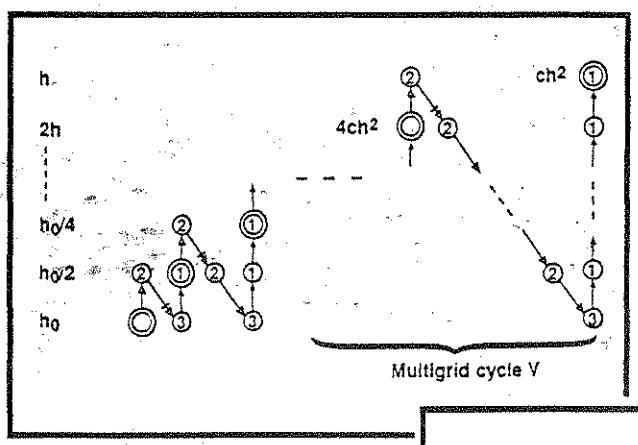
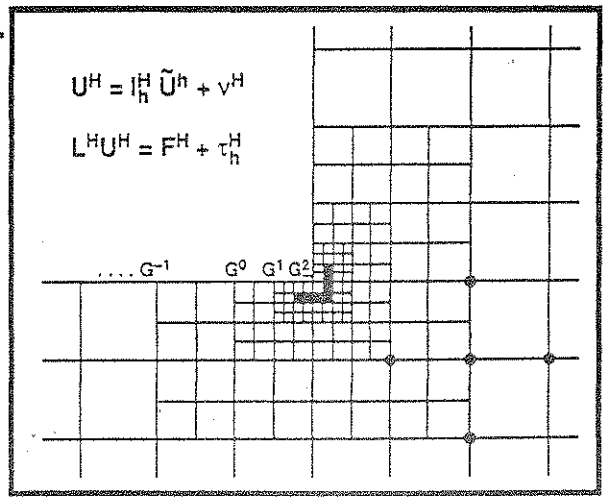
NASA Conference Publication 3224
Part 1

Sixth Copper Mountain Conference on Multigrid Methods





Proceedings of a workshop held at
Copper Mountain, Colorado
April 4–9, 1993

NASA

APPLICATION OF MULTIGRID METHODS TO THE SOLUTION OF LIQUID CRYSTAL EQUATIONS ON A SIMD COMPUTER

Paul A. Farrell[a], Arden Ruttan and Reinhardt R. Zeller
Department of Mathematics and Computer Science, Kent State University
Kent, OH

SUMMARY

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 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.

INTRODUCTION: LIQUID CRYSTALS

Liquid crystal based technology plays a key role in many devices such digital watches and calculators, active and passive matrix liquid crystal displays in laptop computers, switchable windows using Polymer Dispersed Liquid Crystals (PDLCs), thermometers, temperature sensitive films and materials such as Kevlar which employ high-strength liquid crystal polymers. In addition they are likely to play a key role in developments such as High Definition Television (HDTV) and optical communications and computing.

Liquid crystals are so called because they exhibit some of the properties of both the liquid and crystalline states. In fact they are substances which, over certain ranges of temperatures, can exist in one or more *mesophases* somewhere between the rigid lattices of crystalline solids, which exhibit both orientational and positional order, and the isotropic liquid phase, which exhibits neither. Liquid crystals resemble liquids in that their molecules are free to flow and thus can assume the shape of a containment vessel. On the other hand they exhibit orientational and possibly some positional order. This is due to the intermolecular forces which are stronger than those in liquids

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.

[a]This author was supported in part by The Research Council of Kent State University.

and which cause the molecules to have, on average, a preferred direction. Liquid crystals may exist in a number of mesophases, such as the *nematic, smectic, cholesteric* phases (see [3]).

In this paper we shall confine ourselves to nematic liquid crystals, which exhibit orientational but no positional order. We wish to study the orientational order and the inter-molecular forces that are present in a nematic liquid crystal material. To do this we need a quantitative measure of the degree of order and of the total *free-energy* (sum of inter-molecular forces) in the system. A typical liquid crystal molecule is long, rod-like and rigid. Its direction in space is given by the unit vector $n = (n1, n2, n3)$. The molecule points in the n or $-n$ direction with equal probability; therefore, there is no up or down direction. The *director* $\hat{n} = (\hat{n1}, \hat{n2}, \hat{n3})$ is also a unit vector showing the preferred average direction of the molecules at a point in the sample. The *degree of order* of a liquid crystal material at a particular point in the sample can be measured in terms of the statistical average of the angles θ, which molecules make with the director. A more common measure that is used is $S :=< 3\cos^2\theta - 1 > /2$, where $<>$ is a thermodynamic or temporal average. A value close to 1 indicates a strong ordering of the molecules as is present in a crystalline solid. Values near zero indicate random ordering, such as exist in an isotropic liquid. The order parameter S depends on the temperature T.

