The Portable Parallel Implementation of Two Novel Mathematical Biology Algorithms in ZPL

Marios D. Dikaiakos* Calvin Lin†
Daphne Manoussaki‡ Diana E. Woodward§

Abstract

This paper shows that mathematical models of biological pattern formation are ideally suited to data parallelism. We present two new algorithms, one for simulating the dynamic structure of fibroblasts, and the other for studying the self-organization of motile bacteria. We describe implementations of these algorithms using a high level data parallel language called ZPL, and we give performance results for the Kendall Square Research KSR-2 and the Intel Paragon that include comparisons against sequential Fortran.

1 Introduction

Mathematical biology is one of the fastest growing and most exciting applications of modern mathematics. As biology becomes more quantitative, the increased use of mathematical modeling is inevitable. Many of these problems involve extensive numerical computations over large computational domains and are easily amenable to data-parallelism. For example, in the field of pattern formation in biological systems, the equations that one derives are generally continuum models described by nonlinear partial-differential equations that cannot be solved explicitly; thus, numerical methods are crucial in understanding their behavior. The complex dynamics of the continuum often require long calculations for each data point. Moreover, the biological structures often form very complex patterns that require large computational domains for their representation.

This paper makes two primary contributions. First, we describe two new algorithms for solving mathematical biology problems. Second, by using the ZPL programming language [15] to implement these algorithms, we show that ZPL solves many of the problems that inhibit the widespread use of parallel machines.

The first algorithm models the structure of fibroblasts, which are cells of the connective tissue that have been extensively studied by biologists. Fibroblast patterns have been compared to fingerprint patterns using theorems in topology [11], and theories have been devised to describe their movement within the tissue. However, there has been little work in quantifying the influences within culture to validate these theories. In a confluent culture, fibroblasts form dramatic patterns of many parallel arrays of cells meeting at different angles. Our assumptions on how local densities and local orientations interact and change were based on biological observations, and led to the creation of our new mathematical model.

The second algorithm studies the self-organization of motile bacteria, which aggregate in response to gradients of chemical attractants that they themselves excrete. This chemically directed movement, which directs the motion up a concentration gradient, is referred to as chemotaxis. Depending on the conditions under which the cells are cultured, they form a variety of complex spatio-temporal patterns [5, 23]. Based on the solid biological evidence for chemotaxis, we have developed a cell-chemoattractant model mechanism which has enabled us to verify that it is the interaction between the proliferating cells and chemoattractant they produce that is crucial to the formation of the observed geometric patterns.

The numerical solution for the equations of these models provides a realistic litmus test for the use of ZPL, an array sublanguage of the more general Orca

*Department of Astronomy, FM-20, University of Washington, Seattle, WA 98195.

†Department of Computer Science and Engineering, University of Washington.

‡Department of Applied Mathematics, University of Washington.

§Department of Mathematics, Southern Methodist University, Dallas, TX 75275.

ICS '95 Barcelona, Spain © 1995 ACM 0-89791-726-6/95/0007..$3.50

C programming language [14]. We show that ZPL is a convenient platform for implementing these applications, yielding clean solutions and good performance across different parallel machines. In particular, convenience comes from the high level nature of ZPL that frees the programmer from the details of explicit communication and synchronization; from the sequential semantics of the language that allow program development and debugging to proceed on familiar workstation environments; and from source level portability—porting ZPL applications across platforms only requires recompilation of the C code that is produced by the ZPL compiler. The ability to produce *efficient* portable code comes from ZPL's underlying programming model [2], a claim that is supported by the results presented here on two very different parallel computers—the Kendall Square Research KSR-2 and the Intel Paragon.

This paper is organized as follows. Section 2 describes the new algorithm for simulating fibroblast structures. Section 3 discusses the self-organization of motile bacteria and the new model for simulating this process. Section 4 provides some basic background on ZPL. The next two sections describe the ZPL implementation of the two algorithms and give performance results. Concluding remarks are given in Section 7.

2 The Fibroblast Application

2.0.1 Modeling Fibroblasts

Fibroblasts are long, spindle-shaped cells of the connective tissue, the space between organs and tissues. When in culture they interact forming parallel arrays and patterns [11]. This interaction has been described with a model consisting of a parabolic differential equation and an integral equation [8]. When in culture they also acquire a mono- or bipolar shape, which allows us to easily attribute an axis of orientation to each cell. Fibroblasts are highly motile cells and move along this axis of orientation. At a macroscopic level we can assign a local orientation to almost every point in the culture. The few points where an orientation cannot be uniquely assigned are called points of discontinuity. The Fibroblast model is based on the following biological observations:

- *Cells influence the orientation of their neighboring cells:* In particular, upon meeting a cell with an orientation similar to its own, a cell will tend to move, change its orientation, and align itself with its neighbors. If the orientation are dissimilar, cells do not affect each other's movement.
- *Local cell densities are influenced by the local cell orientation distribution:* In areas where all cells are aligned parallel to each other (parallel arrays), we assume that cell movement will be simply diffusive and will depend on the local cell densities. In areas where there is great local variation in cell orientation, we assume that cell movement will be hindered.

