To the Graduate Council:

I am submitting herewith a thesis written by Derek Evan Prowe entitled "The Effect of Reproduction of Algal Cells in Aggregates and Computational Efficiency in an Aggregation Model." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Thomas Hallam, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor and
Dean of the Graduate School

The Effect of Reproduction of Algal Cells in Aggregates and Computational Efficiency in an Aggregation Model

A Thesis Presented in Partial Fulfillment of

the Master of Science Degree Requirements

The University of Tennessee, Knoxville

Derek Evan Prowe

August 1995

ACKNOWLEDGMENTS

I wish to thank the members of my thesis committee, Professors Thomas Hallam, Jerzy Dydak, and Lou Gross, for their time, energy, and concern. My current advisor, Professor Hallam, has my appreciation for committing to this project without any real knowledge about me from experience; it was mostly a matter of faith and willingness to guide. He has since given me as much and as little support, guidance, and direction as I have needed through the course of this project. My past advisor Professor Dydak has my appreciation for the abundance of encouragement, perspective, and insight he has given me in mathematics and in teaching.

Azmy Ackleh, a former student of my advisor who currently holds a post-doctoral position elsewhere, helped a great deal with this project. He explained, guided, suggested, and questioned very effectively all through the development of this project. I couldn't have done it without him. I would also like to thank the Department of Mathematics at large, and the people who make it what it is, including Pam Armentrout and Assistant Head Sam Jordan. The department has treated me well during the past two years.

This study was supported by Office of Naval Research Grant N00014-93-1-1174.

ABSTRACT

The clustering, or aggregation, of algal cells plays an important role in the dynamics of oceanic algal populations. A model of the algal aggregation process was presented by Ackleh, Hallam, and Smith (1994) for a specific phytoplankton species, diatoms. The model represents algal aggregation and several related factors including nitrogen levels, a predator population (zooplankton), and settling or sedimentation of aggregates. The purpose of the current study is twofold: 1) to decrease the time required for a simulation run, and 2) to create a more realistic, flexible mechanism for handling cell reproduction in which aggregates may increase in size due to reproduction of their constituent cells. It was found that these objectives necessitated other minor modifications in the modeling methodology.

TABLE OF CONTENTS

Chapter 1.
Introduction

The situation considered in this study is centered around the clustering or "aggregation" of algal cells in the ocean. A useful image of the aggregation process is to imagine standing on the bottom of the shallow portion of the ocean looking up, during which a phenomenon known as *marine snow* might be witnessed. Small clusters of organic material, consisting largely of algal cells but also including fecal pellets and other debris, descend from the life-supporting surface layer of the ocean. Algae on the surface layer of the ocean are likely to die for one of two reasons. One major cause of mortality is sinking, because algae depend on photosynthesis for their life processes and there is insufficient light to support them below a certain depth. Algae are more likely to sink if tied up in an aggregate. A second major cause of mortality for algae is being eaten by predators known as zooplankton. Algal cells tend to be less vulnerable to predation when in aggregates.

Algal aggregation dynamics are of interest to biologists on at least two levels. Because algae are important populations in the oceans in their own right, biologists are interested in knowing which cause of mortality dominates, if either. There is also a larger picture, and that is the global carbon cycle. Ordinarily, organic material is recycled: when an organism

dies, it is eaten or decomposes to provide nutrition for other organisms. However, phytoplankton cells that settle to the bottom of the ocean stay there, leaving the global carbon cycle until they are recycled. The settling or sedimentation of algal aggregates may be one of the major global carbon sinks, which is a second reason why the dynamics of algal aggregation are of interest.

An experiment was performed in 1993 at Santa Barbara known as the SIGMA Tank Experiment (Alldredge et al., 1994). In this tank, the clustering of a particular algal species, *diatoms*, was tracked over time. As this experiment is an important and useful source of data, the conditions of the tank study drive many of the decisions and assumptions made for the model discussed here.

1.1. The Biology

There are elements included in the algal aggregate model besides the algae itself. The model must consider a zooplankton population, the predators in the system. The nutrient source available to the algae, nitrogen in this case, must be tracked. The algae are modeled on two levels: at the individual level and as aggregates. While all individual cells are assumed to have the same physiological characteristics in the model, different age or maturity cohorts are followed. On another level, the model includes algae as the building blocks for aggregates.

Figure 1 depicts the elements of the model and the interactions between them schematically. The densities of nutrients, zooplankton, singlephytoplankton cells, and phytoplankton aggregates of various sizes are tracked over time. Nitrogen in the water starts at a certain concentration and is reduced over the simulation time through consumption by both free algal cells and algal cells tied up in aggregates. Our model contains no source of additional nitrogen during the simulation; the entire nutrient supply originates in an initial "pulse." This condition was chosen to mimic the conditions in the SIGMA Tank Experiment described above.

The single cell population is influenced by the availability of nutrients for growth and reproduction. The density of these "free" phytoplankton cells is decreased by the process of coagulation, whereby cells move into the class of aggregates. It is positively influenced by the breakup of aggregates, which can produce single cells, and by reproduction, both of free cells and of cells in aggregates--some of the daughter cells produced by cells in aggregates "fall off" to become free cells. Phytoplankton cells are also lost as they sink below the surface layer of the ocean, where there is insufficient light. The last single cell sink is consumption by zooplankton.

Algal aggregates are subject to many of the same factors as are single cells. The density of algal aggregates is reduced by grazing of zooplankton, though larger aggregates are less prone to consumption (depending upon the species of zooplankton present). On the other hand, aggregates are

