Computational and Applied Mathematics, I
C. Brezinski and U. Kulish (Editors)
Elsevier Science Publishers B.V. (North-Holland)

Finite Difference Minimization of the Landau - de Gennes Free Energy for Liquid Crystals in Rectangular Regions

Paul A. Farrell , Arden Ruttan and Reinhardt R. Zeller

Department of Mathematics & Computer Science, Kent State University, Kent, OH 44242, U.S.A.

Abstract

We will describe a finite difference code for computing equilibrium configurations, of the order-parameter tensor field for nematic liquid crystals, in rectangular regions, by minimization of the Landau - de Gennes Free Energy functional. The implementation of the free energy functional described here includes electric and magnetic fields, quadratic gradient terms, and scalar bulk terms through fourth order. Boundary conditions include the effects of strong surface anchoring. The target architectures for our implementation are SIMD machines, with interconnection networks which can be configured as 2 or 3 dimensional grids, such as the Wavetracer DTC. We also discuss the relative efficiency of a number of iterative methods for the solution of the linear systems arising from this discretization on such architectures.

1. INTRODUCTION: THE PROBLEM

We consider a finite difference approximation of the equilibrium configuration of liquid crystals in a slab,

$$\Omega = \{(x, y, z) : 0 \leq x \leq a, 0 \leq y \leq b, 0 \leq z \leq c\}.$$

Following Gartland [4], we will use the Landau-de Gennes formulation, and seek a tensor order parameter field Q that minimizes the free energy of the system. In this case, the free energy can be expressed as

$$F(Q) = F_{\text{vol}}(Q) + F_{\text{surf}}(Q) = \int_{\Omega} f_{\text{vol}}(Q) + \int_{\partial\Omega} f_{\text{surf}}(Q)\,,$$

where Q(x) is a 3×3 symmetric, traceless tensor and where Ω and $\partial\Omega$ represent the interior and surface of the slab respectively. In this implementation we limit ourselves to strong anchoring on the surface of Ω. This corresponds to Dirichlet boundary conditions on $q_i, i = 1, \ldots, 5$, and in such cases

$$F_{\text{surf}}(Q) = \int_{\partial\Omega} f_{\text{surf}}(Q) = 0.$$

This work was supported in part by the Advanced Liquid Crystalline Optical Materials (ALCOM) Science and Technology Center at Kent State University under DMR89-20147.

The term $F_{\text{vol}}(Q)$ represents an approximation of the interior free energy. To describe $F_{\text{vol}}(Q)$, we will use the convention that summation over repeated indices is implied and that indices separated by commas represent partial derivatives. With these conventions $F_{\text{vol}}(Q)$ has the following form, (see, for instance [9]):

$$\begin{aligned} f_{\text{vol}}(Q) \;:=\; & \frac{1}{2}L_1 Q_{\alpha\beta,\gamma}Q_{\alpha\beta,\gamma} + \frac{1}{2}L_2 Q_{\alpha\beta,\beta}Q_{\alpha\gamma,\gamma} + \frac{1}{2}L_3 Q_{\alpha\beta,\gamma}Q_{\alpha\gamma,\beta} + \frac{1}{2}A\,\text{trace}(Q^2) \\ & -\frac{1}{3}B\,\text{trace}(Q^3) + \frac{1}{4}C\,\text{trace}(Q^2)^2 + \frac{1}{5}D\,\text{trace}(Q^2)\text{trace}(Q^3) \qquad (1) \\ & +\frac{1}{6}M\,\text{trace}(Q^2)^3 + \frac{1}{6}M'\,\text{trace}(Q^3)^2 - \Delta\chi_{\max}H_\alpha Q_{\alpha\beta}H_\beta - \Delta\epsilon_{\max}E_\alpha Q_{\alpha\beta}E_\beta\,. \end{aligned}$$

where L_1, L_2, and L_3 are elastic constants, A, B, C, D, M, and M' are bulk constants, and H, $\Delta\chi_{\max}$, E, and $\Delta\epsilon_{\max}$ are the field terms and constants associated with the magnetic and electrical fields respectively.

