Estimating Phytoplankton Growth Rates from Compositional Data

by

Lorraine Thomas

Submitted to the Joint Program in Oceanography / Biological Oceanography
Massachusetts Institute of Technology
and Woods Hole Oceanographic Institution
in partial fulfillment of the requirements for the degree of
Master of Science in Biological Oceanography
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2008

Author .....
Joint Program in Oceanography / Biological Oceanography
Massachusetts Institute of Technology
and Woods Hole Oceanographic Institution
January 18, 2008

Certified by .....
Michael Neubert
Associate Scientist, Woods Hole Oceanographic Institution
Thesis Supervisor

Certified by .....
Heidi Sosik
Associate Scientist, Woods Hole Oceanographic Institution
Thesis Supervisor

Accepted by .....
(for) Edward DeLong
Chair, Joint Committee for Biological Oceanography
Massachusetts Institute of Technology

Estimating Phytoplankton Growth Rates from Compositional Data

by

Lorraine Thomas

Submitted to the Joint Program in Oceanography / Biological Oceanography
Massachusetts Institute of Technology
and Woods Hole Oceanographic Institution
on January 18, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Biological Oceanography

Abstract

I build on the deterministic phytoplankton growth model of Sosik et al. by introducing process error, which simulates real variation in population growth and inaccuracies in the structure of the matrix model. Adding a stochastic component allows me to use maximum likelihood methods of parameter estimation.

I lay out the method used to calculate parameter estimates, confidence intervals, and estimated population growth rates, then use a simplified three-stage model to test the efficacy of this method with simulated observations. I repeat similar tests with the full model based on Sosik et al., then test this model with a set of data from a laboratory culture whose population growth rate was independently determined.

In general, the parameter estimates I obtain for simulated data are better the lower the levels of stochasticity. Despite large confidence intervals around some model parameter estimates, the estimated population growth rates have relatively small confidence intervals. The parameter estimates I obtained for the laboratory data fell in a region of the parameter space that in general contains parameter sets that are difficult to estimate, although the estimated population growth rate was close to the independently determined value.

Thesis Supervisor: Michael Neubert
Title: Associate Scientist, Woods Hole Oceanographic Institution

Thesis Supervisor: Heidi Sosik
Title: Associate Scientist, Woods Hole Oceanographic Institution

Acknowledgments

Throughout this project, I benefited from the advice of many people. I'd like to particularly thank my advisors, Mike Neubert and Heidi Sosik for their help, particularly figuring out what the most recent set of results implied for the debugging process, and assistance and guidance during the writing process. Andy Solow also provided insight into several statistical questions that arose. Mick Follows not only served on my thesis committee, but provided helpful notes in a very short time frame. The Neubert/Caswell lab group was also very supportive, particularly when it came to general tools for doing mathematical modeling research. Beyond the details of this project, I'd like to thank Stephanie Jenouvrier and Carly Strasser, as well as the other Biological Oceanography students for their friendship during my time here, and Erin Dupuis for helping me with several logistical issues. My research was supported by MIT, the Woods Hole Oceanographic Institution, and a grant from the National Science Foundation (OCE-0530830).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Current methods for measuring phytoplankton population growth rates fall into two categories: laboratory and in situ methods. Population growth rates measured in laboratory cultures only reflect population growth in an artificial setting. In situ methods often involve large amounts of time in the field, making them expensive for long term studies. Reynolds [11] describes several modern methods involving large enclosures, hourly sampling, or time-intensive handling of field samples.

Emerging approaches such as automated flow cytometry promise to overcome some limitations of conventional methods. For example, FlowCytobot, a submersible flow cytometer, can autonomously monitor phytoplankton cell size distributions over time [10]. FlowCytobot draws water from a particular depth and measures light scattering and fluorescence of each particle passing through the system. These measurements can be used to identify various taxa and determine cell size. Other instruments deployed at the same study site monitor light level, water temperature, salinity and other environmental characteristics.

Because of these sampling capabilities, submersible flow cytometry offers a new way to monitor phytoplankton population dynamics. Motivated by the availability of this information, we can build new models to estimate population growth rates. For example, Sosik et al. [12] have developed a deterministic model, to estimate population growth rates of the cyanobacteria *Synechococcus* from time series of cell size distributions. Because their analysis relies on relative abundance, rather than

total number of phytoplankton in each size class, they avoid mistaking changes in cell concentration due to advection and patchiness for changes due to population growth. To convert these relative abundances into population growth rates, they used a matrix population model [5]. The model has terms for cell growth, stasis and division, and includes terms that represent the effects of measured light level. This model is described in detail in Chap. 4.

Sosik et al. [12] estimated the parameters in their model by minimizing a weighted sum of squared deviations between the observed and modeled size distributions. To calculate confidence intervals, they generated bootstrapped data sets by sampling from the real data and estimating parameters for each data set.

I build on the deterministic model of Sosik et al. [12] by introducing process error, which simulates real variation in population growth and inaccuracies in the structure of the matrix model [4]. For the sake of simplicity and tractability, I ignore observation error. (See, however, Calder et al. [4], who point out that this can lead to errors in parameter estimation.) Adding a stochastic component allows me to use maximum likelihood methods of parameter estimation. With known asymptotic approximations [3], I can also calculate confidence intervals around the parameter estimates without resorting to the bootstrap.

In Chap. 2, I describe the new stochastic model and the parameter estimation method. Because the estimation of population growth rate is a central problem in population biology, and because time series of proportions are likely to appear in other biological applications (e.g., epidemiology), my presentation in Chap. 2 is as general as possible. I use a simplified three-stage model to test the efficacy of the method. The results appear in Chap. 3. In Chap. 4, I report the results of similar tests I performed with the full model described by Sosik et al. [12]. These tests use simulated data sets to determine the predictive abilities and limitations of the method. I have additionally tested the method using a set of data obtained from a laboratory culture, whose population growth rate was independently determined.

In general, the parameter estimates I obtain for simulated data are better the lower the levels of stochasticity. Despite large confidence intervals around some model

parameter estimates, the estimated population growth rates have relatively small confidence intervals. The parameter estimates I obtained for the laboratory data fell in a region of the parameter space that in general contains parameter sets that are difficult to estimate, although the estimated population growth rate was close to the independantly determined value.

Chapter 2

Methods

2.1 Structure of the model

We begin by dividing the population into m stages. Let the i^{th} component of the vector $\mathbf{w}_t$, $w_t^{(i)}$ be the fraction of cells in stage i. Our population growth model describes the dynamics of $\mathbf{w}_t$.

The model has two components: a projection matrix component and a stochastic component. First we calculate the expected size distribution at time t, $\mathbf{v}_t(\theta)$ from the observed distribution at time $t-1$ via

$$\mathbf{v}_t(\theta) = \frac{\mathbf{B}_{t-1}(\theta)\mathbf{w}_{t-1}}{||\mathbf{B}_{t-1}(\theta)\mathbf{w}_{t-1}||} \tag{2.1}$$

where $||\mathbf{w}|| = \sum_{i=1}^{m} w^{(i)}$. In the projection matrix $\mathbf{B}_t(\theta)$, the entry $b_t^{ij}(\theta)$ in the ith row and jth column gives the number of cells in stage i at time $t+1$ per cell in stage j at time t. These entries are based on formulas for cell growth, stasis and division, which in turn may depend on environmental covariates (e. g., light or temperature) and model parameters θ.

Next, we assume that the actual stage distribution at time t is drawn from a probability density function whose expectation is $\mathbf{v}_t(\theta)$. Because, by definition, $\sum_i w_t^{(i)} = 1$, our choice of probability density function is constrained. The most popular choice [7]

is the Dirichlet distribution [6], whose probability density function is

$$f(\mathbf{w}|\mathbf{v}(\theta),\theta) = \frac{\Gamma(\phi)}{\prod_{i=1}^{m}\Gamma(\phi v^{(i)}(\theta))}\prod_{i=1}^{m}(w^{(i)})^{\phi v^{(i)}(\theta)-1}. \tag{2.2}$$

Thus

$$\mathbf{w}_t \sim \mathrm{Dir}(\phi \mathbf{v}_t(\theta)). \tag{2.3}$$

The initial distribution $\mathbf{w_0}$ is assumed to be known.

The precision parameter ϕ is part of the vector θ of model parmaeters to estimate. It is inversely proportional to the variance of the Dirichlet distribution. In particular, the variance and covariances are given by

$$\mathrm{Var}[w^{(i)}] = \frac{w^{(i)}(1-w^{(i)})}{\phi+1} \tag{2.4}$$

and

$$\mathrm{Cov}[w^{(i)}, w^{(j)}] = \frac{-w^{(i)}w^{(j)}}{\phi+1}. \tag{2.5}$$

A Dirichlet random variate can be generated by first generating gamma random variates (with expected values $\phi v^{(i)}(\theta)$) according to

$$y^{(i)} \sim \mathrm{Gamma}(\phi v^{(i)}(\theta), 1). \tag{2.6}$$

The components of the Dirichlet random variate are then calculated with

$$w^{(i)} = \frac{y^{(i)}}{\sum_{j=1}^{m} y^{(j)}} \tag{2.7}$$

where m is the number of stages [6].

To calculate the population growth rate, the population is first projected forward over the course of an entire day:

$$\mathbf{u}(\theta) = \left[\prod_{t=0}^{23}\mathbf{B}_t(\theta)\right]\mathbf{w}_0. \tag{2.8}$$

We then calculate the population rate, μ, as

$$\mu(\theta) = \ln \sum_{i=1}^{m} u^{(i)}(\theta). \tag{2.9}$$

Note that this deterministic measurement does not directly depend on the precision parameter ϕ, but only on the components of $\mathbf{B}$.

2.2 Parameter estimation

The conditional likelihood of an observed time series $\mathbf{w}_t$ (conditioned on $\mathbf{w}_0$) as a function of the model parameters (θ) can be written:

$$L(\mathbf{w}_t|\mathbf{w}_0, \theta) = \prod_{k=1}^{T} f(\mathbf{w}_k|\mathbf{v}_k(\theta), \theta) \tag{2.10}$$

with $\mathbf{v}_k(\theta)$ given by (2.1). Maximum likelihood estimation amounts to maximizing the conditional likelihood over θ [3]. I denote the estimate $\hat{\theta}$.

In practice, the minimum of the negative log likelihood ($-l(\mathbf{w}_t|\mathbf{w}_0, \theta)$) is typically calculated. From (2.2), the negative log likelihood simplifies to

$$-l(\mathbf{w}_t|\mathbf{w}_0, \theta) = \sum_{k=1}^{T} \left[-\ln \Gamma(\phi) + \sum_{i=1}^{m} \ln \Gamma(\phi v_k^{(i)})(\theta) - \sum_{i=1}^{m} (\phi v_k^{(i)}(\theta) - 1) \ln w_k^{(i)} \right]. \tag{2.11}$$

As mentioned in the introduction, one advantage of the maximum likelihood method is the ease of calculation of confidence intervals for our estimate $\hat{\theta}$. Asymptotically, the estimates have a multivariate normal distribution.

The variance-covariance matrix of this distribution is found by taking the inverse of the observed Fisher information matrix [3]. The observed Fisher information matrix ($\mathcal{I}$) is defined as

$$\mathcal{I}(\hat{\theta}) = -\frac{d^2}{d\theta^2} l(\theta) \mid_{\theta=\hat{\theta}} . \tag{2.12}$$

Conveniently, the maximum likelihood estimation of a function of θ is simply the function of the maximum likelihood estimate of θ. Thus $\hat{\mu} = \mu(\hat{\theta})$.

2.3 Practicalities

In implementing the maximum likelihood procedure above, I encountered various practical issues.

Numerical Optimization In order to use an unconstrained optimization routine, I transformed the model parameters (θ) from their natural ranges ($[0, \infty]$ and $[0, 1]$) to values which range from $-\infty$ and ∞ (Fig. 2-1) via $\theta' = g(\theta)$. I then found the maximum likelihood estimate of the transformed parameter vector, $\widehat{\theta'}$. Finally, I applied the inverse transform $g^{-1}(\hat{\theta'})$ to calculate original parameters $\hat{\theta}$.

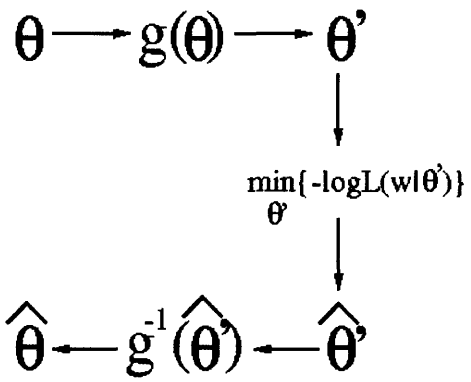

Figure 2-1: Diagram of parameter transformation and maximum likelihood estimation.

Asymptotically, the distributions of $\hat{\theta}$ and $\widehat{\theta'}$ will both be normally distributed and unbiased. In practice these distributions may differ because the rate of convergence to normality will differ [3]. We can take advantage of the fact that $\widehat{\theta'}$ converges faster when we calculate confidence intervals on $\hat{\theta}$.

To calculate these asymptotic confidence intervals, we invert the Fisher information matrix that corresponds to the transformed parameter estimate $\widehat{\theta'}$ to get a matrix of variances and covariances for the transformed parameters. We then draw 500 random variables from a multivariate normal distribution with mean $\widehat{\theta'}$ and corresponding variance/covariance matrix, then untransform them to get an asymptotic distribution of parameter estimates $\hat{\theta}$. We then calculate a 95% confidence interval for

each parameter by finding the 12th smallest and largest values in each distribution. We also calculate the projected population growth rate, $\hat{\mu}$, corresponding to each set of parameter estimates.

Zeros One drawback to the Dirichlet distribution is that by definition the probability of having an empty size class is 0.

Grunwald et al. [7] note that this problem is still present for alternative probability distributions and propose a solution involving conditional distributions on the boundaries as described in a 1982 paper by Aitchison [1]. Aitchison also proposed a simpler ad-hoc solution for situations with only a few zeros [1]. I used Aitchison's solution, adding an arbitrarily small amount ($\epsilon = 10^{-10}$) to each size class when projecting forward, but before renormalizing. This is done at each time step, and changes all zero entries into amounts we assume are below the detection threshold of actual data-gathering devices. I later did the same with the laboratory data used in the full model (See Chap. 4).

In theory, if there are no zeros in the observations or starting points used for the matrix population model, the expected population distribution $\mathbf{v}_t(\theta)$ should never contain zeros. In practice, however, there are times when the projected proportion in a size class is a small fraction of a small initial proportion, and the resulting proportion is rounded to zero because of the way my implementation stores data. I used the same approach of adding $\epsilon = 10^{-10}$ to eliminate these zeros.

Chapter 3

Three stage model

To evaluate the strengths and weaknesses of the maximum likelihood method, I designed a simplified version of the full phytoplankton growth model. This simple model has three size classes of plankton, cell growth dependent on external forcing and size dependent cell division rates. In this section, I examined assumptions about normality of parameter estimate distributions, the effects of removing zeros, and the impact of the starting population distribution. I also examined the accuracy of the parameter and population growth rate estimates over a wide range of parameter values.

3.1 Basic matrix population model

For the three stage model, I use a life cycle scheme that includes cell growth, division and stasis (Fig. 3-1). The cell division rates, δ_2 and δ_3 are parameters to be fit. An additional parameter c reflects how cell growth γ [1] depends on external forcing (e.g., light) that follows a normal distribution over the course of a day (Fig. 3-2):

$$\gamma(t) = \frac{\mathcal{E}(t)}{c + \mathcal{E}(t)}, \tag{3.1}$$

$$\mathcal{E}(t) = \frac{1}{6\sqrt{2\pi}} e^{-(t-12)^2/72}. \tag{3.2}$$

[1] The constant c and function $\mathcal{E}$ are non-physical quantities meant to represent variation, and have no units.

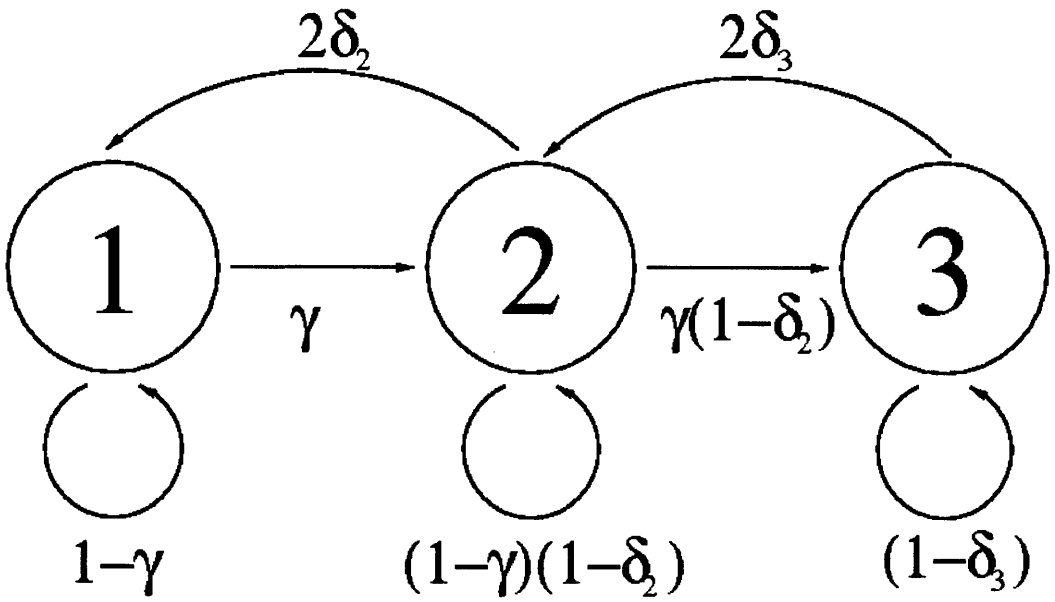

Figure 3-1: Life cycle graph for the three stage model. In our model, stages 1, 2, and 3 represent small, medium and large phytoplankton. Medium cells divide at a rate δ_2, large cells at a rate δ_3. Both small and medium cells grow as a function of c and the level of incident radiation $\mathcal{E}$.

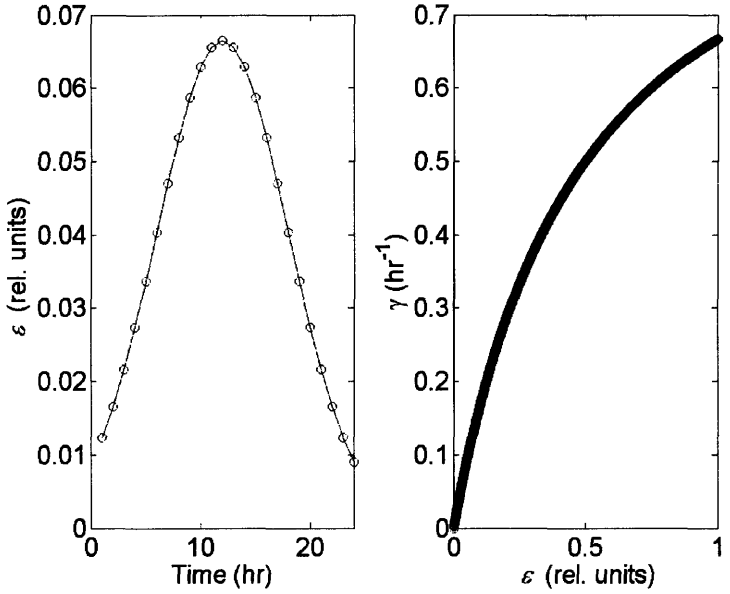

Figure 3-2: Incident radiation (3.2) as a function of time, and cell growth rate (3.1) as a function of incident radiation, with $c = 0.5$, in the three stage model.

The projection matrix $\mathbf{B}_t$, in (2.1), now takes the form

$$\mathbf{B}_t = \begin{pmatrix} 1-\gamma(t) & 2\delta_2 & 0 \\ \gamma(t) & (1-\gamma(t))(1-\delta_2) & 2\delta_3 \\ 0 & \gamma(t)(1-\delta_2) & 1-\delta_3 \end{pmatrix} \tag{3.3}$$

and $\theta = [c, \delta_2, \delta_3, \phi]$.

For a typical realization of the model starting from a uniform initial population distribution $\mathbf{w}_0^{(i)} = 1/3$, the population slowly reaches equilibrium, with approximately 2/3 large, under 1/3 medium, and 1/10 small phytoplankton (Fig. 3-3). When multiple sample observations are considered, we see that the variance in model size distributions increases as ϕ decreases (Fig. 3-4).

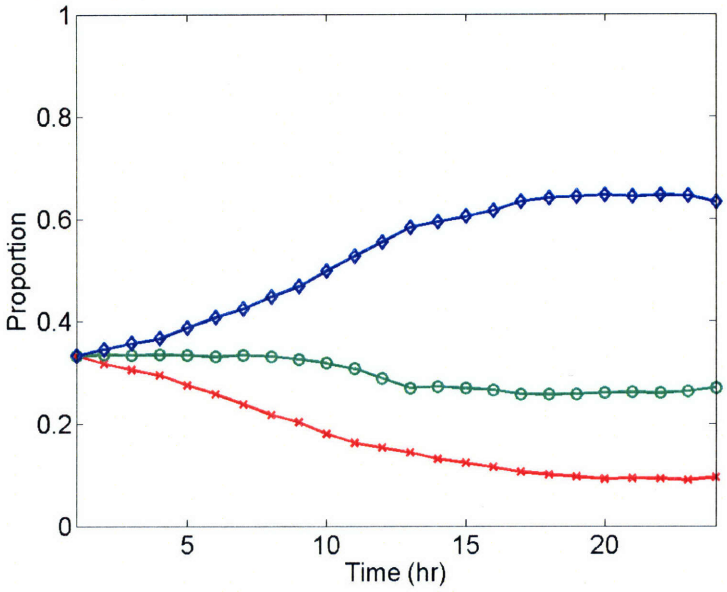

Figure 3-3: Time series of continuous proportions for three stage model. Sample observation generated with $c = 0.5$, $\delta_2 = 0.01(h^{-1})$, $\delta_3 = 0.01(h^{-1})$, and $\phi = 10^4$. The red x line corresponds to small, the green circles to medium and the blue diamonds to large phytoplankton.

3.2 Parameter estimate distribution for an artificial observation

As a first assessment of the maximum likelihood methods described here, we consider parameter estimates ($\hat{\theta}$) for observations generated from a single set of known parameters: $c = 0.5$, $\delta_2 = 0.01(h^{-1})$, $\delta_3 = 0.01(h^{-1})$, and $\phi = 10^4$. These parameters were chosen so that the effect of any single parameter would not predominate when artificial observations were created, and so that the maximum cell growth and division rates did not exceed $0.15(h^{-1})$.

Based on the properties of maximum likelihood estimation, the distribution of parameter estimates found by simulating 500 observations from the same parameters should be asymptotically unbiased and normally distributed. We can calculate the boundaries on a 95% confidence interval from this distribution by finding the 12th largest and smallest values. Such a bootstrap distribution should be well approxi-

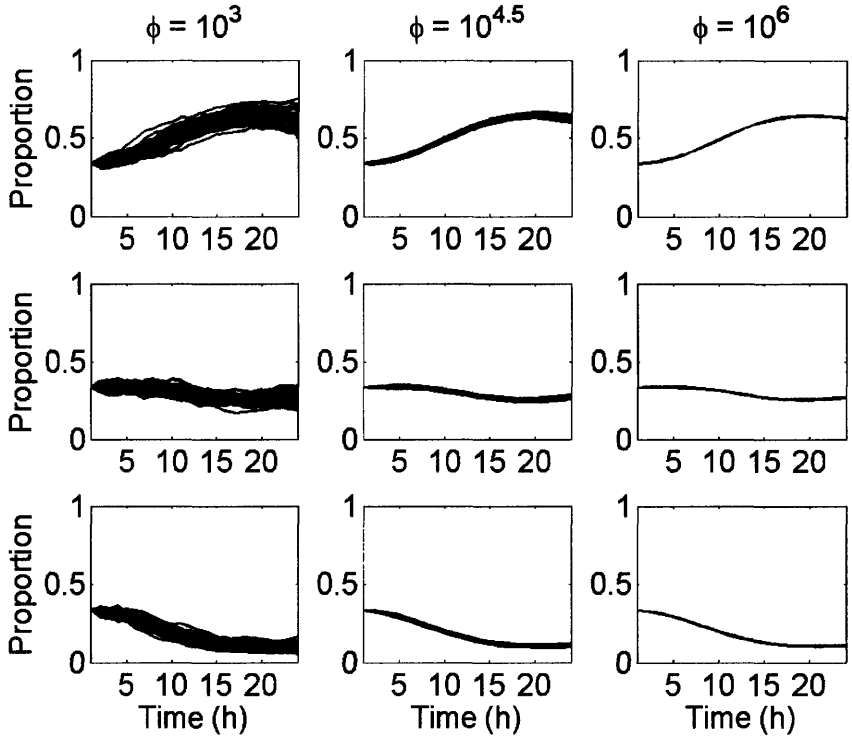

Figure 3-4: Time series of proportion in each size class, with $c = 0.5$, $\delta_2 = 0.01$ and $\delta_3 = 0.01$, for 50 model simulations at each ϕ value; all cases were initialized with $w_0^{(i)} = 1/3$. The top row shows the proportion of large phytoplankton, the middle row the proportion of medium phytoplankton, and the bottom row the proportion of small phytoplankton.