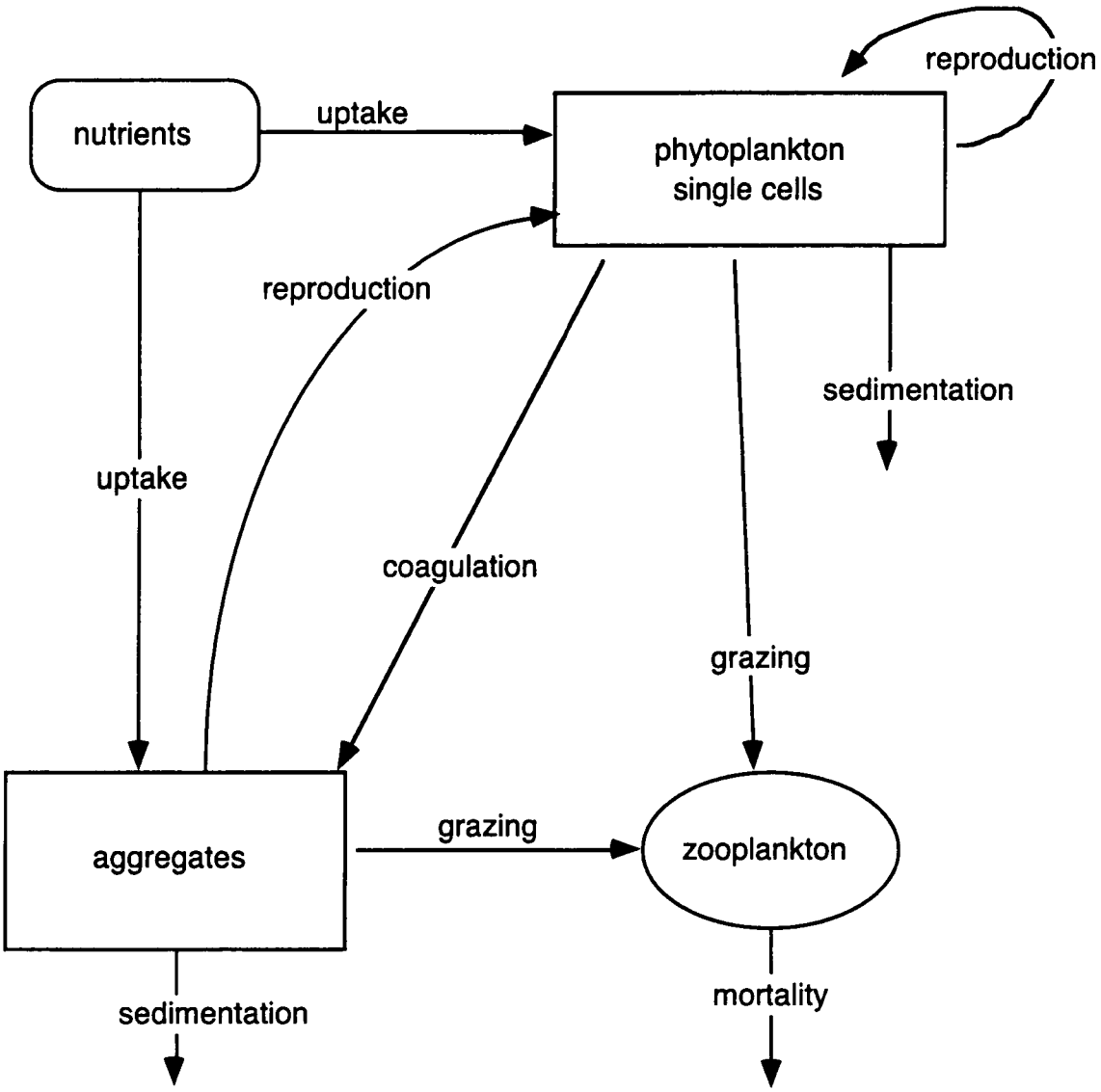

Figure 1. The Algal Aggregation Model

more vulnerable to sinking and sedimentation. The density of aggregates is increased by the process of coagulation of single cells and reduced when single cells result from aggregate breakup.

The zooplankton population model is unstructured. While a structured zooplankton model would be more biologically accurate, at this stage an unstructured model is considered an acceptable approximation. The zooplankton population density is positively influenced by the availability of phytoplankton for nourishment, both in the form of single cells and in the form of aggregates. The zooplankton population is controlled by density-dependent mortality.

Finally, there are several forces operating within the "aggregates" box itself to affect the size distribution of aggregates. Clustering between aggregates from one cell (4000 μm^3) up to a volume of 1 mm^3 tends to push the size distribution towards the larger end of the spectrum. Breaking up or *disaggregation* of aggregates tends to have the opposite effect. Some of the cells in aggregates are alive and reproducing: while some of their offspring fall off the aggregates to become free cells, (in the model as revised in this study) some remain to make individual aggregates larger.

This completes a broad overview of the physical and biological elements of the model and the forces operating on them over time.

1.2. The Components of the Model

To be more specific about the model, an additional assumption needs to be made. A specific species of phytoplankton must be selected; we use the *diatom* species. Two characteristics of diatoms are important for this study: diatom cells form a carapace or shell known as a *siliceous test.* The test has a fixed volume during the lifetime of the cell, so the external size of the cell remains unchanged; however, the carbon content of the cell increases as it matures. The other important characteristic of diatoms is that, due to their physical configuration, when cells join together they form chains. Thus, the aggregates studied here are roughly one-dimensional clusters.

1.2.1. Individual Cells

In this model, the state of an individual phytoplankton cell is characterized by its carbon content $C(t)$, where t represents time. The volume v of a diatom's siliceous test does not change during its lifetime, though its biomass does. This is the model for individual cell growth:

$$\frac{dC}{dt} = \frac{N}{k_N + N}(av^{2/3} - bC).$$

The rate of all cell activity is a function of the availability of nutrient $N(t)$, which explains the factor $\frac{N}{k_N + N}$; k_N is the nitrogen half saturation constant. Growth is constant, in proportion to the cell's (fixed) surface area with constant of proportionality a--hence the first term, $av^{2/3}$. The second term $-bC$ represents cell maintenance costs, which are proportional to the cell's carbon content with constant of proportionality

b. The values, units, and sources for a, b, k_N, and other constants used in the model are presented in Table 1.

According to our characterization of a diatom's life cycle, during its lifetime its organic material or carbon content doubles. When the carbon content reaches twice that of a newborn cell, the cell divides into two cells, each of which starts life with half the carbon content of its "parent."

1.2.2. An Aggregation Model

The first influence on the size distribution of algal aggregates that we consider is the process of clustering, both between individual cells and between aggregates. Ackleh, Hallam, and Smith (1994) use the following equation to describe this effect:

$$\frac{\partial \rho(t,x)}{\partial t} = \frac{1}{2}\int_0^x \beta(x-y,y)\rho(t,x-y)\rho(t,y)dy \qquad (1)$$
$$-\rho(t,x)\int_0^A \beta(x,y)\rho(t,y)dy$$

where $\rho(t,x)$ is the density of aggregates of size x at time t and A is the maximum aggregate size included in the model. Here the first term on the right-hand side represents aggregates gained at size x from encounters between two smaller aggregates. The second term represents a loss in density of aggregates at size x due to encounters with other aggregates, forming a larger aggregate. The function $\beta(x,y)$ incorporates two processes: the physical factors determining the probability that particles of size x and y will collide, and the "stickiness" of these algal aggregates. The representation of the former process is adapted from studies of other aggregation processes--further details are available in Ackleh, Hallam,