For $P \in \Omega$, the tensor $Q(P)$ will be represented in the form,

$$\begin{aligned} Q(P) &= (Q_{\alpha\beta})_{\alpha,\beta=1}^{3} \\ &= q_1(P)\phi_1 + q_2(P)\phi_2 + q_3(P)\phi_3 + q_4(P)\phi_4 + q_5(P)\phi_5 \\ &= q_1(P)\begin{pmatrix} \frac{\sqrt{3}-3}{6} & 0 & 0 \\ 0 & \frac{\sqrt{3}+3}{6} & 0 \\ 0 & 0 & \frac{-\sqrt{3}}{3} \end{pmatrix} + q_2(P)\begin{pmatrix} \frac{\sqrt{3}+3}{6} & 0 & 0 \\ 0 & \frac{\sqrt{3}-3}{6} & 0 \\ 0 & 0 & \frac{-\sqrt{3}}{3} \end{pmatrix} \\ &\quad + q_3(P)\begin{pmatrix} 0 & \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + q_4(P)\begin{pmatrix} 0 & 0 & \frac{\sqrt{2}}{2} \\ 0 & 0 & 0 \\ \frac{\sqrt{2}}{2} & 0 & 0 \end{pmatrix} + q_5(P)\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 \end{pmatrix}, \end{aligned}$$

similar to that in Gartland [5], where $\{q_\ell(P)\}_{\ell=1}^{5}$ are real-valued functions on Ω.

2. THE IMPLEMENTATION

To discretize our problem we begin by dividing the slab Ω into $I \times J \times K$ regions

$$v(i,j,k) = \{(x,y,z) : i\Delta x \le x \le (i+1)\Delta x, j\Delta y \le y \le (j+1)\Delta y, k\Delta z \le z \le (k+1)\Delta z\}$$

for $0 \le i \le I-1, 0 \le j \le J-1$, and $0 \le k \le K-1$, where $\Delta x = a/I, \Delta y = b/J, \Delta z = c/K$.

Using the points $P = (x_i, y_j, z_k)$, where $x_i = i\Delta x, y_j = j\Delta y, z_k = k\Delta z$, which are located in the lower left-hand corner of the $v(i,j,k)$, we represent the discrete interior free energy integral by

$$\int_\Omega f_{\text{vol}}(Q) \approx \sum_{i,j,k} f_{\text{vol}}(Q(x_i,y_j,z_k)) \times volume(v(i,j,k)). \qquad (2)$$

Note that (1) involves derivatives of the functions $\{q_\ell(P)\}_{\ell=1}^5$ with respect to the x, y, and z coordinates. For interior points P, these derivatives can be approximated using central difference approximations.

Thus, we have the following approximation of the **Landau-de Gennes free energy** which is second order accurate:

$$F(Q) \approx \sum_{i,j,k} f_{\text{vol}}(Q(x_i, y_j, z_k)) \times volume(v(i,j,k)) = \sum_{i,j,k} h(x_i, y_j, z_k). \tag{3}$$

With the discretization (3), we have reduced the problem to one of minimizing $\sum_{i,j,k} h(x_i, y_j, z_k)$ over all choices of $\{q_\ell(x_i, y_j, z_k)\}_{\ell=1}^5$. This unconstrained discrete minimization problem can be attacked in the standard way. That is, seek a solution of the non-linear system of equations

$$g(\hat{\ell}, \hat{i}, \hat{j}, \hat{k}) := \frac{\partial \sum_{i,j,k} h(x_i, y_j, z_k)}{\partial q_{\hat{\ell}}(x_{\hat{i}}, y_{\hat{j}}, z_{\hat{k}})} = 0, \tag{4}$$

for $0 \le \hat{i} \le I$, $0 \le \hat{j} \le J$, $0 \le \hat{k} \le K$, and $\hat{\ell} = 1 \ldots 5$. A standard approach, to solving non-linear systems such as these, is to use Newton's method (see [3]).