Most early theoretical and computational results on liquid crystals employed the Oseen-Frank theory. This assumes that the degree of order S is uniform throughout the material and seeks to calculate the equilibrium configuration of the material by obtaining the director field which minimizes the *free energy functional*

$$F(n) := \frac{1}{2}\int_\Omega \{K_1(\nabla \cdot n)^2 + K_2(n \cdot \nabla \times n)^2 + K_3|n \times \nabla \times n|^2\}.$$

In an infinite bulk the preferred configuration for the director field is one of uniform parallel alignment. This will not normally be the case in practice, however, due to the effects of boundaries and external fields. This theory, while instrumental in predicting many important phenomena in liquid crystal physics, has some deficiencies. In particular, it is inadequate to model behavior close to a defect, where the order may not be uniform and the director may not be well defined. For example, in the presence of a radial field about a line defect this theory will exhibit a singularity at the core. For this reason there is increased emphasis on the more computationally complex Landau-de Gennes formulation.

THE LANDAU DE-GENNES FORMULATION

The Landau-de Gennes formulation describes nematic liquid crystals by a 3×3 symmetric, traceless tensor order parameter Q. The local orientational information is given by the eigenvectors and eigenvalues of Q at each point. Several behaviors can be distinguished by considering the relative magnitudes of the eigenvalues. The material is said to be *uniaxial* if Q has a unique largest eigenvalue, with the two other eigenvalues equal to minus half the largest one. The corresponding eigenvector gives the locally preferred direction . Thus this is the case which can be represented by the Oseen-Frank theory and in fact in this case Q can be represented in the form

$$Q = \frac{1}{2}S(3\hat{n}\hat{n}^T - I)$$

where S is the value of the maximum eigenvalue and $\hat{n}$ is the normalized eigenvector associated with it. The Landau-de Gennes formulation, however, is capable of representing more complex behaviors, such as the *biaxial* case, where all three eigenvalues are distinct and the *isotropic* case, where all three eigenvalues are equal and hence, because Q is traceless, all three are 0.

To obtain the equilibrium tensor field again seek a tensor 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 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.

The term $F_{\text{vol}}(Q)$ gives an approximation of the interior free energy and is given by the following expression, (see, for instance [18]):

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

where L_1, L_2, and L_3 are elastic constants, A, B, C, D, M, and M' are bulk constants, and H, $\Delta\chi_{\max}$, and E are the field terms and constants associated with the magnetic field respectively, and the convention is used that summation over repeated indices is implied and that indices separated by commas represent partial derivatives. The surface free density f_{surf} has the form

$$f_{\text{surf}}(Q) := \frac{1}{2}V\text{trace}((Q - Q_0)^2) \tag{2}$$

where Q_0 is a tensor associated with the type of anchoring of the surface elements and V is prescribed constant. In the strong anchoring case presented here Q cannot vary from Q_0 and hence $\int_{\partial\Omega} f_{\text{surf}}(Q) = 0$.

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

The discretization of the full slab problem in which a finite difference approximation of the equilibrium configuration of liquid crystals in a slab

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

is given in [7]. 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:

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

The region is discretized in the standard manner by dividing the rectangle Ω into $I \times J$ regions

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

for $0 \leq i \leq I-1$, and $0 \leq j \leq J-1$, where $\Delta x = a/I, \Delta y = b/J$.

The discrete interior free energy integral is now represented by

$$\int_\Omega f_{\text{vol}}(Q) \approx \sum_{i,j} f_{\text{vol}}(Q(x_i, y_j)) \times volume(v(i, j)), \tag{3}$$

where the points $P = (x_i, y_j)$, for $x_i = i\Delta x$ and $y_j = j\Delta y$, are located in the lower left-hand corner of the rectangle $v(i, j)$. The derivatives with respect to x and y in (1) are approximated using central difference approximations.

With the assumption of strong anchoring, a second order accurate approximation of the Landau-de Gennes free energy density given by

$$F(Q) \approx \sum_{i,j} f_{\text{vol}}(Q(x_i, y_j)) \times volume(v(i, j)) = \sum_{i,j} h(x_i, y_j) \tag{4}$$

is obtained. With the discretization (4), the problem is reduced to one of minimizing $\sum_{i,j} h(x_i, y_j)$ overall choices of $\{q_\ell(x_i, y_j)\}_{\ell=1}^5$. This unconstrained discrete minimization problem can be attacked in the standard way. That is, seek a solution of the non-linear system of equations

$$g(\hat{\ell}, \hat{i}, \hat{j}) := \frac{\partial \sum_{i,j,k} h(x_i, y_j)}{\partial q_{\hat{\ell}}(x_{\hat{i}}, y_{\hat{j}})} = 0, \tag{5}$$

for $0 \leq \hat{i} \leq I$, $0 \leq \hat{j} \leq J$, and $\hat{\ell} = 1 \ldots 5$. A standard approach to solving non-linear systems such as these is to use a modified Newton method (see [6]).