Based on these observations, we conclude that the variables that best describe the cell distribution are the cell density N and the cell orientation θ; θ is the angle that the cell forms with a fixed reference axis. This leads to the following model (in non-dimensional form) [11]

$$\theta_t = \int_{B(\epsilon)} K(\underline{x}, \underline{y}, \theta(\underline{x},t), \theta(\underline{y},t), N(\underline{x},t), N(\underline{y},t))\, dy - \nabla \cdot J_\theta + random \tag{1}$$

$$N_t = -\nabla \cdot J_N \tag{2}$$

where $B(\epsilon)$ denotes a ball of radius ϵ around $\underline{x}$, i.e., a small neighborhood of $\underline{x}$. The kernel K measures the rate of change in angle due to the angle and density of neighboring cell populations. In our numerical solution of the equations we set

$$K = I(\delta\theta)\sin(k_3 \cdot \delta\theta)\frac{k_1 N(\underline{y})}{N(\underline{y})N(\underline{x})+k_2} \tag{3}$$

where $\delta\theta = ||\theta(\underline{x}) - \theta(\underline{y})||$

I is a step function which determines the maximum difference $\delta\theta$ in angle at which cells may influence each other's orientation.

J_θ is the flux of angle, that is, the orientation that the cells tend to carry with them as they move. We assume that cells move slowly so that at each time step the local effects have time to set in and completely determine the cells' orientation. Therefore, we make the approximation $J_\theta = 0$. The diffusive flux of the cells, J_N, is given by:

$$J_N = -[\vec{\theta} \cdot \nabla(ND)]\vec{\theta} \tag{4}$$

The diffusion D is a function of the cell density and the orientation variation ($\frac{\partial\theta}{\partial n}$):

$$D(\frac{\partial\theta}{\partial n}, N) = \frac{a}{N(\frac{\partial\theta}{\partial n})^2 + b} \tag{5}$$

Here, $\vec{\theta} = (cos(\theta), sin(\theta))$ is the unit vector n in the direction of orientation, and

$$\theta_n = \frac{\partial\theta}{\partial n} = \vec{\theta} \cdot \nabla\theta \tag{6}$$

is the rate of change of orientation in the direction of local orientation (the direction along which the cells are able to move).

2.1 Numerical Simulation of Fibroblasts

To calculate equation (3) for a meshpoint (i, j) we approximate the integral using Simpson's rule. Numerical simulations have shown that a reasonable choice for meshsize is $\Delta x = \Delta y = d$, where d is the radius within which a cell will influence a neighboring cell. With such a choice of Δx, the approximation to the integral is written as:

$$\int_{B(\epsilon)} K(\underline{x}, \underline{y}, \theta(\underline{x}, t), \theta(\underline{y}, t), N(\underline{x}, t), N(\underline{y}, t))\, dy \approx$$
$$\Delta x^2 \cdot [K(\theta^k_{i,j}, \theta^k_{i-1,j}, N^k_{i,j}, N^k_{i-1,j}) +$$
$$K(\theta^k_{i,j}, \theta^k_{i,j-1}, N^k_{i,j}, N^k_{i,j-1}) +$$
$$K(\theta^k_{i,j}, \theta^k_{i+1,j}, N^k_{i,j}, N^k_{i+1,j}) +$$
$$K(\theta^k_{i,j}, \theta^k_{i,j+1}, N^k_{i,j}, N^k_{i,j+1})]$$

with

$$K(\theta^k_{i,j}, \theta^k_{i-1,j}, N^k_{i,j}, N^k_{i-1,j}) =$$
$$I(\delta\theta_{ij,i-1j}) \sin(k_3 \cdot \delta\theta_{ij,i-1j}) \frac{k_1 N^k_{i-1,j}}{N^k_{i-1,j} N^k_{i,j} + k_2}$$

Here, $\delta\theta_{ij,i-1j} = \|\theta^k_{i,j} - \theta^k_{i-1,j}\|$. Similar definitions hold for the other three terms. We can thus approximate the integro-differential equation by a system of non-stiff ordinary differential equations for which the time derivative can be discretized using forward Euler:

$$\theta^{k+1}_{i,j} = \theta^k_{i,j} +$$
$$\Delta t \Delta x^2 \cdot [K(\theta^k_{i,j}, \theta^k_{i-1,j}, N^k_{i,j}, N^k_{i-1,j}) +$$
$$K(\theta^k_{i,j}, \theta^k_{i,j-1}, N^k_{i,j}, N^k_{i,j-1}) +$$
$$K(\theta^k_{i,j}, \theta^k_{i+1,j}, N^k_{i,j}, N^k_{i+1,j}) +$$
$$K(\theta^k_{i,j}, \theta^k_{i,j+1}, N^k_{i,j}, N^k_{i,j+1})]$$

The cell density was calculated using first order upwinding for the flux term $\vec{J} = (J_1, J_2)$:

$$N^*_{i,j} = N^k_{i,j} +$$
$$\frac{\Delta t}{\Delta x} \cdot (J_{1Ni+\frac{1}{2},j}(\theta^k_{i,j}, N^k_{i,j}) - J_{1Ni-\frac{1}{2},j}(\theta^k_{i,j}, N^k_{i,j}))$$
$$N^{k+1}_{i,j} = N^*_{i,j} +$$
$$\frac{\Delta t}{\Delta x} \cdot (J_{2Ni,j+\frac{1}{2}}(\theta^k_{i,j}, N^*_{i,j}) - J_{2Ni,j-\frac{1}{2}}(\theta^k_{i,j}, N^*_{i,j}))$$

where $J_{Ni+\frac{1}{2},j}$ is the value of J_N calculated upstream of the point $(i+\frac{1}{2}, j)$ along the x direction (similarly for the other three terms). On the boundary we assume no flux for the cells or their angles.