Each iteration of Newton's method involves solving a linear system, whose matrix is the Jacobian of (4), and then using that solution to update the iterate and the Jacobian, after which the process is repeated. For the three dimensional problem under consideration, this linear system is extremely large. A moderate sized problem would involve 10^7 unknowns. As our ultimate aim is to track the minimal energy state as the temperature varies, we need to be able to solve the minimal energy problem, as described above, rapidly and to moderate accuracy for each of a large number of successive temperatures. As such a task is probably only feasible on a massively parallel computer, we choose to implement our minimization problem on such a computer. In the next section, we will describe the type of computer architecture we use for our implementation, and sketch the details of that implementation.

3. THE ARCHITECTURE

Our target architecture is a massively parallel SIMD computer. This class of computers, is called Single Instruction Multiple Data stream (SIMD), since each of the processors execute the same instruction in lock-step on the data stored in its associated local memory. A facility is also provided to select a subset of processors, by masking the other processors. Only the selected processors execute the instruction, while the masked processors remain idle. This architecture is considerably simpler to implement and program than the alternative Multiple Instruction Multiple Data stream (MIMD) machines, in which each processor can execute a different instruction. The SIMD architecture is normally used for massively parallel machines, having between 4096 and 65536 processors, each with local memory. The processors are normally connected by a special purpose high-capacity communication network and controlled by a front-end processor. Early examples of this architecture included the Thinking Machines Corporation Connection Machine CM-1 and CM-2.

The platform chosen for this implementation was the Wavetracer Data Transport Computer (DTC), situated in the Department of Mathematics and Computer Science at Kent State. The

front-end sends instructions and data to a control unit, which decodes these instructions and broadcasts both instructions and data to the processor array. The array processors are semi-custom 1.5 micron standard cell chips. Each chip contains 32 one-bit processors together with 2 kilobits of fast static RAM for each processor, and associated control and memory error-detection circuitry. In addition, each processor has access to 8 or 32 kilobytes of private external dynamic memory depending on the configuration. Each circuit board consists of 128 chips. The minimal configuration, the DTC-4, has one circuit board and thus 4096 processors. Other configurations are the DTC-8, with 2 circuit boards and 8192 processors, and the DTC-16, with 4 circuit boards and 16384 processors.

The processors on each circuit board of the DTC-4 can be configured either as a $16 \times 16 \times 16$ cube, for three dimensional applications, or as a 64×64 square, for two dimensional applications. The DTC-8, can be configured as a $16 \times 32 \times 16$ cube or as a 64×128 square, and the DTC-16 as a $32 \times 32 \times 16$ cube or as a 128×128 square. The assumption here is that most applications, correspond to physical problems in 2 or 3 dimensions, and thus a 2 and 3 dimensional interconnection network is the most efficient for their solution. This is in contrast to the Connection Machine, in which the processors are connected by a hypercube network.

The traditional mode of solution of problems on a SIMD machine involves assigning one processor of the array per node in the problem space. To provide the ability to consider problems with more nodes than are available in the array, the DTC provides the ability to partition the memory of each processor to provide a larger number of *virtual processors*. There must be the same number of virtual processors for each physical processor. The number of virtual processors per physical processor is called the *virtual processor ratio*. The controller automatically issues instructions to the array once for each partition. Thus the execution time may be expected to increase linearly with the virtual processor ratio.

For the minimization problem we are considering, each discretization point, P, of the slab is associated with a virtual processor. Since the virtual processors are arranged in a rectangle or cube, similar to the actual processors, this provides an entirely natural mapping of the domain onto the cubic grid of the DTC, provided we use an equal number of grid points in each direction.

At each point of the slab the tensor order parameter Q is defined in terms of the 5 unknowns $\{q_\ell(P)\}_{\ell=1,5}$. In our implementation, each set of 5 unknowns $\{q_\ell(P)\}_{\ell=1,5}$ is stored in a single virtual processor. For each $\{q_\ell(P)\}_{\ell=1,5}$ there is also a corresponding row of the Jacobian matrix. The nonzero constants of that row are also stored in the memory of the processor associated with the discretization point P. Each non-zero constant, in a row of the Jacobian associated with P, also corresponds to another virtual processor (which in turn corresponds to a discretization point) with which the values of $\{q_\ell(P)\}_{\ell=1,5}$ at P must be communicated when the Jacobian matrix is updated. The set of processors with which a given processor, P, must communicate in order to update its row of the Jacobian is called the stencil of P. If the stencil of any processor is large, then the process of updating the Jacobian at each step of Newton's method will be expensive. Fortunately, the finite difference approximation described here yields a relatively small and compact stencil. In the problem discussed below, the stencil will consist of the all processors which are at most one step away from the given processor.

