XSEDE allocation proposal

Lucas Myers

March 8, 2022

1 Order parameter model

Nematic liquid crystal systems translationally act like a fluid with molecules that are able to flow around one another. However, the rod-like structure of their molecules causes alignment along some preferred axis at low temperatures or high densities, depending on the specific system. Unlike polar systems, the nematic molecules have no preference to pointing parallel, or antiparallel along the axis [1].

For equilibrium systems we use a symmetric, tracelss tensor to represent the alignment direction and degree of order. Given a probability distribution function $p : S^2 \to [0, 1]$ which represents the probability of finding a molecule oriented in some direction, we may write down the order parameter tensor as:

$$Q_{ij} = \int_{S^2} \left[\xi_i \xi_j p(\xi) - \tfrac{1}{3}\delta_{ij}\right] d\xi \tag{1}$$

Note that $p(\xi) = p(-\xi)$ given the constraint that molecules are indifferent to aligning vs. antialigning.

Supposing an orientation-dependent molecular energy, one can calculate to lowest order using a mean-field approximation, that the average energy over the ensemble is given by:

$$\langle E \rangle = -\alpha Q_{ij} Q_{ji} \tag{2}$$

for some interaction parameter $\alpha > 0$. Given this, a free energy may be constructed in the usual way:

$$F = \langle E \rangle - TS \tag{3}$$

with T the temperature and S the system entropy given in terms of the molecular probability distribution function:

$$S = -nk_B \int_{S^2} p(\xi) \log\left[4\pi p(\xi)\right] d\xi \tag{4}$$

By introducing a Lagrange Multiplier tensor, one may find the probability distribution which maximizes entropy for a fixed Q-tensor. The value of the Lagrange Multiplier is then given implicitly by:

$$Q_{ij} = \frac{\partial \log Z}{\partial \Lambda_{ij}} - \frac{1}{3}\delta_{ij} \tag{5}$$

To find the equilibrium state, one must then find the Q-tensor which minimizes the free energy for a particular temperature [2].

For non-equilibrium states, we take a mesoscopic approach wherein each point in our domain represents some quasi-isolated system in equilibrium, thereby being assigned a Q-value. For this, each of the quantities above become densities which are functions of space. As a result, the

Q-tensor field allows for spacial variations which must be penalized by an elastic free energy density, given by:

$$f_{el} = L_1(\partial_k Q_{ij})(\partial_k Q_{ij}) + L_2(\partial_j Q_{ij})(\partial_k Q_{ik}) + L_3 Q_{kl}(\partial_k Q_{ij})(\partial_l Q_{ij}) \tag{6}$$

For a purely thermodynamic model, one may find the time evolution of the given system by taking the variation of the total free energy density (bulk and elastic). To include hydrodynamics as we do, one may derive various hydrodynamic stresses by considering the effect of virtual distortions of the nematic field. Further, one my come up with constraints by considering the rotational symmetry of the system, as well as local conservation laws. There are several ways to do this, but we choose the model presented by Qian and Sheng [3]:

$$\begin{aligned} h_{ij} + h'_{ij} - \lambda\delta_{ij} - \epsilon_{ijk}\lambda_k &= 0 \\ \partial_j\left(-p\delta_{ji} + \sigma^d_{ji} + \sigma'_{ji}\right) &= 0 \end{aligned} \tag{7}$$

The first equation corresponds to a generalized force-balance law for which h_{ij} is the force resulting from the free energy, and h'_{ij} is the viscous force from the fluid:

$$\begin{aligned} h_{ij} &= -\frac{\delta f}{\delta Q_{ij}} \\ h'_{ij} &= -\frac{1}{2}\mu_2 A_{ij} - \mu_1 N_{ij} \end{aligned} \tag{8}$$

where μ_1 and μ_2 are viscosities, $A_{ij} = \frac{1}{2}(\partial_i v_j + \partial_j v_i)$ is the symmetric gradient of the flow field, and $N_{ij} = \frac{dQ_{ij}}{dt} + (W_{ik}Q_{kj} - Q_{ik}W_{kj})$. Here $d/dt = v_i\partial_i + \partial/\partial t$ is the convective derivative, and $W_{ij} = \frac{1}{2}(\partial_i v_j - \partial_j v_i)$ is the antisymmetric gradient of the flow field. The Lagrange multipliers λ and λ_k are to make sure that the force equation remains traceless and symmetric.