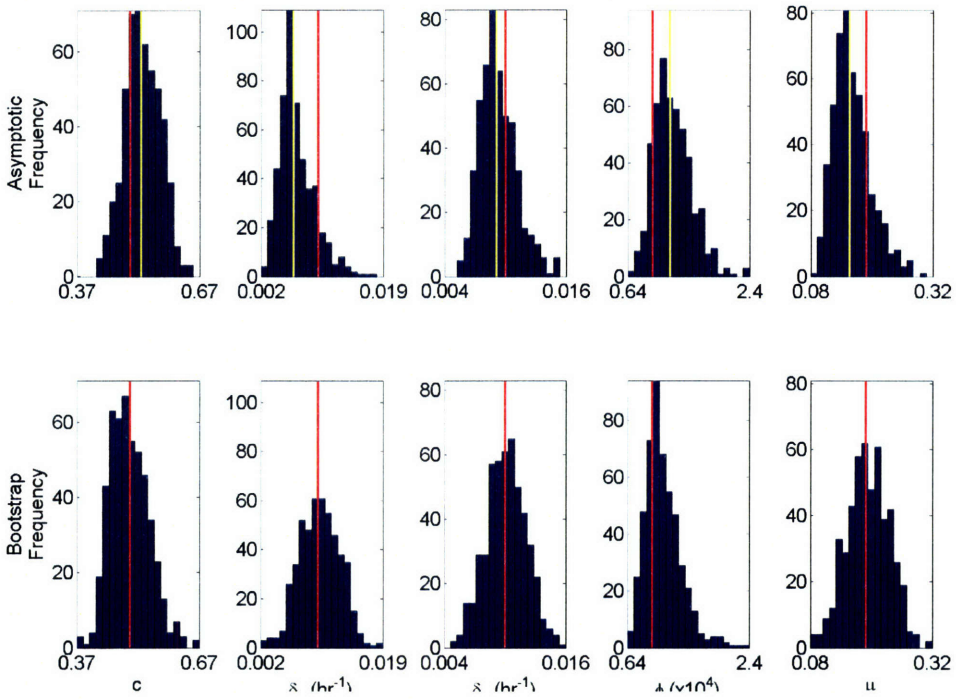

Figure 3-5: Histogram of asymptotic (from Fisher information matrix) and bootstrap parameter distributions and population growth rates for the observations generated with $c = 0.5$, $\delta_2 = 0.01(h^{-1})$, $\delta_3 = 0.01(h^{-1})$, and $\phi = 10^4$. The red lines correspond to the true parameter values, the yellow lines to the maximum likelihood estimates.

mated by the one generated from the observed Fisher information matrix (2.12).

I found that the means of the asymptotic distributions are not centered on the true value, but rather on the single maximum likelihood estimate of the parameters (Fig. 3-5). In contrast, the means of the bootstrap distributions, which are generated from independent realizations based on the "true" values, are close to the true values.

All of the asymptotic and bootstrap confidence intervals contain the true parameter values (Table 3.1). Also, the mean bootstrap parameter estimates of c, δ_2, δ_3 and the population growth rate are all accurate within 3%, while the estimate of ϕ is off by just over 15%. The asymptotic parameter estimates are less accurate, with estimates of δ_2, ϕ and the population growth rate all more than 15% off from the true

		Asymptotic Approximation		Bootstrap	
Parameter	True value	Max. Like.	95% confidence interval	Mean	95% confidence interval
c	0.50	0.53	[0.44, 0.61]	0.50	[0.43, 0.60]
$\delta_2(h^{-1})$	0.010	0.0070	[0.0035, 0.013]	0.010	[0.0053, 0.015]
$\delta_3(h^{-1})$	0.010	0.0093	[0.0064, 0.013]	0.010	[0.0064, 0.014]
ϕ	1.0e4	1.3e4	[8.5e3, 1.9e4]	1.2e4	[7.8e3, 1.8e4]
$\mu(d)^{-1}$	0.19	0.16	[0.10, 0.25]	0.19	[0.12, 0.27]

Table 3.1: Asymptotic and bootstrap mean parameter and population growth rate estimates and confidence intervals for the observation shown in Fig. 3-3.

Parameter	Asymptotic skew	Bootstrap skew
c	0.017	0.46
$\delta_2(h^{-1})$	1.1	0.0073
$\delta_3(h^{-1})$	0.67	0.044
ϕ	0.76	1.1
$\mu(d)^{-1}$	0.81	-0.034

Table 3.2: Skew of distribution of parameter estimates with bootstrap distribution and distribution from Fisher information matrix based on the observation in Fig. 3-3.

value. For a single parameter (δ_2), the asymptotic and bootstrap estimates differ by as much as 30%. The percent difference of the lower and upper bounds of the confidence intervals generated by the two methods ranges from under 5% for c to over 30% for the lower bound on δ_2.

The other factor to consider is normality. Visually, several of the the parameter estimate distributions are non-normal (Fig. 3-5). I quantified this by calculating the skew of the parameter distributions (Table 3.2). The asymptotic parameter and population growth rate estimate distributions all show high skew (over 0.5) except in the c parameter, while for the bootstrap estimate distributions, only ϕ shows high skew. I discuss the causes of this skew in the next section.

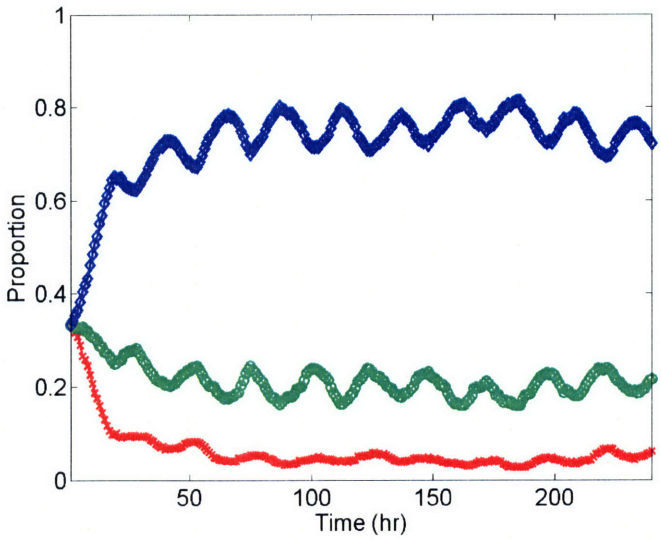

Figure 3-6: A sample observation of the proportion of plankton in each size class over 10 days, generated from the three stage model with $c = 0.5$, $\delta_2 = 0.01(h^{-1})$, $\delta_3 = 0.01(h^{-1})$, $\phi = 10^4$. The red x line corresponds to small, the green circles to medium and the blue diamonds to large phytoplankton.

3.3 Bias and normality

How do the results fit with the fact that the maximum likelihood method promises asymptotically unbiased and normally distributed parameter estimates? These parameter estimates were based on only 24 size distributions, corresponding to one day's worth of hourly observations. As the length of the observation increases, bias in parameter estimates and skew in the parameter distribution should decrease. To test this, I repeated the calculations of Sec. 3.2 and generated asymptotic and bootstrap parameter distributions and confidence intervals for a 10-day long simulated observation (Fig. 3-6). The incident light function $\mathcal{E}(t)$ was repeated each day.

The bias in parameter estimates decreases compared to the 1-day observations, with the largest bias just over 10% (Table 3.3). The confidence interval size also decreases, although in the case of δ_2, this means it no longer contains the true parameter value. As in the 1-day case, the other confidence intervals do all contain the true parameter values.

For the 10-day long observation, there was a significant decrease in skew for all of

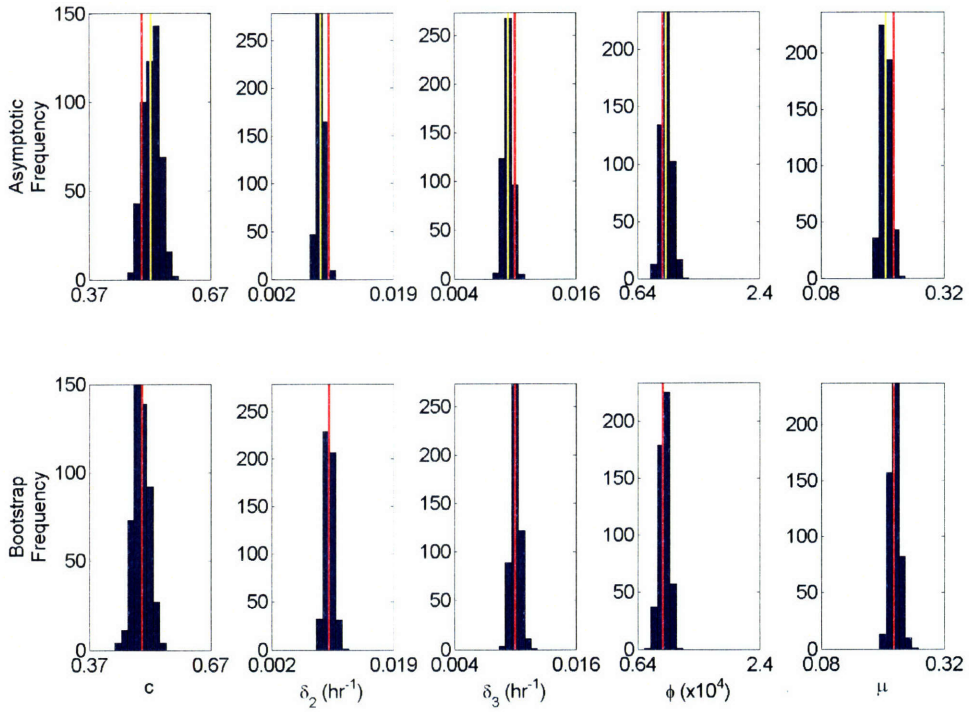

Figure 3-7: Histograms of asymptotic parameter distributions when the observations are lengthened to 10 days, with parameters $c = 0.5$, $\delta_2 = 0.01(h^{-1})$, $\delta_3 = 0.01(h^{-1})$, and $\phi = 10^4$.

		Ten day asymptotic	
Parameter	True value	Max. Like.	95% confidence interval
c	0.50	0.52	[0.49, 0.56]
$\delta_2(h^{-1})$	0.010	0.0090	[0.0080, 0.0099]
$\delta_3(h^{-1})$	0.010	0.0093	[0.0085, 0.010]
ϕ	1.0e4	1.1e4	[9.2e3, 1.2e4]
$\mu(d)^{-1}$	0.22	0.21	[0.19, 0.22]

Table 3.3: Asymptotic parameter estimates and confidence intervals when the observations are lengthened to 10 days.

Parameter	One day skew	Ten day skew
c	0.017	-0.0074
$\delta_2(h^{-1})$	1.1	0.067
$\delta_3(h^{-1})$	0.67	0.20
ϕ	0.76	0.34
$\mu(d)^{-1}$	0.81	0.17

Table 3.4: Skew of asymptotic parameter estimate distributions for 1- and 10-day long observations, based on $c = 0.5$, $\delta_2 = 0.01(h^{-1})$, $\delta_3 = 0.01(h^{-1})$, and $\phi = 10^4$.

the parameters compared to the 1-day case, particularly δ_2, whose distribution was the most non-normal in the 1-day case (Table 3.4).

The decrease in skew (Table 3.4) and decrease in bias (Table 3.3) of the ϕ estimate matches expectations about the asymptotic behavior of the model. These findings confirm that the using ten times as much data substantially improves the parameter estimate accuracy. Since the 1-day estimate of the population growth rate was close to the true value, differing by 15% (Table. 3.1), I continue to use the 1-day observations for my analysis to avoid the computational cost of using 10-day observations.

3.4 Effect of epsilon

In Sec. 2, I made the assumption that adding $\epsilon = 10^{-10}$ to each vector of proportions and renormalizing would have negligible impact on the parameter estimates. To test this, I repeated the same calculations of asymptotic parameter estimate distributions with $\epsilon = 0$ (Table 3.5). I was able to do this because the observation in question (Fig. 3-3) had no zeros in it.

Note that the while the maximum likelihood estimates of the parameter values differ between the two methods, the confidence intervals all contain the true parameter values. It is also worth noting that the estimates where $\epsilon = 10^{-10}$ are as accurate as the estimates where $\epsilon = 0$. From this, I conclude that removing zeros with $\epsilon = 10^{-10}$ does not significantly decrease the accuracy of the parameter estimates, and that this approach is therefore safe to use.

		$\epsilon = 10^{-10}$		$\epsilon = 0$	
Parameter	True value	Max. Like.	95% confidence interval	Max. Like.	95% confidence interva
c	0.50	0.53	[0.44, 0.61]	0.47	[0.40, 0.53]
$\delta_2(h^{-1})$	0.010	0.0070	[0.0035, 0.013]	0.012	[0.0082, 0.017]
$\delta_3(h^{-1})$	0.010	0.0093	[0.0064, 0.013]	0.012	[0.0096, 0.016]
ϕ	1.0e4	1.3e4	[8.5e3, 1.9e4]	1.4e4	[9.3e3, 1.9e4]
$\mu(d)^{-1}$	0.19	0.16	[0.10, 0.25]	0.23	[0.18, 0.31]

Table 3.5: Maximum likelihood parameter estimates and asymptotic confidence intervals when $\epsilon = 10^{-10}$ and when $\epsilon = 0$.

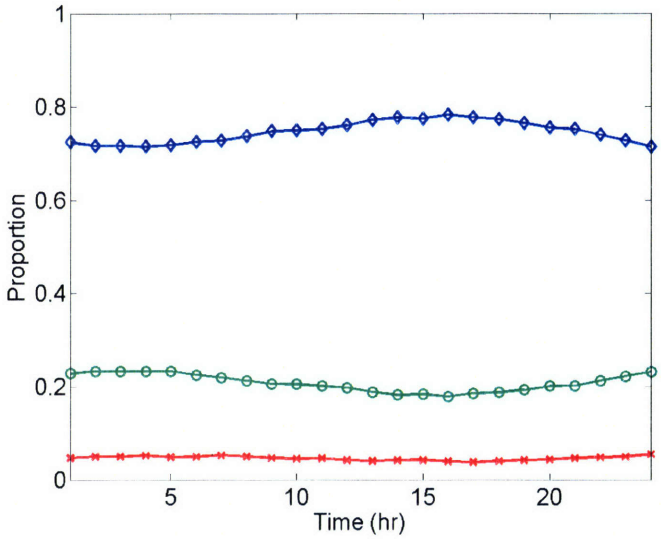

Figure 3-8: A sample observation of the proportion of plankton in each size class with $\epsilon = 10^{-10}$, $c = 0.5$, $\delta_2 = 0.01(h^{-1})$, $\delta_3 = 0.01(h^{-1})$, and $\phi = 10^4$, from the last day of a 100-day observation. The red x line corresponds to small, the green circles to medium and the blue diamonds to large phytoplankton.

3.5 Equilibrium

In the previous analyses, the initial population composition is unrealistic, with an equal proportion in each size class. By generating a long (100 day) observation and using only the last day, I create an observation whose population composition begins at equilibrium (Fig. 3-8). In contrast with the transient observation (Fig. 3-3), the size distribution changes very little over the course of a day. Overall, the parame-

Parameter	True value	Transient Max. Like.	95% confidence interval	Equilibrium Max. Like.	95% confidence inter
c	0.50	0.53	[0.44, 0.61]	0.45	[0.34, 0.59]
$\delta_2(h^{-1})$	0.010	0.0070	[0.0035, 0.013]	0.011	[0.0085, 0.015]
$\delta_3(h^{-1})$	0.010	0.0093	[0.0064, 0.013]	0.012	[0.0090, 0.015]
ϕ	1.0e4	1.3e4	[8.5e3, 1.9e4]	1.5e4	[9.3e3, 2.1e4]
$\mu(d)^{-1}$	0.19	0.16	[0.10, 0.25]	0.22	[0.17, 0.29]

Table 3.6: Maximum likelihood parameter estimates and asymptotic confidence intervals from the first (transient) and 100th (equilibrium) day of an observation whose population distribution begins with a uniform distribution and reaches equilibrium.

ter estimates are as accurate as they were when the initial population distribution was uniform, and the confidence intervals still contain the true parameter values (Table 3.6).

3.6 Accuracy across parameter space

The results in the previous section were all based on a single set of parameters. To evaluate whether these methods work in general, we need to test other points in the parameter space. To accomplish this, I created a grid of parameter values. These values are equally spaced on a log scale, so that they cover a large part of the parameter space. The grid consists of all combinations of the following parameter values:

$$c = 0.01,\ 0.0251,\ 0.0631,\ 0.1585,\ 0.3981,\ 1.0 \tag{3.4}$$

$$\delta_2 = 0.001,\ 0.0032,\ 0.01,\ 0.0316,\ 0.1,\ 0.3162 \tag{3.5}$$

$$\delta_3 = 0.001,\ 0.0032,\ 0.01,\ 0.0316,\ 0.1,\ 0.3162 \tag{3.6}$$

$$\phi = 10^3,\ 3.16 \times 10^4,\ 10^6 \tag{3.7}$$

For each parameter combination, I generated an artificial observation based on those values, found the maximum likelihood parameter estimate and generated a confidence interval with the Fisher information matrix. I also used the maximum

likelihood estimate to calculate a population growth rate μ.

3.6.1 Total Error

To get an overall picture of the accuracy of the parameter estimates, I calculated a total error term, that measures the distance between the true parameter values and the maximum likelihood estimate. Because the parameters cover several orders of magnitude, I use relative error, rather than absolute error. This total error is given by

$$T_{err} = \sqrt{(|c - \hat{c}|/c)^2 + (|\delta_2 - \hat{\delta_2}|/\delta_2)^2 + (|\delta_3 - \hat{\delta_3}|/\delta_3)^2 + (|\phi - \hat{\phi}|/\phi)^2}. \quad (3.8)$$

T_{err} is largest when ϕ is small (Fig. 3-9). Error is also higher when δ_2 and δ_3 are small, and when c is large, but these effects are less prominent. There are also a few parameter combinations, which appear to be randomly distributed, that have much higher total error than their neighbors. These estimates are likely inaccurate due to difficulties with the optimization routine, and do not represent a true pattern.

3.6.2 Relative error and confidence interval size

In addition to overall trends in error, I wanted to examine trends in error of individual parameters. To do this, I calculated the error and confidence interval size for each parameter, again with the relative error rather than absolute error.

The relative error formula is

$$p_{err} = (p - \hat{p})/p, \quad (3.9)$$

where p is the true parameter value, and $\hat{p}$ is the parameter estimate.

I also wanted to capture the precision of each estimate. To do this, I calculated the relative size of the confidence interval for each parameter, with

$$p_{ci} = (p_{high} - p_{low})/p, \quad (3.10)$$

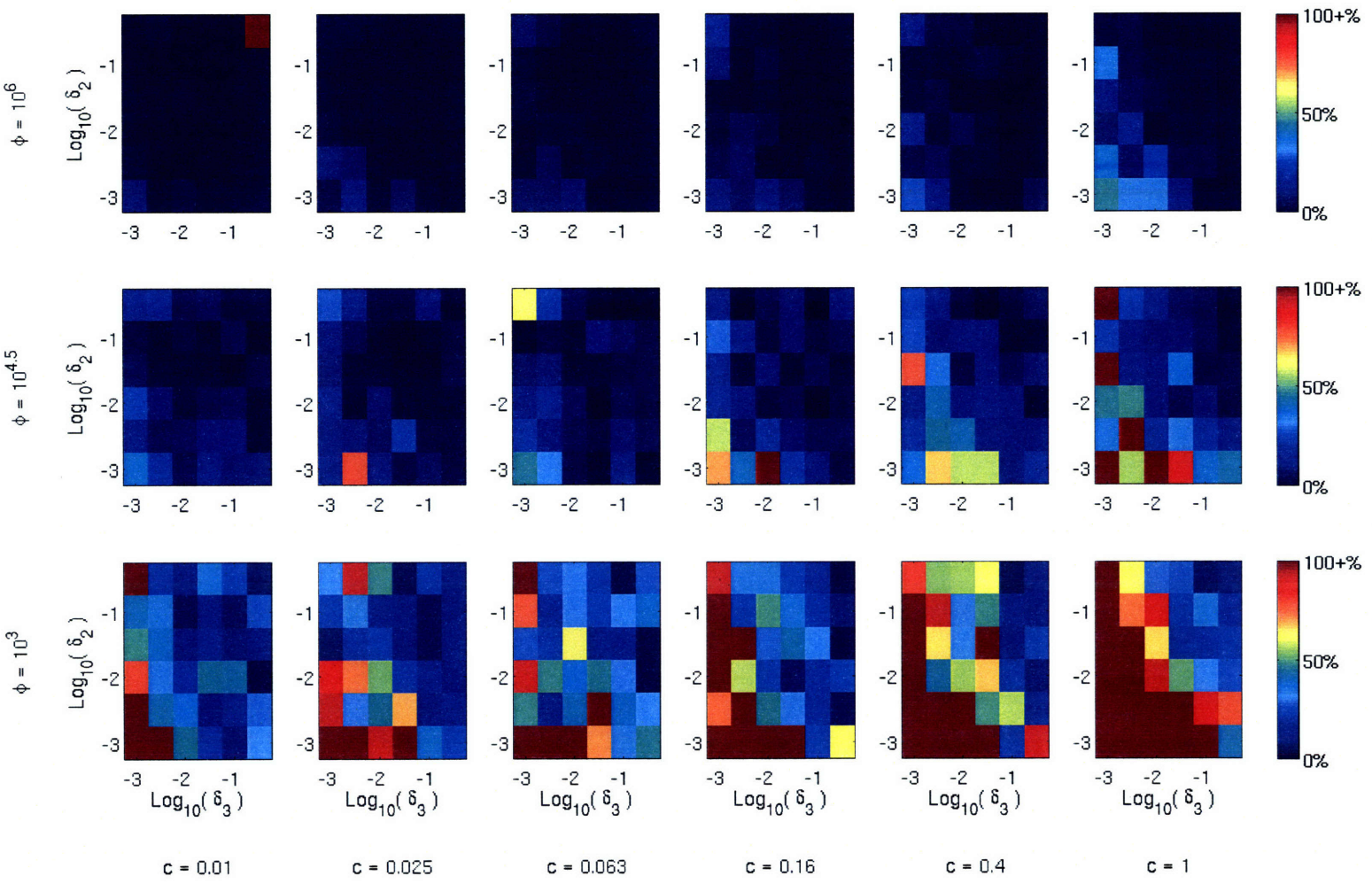

Figure 3-9: Total relative error of the parameter estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

where p_{high} and p_{low} are the high and low ends of the confidence interval and p is the true parameter value.

For each of the model parameters and for μ, the pattern of relative errors shows a similar response to the parameters as the pattern of confidence interval sizes (Figs. 3-10 to 3-17).

For estimates of c, δ_2 and δ_3, the relative error increases with increases in ϕ and c, though the effect of c on the relative errors of δ_2 and δ_3 is small.

Relative errors increase as δ_3 decreases for both δ_2 and δ_3, while only δ_2 has relative error that increases in response to decreases in δ_2.

In the case of the parameter ϕ, relative errors appear to be large and highly variable, but uncorrelated with the parameters themselves, including ϕ. This large relative error is not surprising, given that the ϕ confidence intervals are very large throughout the parameter space.

The relationships between the relative error of μ and the model's parameter values are similar to the ones described above. Relative errors appear to be correlated with ϕ, c, and to a lesser extent negatively correlated with δ_2 and δ_3 (Fig. 3-18 and 3-19). In addition, the relative error of the population growth rate μ appears to be correlated with the magnitude of the population growth rate itself (Fig. 3-20 and 3-21). This is unsurprising, since μ is positively correlated with δ_2 and δ_3, and (when c is small) negatively correlated with c. The precision parameter ϕ is positively correlated with the accuracy of μ, even though the formula for μ does not depend on ϕ. This is not surprising, since ϕ can affect μ through its influence on the accuracy of the other parameter estimates.

