ACCESS allocation proposal

Lucas Myers

June 1, 2023

1 Research objectives

Nematic liquid crystal (LC) systems are microscopically composed of rod-like molecules whose interactions depend on their relative angle. At low temperatures or high densities (in the case of thermotropic or lyotropic LCs respectively) the molecules tend to align along some preferred axis. Additionally, molecules are able to flow past one another, thereby behaving as an intermediate between an isotropic liquid and a crystalline solid.

One method to describe such systems in equilibrium is to introduce a tensorial order parameter Q which depends on a probability distribution function of molecular orientations. Q is traceless and symmetric with eigenvectors indicating preferred axes along which molecules align, and eigenvalues indicate propensity to align along one axis relative to the others.

For non-equilibrium systems, one takes $Q(\mathbf{r})$ to be a function of space which is in local equilibrium at each point. To evolve this field in time, one must introduce a free energy density to be minimized. Our model combines the Ball-Majumdar free energy with a Qian-Sheng hydrodynamic theory to model the flow of the materials. The result is a pair of coupled partial-differential equations describing the time evolution of the Q-tensor and the instantaneous velocity corresponding to some Q-tensor configuration.

We solve these equations numerically with a finite-element solver based on the deal.II C++ library. The code is available online at the *lucasmyers97/maier-saupe-lc-hydrodynamics* repository on Github. To evolve the Q-tensor we use a Crank-Nicolson time-stepping scheme, while at each time-step solving a generalized Stoke's equation for the instantaneous velocity configuration. Because the Q-tensor equation is nonlinear, we must use a Newton-Rhapson scheme to iteratively solve for each timestep. To solve the linear system at each Newton-Rhapson step we precondition with the Trilinos ML Algebraic Multigrid preconditioner and solve with a GMRES method. To solve for velocity we use an Algebraic Multigrid block-preconditioner method proposed in step-55 of the deal.II tutorial library. The code is based on MPI and Trilinos, and is thus massively scalable.

With this software we will simulate three-dimensional nematic configurations which contain so-called disclination lines that annihilate. Typically for the bulk of a nematic system, the Q-tensor field will have some largest eigenvalue which is largely constant and corresponds to an eigenvector which varies smoothly in space. However there are regions (points or lines) for which the largest eigenvector is pointed in different directions depending on the direction of approach. Such regions are called "disclinations" or "topological defects", and their motion in different media is of interest. In particular, LC systems resist different types of distortion from uniform configurations in different ways. It has been observed experimentally that, somewhat suprisingly, disclinations in systems which prefer a particular type of distortion over one another attract and annihilate one another in a manner identical to those in systems which have no such preference. We will observe the mesoscopic details of the Q-tensor close to the annihilation sites to provide an explanation for this phenomenon. Additionally, we will investigate the effect that hydrodynamics have on altering a system's propensity for different modes of distortion.

2 Description of resource needs

Preliminary benchmarking of the time-evolution of Q with flow velocity fixed at zero indicates good scalability. In Fig. 1 we have run a three-dimensional configuration of two disclination lines for 10 timesteps at $\delta t = 0.1$.

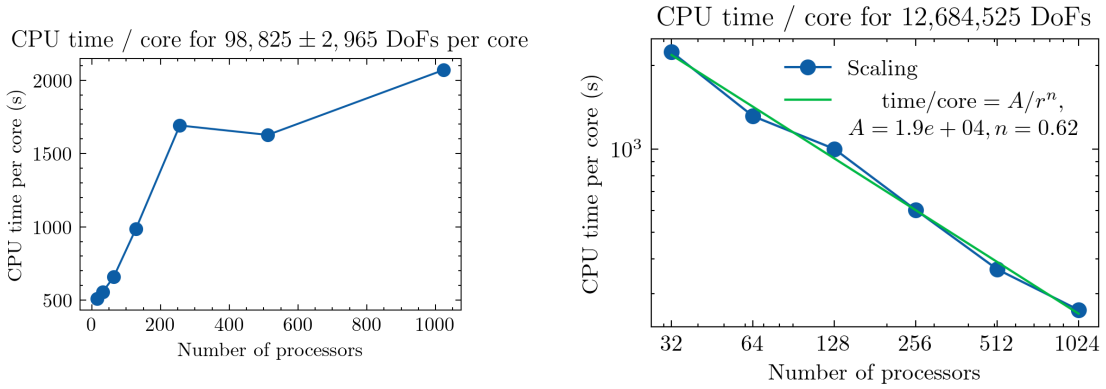

Figure 1: Strong and weak scaling for 3D Q-tensor system given particular numbers of degrees of freedom. Strong scaling plotted on a log scale.

Indeed, despite initial jumps in the range of 128-256 cores – likely due to an increase in MPI communications due to multiple nodes – this indicates that the code is able to scale to very large systems.

To simulate disclination lines annihilating, we would like to use a $256 \times 256 \times 256$ mesh in order to resolve disclination cores. For a grid of this size, a timestep of $\delta t = 0.1$ in nondimensional units is reasonable to bound numerical error, and a total simulation time of $t_{\text{total}} = 1,000$ is necessary to observe annihilation. Given the benchmarking data, it takes $\approx 2,000$ seconds per core to run $\approx 100,000$ degrees of freedom per corefor 10 timesteps. This works out to $\approx 0.002 \frac{\text{seconds}}{\text{core} \cdot \text{DoF} \cdot \text{timestep}}$. Then for the simulation size given above with 5 degrees of freedom per vertex, we get about $100,000$ core hours per run. Because we would like to look at several configurations and several differerent elastic constant values, we would like the capacity to run at least 15 different simulations. This gives a total time request of 1.5 million CPU hours.

For storage, each fixed configuration will take up $\sim$ 42Gb given double-precision floating point degree-of-freedom values. We would like to store 100 configurations per simulation, which gives $\sim$ 70Tb of storage total.

Given our current allocation on Expanse at the San Diego Supercomputing Institute, all of the dependencies are already installed. These include the deal.II library, which is in turn built on top of the p4est and Trilinos libraries. All of these are somewhat involved to install, so it would be a simpler transition to continue with this allocation.