Table 1. Parameter Values and Units

Symbol	Interpretation	Value (Unit)	Reference
v	volume of cell	4000 (μm^3)	
x	volume of aggregate	(mm^3)	
t	time	(day)	
N	nitrate concentration	(μM)	
N_0	initial nitrate concentration	20 (μM)	
k_N	half saturation	0.5 (μM)	Scheffer 1991 Jackson and Lochmann 1992
a	cell growth rate	($\mu m\, \mu m\, m^{-2/3}\, d^{-1}$	derived in Ackleh, Hallam and Smith 1994
b	cell maintenance rate	0.0084 (d^{-1})	Lancelot et al. 1991
v_{min}	initial carbon mass of cell	208.4 (pg)	
$\rho(t,x)$	density of aggregates	(# $mm^{-3}\, l^{-1}$)	
$\beta(x,y)$	stickiness times coagulation kernel	0.1 ($L\, \#^{-1}\, t^{-1}$)	
(none)	portion of active cells in aggregates	0.3	
q	portion of new cells sticking to aggregate	0.3	
D	depth of surface layer	50 (m)	
Z_0	initial density of zooplankton	.00003 (# mm^{-3})	
Z	density of zooplankton	(# mm^{-3})	
p	zooplankton ingestion rate	10^4 (cells d^{-1} zooplankter^{-1})	Raymont 1983
δ	zooplankton assimilation efficiency	0.6	Raymont 1983
m	zooplankton mortality rate	0.105 (d^{-1})	Lancelot et al. 1991
$f(x)$	grazing probability		
x_{max}	maximum aggregate size ingested by zooplankton	300 (μm diameter)	Lancelot et al. 1991
k_ρ	algal half saturation constant for feeding	0.12 (# mm^{-3})	Steele and Henderson 1981 Frost 1987

and Smith (1994). "Stickiness" refers to the probability that two particles will adhere to one another if they collide. In the case of algal aggregation, there is no direct experimental value available for "stickiness" from the literature. In one study, Ackleh, Hallam, and Muller-Landau (1995) utilize parameter estimation techniques in conjunction with the SIGMA tank data to obtain an estimate of the stickiness constant.

Equation (1) describes an essential dynamic affecting ρ, but there are other factors which come into play: disaggregation or breaking up of aggregates, the effect of predation by zooplankton, and sinking or sedimentation. These elements will be included in a comprehensive version of (1) below.

1.2.3. Zooplankton

Zooplankton can eat both individual diatom cells and algal aggregates. An unstructured model is used to model zooplankton population dynamics:

$$\frac{dZ}{dt} = p\delta \frac{\int_0^A f(x)\rho(t,x)dx}{\int_0^A f(x)\rho(t,x)dx + k_\rho} Z - mZ^2 \qquad (2)$$

Here, Z(t) is the density of zooplankton while f(x) is the probability density function for an individual zooplankter eating an aggregate of various sizes. Thus, the integral $\int_0^A f(x)\rho(t,x)dx$ represents, in some sense, the effective amount of food available to these grazers.

It is assumed that there is an aggregate size above which zooplankton will not ingest aggregates, so $f(x)$ is defined to be 0 for $x > x_{max}$; we take

$x_{max} = 0.04\, mm^3$. In equation (2), p represents an average zooplankter's ingestion rate while δ is its assimilation efficiency, m is a density-dependent mortality coefficient, and k_ρ is the trophic response half saturation constant.

1.2.4. A More Complete Aggregate Growth Model

A partial differential equation for aggregate dynamics containing representations for the processes of aggregation, disaggregation, sinking, and grazing is:

$$\begin{aligned}\frac{\partial \rho(t,x)}{\partial t} = & \frac{1}{2}\int_0^x \beta(x-y,y)\rho(t,x-y)\rho(t,y)dy \\ & -\rho(t,x)\int_0^A \beta(x,y)\rho(t,y)dy \\ & -\frac{1}{2}\int_0^x \gamma(x,y)\rho(t,y)dy \\ & +\int_0^{A-x} \gamma(x,y)\rho(t,x+y)dy \\ & -\rho(t,x)\frac{w(x)}{D} \\ & -p\delta\frac{f(x)\rho(t,x)}{\int_0^A f(y)\rho(t,y)dy + k_\rho}Z\end{aligned} \tag{3}$$

The fifth term on the right-hand side represents sinking or sedimentation: $w(x)$ is the settling velocity of an aggregate of size x and D is the depth of the surface layer. Thus, an aggregate is assumed to die and leave the system once it drops below a depth of D. $w(x)$ is an increasing function in x, as larger aggregates sink faster. The sixth term represents loss of aggregates to zooplankton grazing. The elements of this term were introduced above.

The third and fourth terms represent aggregate fragmentation, as introduced by Ackleh (1995). The third term represents loss of aggregates of size x due to breakup into smaller aggregates while the fourth represents gains at size x due to breakup of larger aggregates.

This completes the overview of the mathematical model employed.

1.3. Numerical Methods and Issues

To analyze the representation of the aggregation process, the set of ordinary and partial differential equations which constitute the model must be solved. Previously, a linear spline-based collocation method (Gelbard and Seinfeld, 1978) was used to reduce the system to a set of ordinary differential equations. In the course of addressing the issues of this study, it was found that a different procedure is more appropriate, a *step function-based* collocation method. In either case, the objective is to approximate the partial differential equation describing the aggregate dynamics (3) by a set of ordinary differential equations with respect to time. Then, a package obtained from NETLIB called LSODE (August 13, 1981 version; Hindmarsh, 1980) is used to solve the resulting set of ordinary differential equations.

In the linear spline-based collocation method, $\rho(t,x)$ is approximated by a linear spline, which may be written in the form:

$$\rho'(t,x) = \sum_{k=1}^{N} \alpha_k(t)\pi_k(x)$$

where the π_k are the linear spline basis functions and N is the number of points at which collocation will occur.

A special property of the basis functions π_k is that, if $\{x_1,\ldots,x_N\}$ are the collocation points, $\pi_k(x_k)=1$ but $\pi_k(x_j)=0$ for each $j\neq k$. This set of collocation points $\{x_1,\ldots,x_N\}$ is referred to as the *grid*. Thus, if we substitute ρ' for ρ in (3), we get:

$$\begin{aligned}\frac{\partial\rho'(t,x)}{\partial t} &= \frac{1}{2}\int_0^x \beta(x-y,y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(x-y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(y)dy \\ &\quad -\sum_{k=1}^{N}\alpha_k(t)\pi_k(x)\int_0^A \beta(x,y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(y)dy \\ &\quad -\frac{1}{2}\int_0^x \gamma(x,y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(y)dy \\ &\quad +\int_0^{A-x}\gamma(x,y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(x+y)dy \\ &\quad -\sum_{k=1}^{N}\alpha_k(t)\pi_k(x)\frac{w(x)}{D} \\ &\quad -p\delta\frac{f(x)\sum_{k=1}^{N}\alpha_k(t)\pi_k(x)}{\int_0^A f(y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(y)dy+k_\rho}Z.\end{aligned} \tag{4}$$