Each iteration of the modified Newton method involves solving a linear system, whose matrix is the Jacobian of (5), and then using that solution to update the iterate and the Jacobian, after which the process is repeated. The system in question is a large symmetric system, but for certain values of the temperature it may become indefinite. In addition, it may be expected to exhibit multiple solutions, which may be either stable or unstable. The ultimate aim of this research is to track the

minimal energy states as the temperature varies and to model the resulting bifurcations and phase transitions.

THE WAVETRACER DTC ARCHITECTURE

The target architecture for this application is a massively parallel SIMD computer. A SIMD computer uses multiple synchronized processing elements that operate in a lock-step fashion to achieve parallelism. Each processing element (PE) performs the same operation at the same time on its local data which is either stored in its own local memory or in a shared memory. A control unit (CU) broadcasts instructions to the processing elements for execution. Each PE can be either active or inactive during a particular operation. The control unit determines which PEs are to participate by means of a masking function that either turns a PE on or off. Only the selected processors execute the instruction, while the masked processors remain idle. The control unit normally buffers data and instructions that will be broadcast to the processor array. A front-end computer provides the programming environment along with the usual programming utilities such as a debugger and a compiler. Program code is compiled and separated into scalar and parallel instructions. Scalar operations are usually executed on the front-end, thus freeing the processor array to perform only parallel computations. 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, connected by a special purpose high-capacity communication network. Early examples of this architecture included the MASPAR MP-1 and MP-2 and 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. This has a number of unique features compared with previous SIMD computers. It was designed as a low cost massively parallel processor, which can deliver "super-computing" levels of performance at relatively low cost. Unlike previous SIMD machines, which had dedicated front-end processors for storing scalar data and performing uni-variable (scalar) computations, the DTC uses a standard workstation for this purpose as well as for compilation and storage of the program. Among front-ends supported were the Sun 3, Sparc and Hewlett-Packard/Apollo workstations.

The DTC is connected to the front-end by means of the industry standard Small Computer System Interface (SCSI), which is normally used to connect hard disks. The maximum bandwidth of this interface is 5 Mbytes per second. 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 RAM for each processor, and associated control and memory error-detection circuitry. In addition, each processor has access to between 8 and 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 application, or a 64×64 square, for two dimensional applications. The DTC-8, can be configured as $16 \times 32 \times 16$ cube or a 64×128 square, and the DTC-16 as a $32 \times 32 \times 16$ cube or 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.

There are a number of factors which affect the DTC's performance. Firstly, the speed of the front-end is a determining factor in the overall performance of the DTC, since all uni-variable expressions are processed on the front-end and, in addition, all instructions are passed from the front-end to the control unit. In addition, although the DTC provides efficient data movement along the grid, the results of propagating data to the left, for example, are undefined at the right boundary nodes. In addition, for problems with periodic boundary conditions it is desirable that the interconnection network have *wraparound*, in which one can propagate values from one boundary to the other. This is not provided. This also poses a problem for periodic geometries such as spherical or ellipsoidal. One other inconvenience is that there is no microsecond timer on the DTC and all timings must be done on the front end.

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.

The Wavetracer used in the results presented here was a Wavetracer DTC-4 with a Sun 3/50 front end. Current codes are bring run on a Wavetracer DTC-16 with a Hewlett-Packard/Apollo 705 front end. 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 rectangular grid of the DTC, provided an equal number of grid points are used in each direction.

At each point P 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. Associated with each unknown $q_\ell(P)$ 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 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) to 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 at worst consist of the nine points which correspond to processors at most two steps away from the given processor.

SOLUTION OF THE MINIMIZATION PROBLEM

Non-linear iterations

The minimization of the free energy can be carried out by solution of the corresponding discrete Euler-Lagrange equations given by (5). These give rise to a coupled system of five non-linear elliptic partial differential equations.

An alternative approach is to compute the Euler-Lagrange equations from $\int_\Omega f_{\text{vol}}(Q)$. Discretizing these produces a system similar to (5). In this case central differences are used for the unidirectional partial derivatives. Two alternative choices of discretizations for the mixed derivatives, both having the same accuracy, are considered. One produces a seven and the other a nine 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. A reduced model in which $L_2 = L_3 = D = M = M'$ was also considered. This is significantly less complex and gives rise to a five point scheme. Results for this case were considered in [7].

In all cases the resulting non-linear system of equations was solved using a (modified) inexact Newton method. Let $G : R^n \rightarrow R^n$ be a function representing the discrete Euler-Lagrange equations. There are a total of $5(I-1)(J-1)$ non-linear equations in this system. The function G depends on the $5n$ unknowns

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

where $n = (I-1)(J-1)$ is the number of nodal points. Let $G'(\mathbf{x})$ be the Jacobian of the system of equations. Newton type methods require solving a large sparse linear system $G'(\mathbf{x}_k)\mathbf{s}_k = -G(\mathbf{x}_k)$ and then updating the unknowns appropriately.