The numerical domain for our simulations consists of a square grid of size at least 40×40. Being explicit, the methods require $O(n^2)$ flops for each time step, where $n \times n$ is the number of points in the mesh. For our simulation, we set the distances between mesh-points to $\Delta x = \Delta y = 0.01$. For our parameter values, the system shows stable solutions if $\Delta t \leq 0.001$. We keep Δt constant throughout the calculation to produce a predictable number of iterations. The number of iterations determines the number of angle discontinuities within the dish. For $t > 100$ the cells become pretty much aligned throughout the dish. The results in this paper correspond to runs of $50,000$ iterations.

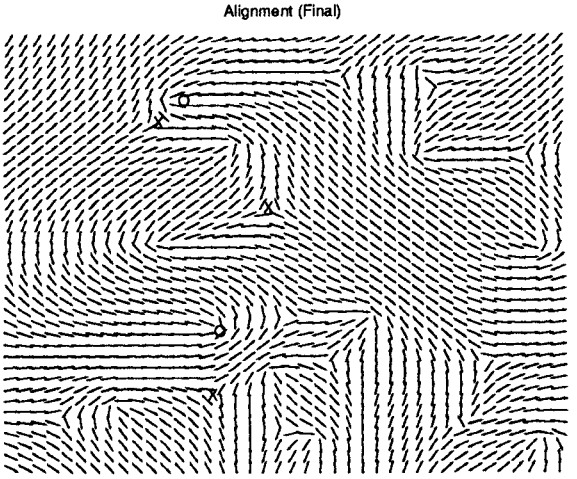

Figure 1: Fibroblast cell-orientation after 50,000 steps.

2.2 Numerical Simulation Results

In our parallel simulations, cell densities are initially uniform throughout the grid. Local cell orientations are assigned randomly at each meshpoint. After a short time (t=0.1) cells begin to align. Moreover, we notice small density variations which are due to the large angle variations that are still persistent at this stage. As time progresses, cells further align themselves, while the number of discontinuities decreases.

Figure 1 shows the cell orientations at the end of the simulation. On this plot, we have marked some discontinuities as arches and others as triradii. Triradii are 3-point-star patterns which appear at the confluence of three parallel arrays of different average orientation near the point of confluence. From Figure 2 we see that cell densities are uniform except at the points of discontinuity. Just below the arches, cell densities are lower, as observed in real cell cultures, with the cell density being higher on the arch (curve above the discontinuity). At the triradii, cell densities appear higher, too, due to the decreased diffusion predicted by our model, resulting in cell clumping. This clumping occurs in our model because we allow cell movement towards the discontinuity. In the actual cell cultures, cells will stay away from areas of large angle variations. Hence, they

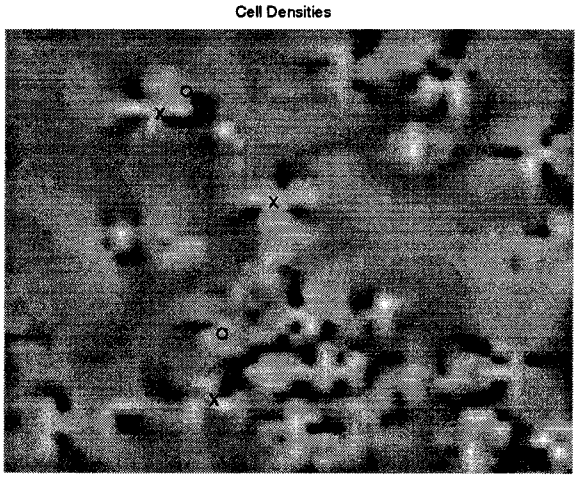

Figure 2: Fibroblast cell-density after 50,000 steps.

appear at lower densities at the center of a triradius. We are currently refining our model to address this problem.

Our numerical simulations of the model show patterns that closely resemble the patterns observed in culture [3]. The similarity between the biological and numerically found patterns provides strong indication that our model captures important characteristics of the cell interactions. Further biological experiments could provide us with accurate parameter values for our simulations.

3 Bacteria Application

3.1 Modeling Bacteria

Conditions have been found under which chemotactic strains of different bacterial species aggregate to form geometric patterns of different complexity. These patterns form when motile cells, inoculated on semisolid agar, respond to gradients of chemical attractants that they themselves excrete [5, 23]. These fascinating experimental patterns are a new class of kinetic patterns [22] in which both random motion (diffusion) and chemically directed movement (chemotaxis) are essential. The simplest patterns, and thus, those most amenable to mathematical analysis, are the periodic arrays of continuous or perforated rings generated by the bacterium *Salmonella typhimurium*. A reaction-diffusion-chemotaxis model based on the Oster-Murray mechanism [18] provides a description of the essential components required to generate simple periodic patterns. This mathematical model consists of a system of coupled nonlinear partial differential equations for the bacteria-chemoattractant- substrate interactions; however, such continuum models can be shown to be the result of using behavioral 'rules' in 'random walk' individuals [1, 4]. The model is based on the following known biological fact:

- *It is the interaction between the cells and chemoattractant that causes self-organization into the observed patterns.* In the semi-solid medium, the cells start out at the center of the dish, proliferate, and produce and degrade chemoattractant; they only sense the chemoattractant and not the substrate. The cells and chemoattractant are both diffusive.