If one is interested in finding the rate of change at a particular gridpoint x_j, the equation simplifies:

$$\begin{aligned}\frac{\partial \rho(t,x_j)}{\partial t} &= \frac{1}{2}\int_0^{x_j}\beta(x_j-y,y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_j-y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(y)dy \\ &-\alpha_j(t)\int_0^{A}\beta(x_j,y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(y)dy \\ &-\frac{1}{2}\int_0^{x_j}\gamma(x_j,y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(y)dy \\ &+\int_0^{A-x_j}\gamma(x_j,y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_j+y)dy \\ &-\alpha_j(t)\frac{w(x_j)}{D} \\ &-p\delta\frac{f(x_j)\alpha_j(t)}{\int_0^{A}f(y)\sum_{k=1}^{N}\alpha_k(t)\pi_k(y)dy+k_\rho}Z.\end{aligned} \tag{5}$$

Using trapezoidal approximation for the integrals, i.e. the approximation $\int_{x_m}^{x_n} f(y)dy \approx \sum_{l=m}^{n}\frac{f(x_l)+f(x_{l+1})}{2}(x_{l+1}-x_l)$, the first term on the right-hand side of (5), for instance, becomes:

$$\frac{1}{2}\sum_{l=1}^{j}\frac{\left[\beta(x_j-x_l,x_l)\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_j-x_l)\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_l)+\beta(x_j-x_{l+1},x_{l+1})\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_j-x_{l+1})\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_{l+1})\right]}{2}(x_{l+1}-x_l). \tag{6}$$

In the case of a uniform grid, this simplifies to:

$$\frac{1}{2}\sum_{l=1}^{j}\frac{\left[\beta(x_{j-l},x_l)\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_{j-l})\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_l)+\beta(x_{j-l-1},x_{l+1})\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_{j-l-1})\sum_{k=1}^{N}\alpha_k(t)\pi_k(x_{l+1})\right]}{2}x_1,$$

or

$$\frac{1}{4}\sum_{l=1}^{j}\left[\beta(x_{j-l},x_l)\alpha_{j-l}(t)\alpha_l(t)+\beta(x_{j-l-1},x_{l+1})\alpha_{j-l-1}(t)\alpha_{l+1}(t)\right]x_1.$$

The application of the trapezoidal approximation and similar simplifications to the other terms of equation (5) results in:

$$\begin{aligned}\frac{d\alpha_j(t)}{dt} &= \frac{1}{4}\sum_{l=1}^{j}\left[\beta(x_{j-l},x_l)\alpha_{j-l}(t)\alpha_l(t)+\beta(x_{j-l-1},x_{l+1})\alpha_{j-l-1}(t)\alpha_{l+1}(t)\right]x_1\\ &-\alpha_j(t)\frac{1}{2}\sum_{l=1}^{N}\left[\beta(x_j,x_l)\alpha_l(t)+\beta(x_j,x_{l+1})\alpha_{l+1}(t)\right]x_1\\ &-\frac{1}{4}\sum_{l=1}^{j}\left[\gamma(x_j,x_l)\alpha_l(t)+\gamma(x_j,x_{l+1})\alpha_{l+1}(t)\right]x_1\\ &+\frac{1}{2}\sum_{l=1}^{N-j}\left[\gamma(x_j,x_l)\alpha_{j+l}(t)+\gamma(x_j,x_{l+1})\alpha_{j+l+1}(t)\right]x_1\\ &-\alpha_j(t)\frac{w(x_j)}{D}\\ &-p\delta\frac{f(x_j)\alpha_j(t)}{\frac{1}{2}\sum_{l=1}^{N}\left[f(x_l)\alpha_l(t)+f(x_{l+1})\alpha_{l+1}(t)\right]x_1+k_\rho}Z,\ j=1,\ldots,N.\end{aligned} \tag{7}$$

This system is a set of N ordinary differential equations. These together with the differential equations for zooplankton dynamics form the stiff system which is solved using the package LSODE. Notice that $\rho(t,x_j)=\alpha_j(t)$, so we are in effect solving for the value of ρ at the gridpoints.

1.4. Decisions Made in Implementation

In the sample computer implementation used for comparative studies, ten separate "cohorts" of individual cells are followed. The biological characteristics of the cohorts are the same (i.e. as described earlier, the model is unstructured on an individual physiological level); the

difference lies in their level of maturity (they are of different ages). Initially, members of the first cohort are young, while members of the tenth cohort are on the verge of reproducing.

Cell reproduction is a not a continuous process, thus the simulation over time of the continuous processes must be interrupted periodically to check for reproduction. The individual cell model is sufficiently simple that it may be solved explicitly and thus need not be numerically solved using LSODE (Ackleh, Hallam, and Smith, 1994). When numerical solution by LSODE of the continuous processes is periodically interrupted (we use a "period" of one hour of simulation time), the nutrient level $N(t)$ and the carbon content $C(t)$ for each cohort of individual cells are updated using the explicit solution of $C(t)$. If the carbon content has increased sufficiently in members of one of the cohorts to trigger reproduction, cell counts are updated. Previously, due to the simplifying assumption mentioned earlier regarding the behavior of reproducing cells within aggregates, this affected only the number of free cells. The new method, where both the number of free cells and the aggregate size distribution may be affected, is described in detail in the methodology section.

1.5. Objectives

This study had two major objectives. The first was to reduce the execution time required to run a simulation. Previously, a simulation run used by the researchers could easily take two hours (Ackleh, personal

communication). The speed of running the simulation is particularly important in parameter estimation studies (Ackleh, Hallam, and Muller-Landau, 1995 and Ackleh, 1995) because results from a number of simulations are required to get one set of estimates. The second major objective was to make the simulation more realistic by eliminating the simplifying assumption that the size of aggregates will be unaffected by reproduction of their constituent cells. The additional biological accuracy gained from this modification continues the series of incremental improvements to the model that have been and continue to be done by other researchers working on this algal aggregation model.

Two different strategies were used to attack on the problem of speeding up the simulation. The first was to eliminate the necessity of using a uniform grid of aggregate sizes. A uniform grid is inefficient because there is much more "activity" in the smaller grid sizes. It is important to track the aggregation at smaller sizes closely, as the densities of these aggregates change very quickly, while the changes in densities of larger aggregates are slower and of smaller magnitude. The uniform grid restriction was initially introduced because some of the calculations are simpler with a uniform grid, and because experimental data is often categorized along a uniform grid. Using a non-uniform grid instead allows the simulation to cover the entire range of aggregate sizes of interest with a smaller set of equations, while retaining adequate resolution at the smaller aggregate sizes. A smaller set of equations can be solved much faster. The second strategy for attacking the issue of

program speed was the design and implementation of a parallel version of the computer program.