3.7 Implications

Although the errors and confidence intervals vary depending on the parameter, there are some clear patterns. The most obvious is the correlation between smaller ϕ and larger confidence intervals/relative errors. This makes intuitive sense, since the ϕ parameter is inversely related to the variance of the Dirichlet distribution. Smaller

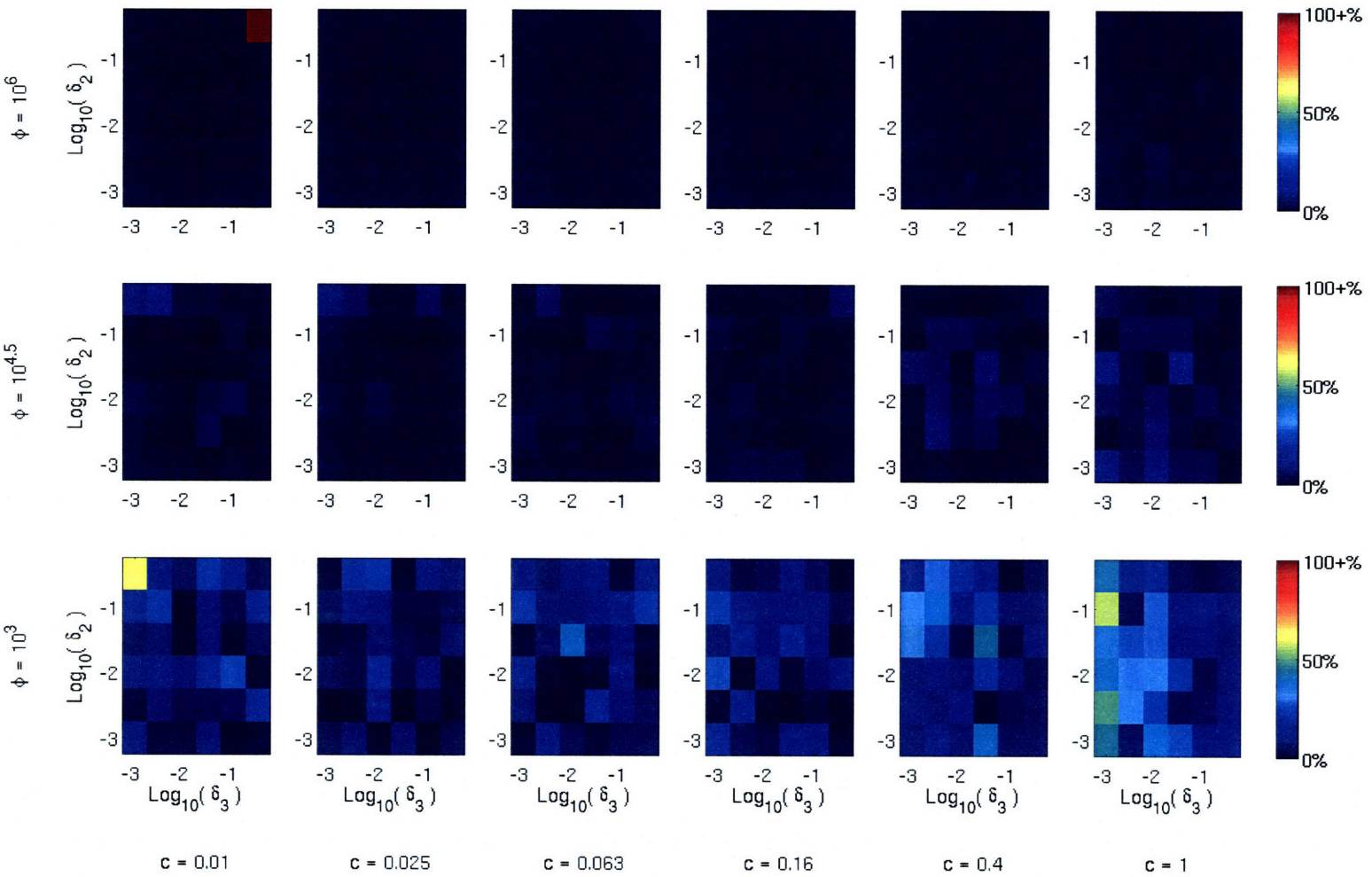

Figure 3-10: Relative error of c estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space. In each sub-plot c and δ_2 are held constant. The c parameter increases as you move up and the δ_2 parameter as you move right across the sub-plots. The color of each square represents the relative error, ranging from 0% (dark blue) to at least 100% (dark red).

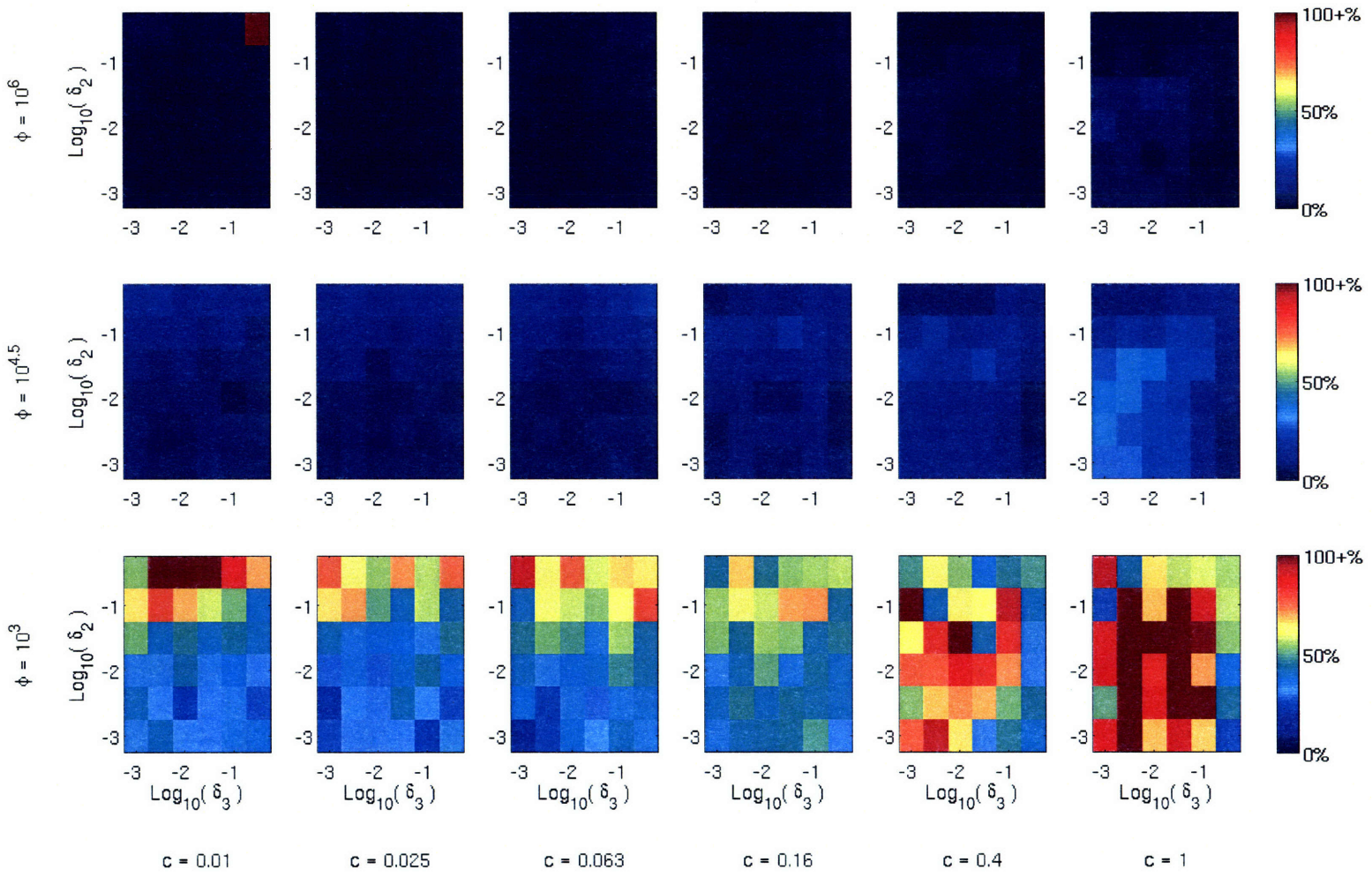

Figure 3-11: Relative confidence interval size of the c estimates, details as in Fig. 3-10

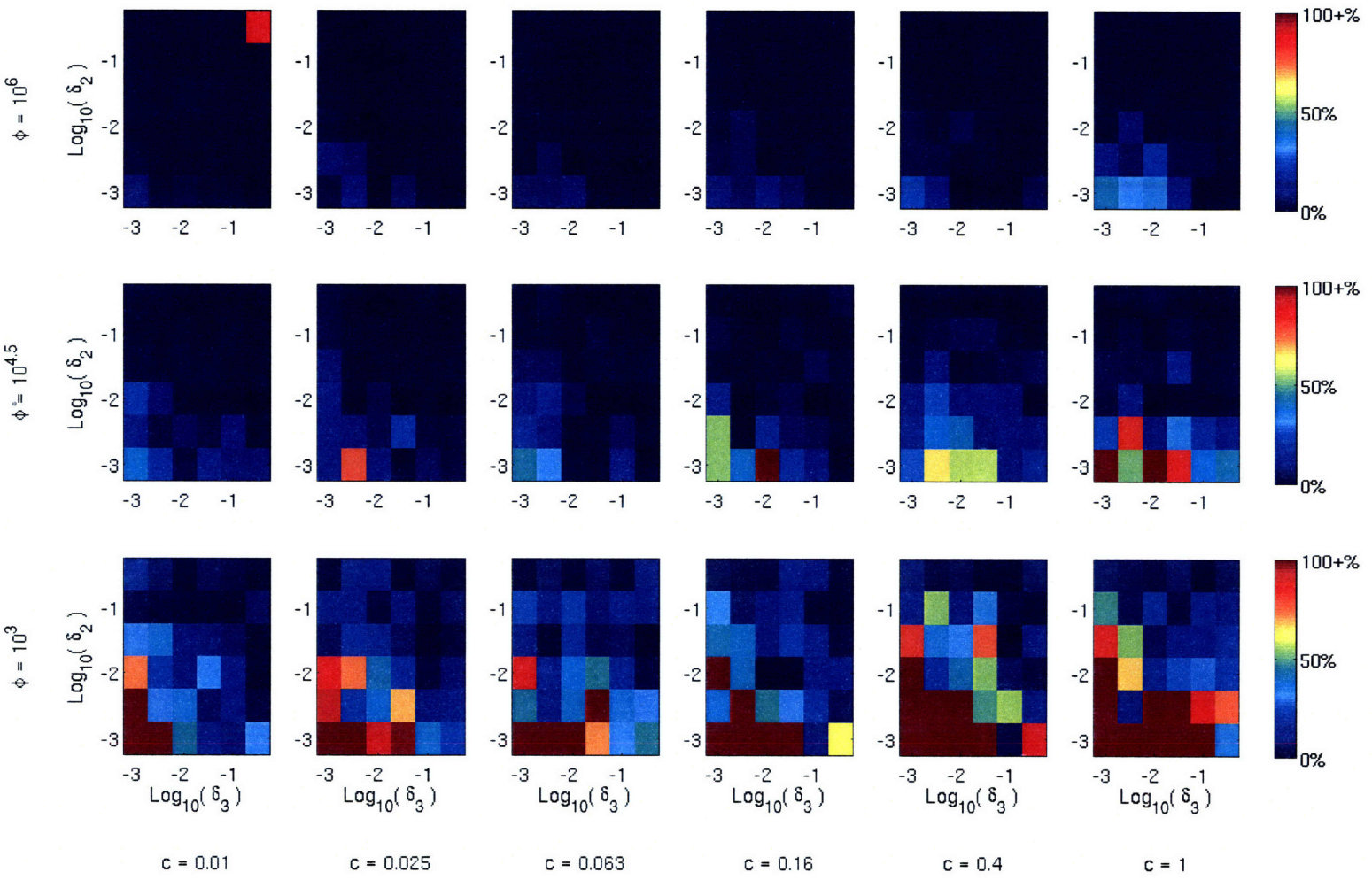

Figure 3-12: Relative error of δ_2 estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

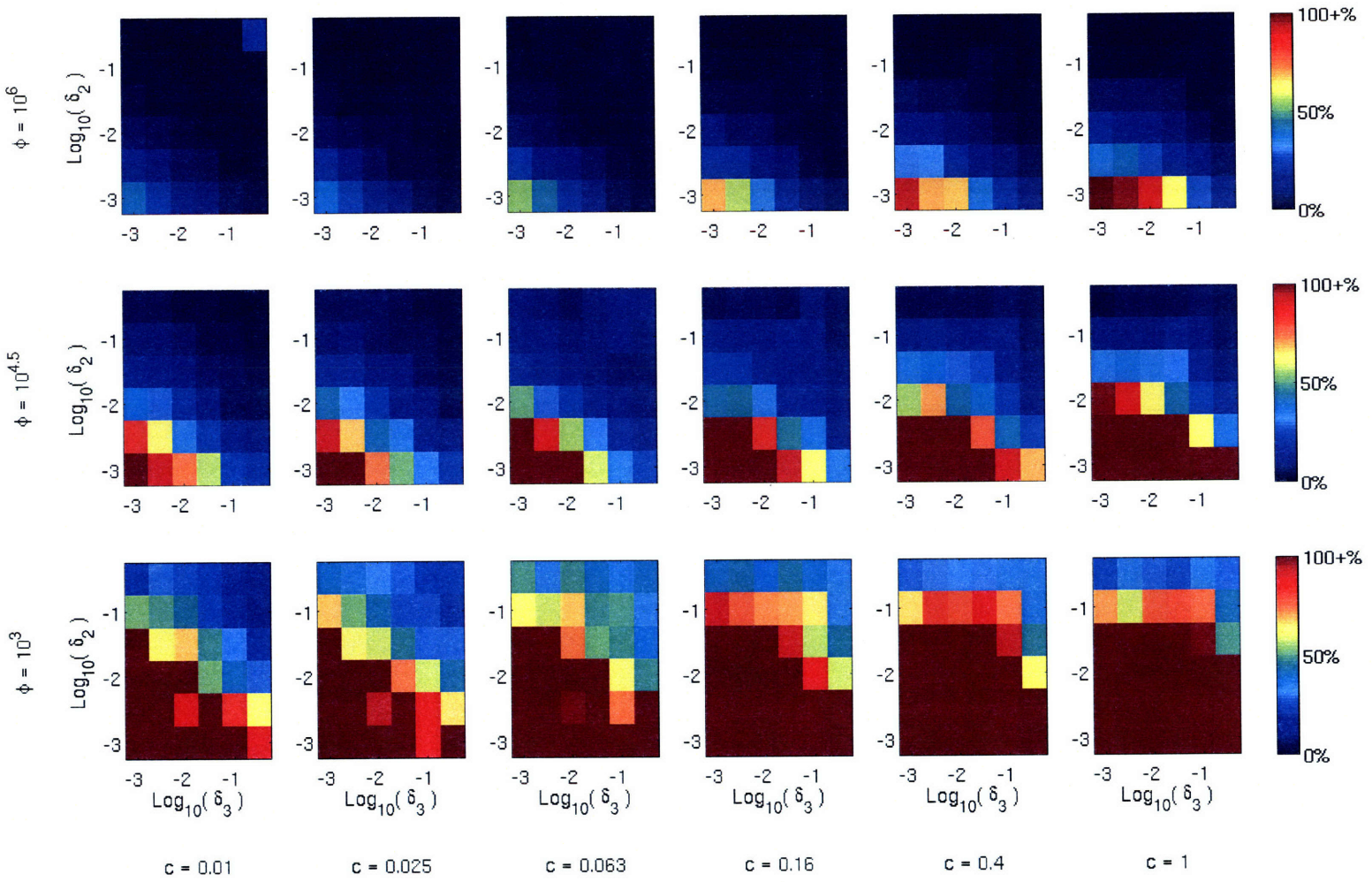

Figure 3-13: Relative confidence interval size of the δ_2 estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

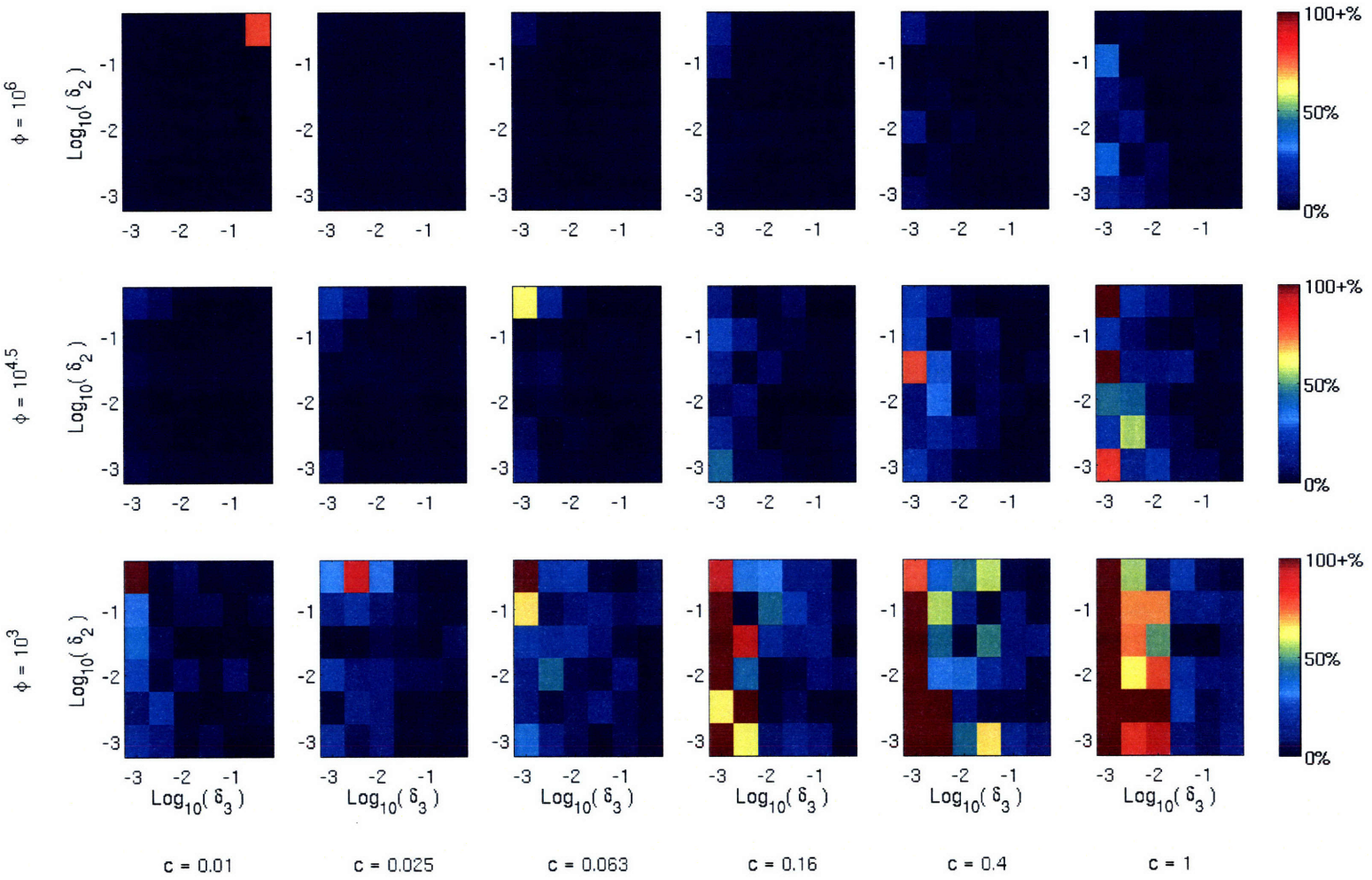

Figure 3-14: Relative error of δ_3 estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

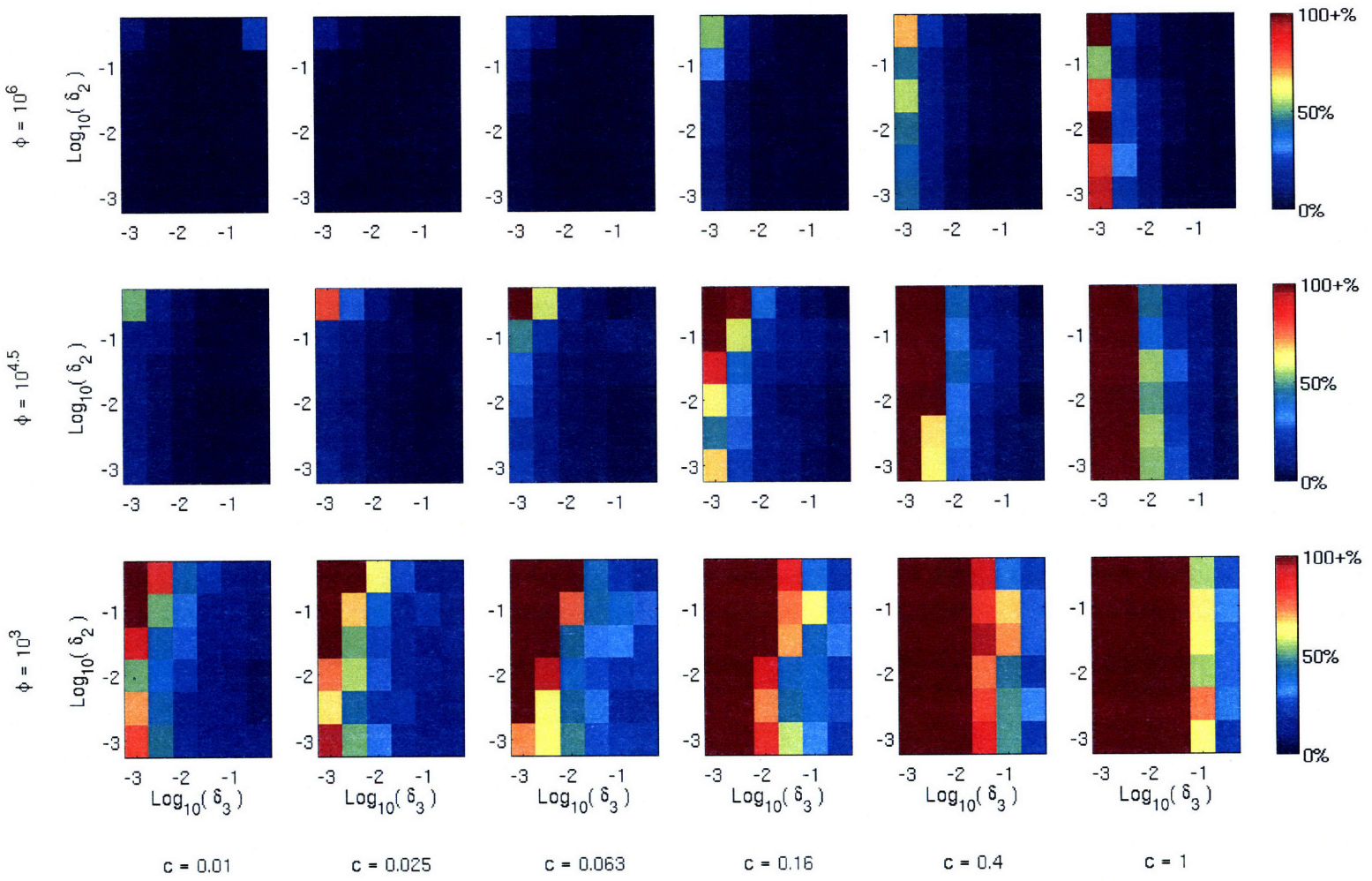

Figure 3-15: Relative confidence interval size of the δ_3 estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