We denote the density of (motile) cells by n, the concentration of chemoattractant by c, and the concentration of substrate by s. In the *S. typhimurium* experiments, the consumption of substrate is negligible. Therefore, we assume that the substrate concentration is constant, and thus s is just a parameter. It is convenient to cast the model in non-dimensional terms (Murray [17] provides a general discussion):

$$\frac{\partial}{\partial t}n = -\nabla \cdot J_n - \nabla \cdot J_{chemo} + f_1(n,c,s) \quad (7)$$

$$\frac{\partial}{\partial t}c = -\nabla \cdot J_c + f_2(n,c,s) - f_3(n,c,s) \quad (8)$$

where:

- J_n = -$D_n \nabla n$ and J_c = -$D_c \nabla c$ are the diffusive fluxes of the cells and chemoattractant, respectively.

- $J_{chemo} = \chi(n,c)\nabla c$ is the chemotactic flux. The functional form for the chemotactic response is the one determined experimentally, $\chi(n,c) = \alpha n/(1+\beta c)^2$, where α measures the strength of chemotaxis, and β the saturation of the chemotactic sensitivity (at high levels of chemoattractant).

- $f_1(n,c,s)$, $f_2(n,c,s)$, $f_3(n,c,s)$ are proliferation of cells, and production and degradation of chemoattractant. We choose the simplest forms possible which are consistent with the experimental observations:

$$f_1(n,c,s) = \rho n(1-\frac{n}{s}) \quad (9)$$

$$f_2(n,c,s) = \frac{sn}{1+\gamma n} \quad (10)$$

$$f_3(n,c,s) = -c \quad (11)$$

The parameter ρ measures the rate of proliferation, and γ the saturation of the chemoattractant production (at high levels of cells).

In the biological experiments, patterns are formed from an initial inoculum of cells at the center of a petri dish, with no cells elsewhere. These cells then diffuse and proliferate, spreading out radially. At the same

time, they produce chemoattractant which causes them to aggregate. It is the interplay between this aggregative destabilizing effect, reflected in the chemotactic coefficient (α), and the stabilizing effects of the diffusion of the cells (D_n) and of the chemoattractant (D_c), that is crucial to the formation of spatially heterogeneous pattern.

3.2 Numerical Solution of the Bacteria Model

In our numerical investigations of the behavior of the model, we focus on cell density (rather than chemoattractant (or substrate) concentration) as this is the quantity of primary interest. Moreover, since there is no variation in the patterns through the thickness of the agar, equations (10)-(11) for the cell density and chemoattractant concentration can be solved on a two-dimensional grid. To develop and analyze the model equations we use an explicit forward Euler finite difference scheme. The experimental patterns are generated from an initial inoculum at the center of a dish, so we choose as our initial conditions a 10 $\times$ 10 mesh point area at the center of the domain in which cell density $n = 1.0$, with $n = 0.0$ elsewhere. Initially $c = 0$ everywhere. These conditions are then perturbed with $\pm$ 1% random noise in order to break the symmetry. Experimentally the cells are confined to a dish, so we impose zero flux boundary conditions. However, pattern formation (in both the experiments and the simulations) takes place well before the leading edge of the perturbation reaches the domain boundary, so the boundary conditions are not relevant, nor is the shape of the mesh, which is a square containing at least 301 x 301 grid points. The time step used in the integrations is 0.005. The reliability of this numerical method was monitored by doubling the mesh size and halving the time step, which produced qualitatively similar results. We have also tested an implicit Crank-Nicholson method which was successfully used by Scribner *et al.* [20] to investigate traveling bands of bacteria, and the method of lines to reduce the partial differential equations to ordinary differential equations. However, because of the stiffness of the model equations, there is little or no gain in using these implicit methods.

To examine explicitly the role of the chemotaxis coefficient, α, in the self-organization of bacteria, we fix $D_n = 0.1$, and $D_c = 0.3$, together with $\beta = 1.0$, $\gamma = 0.2$, $\rho = 0.03$, and $s = 1.0$. With these parameters, linear analysis shows that the non-trivial uniform steady state $(n,c) = (1,1/1.2)$ can be driven unstable by spatial perturbations when $\alpha > \alpha_{crit} = 0.818$. This is the usual way that spatial patterns are generated in most models for biological pattern formation [17]. For a given initial condition, rings are most likely to form when the chemotactic response (α) is low (but above the critical value for patterns to propagate). This also corresponds in the dimensional problem to slow production or rapid consumption of chemoattractant (or, when s is non- constant, to rapid consumption of the substrate). However, spots will be more likely than rings when α is sufficiently large that gradients in chemotactic concentration do not need to be so steep for the recruitment of cells into clusters.

