Non-monotonic, lily-like twist distribution in toroidal nematics

Accompanying ESI

Andrea Pedrini[1] andrea.pedrini@unipv.it
Marco Piastra[2] marco.piastra@unipv.it
Epiganio G. Virga[1] eg.virga@unipv.it

[1]Dipartimento di Matematica, Università di Pavia, via Ferrata 5, 27100 Pavia, Italy
[2]Dipartimento di Ingegneria Industriale e dell'Informazione, Università di Pavia, via Ferrata 5, 27100 Pavia, Italy

This code is companion to the article

"Non-monotonic, lily-like twist distribution in toroidal nematics"

published in *Soft Matter*.

Purpose

The main purpose of this code is to find a numeric approximation to the global minimizer of Frank's elastic free-energy functional, in the non-trivial geometric setting of toroidal nematics.

For any further detail about the mathematical methods and for a discussion of the results, we refer the reader to the associated article.

Prerequisites

Use `python 3.x`.
See also `requirements.txt`.

Files

The following files are included in the online repository:

- `README.md` – this readme
- `run.py` – the main file
- `lib` – folder containing:
  - `config.py` – parameters for advanced configuration
  - `functions.py` – source code for functions
  - `plot.py` – source code for plots
  - `varia.py` – source code for utilities
- `requirements.txt`

Test

To test that everything is working properly, execute:

```
python3 run.py
```

A default experiment will start and a plot window will open.
The plot window contains three plots, which are updated first every 50 steps, and then every 500 steps – the step counter is displayed on the left side of the window, below the first plot.
The run is complete when the step counter disappears.

Three new folders are created during the run:

1. results – contains the results of the experiment:
   - a `.svg` file for the final plot
   - a `.csv` file for the minimizer (1001 values)
2. logs – contains the log file of the experiment
3. graphs – contains the graph of the experiment that can be read by TensorBoard (see TensorFlow documentation)

Experiments

File `run.py` can be run with different parameters:

- eta – geometric parameter: ratio between the radii of the torus (default=0.5)
- k3 – elastic constant: ratio of the bend to twist elastic constants (default=8.5)
- k24 – elastic constant: ratio of the saddle-splay to twist elastic constants (default=0.1)
- steps – minimal number of iterations performed (default=6000)
- plot – if True: the experiment results are displayed in a plot window (default=True)
- save – if True: the experiment results and plots are saved (as `.csv` and `.svg` files, in the results folder) (default=True)

Example: the code

```
python3 run.py --eta 0.5 --k3 7 --k24 0.8 --steps 7000 --plot True --save True
```

starts an experiment where eta=0.5, k3=7, k24=0.8, the minimal number of iterations performed is 7000, the plot is displayed and both the minimizer and the final plot are saved.

Plot window

We give here a brief description of what is shown in the plot window.

The title contains the values of the geometric and elastic constants eta, k3, and k24 for the experiment. While the optimization is running, the title starts with the tag 'OPTIMIZATION PHASE', the tag disappears when the optimization is complete.

The first plot (top left) displays the minimizer a_{min} and the test functions a_{lin} and a_{qua} (for k24$\leq$1) or a_{cyl} (for k24>1):

- a_{min} changes color, from purple to blue when the optimization is complete;
- a_{lin} is the linear test function used in Koning et al. (Ref. 6 in our article);
- a_{qua} is the rational test function used in Pedrini and Virga (Ref. 14);
- a_{cyl} is the exact solution found in a cylinder by Burylov (Ref. 15).

The second plot (right) displays the angle $\alpha_{\min}$ computed according to $a_{\min}$ [see equation (12) in our article].

The third plot (bottom-middle) displays the *rim* (see the article for more details).

On the bottom-left corner the step counter and the value of the energy are displayed. The step counter disappears when the computation is complete. The energy of the reference director configuration (for which the integral lines of the director field are the torus' parallels) is set equal to 0.

Examples

The following examples relate to more specific cases described in the article.

Example 1

```
python3 run.py --eta 0.5 --k3 8.5 --k24 0.1 --steps 6000
```

Here eta=0.5, k3=8.5, k24=0.1. This is the example presented in Figs. 5 and 6 of the article.
The result of this experiment is a non-zero function: for this particular choice of the parameters, the reference director configuration is not stable. In fact, both configurations with $\alpha = \alpha_{\min}$ and $\alpha = -\alpha_{\min}$ have less energy (equal for the two configurations).

Example 2

```
python3 run.py --eta 0.5 --k3 2 --k24 0.5 --steps 6000
```

Here eta=0.5, k3=2, k24=0.5.
The result of this experiment is the null function – up to a negligible error due to numerical approximation: for this particular choice of the parameters, the reference director configuration is stable.
The energy given by the minimizer is essentially 0 and the second plot ($\alpha_{\min}$) is the null function. The third plot here is not relevant.

Example 3

```
python3 run.py --eta 0.01 --k3 1 --k24 2 --steps 10000
```

Here eta=0.01, k3=1, k24=2. This is the example presented in Fig. 11 of the article.
The result of this experiment is a validation of the method. When eta is small, the torus can be approximated with a cylinder and we can compare $a_{\min}$ with the analytical solution a_{cyl} given already by Burylov in Ref. 15 [see, in particular, his formula (51)] and recently re-obtained by Davidson et al. in Ref. 16 [see equation (18) of our article]: as shown by the first plot, they are practically indistinguishable.

Example 4

```
python3 run.py --eta 0.01 --k3 8 --k24 4 --steps 12000
```

Here eta=0.01, k3=8, k24=4. This is the example presented in Fig. 12 of the article.
The result of this experiment provides a further validation to the method, again in the cylinder limit. Note that, for this particular choice of the elastic constants, the graph of a_{cyl} is more elaborate (with also a change in its concavity) than the one in Example 3.

Advanced configuration

File `config.py` collects a number of parameters that can be used for advanced configuration of the program.
Here is a brief description of the most relevant ones.

a_0

By default, the minimizer $a_{\min}$ is forced to be 0 at the origin (see the article for more details). This constraint is obtained by setting `a_0 = True`.
Setting `a_0 = False` makes the minimizer free at 0.

s_step

`s_step` is the sampling step for the values of sigma in the interval $[0, 1]$. Increasing this value makes the computation faster but less precise.

depth and size, epsilon

`depth` is the number of layers of the approximator (i.e. the deep neural network).

`size` is the dimension of vectors and square matrices in which the scalar parameters of the approximator are arranged [i.e. the trainable parameters of the approximators – see equation (20) in the article].

`epsilon` determines the range for the random initialization of vectors and square matrices (i.e. at the beginning of the training process).

update_step

When `update_step` is not 0, the optimization performs the minimal number of iteration given in `steps`, and then continues until the minimizer $a_{\min}$ is not updated for `update_step` iterations (early stopping).

learning_rate

`learning_rate` is the speed of the optimization process. In general, a higher value produces better results in the first steps (i.e. it approaches the minimal energy more quickly), but reaching the optimal shape requires more steps afterwards.

optimizer_method

`optimizer_method` is the method used to perform the gradient descent (GD) optimization. In this code, three different methods are implemented: plain GD, Adam and Adadelta (default: Adam).

checkpoints

If `keep_checkpoints` is True, checkpoints are saved and kept in the corresponding folder. This means that all parameters for the approximator are saved, so that they can be reused in further computations.

plot utilities

All parameters in this section relate to visualization details. In particular, it is possible to speed up the whole optimization process by turning off the interactive plots or choosing which functions will be actually plotted.