For the flow equation, we have the following definitions:

$$\sigma^d_{ij} = -\frac{\partial f}{\partial(\partial_j Q_{kl})}\partial_i Q_{kl} \tag{9}$$

for the distortion stress, and:

$$\begin{aligned} \sigma'_{ij} = \beta_1 Q_{ij}Q_{kl}A_{kl} + \beta_4 A_{ij} + \beta_5 Q_{ik}A_{kj} + \beta_6 A_{ik}Q_{kj} \\ + \frac{1}{2}\mu_2 N_{ij} - \mu_1 Q_{ik}N_{kj} + \mu_1 Q_{jk}N_{ki} \end{aligned} \tag{10}$$

for the viscous stress. Here, each β and μ is a viscosity. For a simplified problem, we only consider terms in the stress tensors which are linear in v and Q_{ij}, which results in a Stoke's equation governing the hydrodynamics.

2 Algorithm

To numerically solve these PDEs, we employ a finite element method. This was chosen partially due to the increased flexibility of the method (e.g. in the size and shape of the domain), as well as the abundance of mature libraries which efficiently implement various algorithmic aspects. For the simplest possible test case, we neglected the hydrodynamic equations and only consider isotropic elasticity ($L_2 = L_3 = 0$) for the thermodynamic relaxation. Further, we neglected time dependence, instead opting to minimize the free energy by finding a zero of its variation. The resulting nondimensional equation is given by:

$$\alpha Q_i + \nabla^2 Q_i - \Lambda_i = 0 \tag{11}$$

where Q_i is a vector consisting of the 5 degrees of freedom of Q, and Λ_i is likewise for Λ. Because the Lagrange multiplier is a nonlinear function of the Q-tensor, we must solve this problem iteratively using the Newton Rhapson method. The resulting iterative scheme is as follows:

$$\begin{aligned} F'(Q^n)\delta Q^n &= -F(Q^n) \\ Q^{n+1} &= Q^n + \delta Q^n \end{aligned} \tag{12}$$

where $F(Q^n)$ is the left-hand side of (11), and the Jacobian $F'(Q^n)$ is given by:

$$F'(Q^n)\delta Q^n = \alpha \delta Q^n + \nabla^2 \delta Q^n - \left(\frac{\partial Q}{\partial \Lambda}\right)^{-1} \delta Q^n \tag{13}$$

Given this, we may discretize in the usual way by taking an inner product with an arbitrary test function, and then integrating by parts to minimize the order of derivatives in our equations. Taking our test function basis to be the piecewise linear Lagrange elements on an arbitrary grid, and taking our solution variable to be a linear combination of those same elements, we recover a matrix equation for each iteration. Note that we must also numerically solve for Λ and $\partial Q/\partial \Lambda$ using Newton's method, given that it is only defined implicitly.

In both 2D and 3D we solve this equation iteratively using the GMRES method. In 2D we use an algebraic multigrid preconditioner in order to limit the number of iterations needed by the GMRES method. In 3D we find the multigrid method has too much memory and operational overhead, likely caused by the extra coupling between the finite element degrees of freedom. We hope that this overhead can be mitigated either by tuning the algorithmic parameters (e.g. coarsening/interpolation technique, or amount of truncation), or by switching to the comparatively simpler geometric multigrid technique. The latter has the added advantage of being compatible with matrix-free methods, which cut the memory cost of the solver significantly and decrease the number of memory access operations. In any case, the entire algorithm is able to be parallelized across more than 1,000 processors, both in terms of memory and CPU usage.

3 Code

To implement this algorithm, we have relied heavily on the deal.II finite element library written in C++ [4]. For distributing the domain mesh across processors, deal.II interfaces with the p4est quad- and oct-tree library [5]. To numerically invert the Lagrange multiplier terms which are present in the residual $F(Q^n)$ and Jacobian $F'(Q^n)$ we must approximate several integrals over S^2. We choose a Lebedev quadrature scheme, which is implemented in C++ by Burkhardt [6]. Finally, the GMRES operator is implemented in PETSc and the algebraic multigrid preconditioner is implemented in the BoomerAMG class from Hypre [7–11]. All of these operations use MPI to distribute work and memory across several nodes, so the scheme is extremely parallelizable. We have made our code publicly available on Github in the *maier-saupe-lc-hydrodynamics* repository, which includes both the distributed simulation described above, as well as a serial time-dependent simulation and several examples demonstrating various functionalities [12].