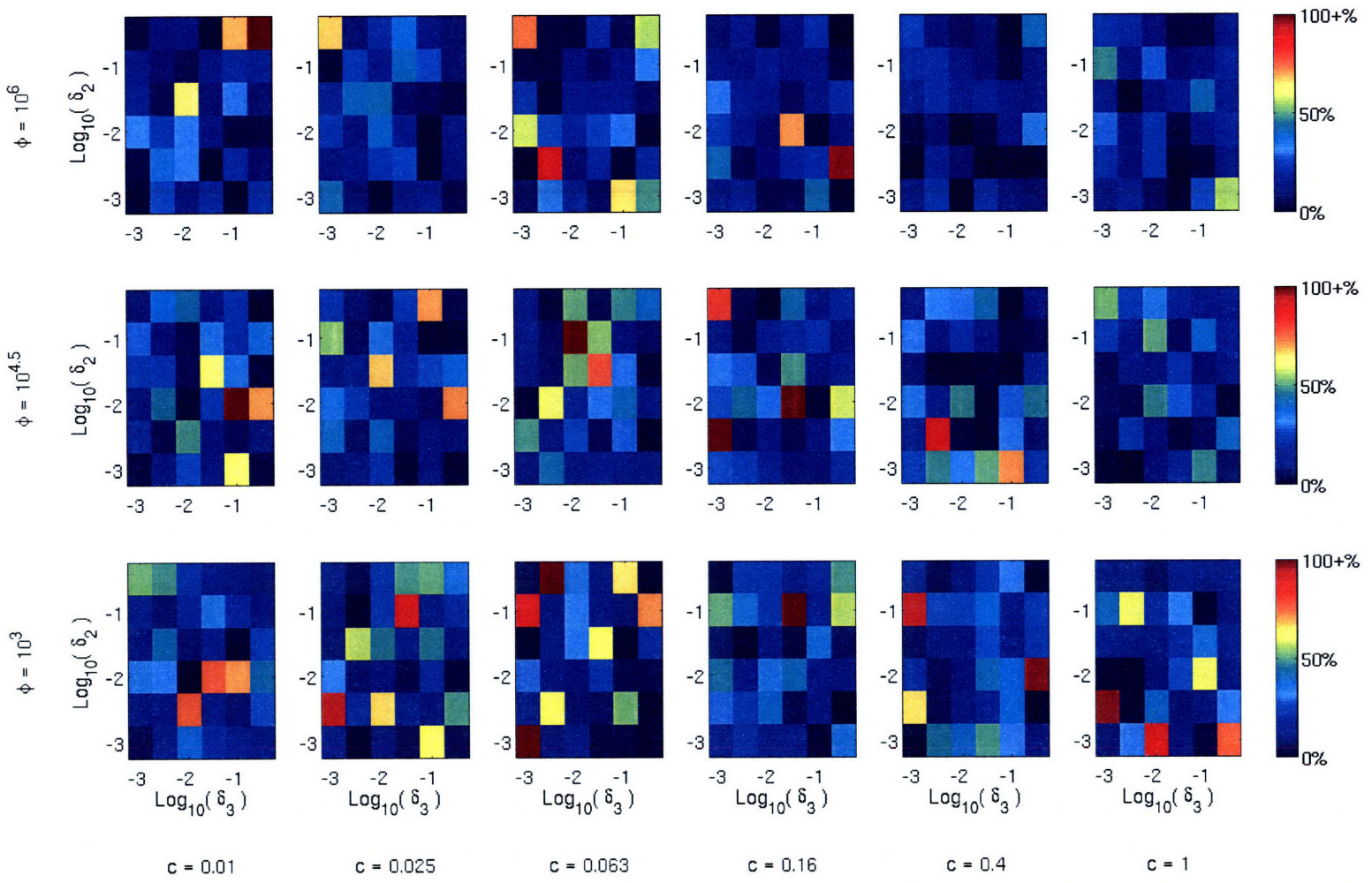

Figure 3-16: Relative error of ϕ estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

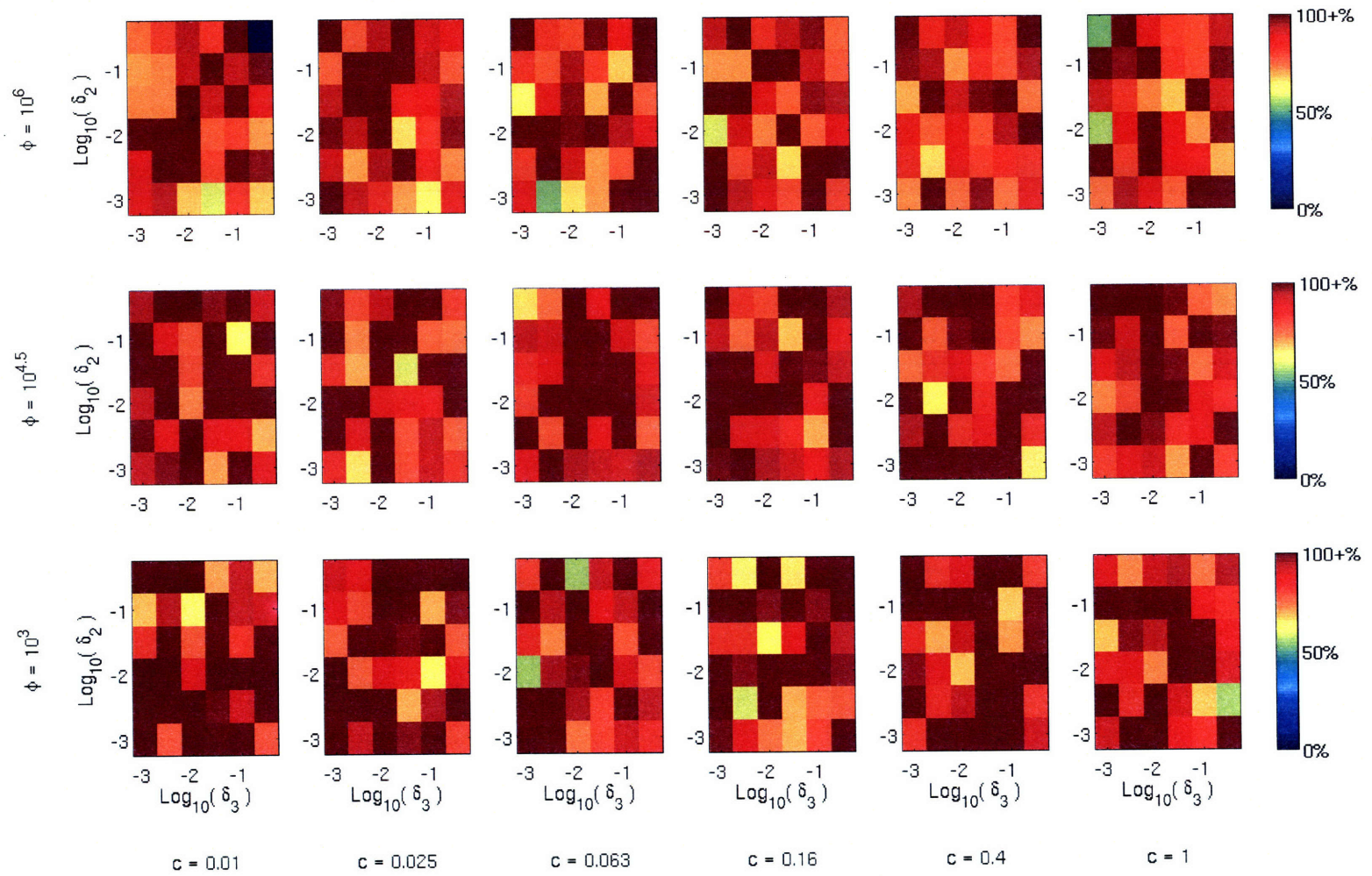

Figure 3-17: Relative confidence interval size of the ϕ estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

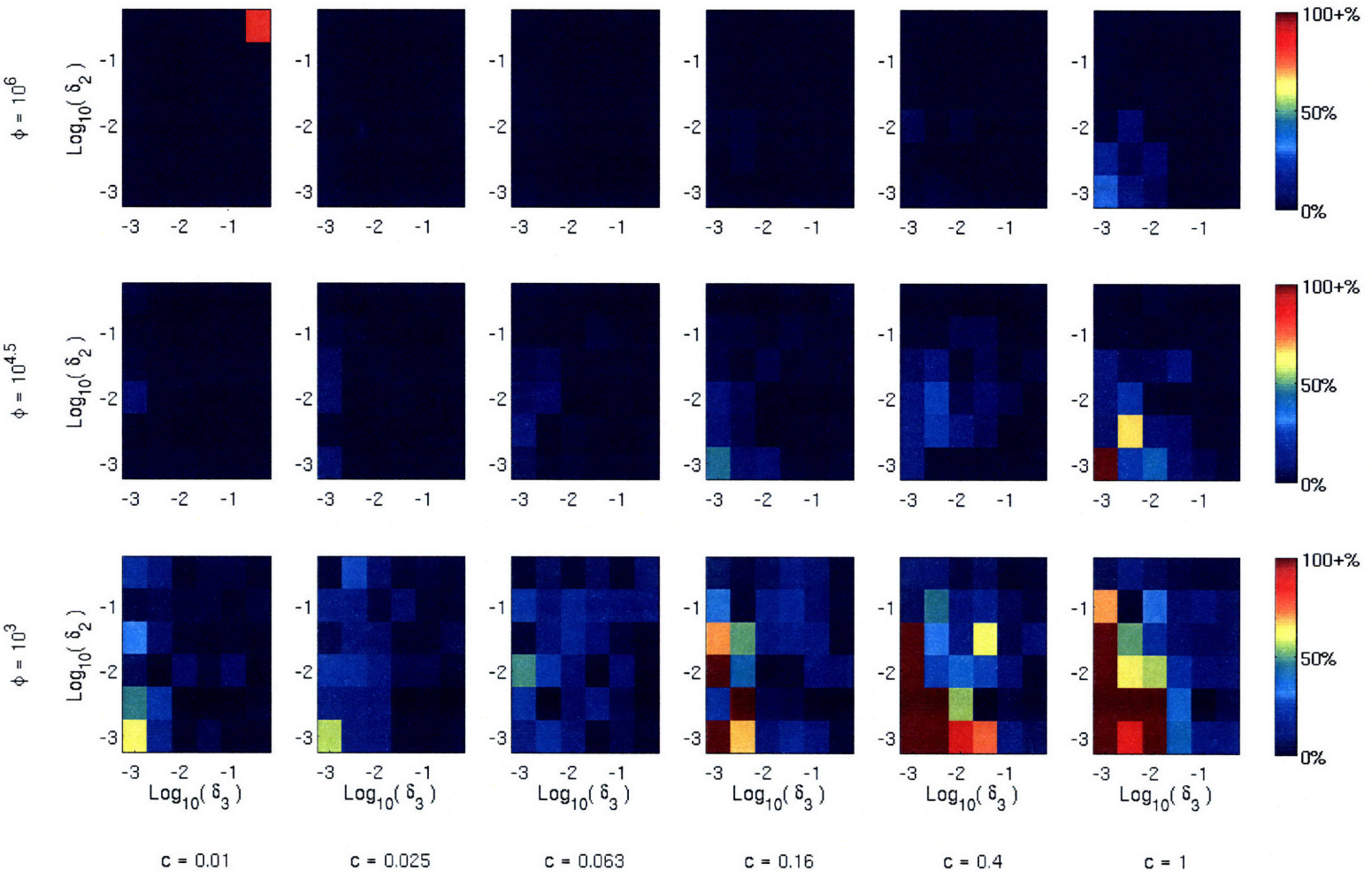

Figure 3-18: Relative error of population growth rate estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

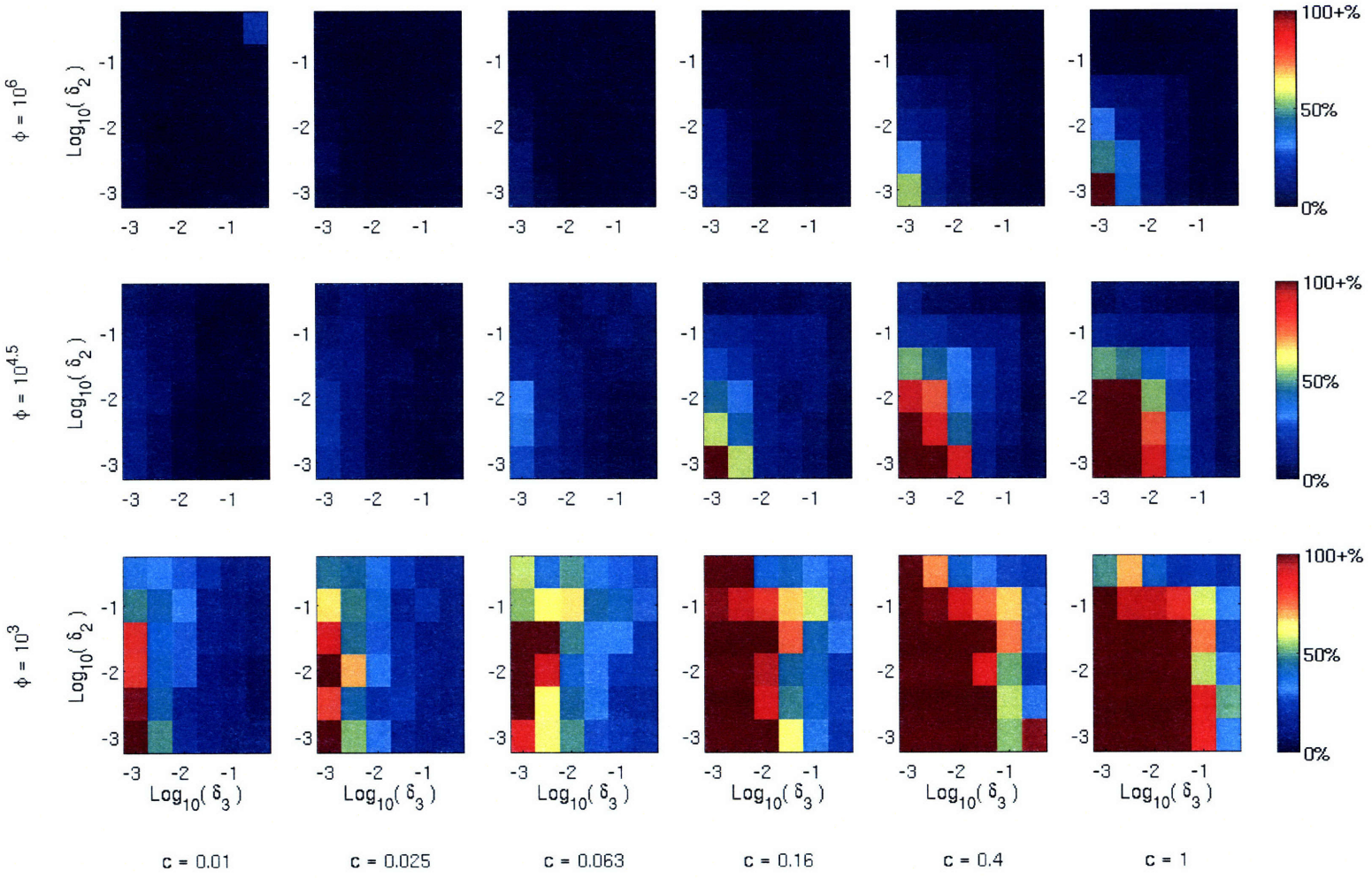

Figure 3-19: Relative confidence interval size of the population growth rate estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

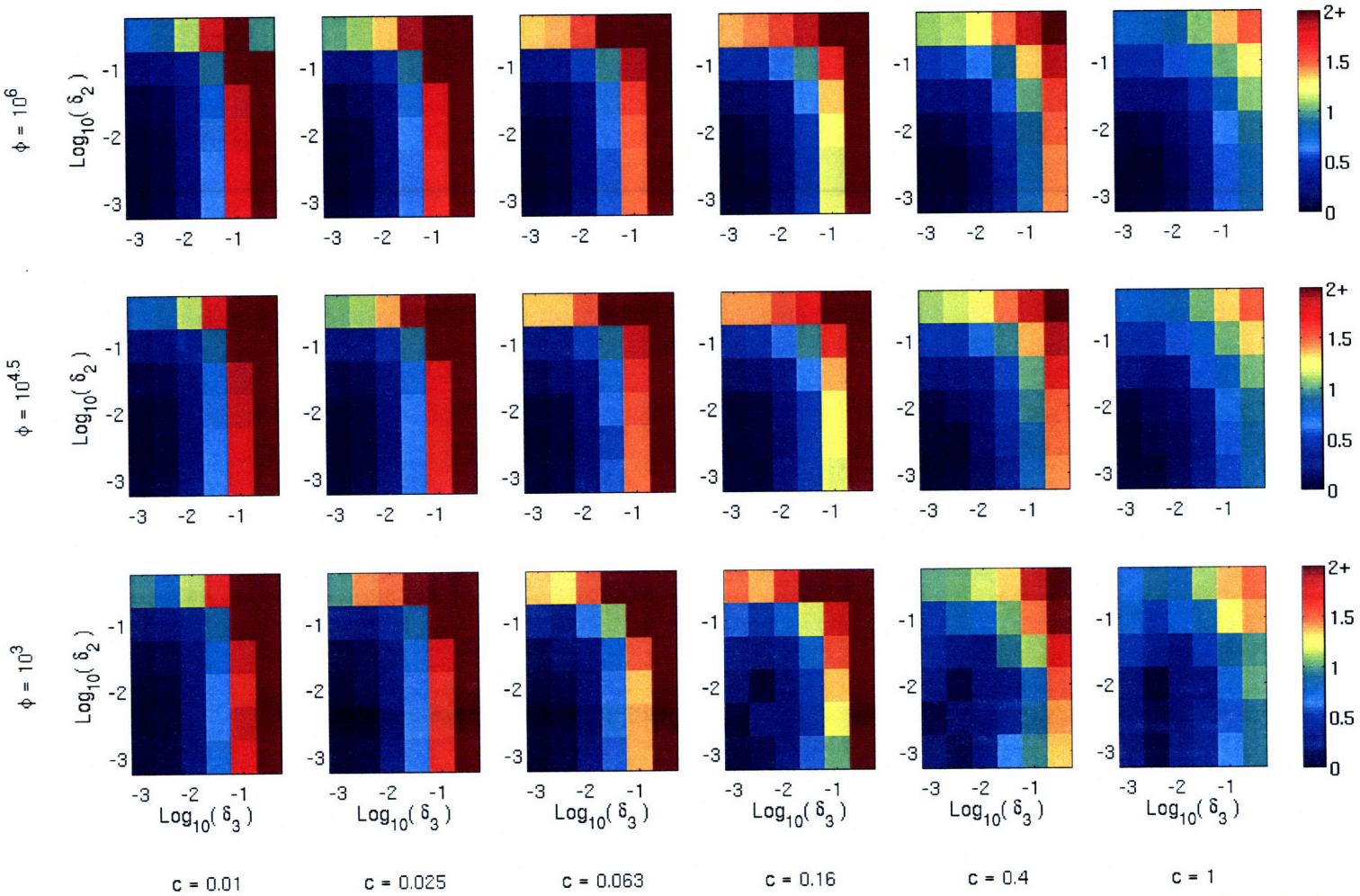

Figure 3-20: Population growth rate estimates, shown as c, δ_2, δ_3 and ϕ vary across the parameter space.

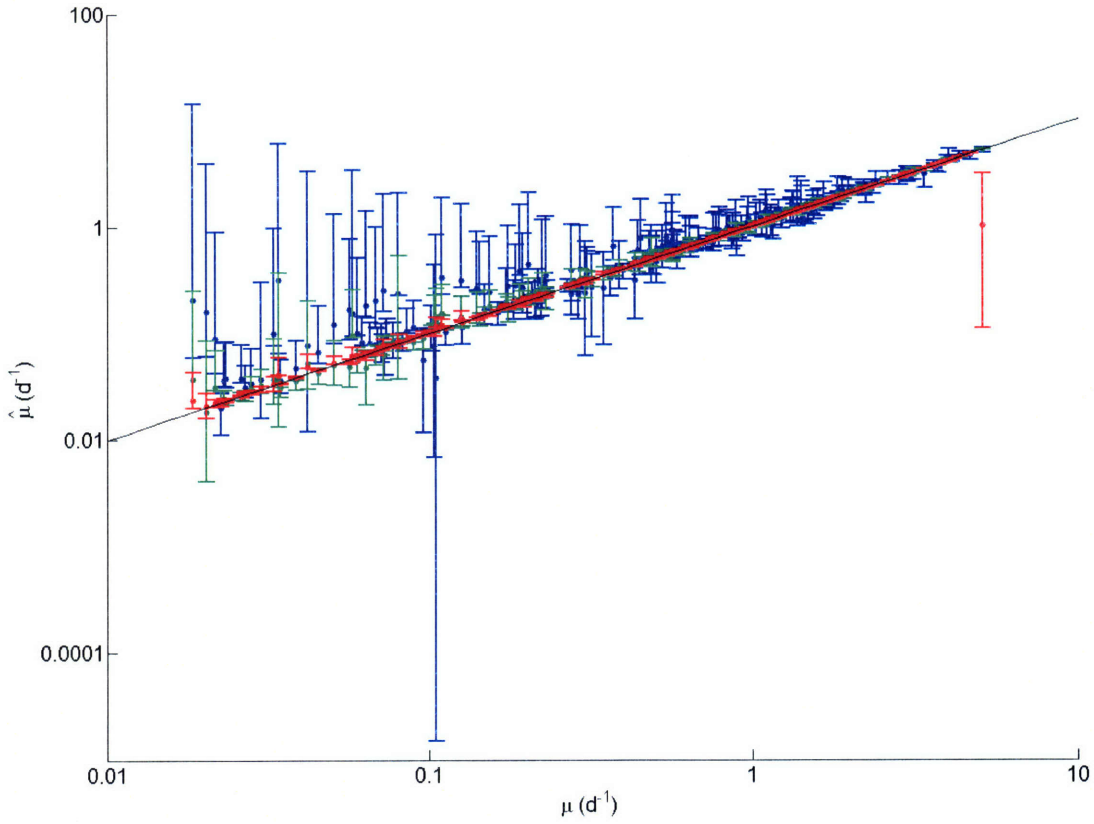

Figure 3-21: Population growth rate estimates $\hat{\mu}$, and confidence intervals as a function of the true growth rate μ. Data is shown on a log-log scale to focus on relative error. Red points correspond to $\phi = 10^6$, green to $\phi = 10^4$ and blue to $\phi = 10^3$.

values of ϕ correspond to higher variance, which effectively adds more "noise" to the artificial observations, making the other parameters harder to estimate.

To a lesser degree, there appears to be a negative correlation between δ_2 and δ_3 and relative model parameter errors. One explanation for this is that δ_2 and δ_3 reflect the phytoplankton cell division rate: when very few plankton are dividing, the population doesn't change much from one time step to the next, making the parameters harder to estimate.

There also appears to be a small correlation between c and relative error. The c parameter is found in the cell growth rate function γ in (3.1). Larger c values correspond to smaller maximum γ and, for the range of parameters in the grid above (Eqns 3.4 to 3.7), smaller differences between minimum and maximum values (weaker forcing). With these data sets, we cannot distinguish the effects of lower total incident radiation, and smaller differences between minimum and maximum light levels. One way to test this would be to compare the effects of high and low light levels which do not change over the course of a day, or use flat and steep incident radiation curves whose average values are the same.

3.7.1 Accuracy of confidence intervals

Another metric to consider is the accuracy of the confidence intervals themselves. In theory, we expect 95% of the confidence intervals to contain the true parameter values. In my results (Fig. 3-22), all four true model parameter values are within their confidence intervals only 73% of the time. The true population growth rate falls within the estimated population growth rate confidence interval 89% of the time (Fig. 3-23). This proportion differs from the proportion of parameter estimate confidence intervals containing the true value because the population growth rate is a function of all 4 parameters, and may be more sensitive to one parameter than another, and therefore may be within the confidence interval even when one of the parameters used to calculate it is outside of its confidence interval. Of the 648 parameter combinations, there are only 2 where the true population growth rate falls outside its confidence interval and the corresponding true parameter values do *not* fall outside of their

confidence intervals.

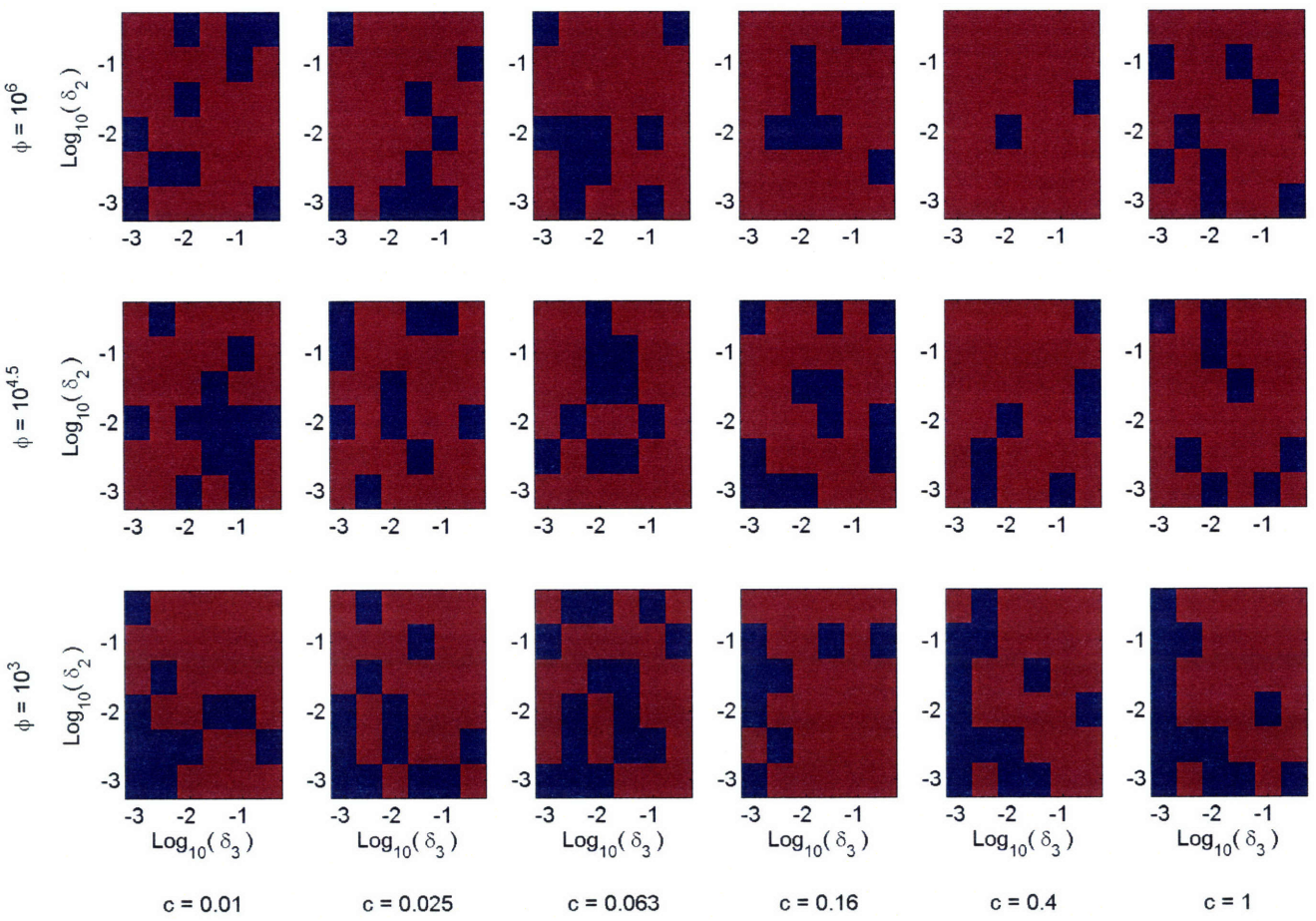

Figure 3-22: Accuracy of parameter estimate confidence intervals as a function of$c, \delta_2, \delta_3, \phi$. Cells where the confidence intervals for each parameter contain the true value are marked as red, cells where at least one confidence interval does not contain the true parameter value are marked as blue.