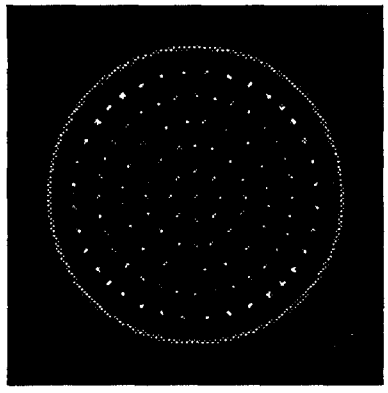

Figure 3: Patterns of bacteria cell-density after 300,000 time steps.

This is illustrated, for $\alpha = 3.0$, in Figure 3 which shows how, after 300,000 time steps, the pattern has spread sequentially outward, with rings forming at a fixed distance from one another, and then breaking up into spots. For even larger values of α (more precisely, for large values of the ratio α/D_n) the numerical calculations ultimately fail because the peaks in the solutions became very sharp and steep. This is more likely to be the limitation of the step size in our numerical scheme rather than chemotactic collapse [7] since the cell removal term, $-\rho n^2$, prevents formation of singularities.

We therefore suggest that the cell-chemoattractant mechanism (10)-(11) is a likely candidate for generating simple periodic patterns found in the *S. typhimurium* experiments. Moreover, by choosing as our bifurcation parameter the chemotactic coefficient, α, we have confirmed the biological observation that it is the sensitivity in chemotactic response that affects the nature of the pattern. Detailed numerical simulations and bifurcation pattern sequences, as well as non-dimensionalized (biological) parameter values, are reported elsewhere [23].

4 The ZPL Array Sublanguage

ZPL is a data parallel language that allows arrays and subarrays to be manipulated as whole entities [15]. The language provides constructs that lead to concise programs while eliminating tedious and error prone array indexing. The language's conciseness was first illus-

trated by the SIMPLE computational fluid dynamics benchmark, which is approximately 5000 lines when coded in C plus message passing, but only 500 lines and considerably more readable when coded in ZPL [16]. In addition, ZPL programs have sequential semantics and control flow, which considerably simplifies the program development and debugging process.

A chief goal of ZPL is to provide portability and efficiency across diverse parallel computers. Our current ZPL compiler produces machine independent ANSI C as object code, which is then compiled on the target machine and linked with machine-specific libraries to produce executable code. Performance matching hand-coded, explicitly parallel C code has been demonstrated on both shared and nonshared memory parallel computers [16]. The mathematical biology applications described here provide further evidence that compiled ZPL code is of high quality.

We now briefly describe the main ZPL constructs and data types. Some of the more powerful features of the language, which were not needed in these two applications, are omitted here but described elsewhere [15, 21]. ZPL supports a typical set of data types (e.g. `real`, `integer`, `char`), dense arrays of arbitrary dimension, the usual arithmetic and logical operators which can be applied to either scalars or arrays, parallel prefix operators (e.g. reduce and scan), and the standard control structures (`if`, `for`, `while`, *etc.*), including recursion. ZPL has two classes of variables that serve as basic units of computation: *parallel arrays* and scalars. In the ZPL model, concurrency is derived from the array and parallel-prefix operators (reductions and scans). Specifically, ZPL's parallel arrays are distributed across physical processors, while all scalar data is replicated. (ZPL also has *indexed arrays*, which are not distributed and do not produce parallelism.) All concurrency is managed by the compiler and run-time system, shielding the application programmer from the tedious details of communication and synchronization.

The ZPL code fragment below is characteristic of both the fibroblast and bacteria codes and illustrates some distinguishing features of ZPL. First, *regions* are central to ZPL. Line 5 shows how regions, which are simply sets of array indices, can be declared. In this case `R` is a square 2D index set. (Regions need not be statically declared, but *dynamic regions* are beyond the scope of this discussion.) Line 11 shows how a region can be used to define the storage of *parallel* arrays, in this case `U` and `V`, whose base-types are double-precision floating point values. Lines 17 and 18 show how regions can be used to specify new regions: `of` is a keyword and `South` is a user-defined *direction* which together define a new region that is disjoint and adjacent to `R`, and offset from `R` by the vector `South`. That is, `[South of R]` is the $(n+1)^{st}$ row of the index space. Finally, line 22 shows how regions are used to specify

Application	Fibroblasts	Bacteria
Fortran	461 lines	255 lines
ZPL	224 lines	197 lines

Table 1: Code sizes for the kernels of the applications under investigation.

array operations: the region `R` specifies the index set over which the statement is to be applied. Thus, line 22 will assign to all elements of the array `Temp` whose indices are in the range `(1..n)` × `(1..n)`. Line 22 also illustrates the use of the At operator (`@`). For example, `U@South` refers to the parallel array whose indices are displaced from `[R]` by the vector `South`. Thus, the `@` operator can be viewed as a "shift" operator.

```
 1  program Bacteria;
 2                        -- Declarations --
 3
 4  constant          beta : integer = 2.0;
 5  region            R = [1..n, 1..n];
 6
 7  direction         North   = [-1, 0];
 8                    South   = [+1, 0];
 9
10  procedure Bacteria();
11  var               U, V: [R] double;
12                    Temp: [R] double;
13
14                      -- Initialization --
15  begin
16    [R]             U := 0.0;
17    [South of R]
18    [North of R]    U := 0.0;
19
20    . . .
21
22    [R]    Temp := (U@South/((1 + beta*V@South) *
23                             (1 + beta*V@South)));
24  end;
```