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 multigrid 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. It is well known that in the early stages of the Newton or *outer* iteration process, the linear system need not be solved to full accuracy, 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, the inner system will need to be solved more accurately. This is precisely the philosophy of the inexact (modified) Newton method. A common criterion used to determine how many inner iterations are needed is as follows. In the k^{th} iteration, compute a value $n_k \epsilon [0, 1)$ which is an acceptable bound on the relative residual. Common choices for this are $n_k := \frac{1}{2^{k+1}}$, $n_k := \frac{1}{k+2}$, and

$n_k := \min\{\|G(\mathbf{x}_k)\|, \frac{1}{k+2}\}$. For these problems the second of the above choices proved on average to give the best results. The update $\mathbf{s}_k$ was then determined by:

$$\frac{\|G(\mathbf{x}_k) + G'(\mathbf{x}_k)\mathbf{s}_k\|}{\|G(\mathbf{x}_k)\|} \leq n_k \tag{6}$$

Expression (6) may be interpreted intuitively as indicating that one should iterate until the inner residual becomes "small" enough, then do an update.

Linear System 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 evaluated :

1. Multi-color SOR
2. Nested (multilevel) multi-color SOR
3. Preconditioned conjugate gradient
4. Multigrid (V-cycle)
5. Nested (multilevel) multigrid

All were implemented as both point-iterative and as block-iterative methods by blocking the $q_1, \ldots, q_5$ at each nodal point. In the point iterative methods one solves for each q_i sequentially, using the best available values for the $q_j, j \neq i$. The block method involves solving a 5×5 dense linear system at each node.

A multi-coloring scheme was used for the SOR iterations [17] 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 multi-colored SOR produces similar benefits. In the case of the reduced model with the five point stencil only two colors were needed. Results for this case are given in [7]. In the full model, three colors are required for the seven point stencil and four colors for the nine point stencil. The parameter ω for the SOR method was chosen as the *optimal* parameter for 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.

Preconditioned conjugate gradient [17] using several pre-conditioners was tried and the performance of all were essentially similar. The results are presented here for symmetric multi-colored SSOR [17], which is simple to implement and easily parallelizable.

Multigrid methods [2, 14, 16] were also implemented for these problems. The multigrid implementation discussed here uses a single V-cycle in the inner iteration for each Newton outer iteration. The Gauss-Seidel iteration is used as the relaxation method on the fine and intermediate coarse grids. The Gauss-Seidel method was chosen over the SOR method for the fine and

intermediate grids because of its better *smoothing* property; that is, it eliminates the high frequency components quicker in the early iterations than the SOR iteration. This is important because a few iterations are performed on these grids per cycle of the multigrid algorithm. The relaxation parameters ν_1 and ν_2 were usually taken to be equal to 3. Multicolored SOR iteration was used to solve the problem on the coarsest grid which was usually taken to be of size $n = 4$. The problem was solved to the level of the truncation error with usually just a few iterations. The numerical simulations were mostly done on the two-dimensional problem of size $n = 64$, meaning 65 grid points in both the x and y directions. Some smaller and larger problems were also examined, but with the minimal configuration of the *Wavetracer*, the DTC-4, available at the time, the $n = 64$ size problem was the largest that could be simulated for the full liquid crystal problem using the multigrid method.

The implementation of the multigrid algorithm on the *Wavetracer* assigns a processor (virtual or physical) to each grid point on the finest mesh, including the boundary grid points. The model simulations all assume Dirichlet (strong anchoring) boundary conditions, so the boundary processors are used mainly to store the boundary data. The *Wavetracer* uses a multi-array data structure to hold the values for each grid level. Because of the restriction in the MultiC language that each multi-variable in the executing program must be of the same size, this implementation was deemed to be the most efficient and easiest to implement. One problem with this implementation is that many processors are idle when solving on the coarser grids. The multigrid is thus not a fully parallelizable method using this implementation because not all processors are being utilized. Alternative variations have been proposed to overcome this problem. Data transfers between grids are fast since they are handled within processor memory and no communications between processors is required. Communications are required when computing the weighted averages for the restriction operation, but the actual transfer of data to the coarser grid is all done within processor memory. Another drawback to this implementation of the multigrid method is evident when one solves the $n = 64$ size problem in two-dimensions. The physical two-dimensional processor grid on the *Wavetracer* contains 64 processors in each dimension for a total of 4096 processors. The $n = 64$ multigrid problem requires 65 mesh points in each of the x and y directions. This causes the *Wavetracer* to operate in *virtual memory* mode. Since each physical processor must contain the same amount of virtual processors, many virtual processors will remain idle during the iterations, resulting in a great loss in efficiency. In addition, since the available memory associated with each physical processor is divided into two halves, one for each of the virtual processors, the maximum problem size, which can be solved, is diminished. Naturally, the solution would be to define a slightly smaller problem of $n = 63$ that would not have this difficulty. The problem then becomes one of how to define the series of coarser grids. In our original definition of the coarse grids we let each grid size be a power of two. This greatly simplifies the construction of the grids and provides the necessary symmetry to allow us to assign processors to the different grid levels in the manner described. Data transfer between grids is also extremely simple, since it is all handled within processor memory. Defining the coarse grids in any other way would greatly complicate the programming process and would require many more computations and inter-processor communications.