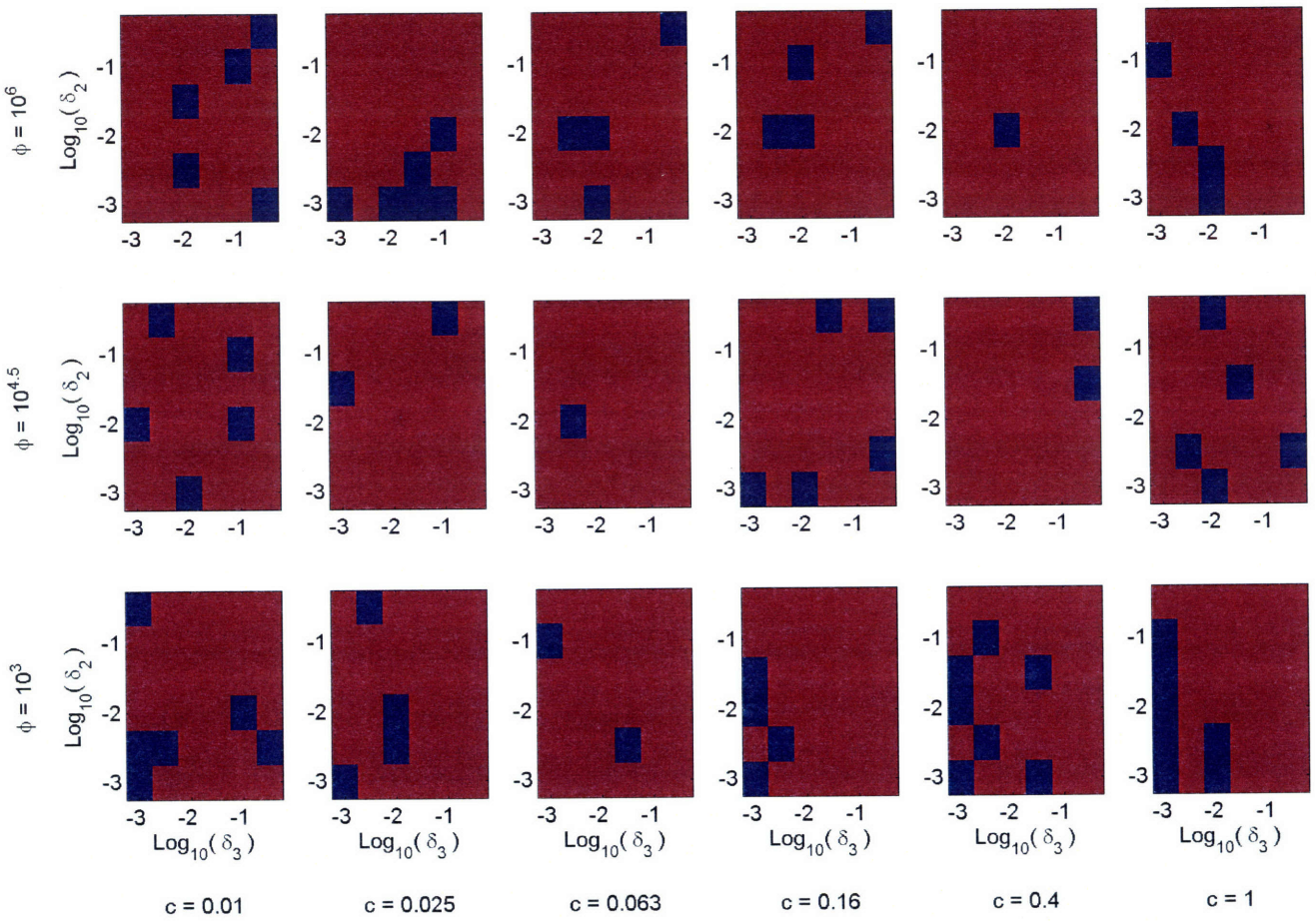

Figure 3-23: Accuracy of population growth rate confidence interval as a function of $c, \delta_2, \delta_3, \phi$. Cells where the population growth rate confidence interval contains the true population growth rate are marked as red, cells where at least one confidence interval does not contain the true population growth rate are marked as blue.

Chapter 4

Full model

Having tested the maximum likelihood methods and assumptions with a basic three stage model, I moved on to the more realistic model of phytoplankton growth laid out by Sosik et al. [12].

4.1 Model description

I classified individuals into 57 size classes of volume v_i (e.g., Fig. 4-4), where

$$v_i = v_{min} 2^{(i-1)\Delta_v},\ v_{min} = 2^{-5}\mu m^3,\ \Delta_v = 0.125. \tag{4.1}$$

For the full model, we use a life cycle scheme which allows one of three things to happen at each time step (Fig. 4-1). An individual phytoplankton cell can grow one size class (growth); remain the same size (stasis); or divide, moving to the size class 1/2 as large as the original.

The fraction of dividing cells in size class i at time t, is given by

$$\delta_i(t) = \begin{cases} 0 & 0 \le t < t_\delta \\ \frac{av_i^b}{1+av_i^b}\delta_{max} & t_\delta \le t < 24 \end{cases} \tag{4.2}$$

a, b, and δ_{max} are parameters that will be estimated in the model. t_δ is set to 6 h, and is the number of hours after dawn that the plankton begin dividing. The values

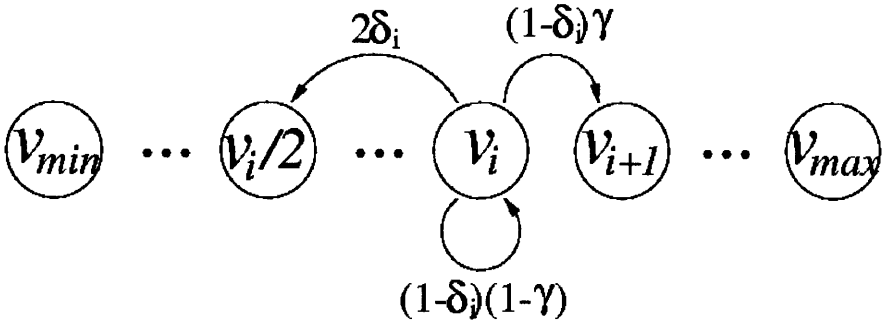

Figure 4-1: Life cycle structure of full model described by Sosik et al. [12].

of a and b define a wide range of division curves.

The fraction of non-dividing cells that grow one size class during one time step dt, $\gamma(t)$ is given by

$$\gamma(t) = (1 - e^{-E(t)/E^*})\gamma_{max} \tag{4.3}$$

Here, E^* and γ_{max} are parameters to be fit. $E(t)$ is the incident radiation, a source of external forcing.

The transition matrix $\mathbf{A}(t)$ summarizes these transitions and can be used to project the population state forward one time step dt:

$$\mathbf{A}(t) = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & \cdots & a_{1,j-1} & & & \\ a_{2,1} & a_{2,2} & & & & a_{2,j} & & \\ & \ddots & \ddots & & & & \ddots & \\ & & \ddots & \ddots & & & & a_{m+1-j,m} \\ & & & a_{i+1,i} & a_{i,i} & & & \\ & & & & \ddots & \ddots & & \\ & & & & & \ddots & a_{m-1,m-1} & \\ & & & & & & a_{m,m-1} & a_{m,m} \end{pmatrix} \tag{4.4}$$

where $j = 1 + \frac{1}{\Delta_v} = 9$.

The growth terms are given by

$$a_{i+1,i}(t) = \gamma(t)[1 - \delta_i(t)],\ i = 1, \ldots, m-1. \tag{4.5}$$

The cell division terms for large ($i \geq j$) phytoplankton are

$$a_{i+1-j,i}(t) = 2\delta_i(t),\ i = j, \ldots, m \tag{4.6}$$

Small phytoplankton (i=2,..., j-1) are less than twice as big as the smallest size class, and so after division are put in the smallest size class. The division term for these size classes is

$$a_{1,i}(t) = 2\delta_i(t). \tag{4.7}$$

The stasis (neither division or growth) terms are given by

$$a_{i,i}(t) = \begin{cases} (1-\gamma(t))(1-\delta_i(t)) + 2\delta_i(t), & i = 1, \\ (1-\gamma(t))(1-\delta_i(t)), & i = 2, \ldots, m-1, \\ 1 - \delta_i(t), & i = m. \end{cases} \tag{4.8}$$

To get an hourly projection from time t to $t+1$, I use

$$\mathbf{B}(t) = \prod_{i=0}^{(1/dt)-1} \mathbf{A}(t + idt) \tag{4.9}$$

In total, there are six parameters to estimate: a, b, δ_{max}, γ_{max}, E^*, and ϕ.

I calculated a new population distribution with the same method described in Chap. 2, by projecting forward one hour with $\mathbf{B}(t)$, normalizing to get $\mathbf{v}_{t+1}$, then adding process error to get $\mathbf{w}_{t+1}$ by drawing from a Dirichlet distribution with expected values of $\mathbf{v}_t$ and a precision parameter ϕ:

$$\mathbf{v}_{t+1} = \frac{\mathbf{B}(t)\mathbf{w}_t}{||\mathbf{B}(t)\mathbf{w}_t||}, \tag{4.10}$$

$$\mathbf{w}_{t+1} = \mathrm{Dir}(\phi \mathbf{v}_{t+1}). \tag{4.11}$$

The deterministic population growth rate is calculated from the hourly projection matrices $\mathbf{B}(t)$, with formula 2.9 in Chap. 2.

4.2 Optimization using maximum likelihood

My goal in the scenarios outlined below is to see if the maximum likelihood methods used in the three-stage case can give good estimates of the population growth rates and parameters in a larger and more complex model, which I hope to use to estimate the population growth rates of real populations. Since the optimization routine I have been using has problems finding the global maximum, I have chosen to use the true parameter values as a starting point for the optimizations with artificial observations and known parameter values. Assuming that this procedure finds near optimal values, this approach will tell us whether the method is working, given a functioning optimization routine.

4.3 Test scenarios

I wanted to test this model's ability to estimate θ and μ over a range of parameter values. In the three stage model, I was able to examine a grid of points covering a large area of the parameter space. The larger number of parameters in the full model makes this method impractical. Instead, I focus on a set of four scenarios, based on a combination of high and low cell growth and division rates.

The high growth scenarios (HG) use $\gamma_{max} = 0.25\ (10\ min)^{-1}$ and $E^* = 200\ rel.\ units.$ The low growth scenarios (LG) use $\gamma_{max} = 0.05\ (10\ min)^{-1}$ and $E^* = 10\ rel.\ units.$ The HG scenario corresponds to high light level adapted phytoplankton and the LG scenario to low light adapted phytoplankton (Fig. 4-2). The high division scenarios (HD) use $a = 1$, $b = 2$ and $\delta_{max} = 0.05\ (10\ min)^{-1}$ (Fig. 4-3). The low division scenarios (LD) use $a = 1$, $b = 2$ and $\delta_{max} = 0.01\ (10\ min)^{-1}$.

I then combined the growth and division scenarios, to get four combinations of high and low growth and division (Table 4.1). I also repeated these four scenarios

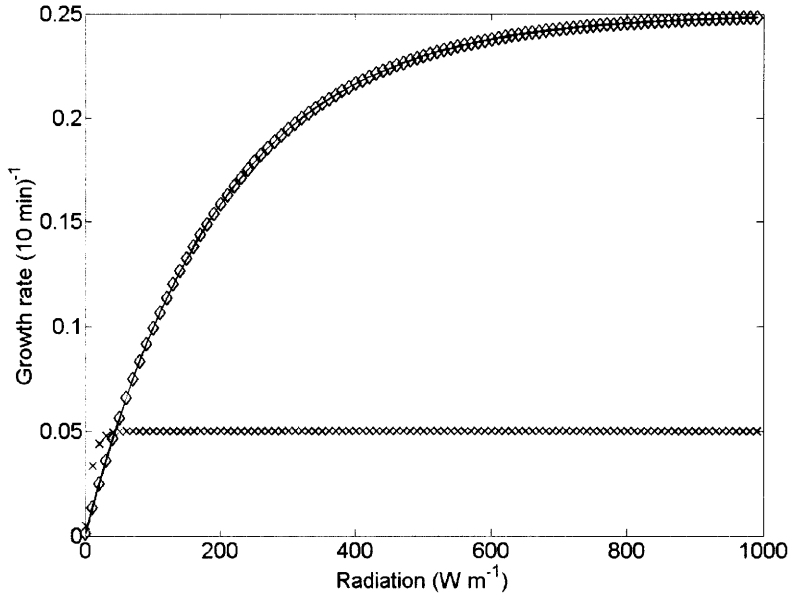

Figure 4-2: Cell growth rates γ as functions of incident radiation. The HG curve is plotted as diamonds and the LG curve as crosses.

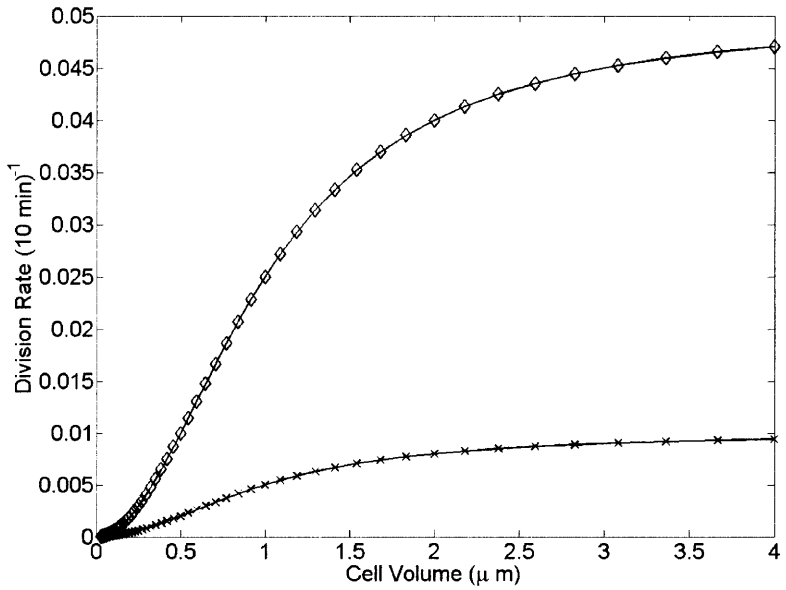

Figure 4-3: Cell division rates δ as functions of cell volume. The HD curve is plotted as diamonds and the LD curve as crosses.

Scenario	a	b	δ_{max} $(10\ min)^{-1}$	γ_{max} $(10\ min)^{-1}$	E^* $(rel.\ units)$	ϕ	μ (d^{-1})
HGHD	1	2	0.05	0.25	200	10^6	1.04
HGLD	1	2	0.01	0.25	200	10^6	0.31
LGHD	1	2	0.05	0.05	10	10^6	0.86
LGLD	1	2	0.01	0.05	10	10^6	0.26

Table 4.1: True parameter values used in the four test scenarios and corresponding deterministic population growth rates μ.

with different precision parameters.

For each scenario, I generated an artificial observation, then estimated the parameters with the maximum likelihood method, with an initial size distribution (Fig. 4-4) and incident radiation over the day (Fig. 4-5) taken from a laboratory observation [9] which will be used later.

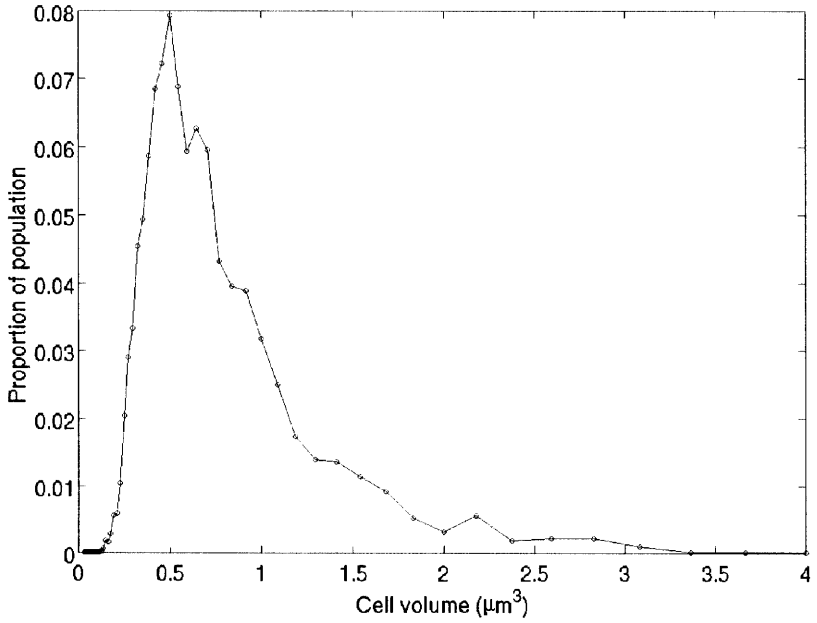

Figure 4-4: Initial population distribution.

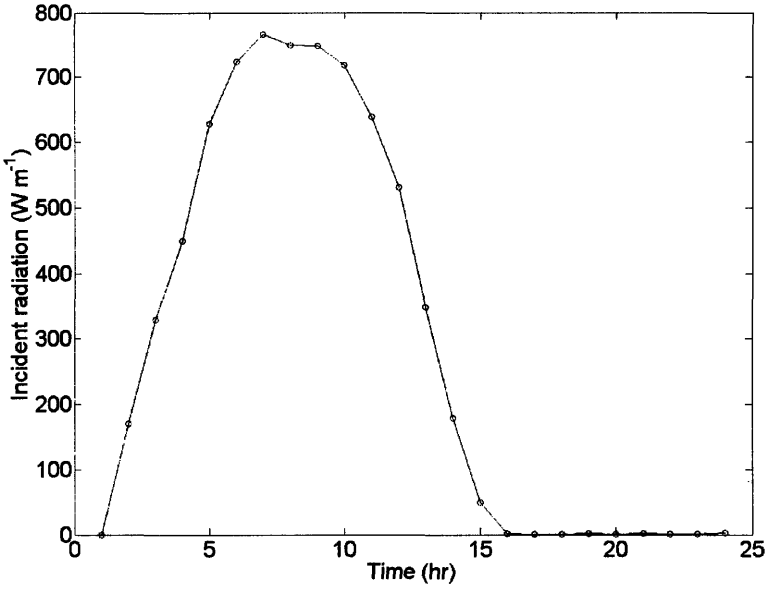

Figure 4-5: Incident radiation as a function of time $E(t)$.

4.3.1 Scenario results

Precision parameter $\phi = 10^6$

When $\phi = 10^6$, most parameter confidence intervals contain the true values (Table 4.2). Exceptions are a and b in the low division (LD) scenarios and ϕ in all cases. In all four scenarios, the ϕ estimates and confidence intervals are consistently high. Despite this fact, the μ confidence intervals contain the true value in all four scenarios. The estimated γ and δ curves are nearly indistinguishable from those calculated from the true parameter values (Fig. 4-6).

Precision parameter $\phi = 10^5$

When ϕ is decreased to 10^5, the parameter estimates become less accurate (Table 4-7). In the HD scenarios, the a, b and δ_{max} confidence intervals do not contain the true values, though the parameter estimates are still within 10%. For the LD scenarios, the confidence intervals are much larger, but still not large enough to contain the true parameter values. The errors are large enough in these LD scenarios that the μ

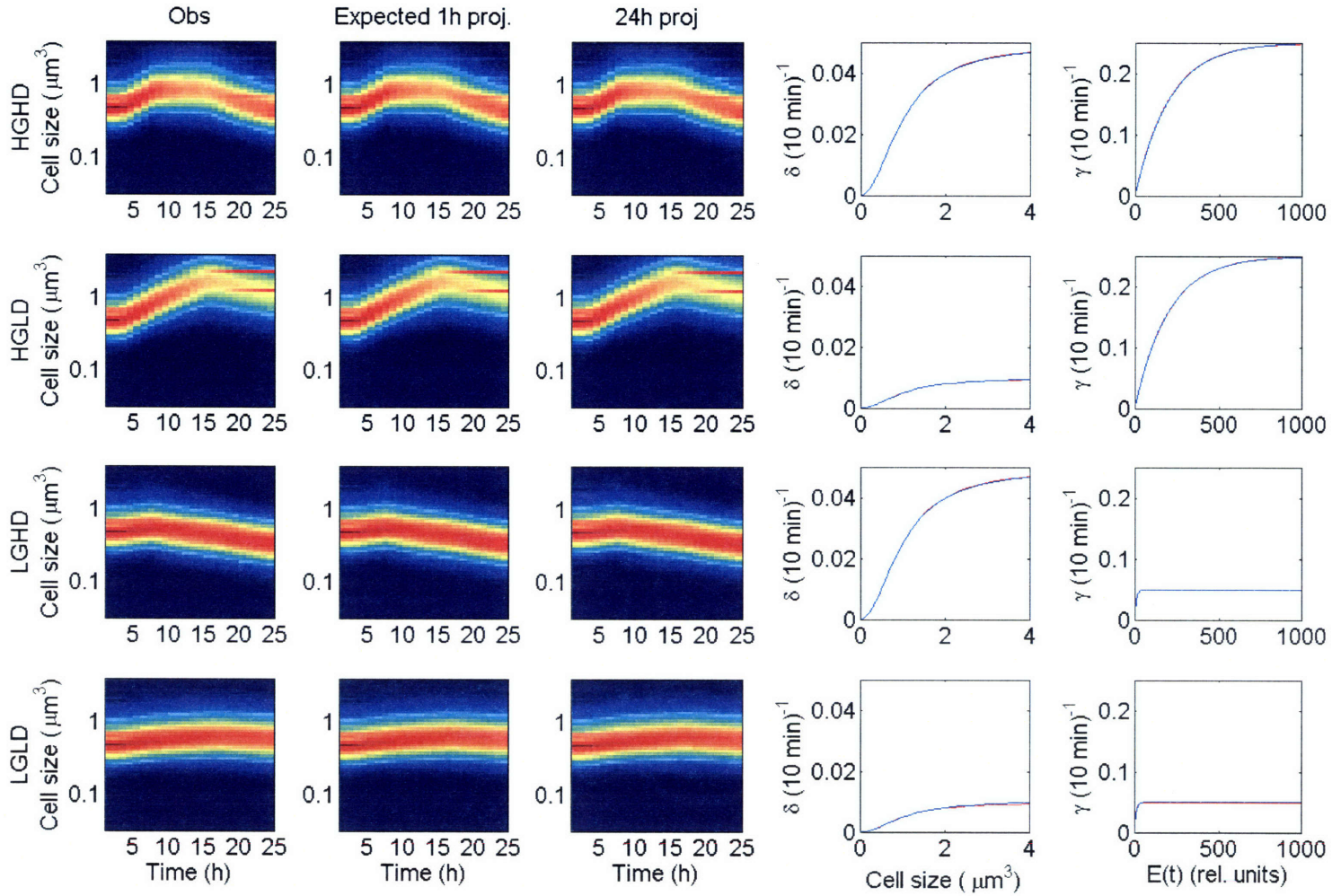

Figure 4-6: Each row corresponds to one of the four growth/division scenarios, with $\phi = 10^6$. The first column shows the artificial scenario used. The second shows the expected population distribution with the parameter estimates $\hat{\theta}$. The third shows a full day's projection from $\mathbf{w}_0$ with $\hat{\theta}$. Time is measured in hours after dawn. The 5th and 6th column show the cell division (plotted vs. volume) and growth (plotted vs incident radiation) curves calculated from θ in red and $\hat{\theta}$ in blue. Cell growth and division are per 10 min time span. In the HGLD scenario, phytoplankton accumulate in the largest size class, which cascades to cause accumulation in the size classes 1/2 and 1/4 as large.

