-isotropic phase transition temperature, respectively. Here we used Einstein summation convention in repeated indexes.

The dielectric energy density is given by:

$$f_e(\mathbf{Q}) = -\frac{1}{3}\epsilon_0\Delta\epsilon^m E_iE_jQ_{ij} + \frac{2}{\epsilon_0}\mathbf{E}\cdot\mathbf{E}, \tag{3}$$

where e_0 is the vacuum dieletric constant, $\Delta\epsilon = \epsilon_{\parallel} - \epsilon_{\perp}$ is the dielectric anisotropy which measures the diference between the dieletric constant parallel($\epsilon_{\parallel}$) and perpendicular ($\epsilon_{\perp}$) to the liquid crystal director. The volume energy density (f_v) will be given by the sum of all energy terms being considered:

$$f_v(\mathbf{Q}) = f_{ldg}(\mathbf{Q}) + f_e(\mathbf{Q}) \tag{4}$$

The liquid crystal also interacts with the confining surfaces, which can induce an order parameter at the surface and a proffered direction for the direction, which is called surface easy axis n_0. One of the simplest form of the surface energy density between a liquid crystal and a surface is given by the Rapini-Papoular (also called Nobili-Durant) which is given by:

$$f_{rp}((Q)) = \frac{1}{2}W_1(Q_{ij} - Q^0_{ij})(Q_{ij} - Q^0_{ij}) \tag{5}$$

where Q^0_{ij} is the surface induced order parameter, which can be given in its tensorial form, or constructed using the induced scalar order parameters $\{S^0, P^0\}$ and the induced easy axis $\{\vec{n}, \vec{l}\}$ using expression 1.

When the liquid crystals boundary is a liquid or gas, instead of inducing a preferred direction the surface may induces a a preferred plane of orientation perpendicular to the surface. Any variation of the director inside this plane gives the same energy. This type of surface energy is described by the Fournier-Galatola anchoring energy given by [1]:

$$f_{FG}(\mathbf{Q}) = W\left(\tilde{Q_{ij}}(\mathbf{Q}) - \tilde{Q}^{\perp}_{ij}(\mathbf{Q})\right)\left(\tilde{Q_{ij}}(\mathbf{Q}) - \tilde{Q}^{\perp}_{ij}(\mathbf{Q})\right) \tag{6}$$

where W is the anchoring strength constant, $\tilde{Q_{ij}}(\mathbf{Q}) = Q_{ij} + \frac{S_0}{3}\delta_{ij}$ and $\tilde{Q_{ij}}^{\perp}(\mathbf{Q}) = (\delta_{ik} - \nu_i\nu_k)Q_{kl}(\delta_{lj} - \nu_l\nu_j)$. Here $\vec{\nu} = \{\nu_1, \nu_2, \nu_2\}$ is the normal surface vector.

The time evolution of the order parameter is given by the Beris-Edwards set of equations. If we neglect the liquid crystal flow, the time evolution of Q_{ij} in the bulk will be given by:

$$\mu\frac{\partial Q_{ij}}{\partial t} = -\left(\frac{\partial f_V(\mathbf{Q})}{\partial Q_{ij}} - \frac{\partial}{\partial x_k}\frac{\partial f_V(\mathbf{Q})}{\partial Q_{ij,k}}\right), \tag{7}$$

where μ is the bulk viscosity and $\partial Q_{ij}\partial Qkl = (\delta_{ik}\delta_{jl} + \delta_{jk}\delta_{kl} - 2\delta_{ij}\delta_{kl}/3)$. Meanwhile the dynamics in the bulk will be given by

$$\mu_s\frac{\partial Q_{ij}}{\partial t} = -\left(\nu_k\frac{\partial f_{LDG}(\mathbf{Q})}{\partial Q_{ij,k}} - \frac{\partial f_{pen}(\mathbf{Q})}{\partial Q_{ij}}\right), \tag{8}$$

here μ_s is the surface viscosity.

3. Software Usage:

Here we present the basic information necessary to install and use the software. The complete information about software usage can be found in the supplementary material "Userguide".

3.1. Installation:

The installation of Marlics in Unix systems is very straightforward. The source code comes with a *makefile* to help user compile it in its computer. Actually the makefile provides an automatic installation for two compilers (being one of them free). The user will need just to assure he/she has the *make* software and the necessary external libraries and their respective developer files installed. These libraries are: GSL (Gnu Scientific library), OpenMp(optional, but highly recomended) and a CBLAS implementation (you can use the GSL implementation for example). If everything is present, the user just need to open a terminal in the program folder and type:

```
make
```

to compile the program. Once the compilation is done, the user will find a executable named *marlics* in the instalation folder. For ease of use, this executable can be added to one of the system search paths for binaries files, or the user can add the installation folder to the list of search-able paths. Its is also possible to run the simulations in the same path that the software is installed, but we strong recommend against, since it can be become very confuse.

3.2. Simulation set up and Execution:

To execute marlics you must call the program passing an input file which sets the simulation parameter as follows:

```
marlics intput_file
```

being *input_file* the file containing the simulation parameters.