The second major objective was to eliminate the simplifying assumption concerning the effect of cell reproduction on the size of aggregates. Previously, it was assumed that when cells in aggregates reproduce (split in two), exactly one of those cells will remain with the aggregate while the other will be released as a free cell. This assumption implies that the size of aggregates will be unaffected by reproduction of their constituent cells, so only the number of free cells is affected by reproduction. It is desirable for the simulation to model the more complex and realistic situation in which some of the cells generated by reproduction in an aggregate would fall off the aggregate while some would stay, resulting in a larger aggregate. Thus, in this improved scenario, reproduction affects not only the number of free cells, but also the size distribution of aggregates. The methods used to address these objectives, as well as issues that arose in designing and implementing the necessary changes, are described in the following chapter.

Chapter 2.
Methodology

2.1. Non-uniform Grid

The use of a uniform grid required approximating ρ, the density of aggregates of different sizes over time, on an ordered set of aggregate sizes $\{x_1,...,x_N\}$ as introduced in Section 1.3. In that method, $\{x_1,...,x_N\}$ are equally spaced, which simplifies the analysis. The first term in the partial differential equation representing aggregate growth (3) is $\int_0^x \beta(x-y,y)\rho(t,x-y)\rho(t,y)dy$, and, as described previously, ρ was initially considered to be a linear spline: we have values for ρ at the gridpoints; between gridpoints the value of ρ lies on the line segment connecting the values at neighboring gridpoints.

The obstacle to using a non-uniform grid lies in the term $\rho(t,x-y)$. We must consider the discrete approximation of this term (refer to equation (6)): $\rho(t,x_j-x_l)$, where l will range over 1,...,j-1. If the grid is uniform, $x_j-x_l=x_{j-l}$ and since that is just a grid point, we have a value for ρ there. If the grid is not uniform, x_j-x_l could lie between gridpoints. So we do not immediately have the value of ρ at that point. The solution is

to figure out which gridpoints lie to either side of $x_j - x_l$ and interpolate linearly between them to find the value of ρ at $x_j - x_l$.

One reason the investigators had not implemented this additional flexibility previously was concern that finding this intermediate value of ρ would require an excessive amount of computer time. One would first think that to evaluate the integral, the program would need to figure out which gridpoints are adjacent to $x_j - x_l$ for each term l--this would involve a loop each time--and determine where $x_j - x_l$ is located between the adjacent gridpoints. But in fact this information can be determined during the program's initialization stage and stored for use during the simulation. The information was stored in a pair of arrays providing the index of the adjacent gridpoint and the relative location of $x_j - x_l$ between the adjacent gridpoints for each pair (j,l). A tradeoff is made here, sacrificing a little additional stored information (memory use) for a significant time savings when calculating the right-hand side of the ordinary differential equation, a calculation which is performed a large number of times during a simulation run.

2.1.1. Distinction Between # of Aggregates And Density

In making this change and understanding the results it yielded, Ackleh (personal communication) pointed out that there is a distinction between the number of aggregates in a particular size class (namely, $\int_{x_{j-1}}^{x_j} \rho(y)dy$, or according to the trapezoidal approximation, $\frac{\rho(x_{j-1})+\rho(x_j)}{2}(x_j - x_{j-1})$) and the density of aggregates of a particular size $\rho(x_j)$. When the grid was

uniform, these quantities were equivalent up to a scaling factor, but this is no longer the case when the grid is non-uniform. The continuous processes simulated only involve the density of aggregates, but the effect of reproduction depends on the numbers of aggregates, as this determines the number of cells present.

Observing this distinction can make a large difference in the results. In particular, at a place in the grid where the space between gridpoints (i.e. size classes) is large, a particular density or value of ρ at a gridpoint will represent a larger number of aggregates than that same density would at a place in the grid where the gridpoints are more closely spaced. For example, if $\rho(x_k)=.1$, $\rho(x_{k+1})=.3$, and $x_{k+1}-x_k=1$, then $\int_{x_k}^{x_{k+1}}\rho(y)dy$, the number of aggregates of size between x_k and x_{k+1}, is 0.2. On the other hand, if $x_{k+1}-x_k$ is only .01, then $\int_{x_k}^{x_{k+1}}\rho(y)dy=.002$. Such a difference in gridpoint separation is realistic, since our choice of non-uniform grid is to have an exponentially increasing distance between gridpoints.

During the simulation time that the continuous processes are operating, only the density, ρ, is important. At those times when the continuous processes are stopped in order to update the available nutrient level, the carbon content of cells in each maturity cohort, and to check for and handle reproduction, the current densities must be used to calculate the number of aggregates in each size class. Based on these numbers, the new numbers of aggregates in each size class is calculated after reproduction

and used to find the new densities. This leads to a numerical approach described in the next section.

2.1.2. Step Functions Rather Than Linear Splines

Typically after the effect of reproduction has been calculated, because a portion of the new cells born to cells in aggregates "fall off" and become free cells, the number of free cells (or aggregates of a single cell) is much larger than the number of aggregates in the next larger aggregate size class. With ρ approximated as a linear spline, this causes a problem when, after reproduction yields new numbers of aggregates in different size classes, we want to translate back to densities ρ at gridpoints.

The number of aggregates in a size class j is, by definition, $\int_{x_{j-1}}^{x_j} \rho(y)dy$, and according to the linear spline approximation, $\frac{\rho(x_{j-1})+\rho(x_j)}{2}(x_j - x_{j-1})$. It is assumed that there are no aggregates of size 0. If we denote the number of free cells, otherwise known as aggregates of size 1, by N_1, we have:

$$\int_0^{x_1} \rho(y)dy \approx x_1\rho(x_1)/2 = N_1$$

And thus $\rho(x_1)$ is determined by:

$$\rho(x_1) = 2 \cdot N_1/x_1 .$$

There are no difficulties thus far. The problem comes when we try to figure out $\rho(x_2)$ based on N_2 and $\rho(x_1)$, which is fixed by the above considerations. As above:

$$\int_{x_1}^{x_2} \rho(y)dy \approx \frac{\rho(x_1)+\rho(x_2)}{2}(x_2 - x_1) = N_2$$

so

$$\rho(x_2) = \frac{2 \cdot N_2}{x_2 - x_1} - \rho(x_1). \tag{8}$$

This can cause $\rho(x_2)$ to be negative in the (typical, if reproduction has occurred) case that $\rho(x_1)$ is larger than the first term. Certainly, a negative density is nonsensical.