Scenario		a	b	δ_{max} $(10\ min)^{-1}$	γ_{max} $(10\ min)^{-1}$	E^* $(rel.\ units)$	ϕ	μ (d^{-1})
HGHD	T	1	2	0.05	0.25	200	10^6	1.04
	MLE	1.0	2.0	0.050	0.25	$2.0e2$	$1.40e6$	1.04
	CI	[0.99, 1.0]	[2.0, 2.0]	[0.049, 0.050]	[0.25, 0.25]	[$2.0e2$, $2.1e2$]	[$1.31e6$, $1.51e6$]	[1.03, 1.04]
HGLD	T	1	2	0.01	0.25	200	10^6	0.31
	M	0.96	1.9	0.010	0.25	$2.0e2$	$1.51e6$	0.31
	CI	[0.93, 0.99]	[1.9, 2.0]	[0.010, 0.010]	[0.25, 0.25]	[$2.0e2$, $2.0e2$]	[$1.41e6$, $1.62e6$]	[0.31, 0.31]
LGHD	T	1	2	0.05	0.05	10	10^6	0.86
	MLE	1.0	2.0	0.050	0.050	10.	$1.35e6$	0.86
	CI	[0.98, 1.1]	[2.0, 2.0]	[0.049, 0.051]	[0.049, 0.051]	[9.2, 11.]	[$1.26e6$, $1.44e6$]	[0.85, 0.86]
LGLD	T	1	2	0.01	0.05	10	10^6	0.26
	MLE	0.87	1.9	0.011	0.050	10.	$1.45e6$	0.25
	CI	[0.79, 0.97]	[1.9, 2.0]	[0.010, 0.011]	[0.050, 0.051]	[9.4, 12.]	[$1.36e6$, $1.54e6$]	[0.25, 0.26]

Table 4.2: Parameter and population growth rate maximum likelihood estimates (MLE) and confidence intervals (CI), for true parameters (T) as in Table 4.1.

confidence intervals do not contain the true values. In contrast, in the HD scenarios, the μ confidence intervals contains the true values, despite the fact that the a, b and δ_{max} confidence intervals do not.

These inaccuracies are particularly visible in the LGLD δ curve (Fig. 4-7). Since the bulk of the plankton have a cell volume between 0.25 and 2, the fit of the lower half of the δ curve is more critical than the upper half. Based on this, the first and third (HD) δ curves are still good fits, even though the upper part of both curves visibly deviates. The LD division curves are worse fits, at least for the smaller cell volumes. This is particularly evident in the fourth scenario (LGLD).

Precision parameter $\phi = 10^4$

When the precision parameter is further decreased to $\phi = 10^4$, the numerical method used to invert the Hessian matrix breaks down in the low division (LD) scenarios, which means confidence intervals cannot be calculated with the asymptotic approach. The HD confidence intervals are still presented, as are the maximum likelihood estimates in the LD scenarios (Table 4.4). In the HGHD scenario, even though only the γ_{max} and E^* confidence intervals contain the true parameter values, the μ confidence interval still contains the true population growth rate. The LGHD confidence interval on μ does not contain the true population growth rate, but it nearly does. Note that the cell division rate curves in the first and third scenarios still fit reasonably well in the relevant cell size range (Fig. 4-8). The fit in the other two scenarios is much worse.

Precision parameter $\phi = 10^3$

When the precision parameter is reduced still further, to $\phi = 10^3$, only the LGHD confidence intervals can be calculated (Table 4.5). Note that while all but the E^* and μ confidence intervals contain the true parameter value, this is in part because the confidence intervals are quite large.

At this point, with $\phi = 10^3$, none of the δ curves, including the HD scenarios, are a close fit with the true ones (Fig. 4-9). Of the γ curves, only the LGHD curve is

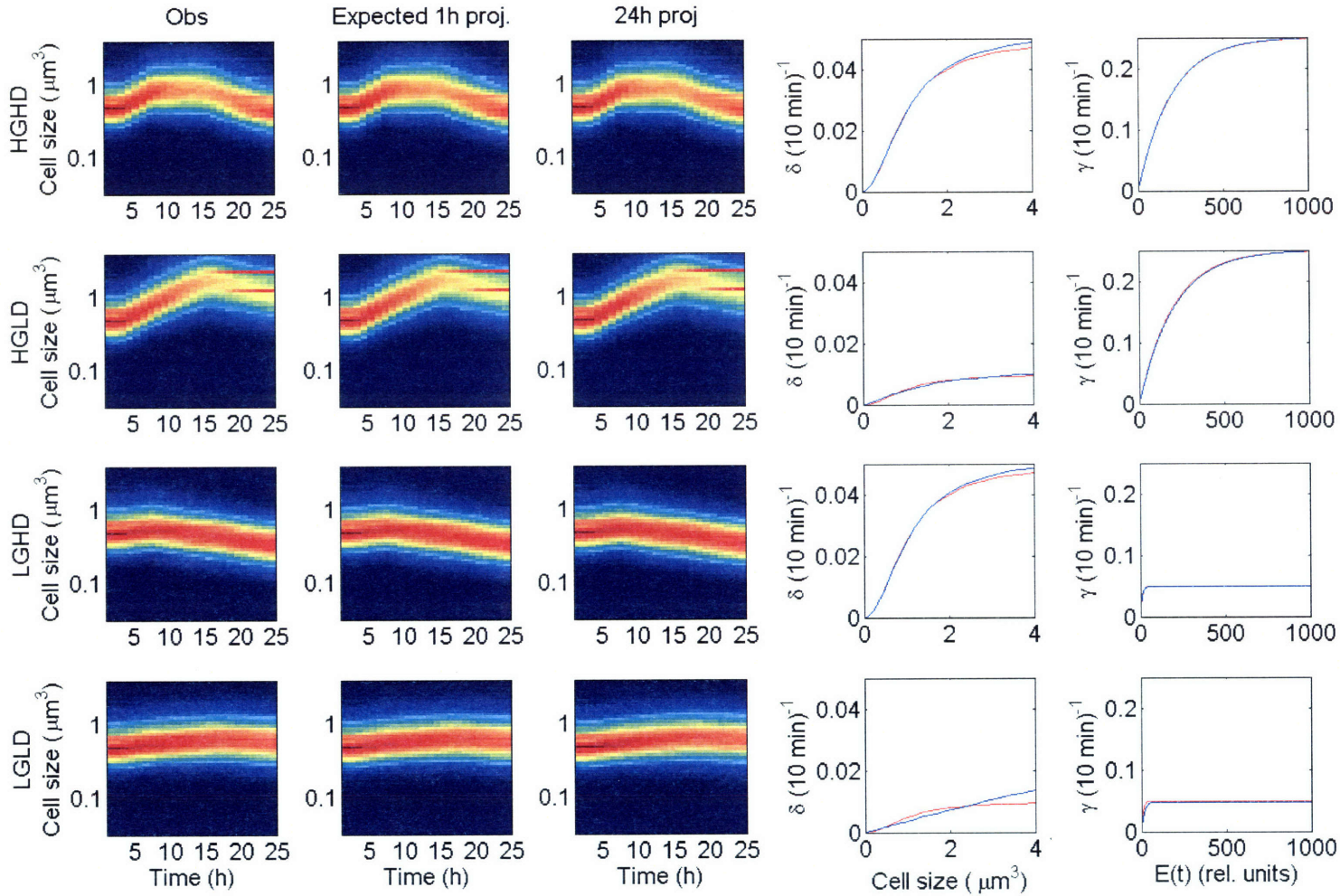

Figure 4-7: Each row corresponds to one of the four growth/division scenarios, with $\phi = 10^5$. Details are as in Fig. 4-6.

		a	b	δ_{max} $(10\ min)^{-1}$	γ_{max} $(10\ min)^{-1}$	E^* $(rel.\ units)$	ϕ	μ (d^{-1})
HGHD	T	1	2	0.05	0.25	200	10^5	1.04
	MLE	0.89	1.9	0.053	0.25	2.0e2	1.36e5	1.04
	CI	[0.85, 0.94]	[1.9, 2.0]	[0.051, 0.054]	[0.25, 0.25]	[1.9e2, 2.1e2]	[1.26e5, 1.47e5]	[1.03, 1.05]
HGLD	T	1	2	0.01	0.25	200	10^5	0.31
	MLE	0.61	1.3	0.013	0.25	2.1e2	1.39e5	0.33
	CI	[0.53, 0.71]	[1.2, 1.4]	[0.012, 0.014]	[0.25, 0.25]	[2.0e2, 2.2e2]	[1.28e5, 1.51e5]	[0.32, 0.34]
LGHD	T	1	2	0.05	0.05	10	10^5	0.86
	MLE	0.89	1.9	0.052	0.049	8.5	1.27e5	0.86
	CI	[0.79, 0.98]	[1.9, 2.0]	[0.049, 0.056]	[0.047, 0.051]	[6.4, 11.]	[1.18e5, 1.35e5]	[0.84, 0.87]
LGLD	T	1	2	0.01	0.05	10	10^5	0.26
	MLE	0.043	1.0	0.091	0.048	16.	1.11e5	0.23
	CI	[0.0063, 0.40]	[0.91, 1.2]	[0.0097, 0.41]	[0.046, 0.050]	[11., 25.]	[1.03e5, 1.20e5]	[0.16, 0.24]

Table 4.3: Parameter and population growth rate maximum likelihood estimates (MLE) and confidence intervals (CI), for true parameters (T) as in Table 4.1, except $\phi = 10^5$.

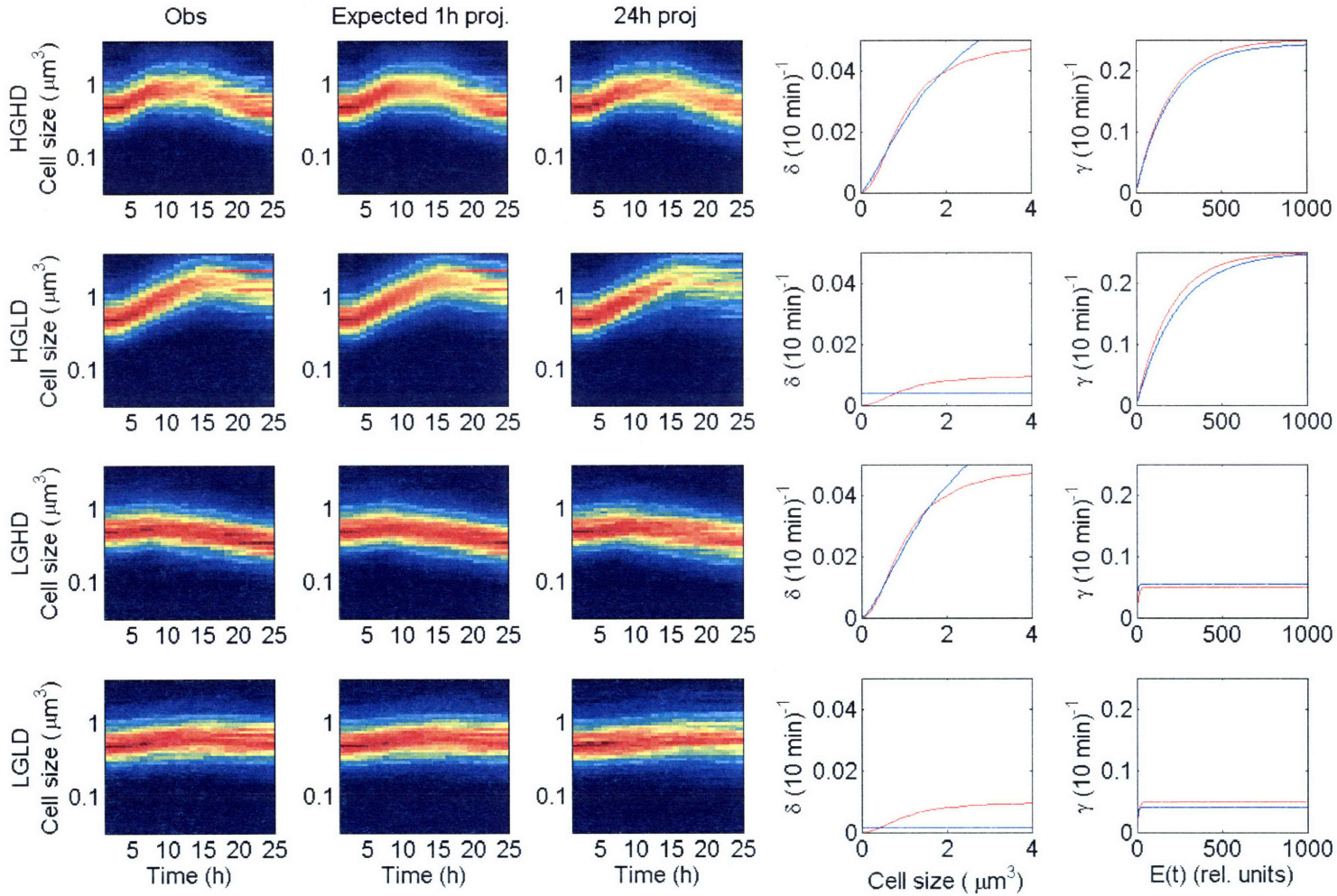

Figure 4-8: Each row corresponds to one of the four growth/division scenarios, with $\phi = 10^4$. Details are as in Fig. 4-6.

Scenario		a	b	δ_{max} $(10\ min)^{-1}$	γ_{max} $(10\ min)^{-1}$	E^* $(rel.\ units)$	ϕ	μ (d^{-1})
HGHD	T	1	2	0.05	0.25	200	10^4	1.04
	MLE	0.38	1.3	0.083	0.24	2.1e02	1.24e04	1.07
	CI	[0.30, 0.48]	[1.3, 1.4]	[0.071, 0.097]	[0.23, 0.25]	[1.7e02, 2.4e02]	[1.14e04, 1.35e04]	[1.04, 1.09]
HGLD	T	1	2	0.01	0.25	200	10^4	0.31
	MLE	4.2	$1.5e-5$	0.0050	0.25	2.4e02	1.02e04	0.43
	CI	-	-	-	-	-	-	-
LGHD	T	1	2	0.05	0.05	10	10^4	0.86
	MLE	0.38	1.5	0.083	0.054	4.1	1.14e04	0.90
	CI	[0.27, 0.56]	[1.4, 1.6]	[0.062, 0.11]	[0.049, 0.059]	[3.0, 5.8]	[1.06e04, 1.24e04]	[0.86, 0.94]
LGLD	T	1	2	0.01	0.05	10	10^4	0.26
	MLE	2.9e04	0.0011	0.0015	0.041	5.4	1.05e04	0.16
	CI	-	-	-	-	-	-	-

Table 4.4: Parameter and population growth rate maximum likelihood estimates (MLE) and confidence intervals (CI), for true parameters (T) as in Table 4.1, except $\phi = 10^4$.

still close to the true one. The accuracy of the growth curve, and its corresponding parameter estimates may explain the ability to calculate confidence intervals for this scenario.

4.3.2 Conclusions regarding artificial observations

Regardless of scenario or precision parameter value, there appears to be a bias in the precision parameter (ϕ) estimates, causing the estimates to be high and the true parameter values to lie outside the corresponding confidence intervals. Despite this, I am able to consistently estimate the population growth rate μ when the precision parameter is high enough ($\phi \geq 10^6$). When ϕ is lower ($\phi = 10^4 - 10^5$), I can still reliably estimate μ in the HD scenarios, but not in all of the LD scenarios. When ϕ is even smaller ($\phi \leq 10^3$), my results show the maximum likelihood approach can be problematic. With my numerical methods, confidence intervals were difficult to determine, and the estimates were often highly inaccurate.

In addition, there appears to be a correlation between the true population growth rate μ and the accuracy of estimates of μ (Fig. 4-10). For $\phi = 10^6$, $\phi = 10^5$, and the HD scenarios with $\phi = 10^4$, the scenarios with higher true growth rates have smaller confidence intervals and smaller relative error for the estimates of μ. More work would be needed to determine if this is a simple function of the cell division curve δ, since μ and δ are strongly correlated.

4.4 Laboratory observation

The initial size distribution and 24 hour incident radiation levels correspond to a phytoplankton population grown in a laboratory setting by Olson et al. [9]. A batch culture of *Synechococcus* were grown under temperature control and artificial light. The population size distribution and abundance were measured every few minutes with a bench top version of the FlowCytobot [10]. This information was then aggregated into hourly size distributions, and zeros were converted to 10^{-10}, as in the artificial observation above (Fig. 4-11). Because the population was well mixed and

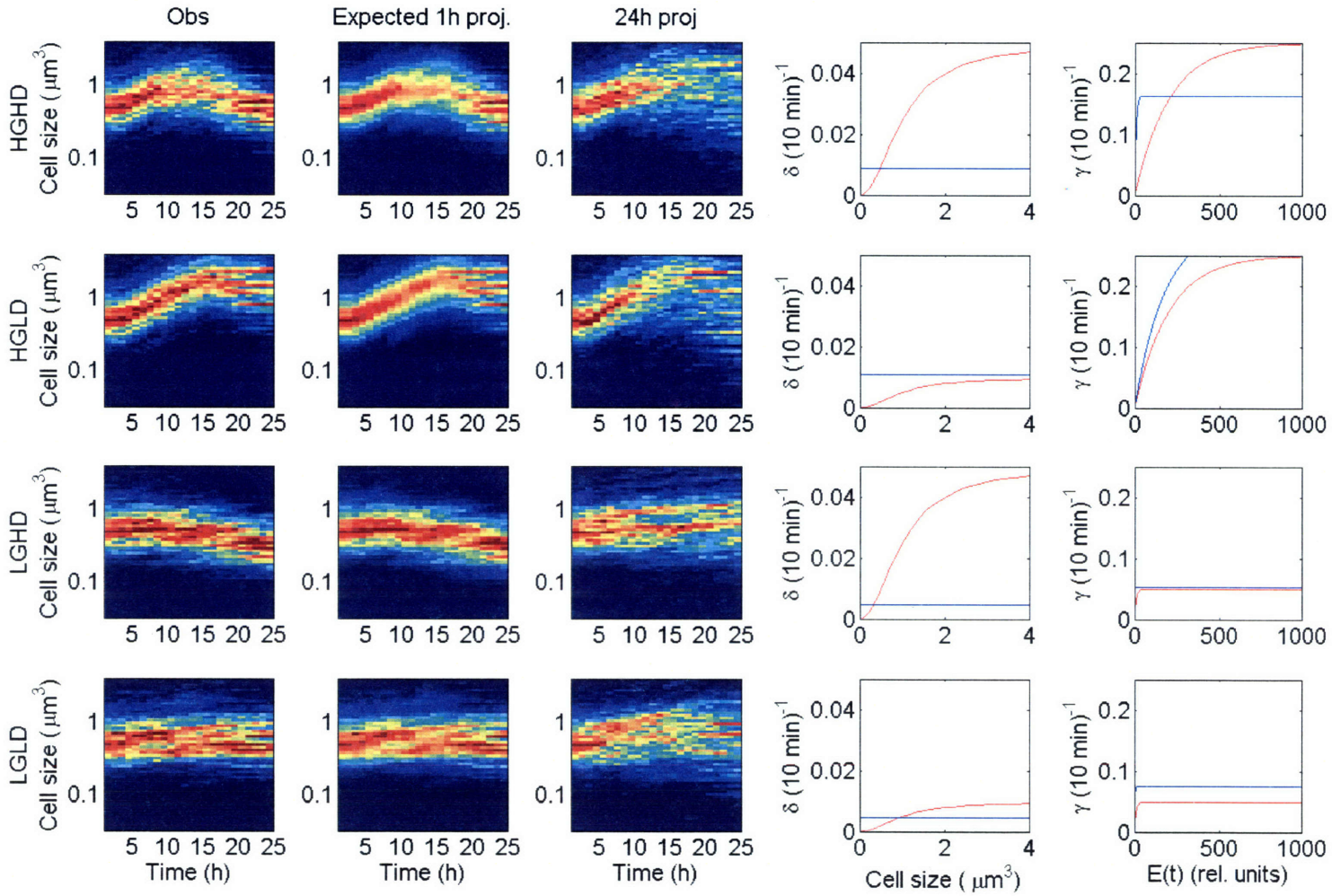

Figure 4-9: Each row corresponds to one of the four growth/division scenarios, with $\phi = 10^3$. Details are as in Fig. 4-6.

Scenario		a	b	δ_{max} $(10\ min)^{-1}$	γ_{max} $(10\ min)^{-1}$	E^* $(rel.\ units)$	ϕ	μ (d^{-1})
HGHD	T	1	2	0.05	0.25	200	10^3	1.04
	MLE	1.2	$1.5e-6$	0.016	0.16	7.9	8.59e02	0.95
	CI	-	-	-	-	-	-	-
HGLD	T	1	2	0.01	0.25	200	10^3	0.31
	MLE	1.1e02	$8.2e-6$	0.011	0.30	1.8e02	8.10e02	1.15
	CI	-	-	-	-	-	-	-
LGHD	T	1	2	0.05	0.05	10	10^3	0.86
	MLE	0.047	$3.8e-6$	0.10	0.052	0.026	9.96e02	0.49
	CI	$[1.0e-5, 5.1e02]$	$[4.4e-129, 2.4e129]$	$[3.5e-6, 1.0]$	[0.046, 0.058]	[0.0051, 0.16]	[9.14e02, 1.08e03]	$[0.00, NaN]$
LGLD	T	1	2	0.01	0.05	10	10^3	0.26
	MLE	6.0e04	$8.6e-5$	0.0047	0.074	2.8	9.12e02	0.50
	CI	-	-	-	-	-	-	-

Table 4.5: Parameter and population growth rate maximum likelihood estimates (MLE) and confidence intervals (CI), for true parameters (T) as in Table 4.1, except $\phi = 10^3$.

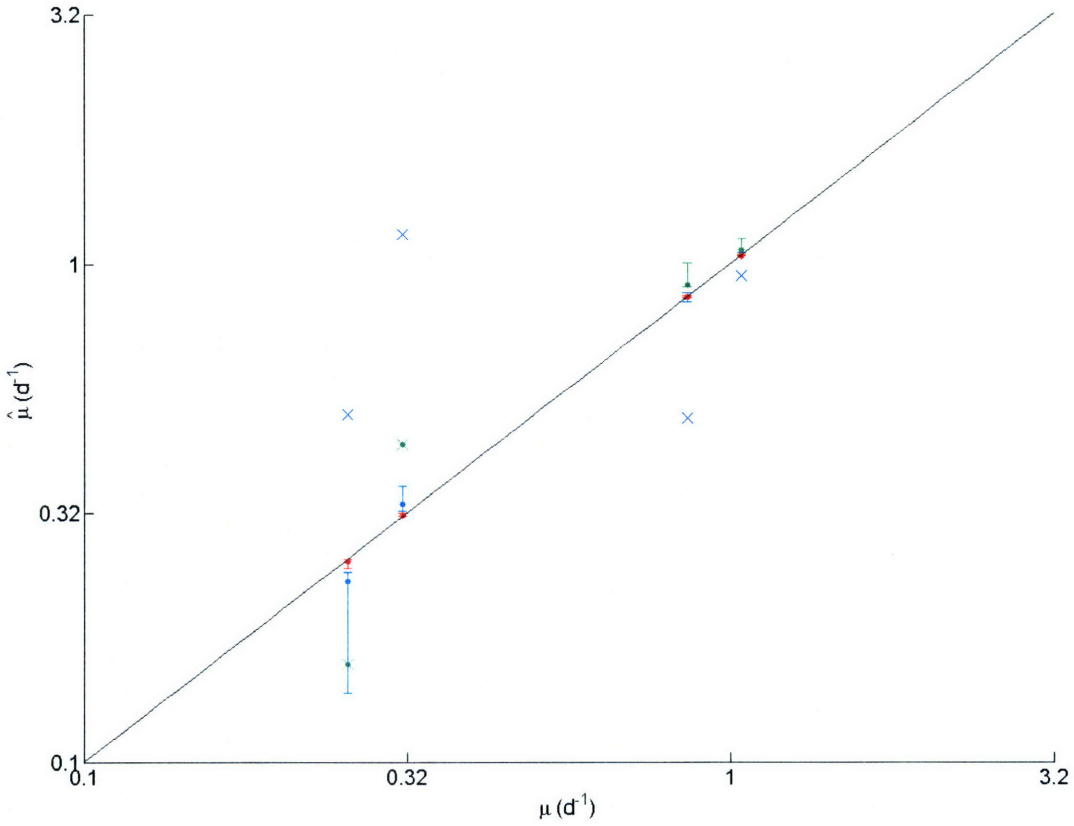

Figure 4-10: Population growth rate estimates $\hat{\mu}$ and confidence intervals as a function of the true growth rate μ, plotted on a log scale. The data is shown on a log-log scale to focus on relative error. Red points correspond to $\phi = 10^6$, cyan to $\phi = 10^5$, green to $\phi = 10^4$ and blue to $\phi = 10^3$. When confidence intervals cannot be calculated, the point is replaced with an x.

a	b	δ_{max} $(10\ min)^{-1}$	γ_{max} $(10\ min)^{-1}$	E^* $(rel.\ units)$	ϕ	μ
0.0054	$2.1e-144$	0.91	0.093	3.0	$5.22e2$	0.52

Table 4.6: Parameter and population growth rate estimates and confidence intervals for a laboratory observation.

had no mortality sources, it is possible to use abundance data to calculate population growth rates. For the experimental condition examined here the population growth rate from cell abundance was 0.64 d^{-1}.