4 Scaling

In 2D and 3D we have run the program described above on evenly partitioned square and cubic lattices respectively. The initial conditions are that of a uniaxial, constant-order topological defect with charge $+1/2$. This gives Q as a function of polar coordinates as:

$$Q = \frac{S}{2}\begin{bmatrix} \frac{1}{3} + \cos\phi & \sin\phi & 0 & \\ \sin\phi & \frac{1}{3} - \cos\phi & 0 & \\ & 0 & 0 & -\frac{1}{3} \end{bmatrix} \tag{14}$$

where here we set the scalar order parameter as $S = 0.6751$ and ϕ is the angular coordinate. We impose Dirichlet boundary conditions by fixing the boundary at the Q-values given above.

The two most computationally expensive steps are the finite element matrix assembly, and the linear solver. The former involves populating a matrix with inner products of Lagrange elements so that each operation only needs to consider a domain cell and its immediate neighbors. Hence, the number of operations scales linearly with the number of degrees of freedom. For the latter step, each iteration of the GMRES technique involves only vector-vector or matrix-vector products. Given a sparse matrix like ours, these operations scale linearly with the number of degrees of freedom. The algebraic multigrid method is also shown (under certain conditions) to scale linearly with the number of degrees of freedom and is further shown to keep the number of GMRES iterations constant [**?**]. Hence, it is reasonable to expect overall program scaling to be linear.

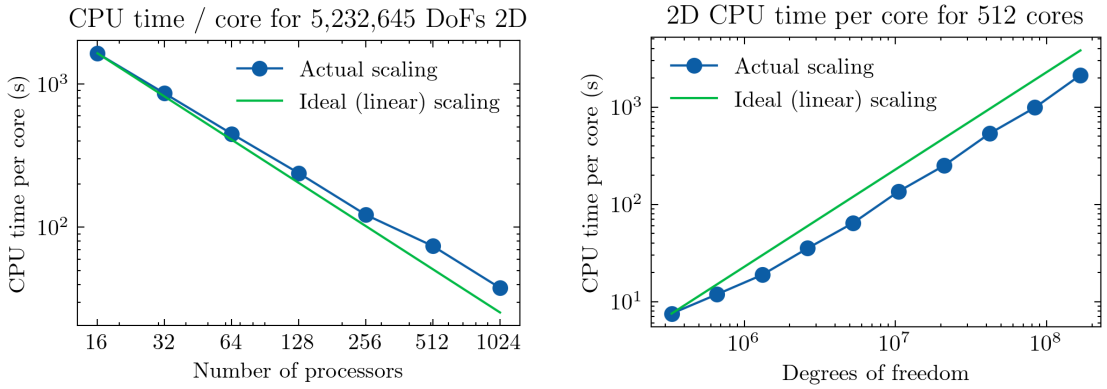

Figure 1: Strong and weak scaling for 2D system using AMG preconditioner, plotted on a logarithm scale

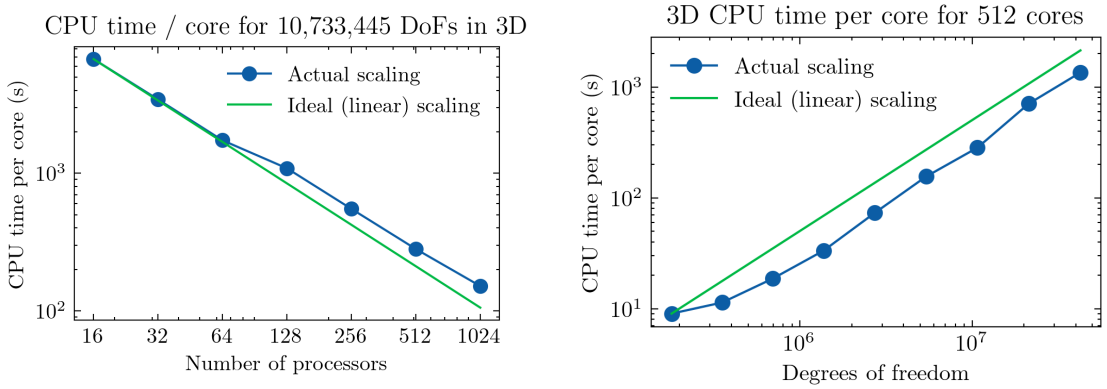

Figure 2: Strong and weak scaling for 3D system with no preconditioner, plotted on a logarithm scale