One might think that this problem could be solved by simply setting $\rho(x_i) = 0$ if (8) results in a negative value. Perhaps this loss of information is an acceptable byproduct of approximating ρ. Unfortunately, besides the compromise in numerical correctness, this only passes the problem of negative densities on to further points on the grid: consider that if $\rho(x_i) = 0$, with a modest number of aggregates in the size class between x_2 and x_3, $\rho(x_3)$ will have to be large to compensate for $\rho(x_2)$ being 0. If there is a smaller number of aggregates in the size class between x_3 and x_4, which is the normal situation, $\rho(x_4)$ will be negative.

Thus a better solution is needed. It turns out that it is appropriate to give up the linear spline approximation in favor of simply treating $\rho(t,x)$ as a step function in x. One reason it is an appropriate choice is that the SIGMA tank data used to provide initial aggregate densities and for comparison in parameter estimation studies (Ackleh, Hallam, and Muller-Landau, 1995), consists of aggregate density information for size classes. A linear spline approximation would be more appropriate if the data were in the form of densities at a set of individual size points.

2.2. Aggregate Growth Due to Cell Reproduction

Another restriction addressed was the simplifying assumption concerning the behavior of cells resulting from reproduction within aggregates. The simplifying assumption was that exactly one of each pair of "daughter" cells leaves the aggregate to become a free cell, while the other stays: this means that each aggregate consists of the same number of cells after reproduction as before. In this case, reproduction only affects the number of free cells; the size of non-trivial aggregates (their size distribution) is unaffected by reproduction. Because it seems unlikely that this simplified situation is an accurate representation of reality, it is desirable to parametrize the probability that new "daughter" cells fall off. This way, if biological data for this probability were to become available, it could be used in the simulation. Alternatively, a future study might estimate this parameter using tank data, as has previously been done with other model parameters (Ackleh, Hallam, and Muller-Landau, 1995). There is more on this topic in the Discussion section.

In the simulation, ten cohorts of cells which are at the same maturity levels are tracked. When the cells in a maturity cohort reproduce, certainly reproduction by free cells increases the number of cells which are free. The number of free cells in that maturity cohort doubles. We must also consider the effect of reproduction of cells which are not free, those which are tied up in an aggregate.

For the moment, restrict consideration to a single size class of aggregates (i.e. a range of aggregate sizes $[x_j, x_{j+1}]$). Just within this aggregate size class, the portion of cells in an aggregate that lie in a specific maturity cohort is tracked. Thus, the number of cells in each aggregate that will reproduce is known. A new model parameter is introduced: q, which controls what proportion of the new daughter cells stick to the aggregate. The rest fall off. Thus $(1-q)\times 100$ percent of the new daughter cells become free cells, and this number of cells is added to the free cell count. $q\times 100$ percent of the new daughter cells stay with the aggregate, causing it to increase in size. Note that this does not affect the number of aggregates in the size class; rather, it increases the size of the aggregates which were (formerly) in the size class under consideration.

We know how many aggregates there are, and we know the new size of this size class. But we can not change the value of x_j, which would mean changing the grid distribution, to suit this new size. Somehow this effect must "shift" the size distribution numbers (i.e. $\rho(x_j)$ for $1 \le j \le N$) to the right. In fact what is done is to first identify the two gridpoints on either side of the new aggregate size: if z is the new aggregate size, we locate the two neighboring gridpoints, that is, k such that $x_k \le z \le x_{k+1}$. Then we split the aggregates, whose number is the same as before reproduction, but which are of a new size, proportionally between $\rho(x_k)$ and $\rho(x_{k+1})$.

Since ρ is generally smaller for larger aggregate sizes, the effect of this right shift caused by reproduction of cells within aggregates is, on the whole, to increase the number of larger aggregates. A small amount of

aggregate biomass or information may be lost to the model in this process, if the increase in size of the largest aggregates due to reproduction of some of their constituent cells causes their new size to be larger than the largest aggregate size class that we track. If this occurs, the portion of the aggregates which are larger than x_N will be "thrown away." In the continuous model language, this largest aggregate size which we track is A. A similar loss to the model of a small amount of aggregate biomass occurs in the continuous processes, when aggregation causes some aggregates to become larger than size A.

2.3. Parallel Implementation

2.3.1. Introduction

In discussing parallelization of an existing non-parallel code, the first question is where potential for parallel execution exists. A related question is whether the algorithm as written contains enough inherent parallelism, or whether a different algorithm should be sought or created. In the implementation described here, the algorithm of the program is essentially not changed. The potential for alternative approaches to parallelization is deferred to the discussion section.

2.3.2. Description Of Sequential Code

The sequential simulation code is built around a public domain package for solving systems of ordinary differential equations called LSODE. Essentially, after some initialization is done, the package LSODE is

repeatedly called over a period of simulation time to numerically solve the set of ordinary differential equations in time which make up the continuous part of the model. Between calls to LSODE, the discontinuous part of the model, reproduction of individual cells, is handled. However, in discussing speed of execution, it is important to realize that the lion's share of processing time is spent by LSODE solving these differential equations.

When LSODE is solving the differential equations, it needs numerical values for the right-hand sides of the ordinary differential equations. The way this is accomplished is that LSODE, which has been asked by the main program to solve the equations for a specified amount of simulation time, in turn calls a subroutine we have written which, given values for all the necessary variables, returns a value for each derivative. In other words, LSODE asks our subroutine to calculate the right-hand side of equations (3) and (2) (or their discrete approximations). This set of calculations must be done many times over the course of an entire simulation: in one sample run they were done 18,531 times. This accounted for 95% of the total runtime. In fact, 75% of the execution time was spent calculating just the first term on the right-hand side of (3) alone. Because the time usage is so heavily concentrated here, it is a good place to look for potential speed-up.

2.3.3. Description Of Strategy/Implementation

When LSODE calls our subroutine to calculate the right-hand sides, for each gridpoint there is a corresponding right-hand side of equation (3) to

be calculated. In a typical simulation we use twenty gridpoints, which means that there are twenty right-hand sides to be calculated for equation (3). Equation (2) represents one additional calculation, but the time required is negligible in comparison.

The parallel machine chosen for the first implementation of a parallel algorithm is the Thinking Machines CM-5 located at the University of Tennessee. It has 32 processors. We only take advantage of twenty of them, one for each right-hand side. Basically, the entire program executes on each processor, which duplicates a lot of work except at the point that LSODE asks for the right-hand sides of the twenty-one differential equations. At that point, each of the first twenty processors calculates one right-hand side concurrently. When they have all finished, the information is shared around. In CM-5 language, each processor that has calculated a right-hand side *broadcasts* its result, while each of the others waits until it receives the result.

2.3.4. Speedup Possible Using This Strategy

Since 95% of the program's execution time is involved in this parallelization, being spread across 20 processors, we might initially hope that the program would run in $0.05 + \frac{0.95}{20} = 0.0975$ times the time the sequential version takes to run. That would be a dramatic speedup--more than tenfold. However, there are a couple of factors that this estimate fails to consider. One is the friction of parallel computation: communications overhead. Recall that the right-hand sides are calculated 18,531 times. Thus, a result must be broadcast by one processor

and received by all the others $18,531 \times 20 = 370,620$ times during a simulation. While each individual communication is extremely fast, the communication overhead does add up.