I repeated the maximum likelihood methods used above, in order to calculate parameter and population growth rate estimates. In this case, I don't know the true parameter values, so I cannot start the optimization routine near the true values. To compensate, I found the maximum likelihood estimate and corresponding likelihood for over 17,000 random starting points. I then used the parameter estimates and Hessian matrix that correspond to the lowest negative log likelihood to attempt to generate confidence intervals (Table 4.6), but was unable to because the matrix was not invertible.

The estimated population growth rate $\hat{\mu}$ is 0.52 d^{-1} . This is close to the true population growth rate of $\mu = 0.64\ d^{-1}$. Although this estimate of μ appears to be fairly accurate, the ϕ estimate is very small, which suggests the other parameter estimates may not be very accurate, particularly considering that the corresponding Hessian matrix is singular. When I plotted estimated cell growth and division curves, I noticed that they showed no response to light levels or cell volume (Fig. 4-11). Note: if one plots the cell growth and division curves for the 200 or so estimates with population growth rates nearly identical to the maximum likelihood estimate, these curves are very similar in appearance, even though the estimates of a and b (for example) may vary by several orders of magnitude. All of the b estimates are very small, which means the a values have little impact. Although the population growth rate estimate appears to be accurate, I could not get confidence intervals on that estimate, making it difficult to decide how well the method is working overall.

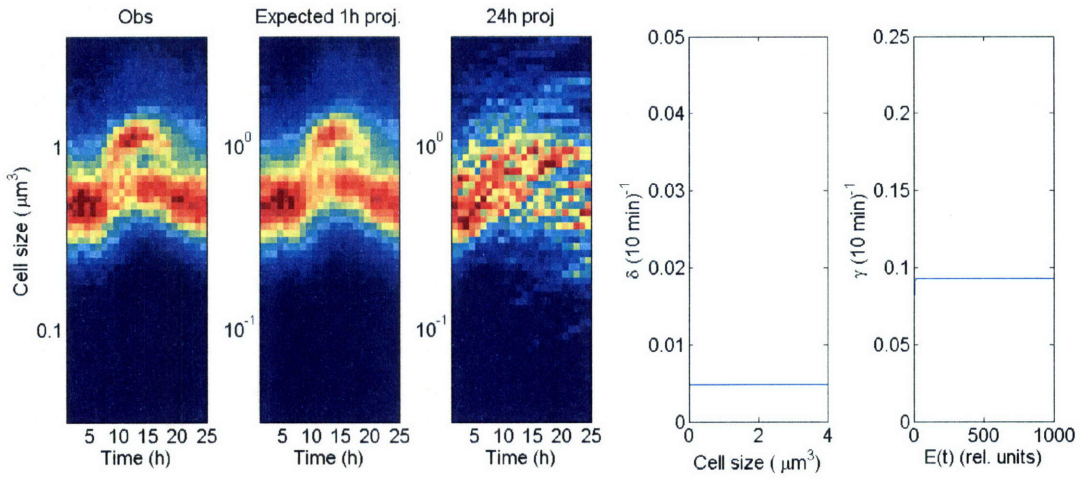

Figure 4-11: Laboratory observation. Details are as in Fig. 4-6

Chapter 5

Conclusions

For the simplified three-stage model (Section 3), one can estimate most of the model parameters using the maximum likelihood method. Estimates of the precision parameter ϕ are consistently too high. Bias in the parameter estimates appears to be negatively correlated with cell division rates and cell growth rates. Because of the structure of the three stage model, this high-bias region also has low daily population growth rates μ.

The bias in population growth rate estimates $\hat{\mu}$ is relatively low, even when the precision parameter ϕ is relatively small and the biases in parameter estimates are large. The corresponding μ confidence intervals contain the true growth rate 89% of the time. Bias and confidence interval size both decrease as the size of the observation increases. Removing zeros does not affect the estimate accuracy. The accuracy of model parameter estimates is high for high δ_2 and δ_3 values (which correspond to high cell division rates) and for low c values (which correspond to high cell growth rates). Since these parameters also correspond to high population growth rates μ, it is difficult to determine if all combination of these parameters that lead to a given μ will have similar accuracy of model parameter estimates.

In the full model, as in the three-stage model, the accuracy of the parameter estimates increased with ϕ. The model also exhibited the same pattern of low bias corresponding to high cell growth and division curves, as well as high population growth rates μ. In cases where ϕ was very small ($\phi = 10^3$), the estimated cell growth

and division curves were almost entirely flat, showing no response to light levels or phytoplankton cell size respectively.

The ϕ value for the laboratory data suggests that this observation falls in the noisy region of the parameter space, and that the the corresponding population growth rate is not likely to be a good estimate. Despite this, the estimated population growth rate is within 13% of the measured population growth rate. A comparison of the laboratory observation with artificial observations generated with low ϕ values shows that the laboratory observation is much less scattered and random than the artificial observation (Fig. 4-6 and 4-11). This suggests that either the parameter estimates are incorrect or that the model is not a good fit for real world data.

5.0.1 Limitations

One limitation of the current implementation of the maximum likelihood method is that the optimization routine used, "fminunc" in Matlab [8], returns different values depending on initial conditions. For the artificial observations, I compensated by including the true parameter values as one starting point, and including several random starting points as well. Starting on the true parameter values is not an option for the laboratory observation, so I resorted to using thousands of random starting points. Use of another optimization routine for this part of the analysis would remove one possible source of bias in the parameter estimates.

Another limitation of the current implementation is that the confidence intervals of the parameter estimates and population growth rates contain the true values slightly less frequently than they should from a statistical standpoint. This could be either because of an unknown source of bias in the estimate around which the confidence interval is generated, a problem with the size of the confidence interval itself, or random chance.

In addition, some of the Hessian matrices returned by the "fminunc" optimization routine cannot be inverted with Matlab's numerical methods. These Hessian matrices correspond to regions with highly biased parameter estimates, which suggests the corresponding confidence intervals will be large, and would be useful for judging

the reliability of a given parameter estimate. This limitation may also disappear with the choice of a new optimization routine. It can also be avoided with a more computationally intensive parametric bootstrap approach.

5.0.2 Future directions

There are several directions this project could head in the future, some involving analysis of the current model, some involving extensions and changes.

Since there are large amounts of laboratory data available, it would be useful to compare estimated and measured population growth rates for multiple laboratory observations. If we assume that multiple cultures grown under the same conditions will have similar parameter values, we can treat them as different samples from a single day and estimate a single set of parameters for all of them. This would increase the number of available data points, and may give better bounds on corresponding parameter estimates, based on the three-stage tests. This would be particularly useful since the estimated precision is very low, at less than 10^3. Making the same assumption that similar cultures have similar model parameters, we can also use the distribution of parameter estimates to get approximate confidence intervals. Either of these methods should give a clearer idea of the accuracy of the individual estimates.

It would also be good to do a sensitivity and elasticity analysis of the full model, to see how a small change in a single parameter affects the population growth rate. Once the sensitivity analysis has been completed, and the model's predictive ability has been further tested with laboratory data, there will be information both on how well the model performs, and on which components of the model have the largest impact. This information can be used to strategically remove model parameters, rather than systematically testing all possible combinations. This has the advantage of reducing the ratio of model parameters to data points (which may not be significant if the parameters removed contribute little to the estimates of μ) as well as clarifying which components of the model are most important and the relationship between cell growth and division and population growth.

In terms of model implementation, it would also be useful to look more closely

at the trade-off between the number of data points into which a day's worth of size distribution data is split, and the accuracy of those data points. In the current implementation, size distribution data is reported for each hour, leading to 24 data points per day. Decreasing this interval, by grouping the data into half-hour bins would double the number of data points, but decrease the accuracy of each point, and vice versa if data were binned into two-hour intervals. Systematic tests with simulated data with an appropriate precision parameter would help to find the ideal trade-off between more information and more accuracy.

Moving on to changes in the underlying model structure, the current implementation does not take into account observation error. As mentioned in the introduction, Calder et al. [4] point out that ignoring observation error can lead to errors in parameter estimation. In this model, the most appropriate form for observation error is likely the multinomial distribution. This could be added as a third step after projecting the model forward and adding process error:

$$\mathbf{z}_{t+1} = Multinom(\mathbf{w}_{t+1}, n)/n. \tag{5.1}$$

The basic model structure in both the three stage and full models has three types of terms: growth, stasis and reproduction. When it comes to modeling organisms other than phytoplankton, new terms, such as mortality, will have to be added. In this model, I have implicitly assumed that any sources of mortality, such as grazing, do not discriminate between size classes, and are constant over time. I have also calculated the population growth rate μ in the absence of mortality. In other types of models, such as human demographics, this assumption is not accurate. These models must directly include mortality terms that take into account the population's age or size structure. Population growth rate formulas will have to be similarly reworked. Other terms will need to be added whenever different sizes or stages of an organism respond differently to their environment.

There are also other statistical approaches to dealing with multivariate distributions and time series of continuous proportions. Aitchison et al. [2] describe an al-

ternate zero-handling method that maintains the proportions between non-zero parts of a population distribution. Grunwald et al. [7] describe a statistical approach with a modified Dirichlet distribution whose results can be interpreted in terms of the odds ratios between different components of the distribution (e.g., size classes), that can also be used when the components of the distribution are dependent. These approaches may be more appropriate than the one laid out here for some types of time series of proportional data.

In future years, another statistical approach may be possible, that naturally handles zeros in the size distributions. In the context of phytoplankton, we can assume any zeros in the size distribution are due to an inability to detect proportions below a certain threshold. Aitchison et al. [2] are developing a method to handle "structural zeros", zeros which are truly zero, which come up in other types of models.

Although the model developed in this paper can successfully estimate population growth rate for artificial observations under low-noise conditions, and shows some ability to estimate population growth rate for a set of laboratory data, there are many possible improvements and extensions which could increase its predictive capabilities. Once the model improvements have been tested with laboratory data, an extended model could be used to calculate growth rates from field data gathered from a FlowCytobot.

Appendix A

Three stage code

A.1 bootstrap_nest_ci

```
function bootstrap_nest_ci(filename, n_obs, true_c, true_delta2, ...
                           true_delta3, true_shape, n_days, epsilon)
% Generate one observation and estimate based on the "true"
% parameter values, and use the corresponding Hessian matrix to
% generate a CI.  (This is called the asymptotic method in the
% writeup, and is called bootstrap_params here.  It is not)  Then
% generate n_obs observations based on the "true" parameter values,
% and calculate parameter estimates for each (This is called the
% bootstrap method in the writeup and is saved as est_params here).
% Save all of these estimates and their associated data in "filename"


n_bootstrap = n_obs;


best_obs = calc_obs(true_c, true_delta2, true_delta3, true_shape, ...
          [1/3,1/3,1/3], n_days, epsilon);
[best_params, best_like, best_flag, best_hess, num_iter] = ...
          recurse_opt_params(best_obs, n_days, epsilon)
```

```
best_c = best_params(1);
best_delta2 = best_params(2);
best_delta3 = best_params(3);
best_shape = best_params(4);

% In transformed param space, normally distributed.  Draw n_bootstrap
% values from multiv. norm. dist.  Then transform these back.
tr_best_params = transform(best_params);
best_var_cov = inv(best_hess);
tr_bootstrap_params = mvnrnd(tr_best_params, best_var_cov, ...
                      n_bootstrap);

for i=1:n_bootstrap
   bootstrap_params(i,:) = untransform(tr_bootstrap_params(i,:));
end

for j=1:n_obs
    j
    all_obs(j,:,:) = calc_obs(true_c, true_delta2, true_delta3, ...
     true_shape, [1/3,1/3,1/3], n_days, epsilon);
    obs_j = squeeze(all_obs(j,:,:));

    [param_est, like_est, flag_est, hess_est, iter_est] = ...
                    recurse_opt_params(obs_j, n_days, epsilon);

    est_params(j,:) = param_est;
est_iterations(j) = iter_est;

save(filename, 'n_obs', 'true_c', 'true_delta2', 'true_delta3', ...
```

```
    'true_shape', 'n_days', 'epsilon', 'best_obs', 'best_params', ...
    'best_hess', 'best_var_cov', 'bootstrap_params', 'all_obs', ...
    'est_params', 'est_iterations')
end
```

A.2 calc_a

```
function A = calc_a(t, c, delta2, delta3)
% Calculate 1-hr projection matrix for three stage model

e_t = calc_e(t);
gamma_t = e_t / (e_t + c);
A = [1-gamma_t 2*delta2 0;
    gamma_t (1-gamma_t)*(1-delta2) 2*delta3;
    0 (1-delta2)*gamma_t 1-delta3];

% Generate radiation curve following normal dist. centered half-way
% through the day.
function e = calc_e(t_multi)
t = mod(t_multi,24);
sigma = 6; % Variance of the normal distribution
t_hat = 12; % Mean of the normal distribution
e = 1/(sigma * sqrt(2 * pi)) * exp(-(t-t_hat).^2 / (2 * sigma^2));
```

A.3 calc_big_acc

```
function calc_big_acc(filename, n_days, epsilon, h_low, h_high, ...
                      i_low, i_high)
% Generate a grid of parameter values (spaced on a log scale).  For
% each point, use the true parameter values to create an observation,
```

```
% calculate best estimates of the parameters, growth rate and high
% and low ends of paramter and growth rate confidence intervals.
% Because this can take a long time to run, use indices h_low and
% h_high to specify the portions of the grid to calculate, if one
% needs to recombine later.

n_c = 6;
n_d2 = 6;
n_d3 = 6;
n_s = 3;
c_vect = logspace(-2, 0, n_c)
delta2_vect = logspace(-3, -0.5, n_d2)
delta3_vect = logspace(-3, -0.5, n_d3)
shape_vect = logspace(3, 6, n_s)

like_arbitrary = 10^15;

for h=h_low:h_high;
   for i=i_low:i_high;
      for j=1:n_d3;
         for k =1:n_s;
            [h,i,j,k]
            true_params(h,i,j,k,:) = [c_vect(h), delta2_vect(i), ...
                                      delta3_vect(j), shape_vect(k)];
            good_est = 0
            n_tries = 0;
            while(good_est == 0)
               n_tries = n_tries+1

               obs_hijk = calc_obs(c_vect(h), delta2_vect(i), ...
```

```
                delta3_vect(j), shape_vect(k), ...
                [1/3,1/3,1/3], n_days, epsilon);
   obs_m(h,i,j,k,:,:) = obs_hijk;
   [best_est, best_like, best_flag, best_hess, ...
            num_iter] = recurse_opt_params(obs_hijk, ...
            n_days, epsilon)

   % Check if the likelihood value is "good", and one can
   % stop trying this parameter combo.
   if(~isnan(best_like)&&(~(best_like==like_arbitrary)))
      good_est = 1
   end

end

tries_total(h,i,j,k,:) = n_tries
param_est(h,i,j,k,:) = best_est
growth_est(h,i,j,k) = calc_growth(best_est, n_days);
[c_low, c_high, d2_low, d2_high, d3_low, d3_high, ...
        s_low, s_high, g_low, g_high] = ...
        calc_ci(best_est, best_hess, n_days);
param_low_ci(h,i,j,k,:) = [c_low, d2_low, d3_low, s_low];
param_high_ci(h,i,j,k,:) = [c_high, d2_high, d3_high, ...
                            s_high];
growth_low_ci(h,i,j,k) = g_low;
growth_high_ci(h,i,j,k) = g_high;

save(filename, 'n_days', 'epsilon', 'c_vect', ...
     'delta2_vect', 'delta3_vect', 'shape_vect', ...
     'true_params', 'obs_m', 'tries_total', ...
```

```
                'param_est', 'growth_est', 'param_low_ci', ...
                'param_high_ci', 'growth_low_ci', 'growth_high_ci');
        end
    end
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate confidence intervals using asymptotic approach.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [c_low, c_high, d2_low, d2_high, d3_low, d3_high, s_low, ...
          s_high, g_low, g_high] = calc_ci(est, hess, n_days)

num_bootstrap = 500;
ci_size = 0.95;
idx_low_est = round((1-ci_size)/2*num_bootstrap);
idx_high_est = num_bootstrap - idx_low_est;

% In transformed param space, normally distributed.  Draw
% num_bootstrap values from multiv. norm. dist.  Then transform these
% back.
tr_est = transform(est);
tr_var_cov = inv(hess);
tr_bootstrap_params = mvnrnd(tr_est, tr_var_cov, num_bootstrap);

for i=1:num_bootstrap
  bootstrap_params(i,:) = untransform(tr_bootstrap_params(i,:));
end

bootstrap_growth = calc_growth(bootstrap_params, n_days);
```

```
ctheor_ci = sort(bootstrap_params(:,1));
c_low = ctheor_ci(idx_low_est);
c_high = ctheor_ci(idx_high_est);

d2theor_ci = sort(bootstrap_params(:,2));
d2_low = d2theor_ci(idx_low_est);
d2_high = d2theor_ci(idx_high_est);

d3theor_ci = sort(bootstrap_params(:,3));
d3_low = d3theor_ci(idx_low_est);
d3_high = d3theor_ci(idx_high_est);

shapetheor_ci = sort(bootstrap_params(:,4));
s_low = shapetheor_ci(idx_low_est);
s_high = shapetheor_ci(idx_high_est);

growththeor_ci = sort(bootstrap_growth);
g_low = growththeor_ci(idx_low_est);
g_high = growththeor_ci(idx_high_est);
```

A.4 calc_dir_like

```
function like = calc_dir_like(tr_param_v, obs, n_days, epsilon)
% Calculate the likelihood of 'obs', given a set of transformed model
% parameters tr_param_v, the number of days in the observation, and
% the size of epsilon used.

param_v = untransform(tr_param_v);
c = param_v(1);
```

```
delta2 = param_v(2);
delta3 = param_v(3);
shape = param_v(4);

like_arbitrary = 10^15;
t_end = n_days*24;
norm_v(:,1) = obs(:,1);

for t=2:t_end
    A = calc_a(t, c, delta2, delta3);
   proj_v = A * obs(:,t-1);
   norm_v(:,t) = (proj_v + epsilon) ./ sum(proj_v + epsilon);
end

like_all = dir_calc(obs, norm_v, shape);
like = sum(like_all);

% If the parameter values are extreme enough , the likelihood
% calculation breaks down.  In order to keep the optimization routine
% running, we set the likelihood arbitrarily high.
if (isnan(like) || isinf(like))
     ['Like was NaN or Inf, set to ' num2str(like_arbitrary)]
     like = like_arbitrary;
end
```

A.5 calc_equilib_obs

```
function obs_m = calc_equilib_obs(c, delta2, delta3, shape, ...
                 time1_v, n_days, epsilon, t_equilib)
% Generate a starting distribution by first calculating an
```

```
% observation of length t_equilib and saving the ending distribution.
% Then generate an observation of length n_days, starting with the
% ending distribution of the first one.  By picking large enough
% t_equilib, the population effectively starts at equilibrium, and
% transient information is ignored.

% Start projection at time1_v, project out until t_equilib.
tmp_obs(:,1) = time1_v;
equi_m(:,1) = (tmp_obs(:,1)+epsilon) ./ sum(tmp_obs(:,1) + epsilon);
equi_end = t_equilib*24;

for i=2:equi_end
   A = calc_a(i, c, delta2, delta3);
   proj_v = A * equi_m(:,i-1);
   norm_v = proj_v / sum(proj_v);
   tmp_equi(:,i) = draw_from_dir(norm_v, shape);
   equi_m(:,i) = (tmp_equi(:,i)+epsilon)./sum(tmp_equi(:,i)+epsilon);
end

% Start observation at end of equilibrium period and project from
% there.
t_end = n_days*24;
obs_m(:,1) = equi_m(:,equi_end);

for i=2:t_end
   A = calc_a(i, c, delta2, delta3);
   proj_v = A * obs_m(:,i-1);
   norm_v = proj_v / sum(proj_v);
   tmp_obs(:,i) = draw_from_dir(norm_v, shape);
   obs_m(:,i) = (tmp_obs(:,i)+epsilon)./sum(tmp_obs(:,i)+epsilon);
```

```
end
```

A.6 calc_growth

```
function growth = calc_growth(param_vec, n_days)
% Given a vector of parameters [c, delta2, delta3, shape], and the
% anumber of days to project calculate the daily specific growth rate
% assuming the population distribution starts at equilibrium.

w_0 = [1/3,1/3,1/3]';
n_obs = size(param_vec,1);
t_end = n_days*24;

for i=1:n_obs
   c_i = param_vec(i,1);
   delta2_i = param_vec(i,2);
   delta3_i = param_vec(i,3);
   shape_i = param_vec(i,4);

   u_prod(i,:,:) = eye(3);
   for j=1:t_end
      u_tmp = calc_a(j, c_i, delta2_i, delta3_i);
      u_prod(i,:,:) = squeeze(u_prod(i,:,:)) * u_tmp;
   end
   % Multiply together all j entries for this i.
   u(i,:) = squeeze(u_prod(i,:,:))*w_0;
   growth(i) = log(sum(u(i,:)))/n_days;
end
```

A.7 calc_obs

```
function obs_m = calc_obs(c, delta2, delta3, shape, time1_v, ...
                 n_days, epsilon)
% Given a set of model parameters c, delta2, delta3 and shape, an
% initial population distribution, the number of days to simulate and
% an epsilon value, project forward, renormalizing the population
% distribution and drawing from a Dirichlet distribution at each time
% step.

tmp_obs(:,1) = time1_v;
obs_m(:,1) = (tmp_obs(:,1) + epsilon) ./ sum(tmp_obs(:,1) + epsilon);
t_end = n_days*24;

for i=2:t_end
   A = calc_a(i, c, delta2, delta3);
   proj_v = A * obs_m(:,i-1);
   norm_v = proj_v / sum(proj_v);
   tmp_obs(:,i) = draw_from_dir(norm_v, shape);
   obs_m(:,i) = (tmp_obs(:,i)+epsilon)./sum(tmp_obs(:,i) + epsilon);
end
```

A.8 dir_calc

```
function neg_like = dir_calc(obs_vect, theor_prob_vect, dir_shape)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate likelihoods given an observation, an expected
% distribution and a shape parameter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Prob is \Pi x^(a-1) * \gamma(\Sigma(a))/ \Pi(\gamma(a))
```

```
% Like is -[\Sigma (a-1)log(x) + log(gamma(\Sigma(a))) -
% \Sigma(log(gamma(a))]
% Which can be rewritten: -\Sigma (a-1)log(x) - gammaln(\Sigma(a)) +
% \Sigma(gammaln(a))

if (~isreal(dir_shape))
    dir_shape
end

c_vect = theor_prob_vect .* dir_shape;
term1 = sum((c_vect - 1) .* log(obs_vect));
term2 = gammaln(dir_shape);
term3 = sum(gammaln(c_vect));
neg_like = -term1 - term2 + term3;
```

A.9 draw_from_dir

```
function obs_vect = draw_from_dir(theor_pi_vect, dir_shape)
% Given a vector of expected values and a shape parameter, draw an
% "observed" distribution from the Dirichlet
% Note: The larger 'phi' is, the smaller the variance, and
% vice-versa.   Also, phi must be strictly greater than zero.

Y_i_vect = gamrnd(dir_shape .* theor_pi_vect, 1);
Y = sum(Y_i_vect);
obs_vect = Y_i_vect ./ Y;
```