As seen from figures 1 and 2, the strong and the weak scaling follow the linear trend closely. In the regime of low number of degrees of freedom per processor, we note some slight jumps, likely due to the fact that there is some fixed overhead associated with setting up the program which contributes less when each processor has enough work allocated. This scaling is comparable to that demonstrated by the deal.II developers using the algebraic multigrid technique and an iterative solver, though their simulations are somewhat faster due to the simpler computations required to build the finite element matrices, and fewer couplings between degrees of freedom [**?**].

As mentioned above, there are plans to reduce the overall run time while maintaining the scaling by tuning the algebraic multigrid preconditioner. Other deal.II users have informally reported

significant ($\sim 4\times$) speedups of the solvers by careful tuning. Additionally, deal.II has recently implemented a geometric multigrid preconditioner which is able to utilize matrix-free methods, which can cut down on the number of memory requests thereby granting speed-ups. Finally, we are currently developing a method to optimize the Lagrange multiplier calculation which will drastically cut down the number of Lebedev quadrature operations that are necessary which we hope will speed up matrix assembly by an order of magnitude or so. Hence, there is still room for optimizing this method.

References

[1] J. V. Selinger, *Introduction to the Theory of Soft Matter: From Ideal Gases to Liquid Crystals*, ser. Soft and Biological Matter. Cham: Springer International Publishing, 2016. [Online]. Available: http://link.springer.com/10.1007/978-3-319-21054-4

[2] C. D. Schimming and J. Viñals, "Computational molecular field theory for nematic liquid crystals," *Physical Review E*, vol. 101, no. 3, p. 032702, Mar. 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.101.032702

[3] T. Qian and P. Sheng, "Generalized hydrodynamic equations for nematic liquid crystals," *Physical Review E*, vol. 58, no. 6, pp. 7475–7485, Dec. 1998. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.58.7475

[4] D. Arndt, W. Bangerth, B. Blais, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, U. Köcher, M. Kronbichler, M. Maier, P. Munch, J.-P. Pelteret, S. Proell, K. Simon, B. Turcksin, D. Wells, and J. Zhang, "The deal.II library, Version 9.3," *Journal of Numerical Mathematics*, vol. 29, no. 3, pp. 171–186, Sep. 2021. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/jnma-2021-0081/html

[5] C. Burstedde, L. C. Wilcox, and O. Ghattas, "p4est : Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees," *SIAM Journal on Scientific Computing*, vol. 33, no. 3, pp. 1103–1133, Jan. 2011. [Online]. Available: http://epubs.siam.org/doi/10.1137/100791634

[6] "Quadrature rules for the unit sphere," https://people.math.sc.edu/Burkardt/cpp_src/sphere_lebedev_rule/sphere_lebedev_rule.html, accessed: 2022-03-08.

[7] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang, "PETSc Web page," 2021. [Online]. Available: https://petsc.org/

[8] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang, "PETSc/TAO Users Manual," Argonne National Laboratory, Tech. Rep. ANL-21/39 - Revision 3.16, 2021.

[9] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries," in *Modern Software Tools in Scientific Computing*, E. Arge, A. M. Bruaset, and H. P. Langtangen, Eds. Birkhäuser Press, 1997, pp. 163–202.

[10] R. D. Falgout, J. E. Jones, and U. M. Yang, "The Design and Implementation of hypre, a Library of Parallel High Performance Preconditioners," *Lecture Notes in Computational Science and Engineering*, vol. 51, Jul. 2004. [Online]. Available: https://www.osti.gov/biblio/875356

[11] V. E. Henson and U. M. Yang, "BoomerAMG: a Parallel Algebraic Multigrid Solver and Preconditioner," *Applied Numerical Mathematics*, vol. 41, no. 5, pp. 155–177, 2002.

[12] L. Myers, "maier-saupe-lc-hydrodynamics," 3 2022. [Online]. Available: https://github.com/lucasmyers97/maier-saupe-lc-hydrodynamics