Another solution to this problem would be to use the boundary processors to not only store the boundary data but also to take part in the iteration process. This means that now each boundary processor would really represent two grid points in the mesh instead of only one. This would solve

the virtual processing problem because one would actually need only 63 physical processors in each direction for the $n = 64$ problem. However, another problem presents itself because of the *SIMD* nature of the *Wavetracer*. In a *SIMD* environment each processor must perform exactly the same operation as all the other processors, except on a different set of data. The boundary processors as defined above would have to be treated separately from the interior processors because in the communications stage of the algorithm they are not performing the same operation. An interior processor must communicate with its four nearest-neighbors in a five-point stencil scheme, whereas a boundary processor would only have to communicate with a subset of its neighbors since the boundary data that it needs to do its update is stored in its own memory. In the two-dimensional mesh the processors on each of the four edges of the grid must be treated separately as must the four corner processors. In a naive implementation these sets of processors would be handled sequentially in the iteration process, greatly slowing down the computations. In fact, if the obvious choice is made, this could increase the update time nine times, which is considerably more than the increase incurred by virtual processing. Unfortunately this problem is not so easily avoided when one considers general boundary conditions rather than Dirichlet conditions.

Another alternative approach is to use a Black Box multigrid method similar to that in Dendy [4, 5]. This eliminates the restriction that the number of unknowns in the finest grid should be $2^k + 1$; for some k. In addition, by storing the interpolation operators explicitly, it allows the incorporation of the boundary conditions, for example, by using extrapolation at the points closest to the boundaries. Thus the boundary conditions are incorporated algebraically rather than by using the difference equations directly. This does involve extra storage and in the SIMD case loss of parallelism due to grid point dependent code. However judicious coding, involving initialized multipliers, can reduce the latter effect at the expense of some further storage. There is reason to believe that, for most problems of this type and most geometries, the increased storage will be less than 100% and thus that a code of this type will consume less storage overhead than one involving virtual processing.

The philosophy behind the nested or multilevel schemes [1, 13] 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 the solution process there. A sequence of successively finer grids is used, the finest is the one on which the result is required. 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. The multilevel methods suffer the same kinds of problems that the multigrid iterations suffered when implemented on the *Wavetracer*. The different levels are implemented using a multi-variable array (in the MultiC language) with the physical (or virtual) processors assigned to the grid points on the finest grid level. This means that when one iterates on the coarsest level, many virtual processors will be idle. The interpolation of results between grids is fast because it is all done within the processor and no inter-processor communications are necessary.

NUMERICAL RESULTS

Laplacian and Scalar Liquid Crystal Problem

Laplacian in Two Dimensions

The model Laplacian problem in two-dimensions is given by:

$$-u_{xx} - u_{yy} = f(x,y), \qquad u = g(x,y) \quad \textit{on the boundary of } \Omega. \tag{7}$$

Dirichlet boundary conditions are assumed and Ω is taken to be the *unit square*. The performances of the various iterative methods previously discussed are compared for problems of size $n = 63$ for the one-level schemes and $n = 64$ for the methods using more than one level. For these simulations we also assume a known true solution given by

$$u = x^2 y^2 \tag{8}$$

which makes the right-hand side of equation (7)

$$f(x,y) = -2.0 * (x^2 + y^2). \tag{9}$$

With this known solution one can compute the *error* as well as the *residual* after each iteration in order to observe the convergence. The boundary values are set to the known true solution and an initial guess of $u = 0.0$ is used at all interior grid points to start the iterations. At each iteration the maximum absolute error and residual (infinity norms) calculated over all interior grid points are monitored.

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.

The results of these simulations are given in Table 1. Given the initial guess $u = 0.0$, the maximum initial error is 1.0 and the maximum initial residuals are approximately 7934 and 7684 for the $n = 64$ and $n = 63$ size problems, respectively. The iterations are continued until the maximum absolute error is reduced by about a factor of 10^5. A red-black scheme is implemented for all the iterations (except Jacobi) to induce parallelism into the methods. The red-black coloring scheme is appropriate since the model Laplacian problem uses a 5-point stencil for processor communications. The iterations are done on the *Wavetracer* using a 64×64 physical two-dimensional grid of processors.