5 Implementation Issues

Implementing the mathematical biology applications with ZPL proved to be easy, even for users with limited parallel-programming experience. The algorithms described in Sections 2 and 3 are both inherently data parallel. In the fibroblast simulation the cell orientations and densities are determined by information from neighboring cells. Similarly, the bacteria formation patterns are caused by diffusion in which cell densities for one time step can be computed from localized information of the previous time step. We found the ZPL programs for the Fibroblast and the Bacteria models to be more readable (and shorter—see Table 1) than the original *sequential* Fortran programs. Implementing the same applications with explicit par-

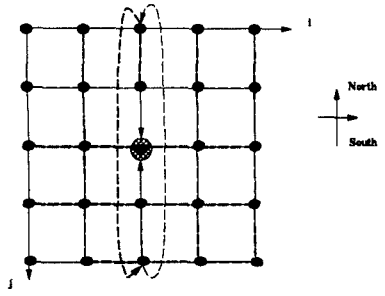

Figure 4: Data-dependences and boundary conditions for the Fibroblast example.

allelism using either message-passing or shared memory constructs would have been much more tedious. The conciseness of ZPL is attributed mainly to:

1. The implicit handling of data decomposition, communication and synchronization;
2. The provision of specific support for common types of boundary conditions;
3. The provision of high-level constructs that are known to have efficient parallel implementations, such as array operations and *Reductions* and *Scans*.

As an example, we present a short Fortran excerpt from the Fibroblast application that initializes periodic boundary conditions on the two-dimensional array of cell-angles and then calculates angle derivatives along the y-axis:

```
do 78 j=1,n+2
   theta(1,j) = theta(n+2,j)
   theta(n+2,j) = theta(2,j)
78 continue

do 11 i=2, n+1
   do 12 j=2, n+1
      thy(i,j) =
      angle_subtr(theta(i-1,j), theta(i+1,j))
                                        /(2.0d0*h)
12 continue
11 continue
```

A pictorial description of the corresponding data-dependencies and boundary conditions is given in Figure 4. The ZPL implementation includes definitions for the region of the parallel arrays upon which the computation is being carried out, and for the *directions* of data-flow:

```
region      R = [1..n,1..n];
direction   North = [-1,0];
            South = [+1,0];
var
         /* declare parallel arrays */
            theta, thy : [R] double;
         /* declare of h as scalar  */
            h: double;
```

The actual code consists of a high-level implementation of the periodic boundary conditions and the sequential semantics of the Fortran code, applied to *parallel* arrays:

```
[North of R]
[South of R] wrap theta;

[R]     thy :=
        angle_subtr(theta@South, theta@North)
                                        /(2*h);
```

The last line illustrates another convenient feature of ZPL: *scalar promotion*. The `angle_subtr()` function was written to accept scalar parameters but here is passed `theta@South` and `theta@North` as parameters. This type of promotion can be significant in allowing code re-use from sequential programs, and also illustrates how parallel arrays in ZPL have been elevated to the same status as the scalar data types. (The scalar expression `(2*h)` is also trivial promoted to an array expression of the same size and shape as the other arrays in the statement.)

6 Performance Results

To study the efficiency of the ZPL codes we performed a large number of runs on three different platforms: the Kendall Square Research KSR-2 [6], the Intel Paragon [9], and DEC-Alpha workstations running PVM [13]. The KSR-2 is a COMA (Cache Only Memory Architecture) multiprocessor with 40MHz custom processors configured as a hierarchy of slotted packetized rings. Each leaf-level ring contains 32 processors. The KSR architecture provides a shared address space with physically distributed memory. Memory modules of each node play the role of a very large hardware-managed cache. Cache coherence is provided through a hierarchical directory scheme which enforces sequential consistency. The Intel Paragon is a message-passing system based on the Intel i860XP, a 50MHz microprocessor. Communication between the processors is carried through a mesh interconnection network. PVM is a message passing interface for distributed computing that uses the TCP/IP communication protocols to connect, in our case, a networks of workstations. The high availability of workstations makes this an attractive development environment. Our workstations are DEC 3000/400 AXP's with 133MHz clocks.

Our study focuses on two measures of performance: a comparison of the execution times for the ZPL codes and their corresponding sequential Fortran programs, and an assessment of the scalability of the ZPL codes as the numbers of processors is increased. The first

	Fibroblasts		
	KSR	*Intel Paragon*	*DEC*
Fortran	-	2114.75	-
ZPL	3159.7	2914.17	1675.54
	Bacteria		
	KSR	*Intel Paragon*	*DEC*
Fortran	46919.16	20198.88	-
ZPL	47519.79	18946.03	9754.75

Table 2: Comparison of Fortran and ZPL execution times on one processor.

metric provides a measure of the sequential efficiency of the ZPL compiler. We would expect the code generated by the ZPL compiler to be less efficient than the hand-written Fortran code because of the extra bookkeeping that is introduced to exploit parallelism. This extra overhead, however, must not be large if this parallel solution is to represent an alternative to sequential implementations.