There is another consideration which prevents the speedup from being as good as one would initially hope. We must examine the right-hand side of equation (3) more closely, specifically the first term. Since we are considering the discrete approximation, in fact equation (7) is more to the point. We see that to calculate the first summation on the right-hand side, for a specific α_j, there are j terms in the sum. Thus, calculating the right-hand side of $\frac{d\alpha_{20}(t)}{dt}$ takes approximately twenty times longer than calculating the right-hand side of $\frac{d\alpha_1(t)}{dt}$, as there are twenty terms in the first sum on the right-hand side of $\frac{d\alpha_{20}(t)}{dt}$ but only one in $\frac{d\alpha_1(t)}{dt}$. So the processor which must calculate $\frac{d\alpha_{20}(t)}{dt}$ will be the limiting factor, the processor on which the others must all wait.

We can make a rough calculation of how much of the work this processor must do. The total work is on the order of $20+19+...+2+1 = 10 \times (20+1) = 210$ additions, as there are that many terms in all of the first sums on the right-hand sides put together. One processor must do 20 parts of this work, so the maximum speedup in calculating the right-hand sides considering this limitation would be $\frac{20}{210} = 2/21$. Overall, the best possible time would thus be $0.05 + \frac{0.95}{21/2} = 0.14$ times the sequential speed, or a roughly sevenfold

speed-up. This is only a rough estimate because we are disregarding the other terms of the right-hand sides, as well as all the bits of constant overhead which must be done for each summation, regardless of how many terms it has.

The FORTRAN code for both the sequential and the parallel CM-5 versions of the simulation described above are available. Contact Professor Tom Hallam, Mathematical Ecology, Department of Mathematics, University of Tennessee, Knoxville, TN.

Chapter 3.
Results

3.1. Effect Of Aggregate Growth Due To Cell Reproduction

Table 2 shows the effects of varying two model parameters related to reproduction. The first column shows what portion of the cells in aggregates are growing and reproducing. The remaining cells in aggregates are inactive. The second column shows what portion of the additional cells resulting from reproduction in aggregates stay on the aggregate. This is the parameter introduced in this study, referred to as q. The other portion of the new cells, as explained above, "fall off" to

Table 2. Dependence Of Nutrient Depletion On Model Parameters

Portion Of Cells In Aggregates That Can Reproduce	Portion Of New Cells Sticking to Aggregate, q	% of Nutrient Remaining After:	
		10 days	20 days
0.8	0.8	64	0
0.8	0.2	18	0
0.2	0.8	86	60
0.2	0.2	81	41

become free cells. By new cells, we mean the additional cells resulting from reproduction, above and beyond the cells that were already in the aggregate. The third and fourth columns show the dependent variable, the percent of the initial nutrient pulse remaining after ten and twenty days, respectively. The initial nitrogen concentration in these simulations was 45 μM.

We see that if a larger portion of the cells in aggregates are growing and reproducing, the nutrient is depleted faster. This is because the cells in aggregates which are inactive do not consume nitrogen, allowing the initial nutrient pulse to last longer. The table also indicates that if more new cells stay with the aggregate (i.e. q is greater), nutrient is depleted more slowly. This may be explained by the fact that if more cells stay with the aggregate, more of them are inactive, so fewer of the new cells are growing, reproducing, and consuming nutrient. All of the new cells that are released as free cells will continue to consume and reproduce, while only a portion (as indicated in the first column) of those that stay with the aggregate will continue to consume and reproduce. Thus in the case of lower q, there will be more consuming cells and the nutrient will be depleted sooner.

3.2. Timing of Non-uniform Grid Version

The times given in Table 3 are for a 25 day simulation performed on a Sparc 20 Model 61. For the uniform grid simulation, 100 gridpoints were used; their separation was 0.01 mm^3, so the volume of the largest

Table 3. Execution Time Required For Different Grid Types

	Uniform Grid	Non-uniform Grid
CPU Time (Minutes)	126.2	1.4

aggregate size tracked was 1 mm^3. In the case of the non-uniform grid simulation, each of the twenty gridpoints was double the volume of the previous gridpoint. As a result, the largest aggregates tracked were just over 2 mm^3, making the non-uniform grid simulation superior in the sense of range of aggregate sizes tracked. There are other differences between the two implementations, including the fact that the uniform grid simulation is based on a linear spline-based approximation while the non-uniform grid simulation is based on a step function-based approximation. However, the faster non-uniform grid program is the more recent, superior program.

3.3. Timing of Parallel Version

Timing data for the parallel implementation is presented in Table 4. CPU Time is the amount of time the central processor spent executing the

Table 4. Comparative Timing Data

(Minutes)	CM-5 Sequential	CM-5 Parallel	Sun Sequential
CPU Time	6.3	1.3	1.4
Elapsed Time	12.9	2.8	1.5

simulation, while elapsed time is the amount of time that passed between initiation of the execution and its termination. All the simulations had the same input parameters. In this case 5 hours of simulation time were modeled; sinking and predation were active. The Sun simulations were run on a Sun Sparc 5 Model 85, at a time when no other users were making significant demands on the system. The CM-5 simulations were run during the middle of the afternoon on a weekday, and there was at least one other user making demands on the system.

The information in the table shows several things. Elapsed time is how long one must wait for the execution to complete; the entries in this row indicate that for practical purposes, running the simulation on a Sun workstation is the best choice with conditions as described. The CPU times for the CM-5 Sequential and for the Sun programs, 6.3 and 1.4 respectively, represent the difference in speed between individual CM-5 processors and the Sun processor for this task. The more modern Sparc 5 CPU is approximately four times faster on this task than an individual CM-5 CPU.

Another observation worth making is that the discrepancy between CPU time and elapsed time on the CM-5, in both the Sequential and Parallel columns, is clearly much greater than on the Sun workstation. This is due to the fact that other users are making demands on the CM-5, while the Sun workstation is devoted to this task.

There are two encouraging results in the table. First, the parallel program runs much faster than the sequential version on the CM-5. This indicates that communication overhead is not overwhelming the efficiency gains from splitting calculations between processors. In fact, an almost fivefold advantage is gained from the parallelization. While this is less than the roughly sevenfold theoretical upper limit on the efficiency gain discussed above, it is not far off. The other encouraging result in the table is that the CPU time that the parallel program requires is actually less than the Sun simulation requires. Thus the efficiency gained from parallelization can "pay for" the necessary communication overhead and still more than make up for the slower processors on the CM-5.