Table 1. Timings for the Model Laplacian Problem on the Wavetracer DTC

	real	user	syst	max. residual	max. error	iterations
Jacobi (n=64)	150.8	9.9	67.0	1.5(-3)	3.5(-5)	6901
Jacobi (n=63)	132.6	13.5	78.4	2.1(-3)	3.5(-5)	6689
Gauss-Seidel (n=64)	89.1	7.5	40.5	1.5(-3)	3.5(-5)	3451
Gauss-Seidel (n=63)	67.3	8.2	39.3	1.5(-3)	3.5(-5)	3345
SOR (n=64)	3.2	0.4	1.5	6.2(-2)	3.4(-5)	113
SOR (n=63)	3.2	0.4	2.0	5.8(-2)	3.5(-5)	111
Pre-cg (n=64)	6.1	0.5	1.0	7.2(-2)	2.5(-5)	32
Pre-cg (n=63)	2.7	0.5	1.0	6.8(-2)	3.1(-5)	31
Multigrid (n=64)	2.5	0.3	0.5	3.4(-2)	3.5(-5)	3 V-cycles

As expected, the Jacobi iteration is the slowest to converge. Even though it is completely parallelizable on the *Wavetracer*, its slow rate of convergence does not make it competitive. The Gauss-Seidel method converges in about half as many iterations as the Jacobi method. This is expected for the model Laplacian problem. Since the Gauss-Seidel iterations are implemented in a red-black ordering, each iteration takes slightly longer than a Jacobi iteration. For both the Jacobi and Gauss-Seidel iterations the real running times for the $n = 63$ size problem are faster than those for the $n = 64$ problem. This is because the $n = 64$ problem uses virtual processors whereas the $n = 63$ problem fits the physical grid of processors precisely.

The SOR method greatly improved the convergence of the problem. It needed only 113 iterations to get to the same level of error as the previous two iterative methods (for the $n = 64$ problem). The real times, user, and systems have also been significantly reduced. This agrees with the theoretical results for the behaviour of these three iterative methods on the model problem.

The preconditioned conjugate gradient iteration was implemented using a red-black coloring scheme and *Symmetric SOR* as the preconditioner. The method is competitive with the SOR iteration for the $n = 63$ problem. It is, however, slower than SOR for the slightly larger problem.

To make a fair comparison, one must compare the multigrid algorithm with the $n = 64$ size problems of the other four iterative methods, since multigrid was implemented using a finest grid of this size. As one can see from the table, multigrid converges significantly faster than Jacobi, Gauss-Seidel and preconditioned conjugate gradient, and slightly faster than SOR (in real time). It even beats the other four methods when they are run on the smaller problem. This shows that multigrid is a very competitive method even with its limitations as discussed previously. Only three V-cycles are needed to reduce the error to the desired level. Five levels were used ($n = 4$ at the coarsest level) with $\nu_1 = \nu_2 = 3$.

Scalar Liquid Crystal Problem

The scalar analog to the full liquid crystal problem is of interest because it has a similar structure to the full model. Various algorithms for solving the full model are first developed for the scalar problem. The relative performances of these algorithms were basically the same for both models.

The free-energy density for the scalar-field analogue to the full systems model is given by:

$$f(q) = \frac{1}{2}L_1|\nabla q|^2 + \frac{1}{2}Aq^2 - \frac{1}{3}Bq^3 + \frac{1}{4}Cq^4 - H^2q \tag{10}$$

where L_1 is an elastic constant, A,B,C are bulk constants and H is a field term representing an outside field such as a magnetic field. To minimize the free-energy of the system one needs to solve the *Euler-Lagrange* equation

$$-L_1\nabla^2 q + Aq - Bq^2 + Cq^3 = H^2. \tag{11}$$

Equation (11) is non-linear in the scalar variable q. The resulting linear system that needs to be solved at each Newton step is very similar in structure to the *Laplacian* problem. The only difference is the additional terms on the diagonal elements of the A matrix that is a result of the non-linearity of the scalar problem. The discretization of the scalar Euler-Lagrange equation produces a 5-point stencil at each mesh point. The communications pattern is thus the same as it was for the Laplacian problem. A red-black coloring scheme is sufficient to induce parallelism into the iterative solvers used.

For the problem used in these tests $L_1 = 1.0$, $A = B = C = 1.0$, $H = 0.0$, with Dirichlet boundary conditions given by :

$$q = 1 \text{ on } x = 1 \text{ and } y = 1\,, \quad q = x \text{ on } y = 0\,, q = y \text{ on } x = 0\,.$$

The true solution to this problem is not known, therefore the error cannot be computed. The maximum absolute residual at each iteration is used to monitor the convergence. The initial guess is given by q=0.0 at each interior mesh point and iteration proceeds until the maximum absolute residual is reduced by approximately factor of 10^6. The initial residuals for the $n = 64$ and $n = 63$ size problems are 8192 and 7938, respectively. Table 2 gives the results of the simulations.

Table 2. Timings for the Scalar Liquid Crystal Problem on the Wavetracer DTC

	real	user	syst	max. residual	max. error	outer iter.
SOR (n=64)	6.2	0.3	1.4	2.7(-3)	—	9
SOR (n=63)	3.3	0.3	1.4	2.4(-3)	—	9
Pre-cg (n=64)	12.0	0.8	1.2	2.5(-3)	—	8
Pre-cg (n=63)	5.2	0.8	0.9	2.1(-3)	—	8
Multigrid (n=64)	4.0	0.5	0.6	2.3(-3)	—	4
Nested SOR (n=64)	6.8	0.5	1.7	5.9(-3)	—	19(4,3,4,4,4)
Nested Multigrid (n=64)	4.0	0.4	0.7	5.9(-3)	—	8(4,1,1,1,1)

Comparing the real execution times of these algorithms shows again that the preconditioned conjugate gradient method is not competitive on this kind of architecture. The nested (multilevel) methods use five levels with the coarsest level being of size $n = 4$. The numbers in parentheses in the last column of the table are the number of outer Newton iterations needed to achieve convergence at each level.