Table 2 presents the running times of the Fibroblast and Bacteria codes running on *one* processor of the KSR-2, Intel Paragon, and DEC-Alpha architectures. The ZPL codes on the DEC-Alpha workstation run under PVM[1]. The running times reported for the Fibroblast application correspond to a mesh size of 100×100 and 2000 time-steps. The running times reported for the Bacteria application correspond to runs on a 481×481 mesh and 2000 time-steps. Table 2 shows that the ZPL codes running on one processor achieve performance that is competitive with that of the hand-written Fortran programs.

Figures 5 and 6 show the running times for the same instances of the Fibroblast and the Bacteria applications on the KSR-2 and Intel Paragon multiprocessors. From these plots we can see that for both applications the ZPL codes scale very well with the number of available processors. This can also be seen from the relative speedup curves shown in Figure 7. Here, speedup is defined as the ratio of the running time on a given number of processors over the running time of *the same code* on one processor of the same machine. From the plots in Figure 7 we see that the speedups achieved with the Fibroblast code are smaller than those for the Bacteria code. To further investigate this and to learn more about the behavior of the codes, we used the performance monitoring environment on the Intel Paragon [19] to instrument the C code produced by the ZPL compiler. In particular, we used *Xipd* [10] to instrument the program and collect traces, and *ParaGraph* [12] to visualize performance. We produced detailed diagrams showing parallelism profiles, processor uti-

[1]We have no measurements of Fortran codes for the DEC-Alpha because we have no Fortran compiler for this machine.

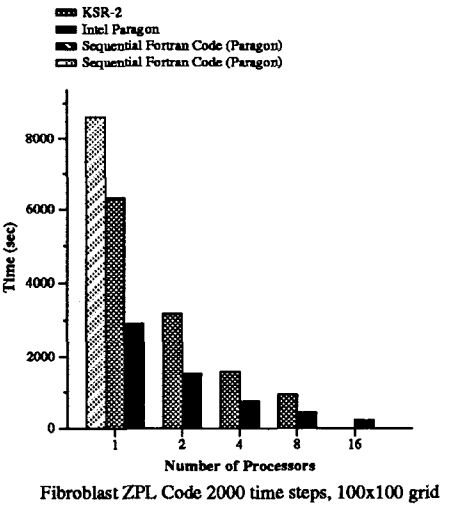

Figure 5: Running Times of the Fibroblast application (100x100 grid, 2000 iterations).

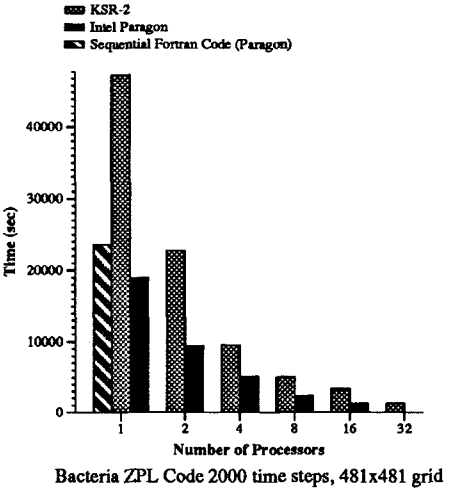

Figure 6: Running Times of the Bacteria application (481x481 grid, 2000 iterations).

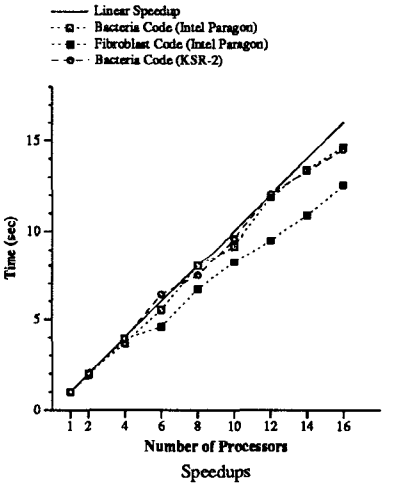

Figure 7: Relative Speedup.

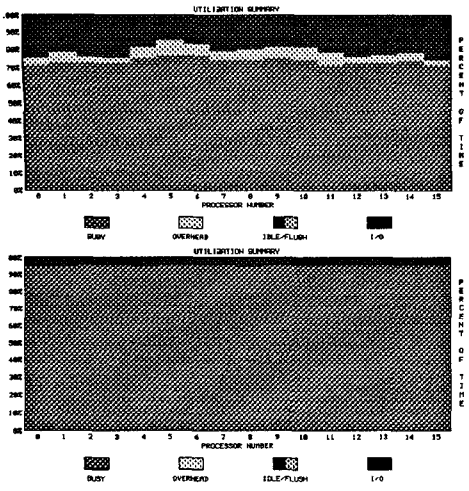

Figure 8: Average utilization of processors for the Fibroblast (top) and the Bacteria codes (bottom) running on Intel Paragon. The dark-grey color denotes the percentage of running time that a processor spends waiting for communication.

lization, communication overhead, communication profiles, and patterns of communication. These profiles confirmed that the lower speedups in the case of the Fibroblast application are due to the higher communication overhead incurred by the Fibroblast runs. For example, Figure 8 illustrates this overhead in terms of average processor utilization during a small number of iterations running on a 16-processor partition of Intel's Paragon. This instance of the Fibroblast application exchanges 23,870 messages (total of 4,138,240 bytes) within a 3.59-second monitoring window, whereas the Bacteria code exchanges only 2,844 messages (total of 2,693,600 bytes) within a 14.33-second monitoring window.