4. INFINITE SLAB CASE

In this paper we shall confine our consideration to the case of an infinite slab. Assuming the slab is infinite in the z-direction and imposing boundary conditions, which do not vary with z, effectively reduces the problem to a two dimensional problem on a rectangle:

$$\Omega := \{(x,y) : 0 \le x \le a,\ 0 \le y \le b\}.$$

To further simplify the problem, we consider a lower order approximation, the so called reduced model, where $L_2 = L_3 = D = M = M' = 0$ and there is no electric field E. Thus we have

$$\begin{aligned} f_{\text{vol}}(Q) &:= \frac{1}{2}L_1 Q_{\alpha\beta,\gamma}Q_{\alpha\beta,\gamma} + \frac{1}{2}A \text{ trace } (Q^2) - \frac{1}{3}B \text{ trace } (Q^3) \\ &\quad + \frac{1}{4}C \text{ trace } (Q^2)^2 - \Delta\chi_{\max} H_\alpha Q_{\alpha\beta} H_\beta. \end{aligned}$$

Using the previously defined representation for the tensor $Q(P)$ the above functional can be written in term of the q_i's:

$$\begin{aligned} f_{\text{vol}}(q_1,\dots,q_5) &:= \frac{1}{2}L_1(|\nabla q_1|^2 + \cdots + |\nabla q_5|^2) + \frac{1}{2}A(q_1^2 + \cdots + q_5^2) \\ &\quad - \frac{1}{3}Bd(q_1,\dots,q_5) + \frac{1}{4}C(q_1^2 + \cdots + q_5^2)^2 \\ &\quad - \Delta\chi_{\max} H_\alpha(q_1\phi_1 + q_2\phi_2 + q_3\phi_3 + q_4\phi_4 + q_5\phi_5)H_\beta, \end{aligned}$$

where d is a polynomial of degree three given by

$$\begin{aligned} d(q_1,\dots,q_5) &= \frac{\sqrt{3}}{6}(q_1^3 + q_2^3) + \frac{\sqrt{3}}{2}(q_1+q_2)(q_3^2 - q_1q_2) \\ &\quad + (-\frac{3+\sqrt{3}}{4}q_1 + \frac{3-\sqrt{3}}{4}q_2)q_4^2 + (\frac{3-\sqrt{3}}{4}q_1 - \frac{3+\sqrt{3}}{4}q_2)q_5^2 + \frac{3}{\sqrt{2}}q_3q_4q_5. \end{aligned}$$

The Euler-Lagrange equations for the reduced model form a system of five non-linear elliptic partial differential equations:

$$-L_1\nabla^2 q_i + Aq_i - BW_i(q_1,\dots,q_5) + C(q_1^2 + \cdots + q_5^2)q_i = \Delta\chi_{\max} H_\alpha[\phi_i]_{\alpha\beta}H_\beta,$$

where

$$\begin{aligned} W_1(q_1,\dots,q_5) &:= \frac{\sqrt{3}}{6}(q_1^2 - 2q_1q_2 - q_2^2 + q_3^2) - \frac{3+\sqrt{3}}{12}q_4^2 + \frac{3-\sqrt{3}}{12}q_5^2, \\ W_2(q_1,\dots,q_5) &:= \frac{\sqrt{3}}{6}(-q_1^2 - 2q_1q_2 + q_2^2 + q_3^2) + \frac{3-\sqrt{3}}{12}q_4^2 - \frac{3+\sqrt{3}}{12}q_5^2, \\ W_3(q_1,\dots,q_5) &:= \frac{\sqrt{3}}{3}(q_1+q_2)q_3 + \frac{1}{\sqrt{2}}q_4q_5, \\ W_4(q_1,\dots,q_5) &:= (-\frac{3+\sqrt{3}}{6}q_1 + \frac{3-\sqrt{3}}{6}q_2)q_4 + \frac{1}{\sqrt{2}}q_3q_5, \\ W_5(q_1,\dots,q_5) &:= (\frac{3-\sqrt{3}}{6}q_1 - \frac{3+\sqrt{3}}{6}q_2)q_5 + \frac{1}{\sqrt{2}}q_3q_4. \end{aligned}$$

Discretizing this system of equations produces the discrete Euler-Lagrange equations. Using standard central-difference approximations for the partial derivatives produces a five-point stencil at each nodal point in the domain. Since nearest-neighbor communications are efficient on the Wavetracer's mesh array of processors, the communication costs are minimal.