We refer to a nonlinear Newton based analogue of the Full Multigrid (FMG) method as nested (multilevel) multigrid. It employs Newton iteration on each mesh level until the desired accuracy is attained. In the case of the example considered here, this required 4 outer iterations on the coarsest mesh and one each of the finer mesh levels. With the exception of the coarsest level, a V-cycle with $\nu_1 = \nu_2 = 3$ is applied at each level to solve the linear system arising from the Newton process. It is considerably faster than nested SOR in achieving the same reduction in the residual. Thus with the exception of the initial 4 Newton cycles, it is a natural nonlinear analogue of the Full Multigrid method (FMG) [16, p.22]. Multigrid (non-nested) seems to perform the best since its timings are essentially the same as nested multigrid but it has greater residual reduction. The SOR ($n = 63$) iteration has the fastest real time but on a smaller problem where no virtual processing is involved.

Note that the *optimal* ω from the Laplacian model was used in the SOR iterations for the scalar problem. The experimental results showed that this was a good choice and gave the best convergence over any other choice. The stopping criteria used to terminate the inner iterations for each Newton step was $n_k = 1/(k+2)$.

Full Liquid Crystal Problem

Table 3 gives the results of the numerical simulations for the full systems model. The same test problem was used as in the case of the reduced model together with the appropriate Dirichlet boundary conditions. Only the size $n = 64$ problem was considered for this set of runs. The following set of parameter values was used: $L_1 = 10.0$, $L_2 = L_3 = 1.0$, $A = B = C = D = M = M' = 1.0$ and outside field parameters are set to zero. Results from both the 7-point and 9-point discretizations are given. Both point and block iterative methods were compared. The initial maximum absolute residuals for the 7-point and 9-point schemes are 8.2(4) and 8.95(4), respectively. The iterations were continued until the maximum residual was reduced by approximately a factor of 10^6. The initial maximum error is 1.0 since initial guesses of $q_i = 0.0$, $i = 1, \cdots, 5$ were used for the interior mesh points. The simulations were all done in single precision.

The SOR methods used 10 inner iterations for each Newton outer iteration. The stopping criteria used for the reduced model ($n_k = 1/(k+2)$) was too restrictive in some cases and caused convergence problems. Using 10 inner iterations avoided these problem areas. As before, the multigrid methods outperformed their SOR counterparts. The 7-point iterative scheme (point method) was competitive with the 9-point scheme for both multigrid and nested (multilevel) multigrid. This was not the case for the SOR methods. The 9-point scheme performed better for the one-level SOR case but did worse for the multilevel iteration. Block methods were not competitive for either multigrid or nested multigrid. The block method performed best for the single-level SOR iterations, and was also competitive in the nested case. The best algorithm for solving the test problem was again nested (multilevel) multigrid using the point iterative approach. The 9-point scheme performed marginally better than the 7-point, producing a slightly smaller residual, upon convergence, in about the same amount of real time. The pre-conditioned conjugate gradient methods were not implemented for the full model since they showed to be not competitive in the reduced model case [7].

Table 3. Timings for the Full Systems Liquid Crystal Problem on the Wavetracer DTC

	real	user	syst	max. residual	max. error	outer iter.
SOR (7p)	277.0	8.2	15.6	8.17(-2)	1.70(-5)	42(10 inner/out)
SOR (9p)	148.7	7.6	12.9	9.10(-2)	2.39(-5)	34(10 inner/out)
Block-SOR (9p)	106.3	6.4	6.5	7.95(-2)	1.27(-5)	17(10 inner/out)
Multigrid (7p)	47.2	2.0	1.6	6.55(-2)	5.36(-5)	4(1 V-cyc/out)
Multigrid (9p)	42.2	2.2	1.9	3.40(-2)	4.72(-5)	4(1 V-cyc/out)
Block-Multigrid (9p)	60.9	4.1	1.6	2.99(-2)	4.15(-5)	4(1 V-cyc/out)
Nested SOR (7p)	85.9	3.6	7.5	8.26(-2)	1.36(-5)	18(3,1,2,4,8)
Nested SOR (9p)	100.5	6.2	9.7	6.14(-2)	1.98(-5)	25(3,1,2,5,14)
Block-Nested SOR (9p)	105.0	7.4	7.6	8.23(-2)	1.05(-5)	18(3,1,2,4,8)
Nested Multigrid (7p)	37.4	2.4	2.8	8.44(-2)	6.88(-6)	8(4,1,1,1,1)
Nested Multigrid (9p)	36.3	2.8	2.8	4.62(-2)	3.86(-6)	8(4,1,1,1,1)
Block-Nested Multigrid (9p)	50.5	3.8	2.7	4.62(-2)	3.89(-6)	8(4,1,1,1,1)

CONCLUDING REMARKS

Multigrid methods work well as inner solvers for liquid crystal problems when implemented on SIMD computers with 2-D grid architectures. Multi-colored SOR methods are also effective, but due to the cost of inner products on such machines pre-conditioned conjugate gradient methods are not. The multigrid algorithms (one-level and multilevel) perform better than their SOR counterparts for the larger $n = 64$ problem.