7 Conclusions

In this paper we have shown that mathematical models of pattern formation in biology are ideally suited to large scale parallel programming. We have presented new algorithms for solving two mathematical biology problems, one in the study of the formation of fibroblast structures, the other in the study of the self-organization of motile bacteria. By focusing on these applications where detailed experimental data are available, we have been able to develop basic mechanisms for pattern formation which should serve as important paradigms in more complex developmental and behavioral systems. Furthermore, we have shown that the ZPL programming language provides an ideal means of expressing these mathematical models, both from a performance aspect and an ease-of-use aspect. The performance of the two codes on the KSR and Intel Paragon were found to be competitive with sequential Fortran programs.

8 Acknowledgments

It is a pleasure to thank Brad Chamberlain and Yi Sun for their help with ZPL. Julian Cook first suggested the problem of modeling Fibroblasts and developed a preliminary model with D. Manoussaki. His contribution is gratefully acknowledged. D.E. Woodward would also like to thank the Department of Mathematics, University of Washington, for its kind hospitality during her visit. This work was supported partly by an HPCC/ESS grant from NASA, a grant from the University of Washington Zoology Dept., and ARPA Grant N00014-92-J-1824.

References

[1] W. Alt. Biased random walk models for chemotaxis and related diffusion approximations. *J. Math. Biol.*, 9:147–177, 1980.

[2] Gail Alverson, William Griswold, Calvin Lin, David Notkin, and Lawrence Snyder. Abstractions for portable, scalable parallel programming. Technical Report 93–12–09, Department of Computer Science and Engineering, University of Washington, submitted to *IEEE Trans. on Parallel and Distributed Systems*, 1993.

[3] Jonathan Bard. *Morphogenesis.* Cambridge University Press, 1991.

[4] H.C. Berg. *Random walks in Biology.* Princeton University Press, 1983 [Expanded edition, 1993].

[5] E.O. Budrene and H.C. Berg. Complex patterns formed by motile cells of Escherichia coli. *Nature*, 349:630–633, 1991.

[6] Henry Burkhardt. Overview of the KSR1 computer system. Technical Report KSR-TR-9202001, Kendall Square Research, February 1992.

[7] S. Childress and J.K. Perkus. Nonlinear aspects of chemotaxis. *Math. Biosci.*, 56:217–237, 1981.

[8] J. Cook and D. Manoussaki. Continuum Model for Fibroblast Pattern Formation: Some Preliminary Ideas. Technical report, Dept. of Applied Mathematics, University of Washington, 1994.

[9] Intel Corporation. Paragon XP/S product overview. Technical report, 1992.

[10] Intel Corporation. *Paragon's Application Tools User's Guide*, June 1994.

[11] Tom Elsdale and Frances Wasoff. Fibroblast Culture and Dermatoglyphics: The topology of two planar patterns. *Roux's Archives of Developmental Biology 180*, pages 121–147, 1976.

[12] M.T. Heath and J.A. Etheridge. Visualizing the Performance of Parallel Programs. *IEEE Software*, 8(5):29–39, September 1991.

[13] Oak Ridge National Laboratory. *PVM 3 User's Guide and Reference Manual*, May 1993.

[14] Calvin Lin and Lawrence Snyder. A portable implementation of SIMPLE. *International Journal of Parallel Programming*, 20(5):363–401, 1991.

[15] Calvin Lin and Lawrence Snyder. ZPL: An array sublanguage. In Uptal Banerjee, David Gelernter, Alexandru Nicolau, and David Padua, editors, *Languages and Compilers for Parallel Computing*, pages 96–114. Springer-Verlag, 1993.

[16] Calvin Lin and Lawrence Snyder. SIMPLE performance results in ZPL. In *7^{th} Workshop on Languages and Compilers for Parallel Computing*, 1994.

[17] J.D. Murray. *Mathematical Biology.* Springer-Verlag, 1989 [2nd corrected edition 1993].

[18] G.F. Oster and J.D. Murray. Pattern formation models and developmental constraints. *J. Exp. Zool.*, 251:186–202, 1989.

[19] B. Ries, R. Anderson, W. Auld, K. Callaghan, E. Richards, and W. Smith. The Paragon Performance Monitoring Environment. In *Supercomputing '94*, pages 850–859, 1994.

[20] T.L. Scribner L.A. Segel and E.H. Rodgers. A numerical study of the formation and propagation of traveling bands of chemotactic bacteria. *J. theor. Biol.*, 56:217–237, 1974.

[21] Lawrence Snyder. A ZPL programming guide. Technical Report 94-12-02, Department of Computer Science and Engineering, University of Washington, 1994.

[22] J.W.T. Wimpenny. Microbial systems: patterns in space and time. *Adv. Micro. Ecol.*, 12:469–522, 1992.

[23] D.E. Woodward, R. Tyson, M.R. Myerscough, J.D. Murray, E.O. Budrene, and H.C. Berg. Spatio-temporal patterns generated by Salmonella typhimurium. *Biophysical J.*, 1994 (submitted).