In order to solve the resulting non-linear system of equations we use Newton's method. Let $G : R^n \rightarrow R^n$ be a function representing the discrete Euler-Lagrange equations. There are a total of $5(I-1) \times (J-1)$ non-linear equations in this system. The function G depends on the $5n$ unknowns

$$G(\mathbf{x}) = G\left(q_1^1, \ldots, q_5^1, q_1^2, \ldots, q_5^2, \ldots, q_1^n, \ldots q_5^n\right),$$

where $n = (I-1) \times (J-1)$ is the number of nodal points. Let $G'(\mathbf{x})$ be the Jacobian of the system of equations. Newton's method is as follows:

1. Choose $\mathbf{x}_0$
2. For $k = 0, 1, 2, \ldots$
   (a) Solve $G'(\mathbf{x}_k)\mathbf{s}_k = -G\left(\mathbf{x}_k\right)$
   (b) Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$

The method requires solving a large sparse linear system $G'(\mathbf{x}_k)\mathbf{s}_k = -G(\mathbf{x}_k)$ and then updating the unknowns. In theory, Newton's method requires the exact solution to the linear system for each Newton iteration. Inexact Newton methods use some form of iterative procedure to solve the linear system approximately. Several iterative techniques such as SOR and multi-grid were tested on this problem with varying success. Note that the matrix $A := G'(\mathbf{x}_k)$ is singular at bifurcation and turning points and can be indefinite near these points. This can cause convergence problems when solving the inner linear system.

A major concern is to determine how accurately should one solve the linear system before doing the Newton update. In the early stages of the Newton or *outer* iteration, the linear system need not be solved too accurately, since $\mathbf{x}_k$ is relatively far from the true solution $\mathbf{x}^*$. Thus only a few *inner* iterations of the linear solver need to be performed. In later stages as $\mathbf{x}_k$ gets closer to the true solution, the inner system will need to be solved more accurately. This is precisely the philosophy of the inexact Newton method. Since inner iterations are expensive, one should not do any more than necessary. A common criterion used to determine how many inner iterations are needed is as follows:

1. Let $\mathbf{x}_0$ be given
2. For $k = 0, 1, 2, \ldots$
3. Determine $n_k \epsilon [0, 1)$ and $\mathbf{s}_k$ such that

$$\frac{\|G(\mathbf{x}_k) + G'(\mathbf{x}_k)\mathbf{s}_k\|}{\|G(\mathbf{x}_k)\|} \leq n_k \tag{5}$$

4. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$

The quantity on the left of inequality (5) is called the relative residual, that is the residual of the inner iteration divided by the residual of the outer. Expression (5) may be interpreted intuitively as indicating that we should iterate until the inner residual becomes "small" enough, then do an update. Common choices for n_k include:

1. $n_k := \frac{1}{2^{k+1}}$
2. $n_k := \frac{1}{k+2}$
3. $n_k := \min\{\|G(\mathbf{x}_k)\|, \frac{1}{k+2}\}$.

Any of the above choices for n_k will guarantee at least linear convergence of the outer iteration. For the numerical results presented later on in this paper the choice of $n_k = \frac{1}{k+2}$ was used, as extensive numerical tests revealed this to be, on average, the best choice.

5. ITERATIVE SOLVERS

Several classical iterative schemes were used to solve the inner sparse linear system for $q_1, \ldots, q_5$ at each nodal point. Each method had certain advantages and disadvantages when used as a solver on the Wavetracer. The following schemes were tried for the numerical simulations:

1. Red-black SOR (sorrb)
2. Multi-level red-black SOR (ml-sor)
3. Multi-grid (m-grid)
4. Multi-level multi-grid (ml-mg)
5. Preconditioned conjugate gradient (pre-cg).