Chapter 4.
Discussion

4.1. Possibility For Further Improvement In Parallelization

4.1.1. Theoretical: Inside or Beyond LSODE

In "The Potential for Parallelism in Ordinary Differential Equations," C. W. Gear (1988) defines two potential sources of parallelism: "across the method" and "across the system." By parallelism across the method he means that "different parts of the method are executed on different processors." That is, ODE solution methods generally involve a number of steps, and in some cases different steps may be executed simultaneously. By parallelism across the system he means that since there is more than one differential equation, operations that must be done to each of them may be done simultaneously. The parallel implementation described above falls in this category--the right-hand side of all of the differential equations in the system are found simultaneously.

An implementation which would take advantage of parallelism across the method was not attempted. Here a further distinction may be made

between two different possible strategies. One strategy would be to locate and harness latent parallelism within LSODE (Hindmarsh, 1981), which is used as an implementation of the Backward Differention Formula (BDF) algorithm (Fatunla, 1988). The other strategy would be to consider alternate algorithms. An important restriction here is that our system of ordinary differential equations is a stiff system. Most of the algorithms available for solving systems of ODEs are inappropriate for use with stiff systems; this is the main reason LSODE is used.

It appears that an efficient parallel algorithm for solving stiff systems of ODEs has yet to be invented. There are other algorithms for solving stiff systems of ODEs, but BDF appears to be the most robust and most widely used method. My search of the world wide web (WWW) and the literature revealed no publicly available "package" for parallel solution of stiff systems of ODEs. There are a number of packages for sequential solutions of such systems--LSODE is but one of many, but absence of a parallel package suggests that a parallel algorithm for this task has yet to be found, as do comments made by Gear (1988): "In large-scale parallelism, communication between processors is costly, yet we have seen that communication is apparently essential in the case of stiff integration." Gear provides justification to support that statement, but a deeper discussion of the topic is beyond the scope of this study.

4.1.2. Fine Tuning

There is clearly still room for improvement in the parallel implementation. In addition to the difficult but potentially more

rewarding broad algorithm changes contemplated above, there are opportunities to "fine tune" the parallelization. While these improvements would probably yield improvements in speed, their value should be judged with the understanding that execution speed is not currently a sticking place. The gains in execution speed resulting from this kind of improvement are also likely to be modest, on the order of tens of percents at best. As was discussed in the parallel implementation section above, some of the right-hand sides take longer to execute than others. One might take these right-hand sides, specifically the first term, and split their calculation across processors. Since the CM-5 at the University of Tennessee has 32 processors and there are only twenty equations, twelve processors are unused when each processor calculates one right-hand side. We might take the twelve equations which take the longest to calculate and let one of these previously unused processors perform half the calculation, then add the two halves to get the full right-hand side. Or we might split three equations into three pieces, which would utilize six of the previously unused processors, and split another six equations in half. There are many possibilities, and one of them is probably optimal in some sense. What holds us back from significant speed gains here is what holds us back overall as well: communication costs. Each additional split requires that more information be shared between processors, which cuts into the gains made by dividing the work more widely.

4.2. Possibility of Estimating Reproduction Parameter

Ackleh, Hallam, and Muller-Landau (1995) estimate several model parameters by using a least squares methodology with experimental data obtained from the SIGMA Tank Experiment. Parameters estimated include "stickiness" and contact efficiencies. At some point in the future, this methodology and data could be used to similarly estimate the new parameter q. Such an analysis would yield information about the physical and biological processes of aggregation that reflects a biological reality, but would be difficult to measure directly experimentally.

BIBLIOGRAPHY

BIBLIOGRAPHY

Ackleh, A.S. Parameter estimation in a structured algal coagulation-fragmentation model. In review.

Ackleh, A. S., B. G. Fitzpatrick and T. G. Hallam (1994) Approximation and parameter estimation problems for algal aggregation models. Mathematical Models and Methods in Applied Sciences **4** 291-311.

Ackleh, A. S., T. G. Hallam and H. C. Muller-Landau (1995) Estimation of Sticking and Contact Efficiencies in Aggregation of Phytoplankton: The 1993 SIGMA Tank Experiment. [CRSC Technical Report 94-21.] Deep Sea Research **42** 185-201.

Ackleh, A. S., T. G. Hallam and W. O. Smith (1994) Influences of aggregation and grazing on phytoplankton dynamics and fluxes: an individual-based modeling approach. Nonlinear World **1** 473-492.

Eldén, L. and Wittmeyer-Koch, L. (1990) **Numerical Analysis: An Introduction**. Academic Press.

Fatunla, S. O. (1988) **Numerical Methods For Initial Value Problems In Ordinary Differential Equations**. Academic Press, San Diego, CA.

Fröberg, C. E. (1985) **Numerical Mathematics: Theory and Computer Applications**. Benjamin/Cummings Publishing.

Frost B. W. (1987) Grazing Control of Phytoplankton Stock in Open Sub-Arctic Pacific Ocean. Marine Ecology Progress Series **39** 49-68.

Gear, C. W. (1988) The Potential For Parallelism in Ordinary Differential Equations. In Computational Mathematics II: Proceedings of the second International Conference on Numerical Analysis and its Applications. Boole Press.

Gelbard, F. and Seinfeld, J. H. Numerical Solution of the Dynamic Equation for Particulate Systems. Journal of Computational Physics **28** 357-375.

Hindmarsh, A. C. (1980) LSODE and LSODI, Two New Initial Value Orindary Differential Equation Solvers. ACM-Signum Newsletter **15** Number 4, 10-11.

Jackson, G. and S. Lochmann (1992) Effect Of Coagulation On Nutrient And Light Limitation Of An Algal Bloom. Limnology and Oceanography 34 514-530.

Lancelot, C., G. Billen and H. Barth (1991) The Dynamics of *Phaeocystis* Blooms in Nutrient Enriched Coastal Zones. Water Pollution Research Report 23, Proceedings of the third workshop held in Plymouth (U.K.).

Ramachandramurthi, S. (1994) A Beginner's Guide to the Connection Machine CM-5. Joint Institute for Computational Science. Knoxville, TN.

Raymont, J. (1983) **Plankton and Productivity in the Oceans** (2nd Edition). Pergamon Press, London.

Scheffer, M. (1991) Fish and Nutrients Interplay Determines Algal Biomass: A Minimal Model. Oikos **62** 271-282.

Steele, J. H. and E. W. Henderson (1981) A Simple Plankton Model. American Naturalist **117** 676-691.

VITA

Derek Prowe was born in Northfield, Minnesota on November 18, 1969. He attended public schools in Northfield and went on to Pomona College in Claremont, CA. He graduated with a major in mathematics in 1992. He attended the University of Wisconsin as a graduate student for a year before transferring to the University of Tennessee.