We already provide input files for some situations that the user can use to test the program, or as a base to their own simulations. All the parameters necessary to set up the simulation must be passed to the program via the input file. An entry in the input file is set by passing the parameter name followed by the parameter required values:

```
parameter value
```

Comments can be placed in the input file starting a line with "#", everything in this line will be ignored by marlics. Also, everything following the required parameters will also be ignored by marlics. We found it quite useful to let the parameters units after its values.

For more details see the supplementary file "Userguide". For reference, we also include an example of input file in the appendix Appendix B.

3.3. Output files:

The software produces two outputs: an log of the program execution, and a series of files containing the spatial distribution of the order parameters. The log has the function of informing the user about the parameters read by the program, so it can be used as reference in the future, and informing the current state of the simulation. The log is printed in the standard output, which can aid the preparation of the input file, but we strong recommend redirecting it to a separate file for reference in future.

The main output of the software are the files containing the spatial distribution of the order parameter: the main LC director director $\vec{n}$, the co-director $\vec{l}$, the uniaxial order parameter S and biaxial order parameter P. We decided to output the order parameters in this form instead of the elements of Q_{ij}, since the former are more ready to use and interpret than the later. We also preferred to refer to the position in space using the lattice numbers instead of the space position in the Cartesian frame. The actual position can be easily reprieved multiplying the column by its referred grid spacing (dx, dy or dz). Although every output file has associate with it an time t, this number is not output in the file. Instead the output file number and is referred output time is informed in the log output.

An example of a truncated output file can be viewed in appendix Appendix C. More information can be found the in the supplementary material"Userguide".

4. Test problems:

To validate our code we performed a few standard simulations.

4.1. Bulk Nematic:

One of the

4.2. Cholesteric Slab:

It is know that a cholesteric liquid crystal with pitch p_0 placed inside slab with planar anchoring in both substrates will organize itself with the profile

4.3. Nematic sphere with strong anchoring

4.4. Cholesteric sphere with weak anchoring

5. Software implementation and extension:

6. Conclusions

Appendix A. List of available parameters and its units:

Parameter name	variable type	units	mandatory/standard value
Geometry	string		Yes
Nx	integer		Yes
Ny	integer		Yes
Nz	integer		Yes
dx	real	nm	Yes
dy	real	nm	Yes
dz	real	nm	Yes
integrator	string		Yes
facmin	real		0.4
facmax	real		3
prefac	real		0.8
Atol	real		0.001
Rtol	real		0.001
a	real	$\mathrm{MJ}/(m^2K)$	Yes
b	real	MJ/m^2	Yes
c	real	MJ/m^2	Yes
K1	real	pN	See 3
K2	real	pN	See 3
K3	real	pN	See 3
L1	real	pN	See 3
L2	real	pN	No
L3	real	pN	No
Ls	real	pN	No
Lq	real	pN/m	No
p0 or q0	real	nm or 1/nm	No
T	real	K	No
Mu or gamma	real	Pa/s	Yes
Mu_s or gamma_s	real	nm Pa m/s	See 3
ti	real	μs	0.0
tf	real	μs	Yes
dt	real	μs	Tf/1e6
timeprint	real	μs	Tf/20
timeprint_type	string		Linear
timeprint_increase_factor	real		Tf/20
output_folder	string		.
output_fname	string		director_field_$$.csv
initial_output_file_number	int		0
initial_conditions	string		yes
initial_file_name	string		See 3
theta_i	real	degrees	See 3
phi_i	real	degrees	See 3
anchoring_type	int + string		See 3
Wo1	int + real		See 3
theta_0	int + real		See 3
phi_0	int + real		See 3

Appendix B. Input File example:

Here you can find the complete input file:

```
#Geometry Parameters:
geometry slab
Nx 200 /* grid size */
Ny 200 /* grid size */
Nz 100 /* grid size */
dx 10.0 /* 10^-9 m */
dy 10.0 /* 10^-9 m */
dz 10.0 /* 10^-9 m */


#Integrator parameters:
integrator DP5
atol 0.005
rtol 0.005
facmax 3.0
facmin 0.4
prefac 0.8


#Liquid crystal parameters:
a 0.182
b -2.12
c 1.73
T -1 Kelvin
k11 16.7 /* pN */
k22 7.8 /* pN */
k33 18.1 /* pN */
k24 0 /* pN */
p0 500
mu_1 0.3 /* Pa s */
mu_1_s 30.0 /* Pa nm s */


#Time parameters:
dt 0.001 /* 10^-6 s */
ti 0.0 /* 10^-6 s */
tf 5000.0 /* 10^-6 s */

#Output Parameters:
time_print_type logarithmic
timeprint 50. /* 10^-6 s */
timeprint_increase_factor 1.16
output_folder .
output_fname output_$$.csv
initial_output_file_number 0

#Initial conditions:
initial_conditions random


#Boundaries conditions

#Bottom boundaries:
anchoring_type 0 Rapini-Papoular
Wo1 0 1000.0
theta_0 0 45.0
```

```
phi_0            0   45.0

#Top boundaries:
anchoring_type 1   Fournier-Galatola
Wo1              1   1000.0
```

Appendix C. Output file:

[1] D. Seč, T. Porenta, M. Ravnik, S. Žumer, Geometrical frustration of chiral ordering in cholesteric droplets, Soft Matter 8 (48) (2012) 11982–11988. doi:10.1039/c2sm27048j.