All were implemented as point-iterative methods. Other natural schemes would involve blocking the $q_1, \ldots, q_5$ at each nodal point and using a block-iterative technique such as block-SOR.

A red-black coloring scheme was used for the SOR iterations [8] in order to introduce parallelism into the method. One should recall that, with red-black ordering, the Gauss-Seidel method decomposes into two Jacobi steps on the half size systems resulting from the coloring. Unlike the original Gauss-Seidel method, the Jacobi method is highly parallelizable. The red-black SOR produces similar benefits. Only two colors were needed in the case considered here, since there is only a 5-point stencil at each node. In the full model more colors would be required, since the stencils are more elaborate. The *optimal* parameter ω was chosen as that from the simple Laplacian model since the matrix in our linear system has a similar structure to the Laplacian matrix. Numerical experimentation showed that this was a good choice for our reduced model and gave good convergence results.

The philosophy behind the multi-level schemes [1, 6] is as follows. The problem is solved on a coarse grid to a certain precision. The results are then interpolated to a finer grid and used as initial starting values for these iterations. A sequence of successively finer grids is used. The

finest grid is the one of interest. It is hoped that providing good initial guesses will reduce the amount of work needed to obtain the desired accuracy on the finer grids. This effect is observed in the numerical simulations.

Multi-grid methods [2, 7] are a class of fast linear solvers which are widely used. These methods performed well in our numerical simulations. The idea again is to solve the problem on a series of coarse and finer grids. When multi-grid was combined with a multi-level scheme, it performed exceptionally well. Multi-level multi-grid uses multi-grid as the linear solver on each successively finer grid. Multi-level V-cycle multi-grid gave the best overall performance in our numerical simulations.

Preconditioned conjugate gradient [8] did not perform as efficiently on this architecture and thus was not as competitive as the other methods discussed. This may be explained by the fact that the algorithm requires many dot products, which are not efficient operations on a SIMD machine like the Wavetracer. Several pre-conditioners were tried and the performance of all were essentially similar. Symmetric red-black SSOR [8] was chosen because of its simplicity and because it was easily parallelizable.

6. NUMERICAL RESULTS

To illustrate the comparative speeds of the different solvers a test problem with known solution was used. The solution was assumed to be

$$q_1 = x,\ q_2 = x^2,\ q_3 = y,\ q_4 = y^2,\ q_5 = xy,$$

the corresponding right hand side was generated, and appropriate Dirichlet (strong anchoring) conditions at the boundary were used. The simulations were performed in double precision on a Wavetracer DTC-4 with a 64×64 grid of processors. The number of points was assumed to be the same in both the x and y directions. In addition to the residuals, the errors can also be computed after each outer iteration, since the true solution is known. Initial guesses in all cases were chosen so that the initial maximum error at all points, except the boundary points, was 1.0. The initial maximum residuals for the $n = 32, 63$ and 64 cases were 2.1(3), 7.9(3), and 8.2(3) respectively. The iterations were continued until the maximum residual for the system was reduced by a factor of 10^6. Table 1 gives the results of these numerical simulations.

The Wavetracer DTC does not itself contain a micro-second timer. Consequently, all timings must be performed on the Sun 3/50 front end. The columns real, user and syst give the real (wall clock) time, the time spent in systems tasks related to the program, including input/output, and the time spent in executing user code on the front end. The input/output time includes time spent accessing the SCSI bus and thus time spent sending instructions from the front end processor to the sequencer of the Wavetracer. User time includes time spent executing the sequential parts of the program. The majority of the remaining real time is time elapsed while the DTC is executing parallel instructions. Maximum residuals and errors are computed over all grid points for $q_1, \ldots, q_5$. For the multi-level schemes the number of outer iterations performed at each level are given in parentheses in the last column of the table along with the total number of outer iterations needed for convergence.

In the sequential case, results in the literature would lead one to expect that the multi-grid method would converge with less iterations. This proves to be the case here also. The multi-grid