Although the Wavetracer's mesh architecture fits the problem (discretization) well thereby making communications between nearest neighbors efficient, it is not as well suited for multigrid algorithms. This is due to the fact that the machine has a physical 2-D grid structure with 64 processors in each dimension. For multigrid and multilevel iterative schemes a grid size of 65×65 is required for an efficient implementation, because of the way the grid refinements are defined. So for an $n = 64$ size problem, the machine must to go into virtual processing mode, thus slowing down the execution time of the algorithm and increasing the storage overhead. One solution would be to generate grids that would not suffer this problem, but this involves considerably more complex coding, which would also increase execution time and storage overhead but not to the same extent as virtual processing. We emphasize that the multigrid implementation employed here is effectively the sequential version of the multigrid method. Thus on the coarsest mesh only 0.4% of the processors were active. Despite this disadvantage multigrid proved the fastest of the algorithms tested. We remark that although these methods worked well for the test problems, where the iteration matrix was positive definite symmetric, convergence problems can be expected, when the system becomes indefinite, due to the coarseness of the coarsest mesh. Use of a coarsest mesh with more points can be expected to remove this problem as well as improving the performance due to higher processor utilization. Further improvements in performance can be anticipated if parallelism were introduced using the method of [4, 5, 8, 9, 15].

REFERENCES

[1] R. E. Bank and D. J. Rose, Analysis of a Multilevel Iterative Method for Nonlinear Finite Element Equations, Math. Comp., 39, no. 160, pp. 453–465, 1982.

[2] William L. Briggs, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1987.

[3] P. J. Collings, *Liquid Crystals, Nature's Delicate Phase of Matter*, Princeton Science Library, Princeton University Press, New Jersey, 1990.

[4] J. E. Dendy, Jr., *Black Box Multigrid*, J. Comp. Phys., 48, pp.366–386, 1982.

[5] J. E. Dendy, Jr., M. P. Ida and J. M. Rutledge, *A Semicoarsening Multigrid Algorithm for SIMD Machines*, SIAM J. Sci. Statist. Comput., 13, no. 6, pp. 1460–1469, 1992.

[6] J. E. Dennis Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood, NJ, 1983.

[7] Paul A. Farrell, Arden Ruttan, and Reinhardt R. Zeller, *Finite Difference Minimization of the Landau-de Gennes Free Energy for Liquid Crystals in Rectangular Regions*, Computational and Applied Mathematics, I, C. Brezinski and U. Kulish eds., North–Holland, pp. 137–146, 1992.

[8] Paul O. Frederickson and Oliver A. McBryan, *Parallel Superconvergent Multigrid*, Lecture Notes in Pure. and Appl. Math., 110, Dekker, New York, 1988.

[9] Paul O. Frederickson and Oliver A. McBryan, *Normalized Convergence Rates for the PSMG Method*, SIAM J. Sci. Statist. Comput., 12, no. 1, pp. 221–229, 1991.

[10] Eugene C. Gartland, An Introduction to Liquid Crystals, SIAM News, 25, no. 6, 1992.

[11] Eugene C. Gartland, On Some Mathematical and Numerical Aspects of the Landau-de Gennes Minimization Problem for Liquid Crystals, Inst. for Computational Mathematics Preprint, Kent State University, Ohio.

[12] E. C. Gartland, Jr., P. Palffy-Muhoray, and R. S. Varga, Numerical Minimization of the Landau-de Gennes free energy: Defects in cylindrical capillaries, Mol. Cryst. Liq. Cryst., 199, pp. 429–452, 1991.

[13] W. Hackbusch, Multigrid Solution of Continuation Problems, in *Iterative Solution of Nonlinear Systems of Equations*, Lect. Notes in Math., 953, pp. 20–45, Springer, Berlin, 1982.

[14] Wolfgang Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985.

[15] Oliver A. McBryan and Paul O. Frederickson, *Multigrid Methods on Parallel Computers – a Survey of Recent Developments*, Impact Comput. Sci. Engrg., 3, no. 1, pp. 1–75, 1991.

[16] Stephen F. McCormick, ed., *Multigrid Methods*, SIAM, Philadelphia, 1987.

[17] James M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1989.

[18] E. B. Priestley, P. J. Wojyowicz, and P. Sheng, eds., *Introduction to Liquid Crystals*. Plenum Press, New York, London, 1975.

[19] Reinhardt R. Zeller, *Ph. D. thesis*, Kent State University, in preparation.