A.10 equi_obs_bootstrap

```
function equi_obs_bootstrap(filename, n_obs, true_c, true_delta2, ...
```

```
        true_delta3, true_shape, n_days, epsilon)
% Do the same thing bootstrap_nest_ci does, but use an observation
% that starts at the size distribution after t_equilib days.

t_equilib = 100;
n_bootstrap = n_obs;

best_obs = calc_equilib_obs(true_c, true_delta2, true_delta3, ...
          true_shape, [1/3,1/3,1/3], n_days, epsilon, t_equilib);

[best_params, best_like, best_flag, best_hess, num_iter] = ...
             recurse_opt_params(best_obs, n_days, epsilon);

best_c = best_params(1);
best_delta2 = best_params(2);
best_delta3 = best_params(3);
best_shape = best_params(4);

% In transformed param space, normally distributed.  Draw n_bootstrap
% values from multiv. norm. dist.  Then transform these back.
tr_best_params = transform(best_params);
best_var_cov = inv(best_hess);
tr_bootstrap_params = mvnrnd(tr_best_params, best_var_cov, ...
                     n_bootstrap);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate n_obs observations centered on best_params, then find the
% best estimate using recurse_opt_params
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:n_bootstrap
   bootstrap_params(i,:) = untransform(tr_bootstrap_params(i,:));
end

for j=1:n_obs
    all_obs(j,:,:) = calc_equilib_obs(best_c, best_delta2, ...
                     best_delta3, best_shape, [1/3,1/3,1/3], ...
                     n_days, epsilon, t_equilib);
    obs_j = squeeze(all_obs(j,:,:));

    [param_est, like_est, flag_est, hess_est, iter_est] = ...
                          recurse_opt_params(obs_j, n_days, epsilon);

    est_params(j,:) = param_est;
    est_iterations(j) = iter_est;

    save(filename, 'n_obs', 'true_c', 'true_delta2', ...
         'true_delta3', 'true_shape', 'n_days', 'epsilon', ...
         'best_obs', 'best_params', 'best_hess', 'best_var_cov', ...
         'bootstrap_params', 'all_obs', 'est_params', ...
         'est_iterations')
end
```

A.11 recurse_opt_params

```
function [best_params, best_like, best_flag, best_hess, num_iter] ...
         = recurse_opt_params(obs, n_days, epsilon)
% Goal: Find at least 3 parameter estimates whose likelihoods are
% within like_tol of the best one found.  Then make sure that there
```

```
% are at least 3 estimates where all the param. estimates are within
% rel_param_tol of each other.  Save the best one (by the likelihood
% metric) and return the results (as well as the number of tries
% needed).  If you reach iter_cutoff, make all the values NaN or
% like_arbitrary and stop.

iter_cutoff = 50;
like_arbitrary = 10^15;
% Likelihood estimates must be within like_tol of each other.
like_tol = 1e-4;
% Param est must be within real_param_tol of each other.
rel_param_tol = 1e-3;

found_best = 0;
num_iter = 0;
tmp_all_like = [];
tmp_all_params = [];

warning('off', 'optim:fminunc:SwitchingMethod');
% Set options for fminunc.
opt = optimset('TolX', 1e-8*n_days,'maxIter', 1000, 'TolFun', ...
             1e-8*n_days, 'LargeScale', 'off');

while (found_best==0)
     rnd_start = calc_rand_start();
     tr_rnd_start = transform(rnd_start);

     [tmp_tr_results, tmp_like, tmp_flag, tmp_obs, tmp_grad, ...
      tmp_hess] = fminunc('calc_dir_like', tr_rnd_start, opt, ...
                  obs, n_days, epsilon);
```

```
tmp_param_est = untransform(tmp_tr_results);
tmp_all_like = [tmp_all_like, tmp_like];
tmp_all_params = [tmp_all_params; tmp_param_est];

% Keep track of best parameter estimates (according to
% likelihood).
if all(tmp_all_like >= tmp_like)
     best_params = tmp_param_est;
     best_like = tmp_like;
     best_flag = tmp_flag;
     best_hess = tmp_hess;
end

% Find the indices of estimates (including the best) that are
% within like_tol of the best estimate
like_err = abs(tmp_all_like - best_like) ./ abs(best_like);
good_like_idx = find( like_err < like_tol);

% Check if at least 3 (plus best_est) within like_tol of best,
% and that best is not the artifical value like_arbitrary
if (length(good_like_idx) >= 4)&&(best_like ~= like_arbitrary)

    % Now we check the parameter estimates are good enough.
    for i=1:length(good_like_idx)
        ith_good = tmp_all_params(good_like_idx(i),:);
        param_err_i(i,:) = abs(ith_good - best_params) ./ ...
                           abs(best_params);

        % If all of the individual parameter errors are within
```

```
            % rel_param_tol, that index is good enough.
            % close_enough has binary values (1 for yes)
            close_enough(i) = all(param_err_i(i,:) < rel_param_tol);
        end

        if (sum(close_enough) >= 4)
             found_best =1;
        end
    end

    % If we've tried iter_cutoff random starting points, give up.
    if (length(tmp_all_like) > iter_cutoff)
       best_params = [NaN, NaN, NaN, NaN];
       best_like = NaN;
       best_flag = NaN;
       best_hess = NaN * ones(4);
       save('err_opt_params', 'obs', 'best_like', 'best_params', ...
            'tmp_all_like', 'tmp_all_params', 'good_like_idx')

       sprintf('Too many attempts, saved some diagnostics, ...
           calling best_like = like_arbitrary, and params all NaN.')
       found_best = 1;
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate a random starting point *somewhere* in the parameter space
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function start_pt = calc_rand_start()
```

```
% If x is in [0,1], x/(1-x) is in [0, Inf]

a = rand;
rand_c = a/(1-a);
rand_delta2 = rand;
rand_delta3 = rand;
b = rand;
rand_shape = b/(1-b);

start_pt = [rand_c, rand_delta2, rand_delta3, rand_shape];
```

A.12 transform

```
function tr_params = transform(param_vect)
% This function takes a vector of parameters [c, delta2, delta3,
% shape] and transforms them from their natural ranges [0,Inf],
% [0,1], [0,1], [0,Inf] to the range [-Inf,Inf] using log or logit
% transformations.  The transform_any function can transform any
% variable in range [a,b] to [-Inf,Inf]

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WARNING: If you change the range for a variable, you MUST change
% it in untransform as well!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c = param_vect(1);
delta2 = param_vect(2);
delta3 = param_vect(3);
shape = param_vect(4);
```

```
tr_c = transform_any(c, 0, Inf); % c > 0
tr_delta2 = transform_any(delta2, 0, 1); % 0 < delta < 1
tr_delta3 = transform_any(delta3, 0, 1); % 0 < delta < 1
tr_shape = transform_any(shape, 0, Inf); % shape > 0

tr_params = [tr_c, tr_delta2, tr_delta3, tr_shape];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function transforms any value from [a,b] to a value in
% [-Inf,Inf].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x_trans = transform_any(x, a, b)

if (x < a || x > b)
    error(['Error in transform_any, ' num2str(x) ' ...
            is larger than ' num2str(b) ' or smaller than ' ...
             num2str(a)]);
end
if (a < 0 || b < 0)
    error(['Error in transform_any, ' num2str(a) ' or ' ...
        num2str(b) ' is less than zero']);
end
if (a > b)
    error(['Error in transform_any, lower bound ', ...
        'is larger than upper bound']);
end

% Special case: a = 0, b = Inf
```

```
% Transform using log(x), untransform using exp(y)
if ( a == 0 && isinf(b));
    x_trans = log(x);
% Special case: a = 0, b > 0
% Transform using log(x/(b-x)), untransform using b*exp(y)/(1+exp(y))
elseif ( a == 0 && ~isinf(b) && b > 0);
    x_trans = log(x/(b-x));
% Special case: a > 0, b = Inf
% Transform using log(x-a), untransform using exp(y) + a
elseif ( a > 0 && isinf(b));
    x_trans = log(x - a);
% General case: a > 0, b > 0
% Transform using log((x-a)/(b-x))
elseif ( a > 0 && b > 0);
    x_trans = log((x-a)/(b-x));
else
    error(['Error in transform_any: no case matches x = ' ...
           num2str(x) ', a=' num2str(a) ', b=' num2str(b)]);
end
```

A.13 untransform

```
function params = untransform(tr_var_vect)
% This function takes a vector of transformed parameters
% [tr_c, tr_delta2, tr_delta3, tr_shape] and un-transforms them from
% [-Inf,Inf] to their natural ranges [0,Inf], [0,1], [0,1], [0,Inf]
% using the inverse of the transformation done in transform. This
% function can un-transform any variable fron [-Inf,Inf] back to its
% range [a,b]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% WARNING: If you change the range for a variable, you MUST change it
% in transform as well!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tr_c = tr_var_vect(1);
tr_delta2 = tr_var_vect(2);
tr_delta3 = tr_var_vect(3);
tr_shape = tr_var_vect(4);

c = untransform_any(tr_c, 0, Inf);
delta2 = untransform_any(tr_delta2, 0, 1);
delta3 = untransform_any(tr_delta3, 0, 1);
shape = untransform_any(tr_shape, 0, Inf);

params = [c, delta2, delta3, shape];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function transforms any value from [-Inf, Inf] to a value in
% [a,b].
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y_untrans = untransform_any(y, a, b)

if (a < 0 || b < 0)
    error(['Error in untransform_any, ' num2str(a) ...
          ' or ' num2str(b) ' is less than zero']);
end
if (a > b)
    error(['Error in untransform_any, lower bound ' ...
        'is larger than upper bound']);
end
```

```
% Special case: a = 0, b = Inf
% Transform using log(x), untransform using exp(y)
if ( a == 0 && isinf(b));
    y_untrans = exp(y);
% Special case: a = 0, b > 0
% Transform using log(x/(b-x)), untransform using b*exp(y)/(1+exp(y))
elseif ( a == 0 && ~isinf(b) && b > 0);
    y_untrans = b*exp(y)/(1+exp(y));
% Special case: a > 0, b = Inf
% Transform using log(x-a), untransform using exp(y) + a
elseif ( a > 0 && isinf(b));
    y_untrans = exp(y)+a;
% General case: a > 0, b > 0
% Transform using log((x-a)/(b-x)), untransform using
% (b*exp(y)+a)/(1+exp(y))
elseif ( a > 0 && b > 0);
    y_untrans = (b*exp(y)+a)/(1+exp(y));
else
    error(['Error in transform_any: no case matches y = ' ...
        num2str(y) ', a=' num2str(a) ', b=' num2str(b)]);
end
```

Appendix B

Full model code

B.1 bootstrap_ci_growth

```
function bootstrap_ci_growth(filename, true_param_v, n_days, epsilon)

% Generate a point estimate of the parameters, then calculate a
% multivariate random distribution of parameter estimates using the
% asymptotic method (mistakenly called bootstrap here)

n_bootstrap = 500;

true_growth = calc_growth(true_param_v, n_days, epsilon);
obs_v = calc_obs(true_param_v, n_days, epsilon);

% Calculate best estimate
[param_est, like_est, flag_est, hess_est, num_iter] = ...
            recurse_cheat_params(filename, obs_v, true_param_v, ...
             n_days, epsilon)

growth_est = calc_growth(param_est, n_days, epsilon);
```

```
% In transformed param space, normally distributed.  Draw
% n_bootstrap values from multiv. norm. dist.  Then transform these
% back.
tr_param_est = transform(param_est);
m_var_cov = inv(hess_est);
tr_bootstrap_params = mvnrnd(tr_param_est, m_var_cov, n_bootstrap);

% Untransform estimates, and calculate growth rate for each
for i=1:n_bootstrap
   bootstrap_params(i,:) = untransform(tr_bootstrap_params(i,:));
   bootstrap_growth(i) = calc_growth(bootstrap_params(i,:), ...
                         n_days, epsilon);
end

save(filename, 'n_bootstrap', 'true_param_v', 'n_days', ...
     'epsilon', 'true_growth', 'obs_v', 'param_est', 'hess_est', ...
     'growth_est', 'm_var_cov', 'bootstrap_params', ...
     'bootstrap_growth')
```

B.2 calc_a

```
function a_matrix = calc_a(param_vec,t)

% The basic growth matrix, A, has three non-zero diagonals, growth,
% stasis and division (plus entries in the first row, corresponding
% to division by small cells).  The formula for growth from class i
% to i+1 is \gamma(t) * [1- \delta_i(t)].  (\gamma is the fraction of
% cells that grow, given they don't divide, and \delta_i is the
% fraction that divide. The formula for division of cells of size i
% is 2 * \delta_i(t).  The formula for stasis is set so that each
```

```
% column sums to 1.  (For i=1, this is [1 - \gamma(t)]*
% [1-\delta_i(t)] + 2 * delta_i(t), for i = m_class this is
% [1- \delta_i(t)], for all other i, it is [1- \gamma(t)]*
% [1- \delta_i(t)].  The formulas for \gamma(t) and \delta_i(t) are
% given in separate functions.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Note: this function assumes load_const has already been called.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Constants
% --------------------------
global m_class Delta_v

size_c = 1:m_class;
a_matrix = zeros(m_class, m_class); % Empty transition matrix.
% Matches size_classes, so delta_i works.
t_vect = t .* ones(size(size_c));
j = 1 + 1/Delta_v; % Where superdiag. starts

% Calculations
% --------------------------
gamma_t_vec = gamma(param_vec, t);
delta_vec = delta_i(param_vec, t_vect(1:m_class), size_c(1:m_class));

% Growth
growth_diag = gamma_t_vec .* [1 - delta_vec(1:m_class-1)];
a_matrix = a_matrix + diag(growth_diag, -1);

% Stasis
```

```
stasis_diag = zeros(1,m_class);
stasis_diag(1) = (1 - gamma_t_vec) * [1 - delta_vec(1)] + ...
                 2*delta_vec(1);
stasis_diag(2:m_class-1) = (1 - gamma_t_vec) .* [1 - ...
                           delta_vec(2:m_class-1)];
stasis_diag(m_class) = 1 - delta_vec(m_class);
a_matrix = a_matrix + diag(stasis_diag, 0);

% Division
div_top_row = 2* delta_vec(2:j-1);
a_matrix(1,2:j-1) = div_top_row;
% Because we don't want to overwrite the row marker.
div_diag(1) = 0;
div_diag(2:length(j:m_class)+1) = 2 * delta_vec(j:m_class);
% The first off-diagonal (a_mtx(1,2)) is 1, not 2, so this diag is
% j-2, not j-1
a_matrix = a_matrix + diag(div_diag, j-2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate fraction of cells in size class i that divide at time t.
% delta_i is a function of \delta_max, a, b and depends on the cell
% volume.  Cells are only allowed after t_delta.  If t and i are
% equal-length vectors, delta_i is calculated for each pair of t and
% i values
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function delta = delta_i(param_vec, t, i)

a_var = param_vec(1);
b_var = param_vec(2);
delta_max = param_vec(3);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Constants loaded in calc_a function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global t_delta v_min Delta_v


% Calculations
% ------------------------
v_i = v_min .* 2.^((i-1) .* Delta_v);

% If 0 < t < t_delta, the division rate should be zero.
is_late_enough = t>t_delta;
tmp_param = a_var .* (v_i.^ b_var);
nonzero_delta = tmp_param .* delta_max ./ (1 + tmp_param);
delta = is_late_enough .* nonzero_delta;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate fraction of cells at time t that grow to the next largest
% size class.  gamma is a function of gamma_max and e_star, and
% depends on incident radiation.  The fraction returned is constant
% for any given call to calc_a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function grow_frac=gamma(param_vec, t);

gamma_max = param_vec(4);
e_star = param_vec(5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Constants loaded in calc_a function
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global model_time model_rad


% We'd like to be able to do "t_idx = find(model_time == t, 1)", but
% matlab is not so good at rounding error, so we need to make sure
% they're within t_dist of each other.  Sorry.

t_dist = 10^-5;

close_enough = (((model_time - t) < t_dist) & ((model_time-t) > ...
                  -t_dist));
t_idx = find(close_enough, 1);
rad = model_rad(t_idx);
grow_frac = (1 - exp(- rad ./ e_star)) .* gamma_max;
```

B.3 calc_b

```
function b_matrix = calc_b(param_vec,t)

% pop_vec(t+1) = calc_b(param, t)*pop(t).  In other words,
% calc_b(param, t) returns a matrix projecting a vector at time t to
% time t+1, where time is measured in hours.  The growth matrix A
% gives population changes over time dt, which is set so that no more
% than one division or size change will occur. (B(t) is the product
% A(t+1-dt)*...*A(t+dt)*A(t).  Element a_i_j represents the
% probability of moving from state j to state i during one full
% timestep 1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Note: this function assumes load_const has alredy been called.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Constants
% ---------------
global m_class dt

t_mod = mod(t, 24);

b_matrix = eye(m_class);
for j = 0:(1/dt-1)
   a_matrix = calc_a(param_vec, t_mod + j * dt);
   b_matrix = a_matrix * b_matrix;
end
```

B.4 calc_dir_like

```
function neg_like = calc_dir_like(tr_var_vect, obs, n_days, epsilon)

% Calculate the likelihood of obs coming from a projecting using the
% untransformed version of tr_var_vect.

like_arbitrary = 10^15;

% Transform back
param_vect = untransform(tr_var_vect);
dir_shape = param_vect(6);

% Calculate neg log likelihood of each observation
```

```
norm_v(:,1) = obs(:,1);
t_end = n_days*24+1;

for t=2:t_end
    B = calc_b(param_vect, t-2);
    proj_v = B * obs(:,t-1);
    norm_v(:,t) = (proj_v + epsilon) ./ sum(proj_v + epsilon);
end

like_all = dir_calc(obs, norm_v, dir_shape);
neg_like = sum(like_all);

if (isnan(neg_like) || isinf(neg_like))
     ['neg_like was NaN or Inf, set to ' num2str(like_arbitrary)]
     neg_like = like_arbitrary;
end
```

B.5 calc_growth

```
function growth = calc_growth(param_vec, n_days, epsilon)

% Calculate the deterministic growth rate associated with a set of
% parameter values

global real_w
load_const

w_0 = obs_w(:,1);
n_obs = size(param_vec,1);
t_end = n_days*24+1;
```

```
% Set up so that you can calculate multiple growth rates at once
%for i=1:n_obs
for i=1:1
   u_prod(i,:,:) = eye(m_class);

   for j=1:t_end
      u_tmp = calc_b(param_vec(i,:), j-1);
      u_prod(i,:,:) = squeeze(u_prod(i,:,:)) * u_tmp;
   end

   % Multiply together all j entries for this i.
   u(i,:) = squeeze(u_prod(i,:,:))*w_0;
   growth(i) = log(sum(u(i,:)));
end
```

B.6 calc_obs

```
function obs_m = calc_obs(param_vec, n_days, epsilon)

% This function calculates an artificial observation from a set of
% parameter values by projecting forward one timestep using calc_b,
% then, removing zeros as appropriate, and drawing from a Dirichlet
% distribution.

dir_shape = param_vec(6);

% pi_vec(i,:) is the population vector for time i-1 (since time goes
% from 0 to 24, but indices must start at 1.)
```

```
global real_w
load_const

obs_m(:,1) = obs_w(:,1);

% We want to project from time 0 to time 24.  This means obs_m(:,1)
% is the time 0 observation.  The time 1 observation we get by
% projecting from time 0 to time 1.  Which means that we want t=0 for
% calc_b with obs_m(:,1) to be saved as obs_m(:,2).  Sensible, isn't
% it?
t_end = n_days*24+1;

for i=2:t_end
    B = calc_b(param_vec, i-2);
    proj_v = B * obs_m(:,i-1);
    norm_v = proj_v / sum(proj_v);
    tmp_obs(:,i) = draw_from_dir(norm_v, dir_shape);
    obs_m(:,i) = (tmp_obs(:,i)+epsilon)./sum(tmp_obs(:,i)+epsilon);
end
```

B.7 dir_calc

```
function neg_like = dir_calc(obs_vect, theor_prob_vect, dir_shape)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate likelihood of drawing obs_vect from a Dirichlet
% distribution with expected value theor_prob_vect and a precision
% parameter dir_shape
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Prob is \Pi x^(a-1) * \gamma(\Sigma(a))/ \Pi(\gamma(a))
% Like is -[\Sigma (a-1)log(x) + log(gamma(\Sigma(a))) -
% \Sigma(log(gamma(a))]
% Which can be rewritten: -\Sigma (a-1)log(x) - gammaln(\Sigma(a)) +
% \Sigma(gammaln(a))

c_vect = theor_prob_vect .* dir_shape;

term1 = sum((c_vect - 1) .* log(obs_vect));
term2 = gammaln(dir_shape);
term3 = sum(gammaln(c_vect));

neg_like = -term1 - term2 + term3;
```

B.8 day733033_2_const

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load 'Vhists', 'volbins' and 'Edata' from day733033_2_forLori.mat,
% or any .mat file containing these variables.  Define several model
% constants.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load('day733033_2_forLori.mat', 'Vhists', 'volbins', 'Edata')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This mat file also contains the following variables, which we
% ignore: a, gmax, b, Estar, c, Vmod

% Constants
m_class = 57; % Number of size classes.  In paper, is 57
dt = 10/60; % dt is 10 minutes, time is measured in hours
```

```
% Defined in load_const as real_w, and used to create obs_w
% --------------------------------
observ = Vhists;

% In delta_i
% --------------------------------
t_delta = 6; % Time after which division is allowed
v_min = volbins(1); % Minimum phytoplankton size

% Also in calc_a
Delta_v = 0.125;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% In incident_rad
% --------------------------------
E = Edata(:,2)';
% Time vector corresponding to E is in hours and starts the hour dawn
% occurs in.
time_days = Edata(:,1)';
```

B.9 draw_from_dir

```
function obs_vect = draw_from_dir(theor_pi_vec, dir_shape)

% This function takes a mean/theoretical pi value (theor_pi) and a
% measure of the variance (phi), and uses gamma random varibles to
% generate an a Dirichlet random variable.  Note: The larger 'phi'
% is, the smaller the variance, and vice-versa.  Phi must be strictly
% greater than zero.
```

```
Y_i_vec = gamrnd(dir_shape .* theor_pi_vec, 1);
Y = sum(Y_i_vec);
obs_vect = Y_i_vec ./ Y;
```

B.10 load_const

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function loads the script day733033_2_const, which defines a
% bunch of model constants (m_class, dt, observ, t_delta, v_min,
% Delta_v, E, time_days). This script takes those constants and
% processes them to clean up the data and makes it more useful
% elsewhere (creates real_w, model_time and model_rad)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%default_const
global m_class dt real_w t_delta v_min Delta_v E model_time model_rad

% Load constants
day733033_2_const

real_w = observ;

for j=1:25
    obs_w(:,j) = (real_w(:,j)+eps)./sum(real_w(:,j)+eps);
end

if (sum(sum(real_w)) ~= 25)
    ['Error in load_const - real_w does not sum to 25']
    sum(sum(real_w))
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate incoming radiation as function of time
% Dawn happens at t=0

sample_interval = 24/length(E); % To get samples/hour, take inverse.

% Change any negative values of E to zero.
pos_E = (E>0);
E = E .* pos_E;

% Figure out when dawn is, and shift so that the first hour is the
% hour where dawn occurs.

% Find index of first element of E greater than 1, and get
% corresponding time.
time_first_light = time_days(find(E>0,1));
dawn_hour = floor(time_first_light);
dawn_index = find((dawn_hour-0.05 < time_days), 1);

length_post_dawn = length(E) - dawn_index + 1;
% t_vext and rad_vect start with dawn_hour as t=0.
rad_vect(1:length_post_dawn) = E(dawn_index:end);
rad_vect(length_post_dawn+1:length(E)) = E(1:dawn_index-1);
% t_vect starts with dawn_hour as t=0, and assumes the original
% sampling happened every time_interval, starting exactly on the
% hour.
t_vect = 0:sample_interval:(24-sample_interval);
```

```
model_time = 0:dt:24;
model_rad = interp1(t_vect, rad_vect, model_time, 'linear', ...
                    'extrap');
```

B.11 recurse_cheat_params

```
function [best_params, best_like, best_flag, best_hess, ...
          num_iter] = recurse_cheat_params(filename, obs, ...
          true_param_v, n_days, epsilon)

% Calculate the best parameter estimate using fminunc on the
% function calc_dir_like.  Use the following workaround to deal with
% fminunc's inaccuracies: first, use the known true parameter values
% as one starting point, then try iter_cutoff random starting
% points, where the space of starting points is specified in
% calc_rand_start.  Return the best estimate.

file_test = ['recurse_cheat_' filename]

iter_cutoff = 10;

like_arbitrary = 10^15;
% Likelihood estimates must be within this tol. of each other.
like_tol = 1e-5;

found_best = 0;
num_iter = 0;
good_like_idx = [];

warning('off', 'optim:fminunc:SwitchingMethod');
```

```
% Set options for recursion.
opt = optimset('TolX', 1e-8*n_days,'maxIter', 1000, 'TolFun', ...
               1e-8*n_days, 'LargeScale', 'off');


% Goal: Just find the best estimate among those generated by
% starting at the true parameter values and iter_cutoff random
% starting points.  Save the one with the best (lowest) negative log
% likelihood.
tmp_all_like = [];
tmp_all_params = [];
tmp_all_start = [];
tmp_all_growth = [];
tmp_all_hess = [];

tr_true_v = transform(true_param_v);

[truest_tr_results, truest_like, truest_flag, truest_obs, ...
        truest_grad, truest_hess] = fminunc('calc_dir_like', ...
        tr_true_v, opt, obs, n_days, epsilon);
truest_params = untransform(truest_tr_results);
truest_growth = calc_growth(truest_params, n_days, epsilon);

     tmp_all_like = truest_like;
     tmp_all_params = truest_params;
     tmp_all_start = tr_true_v;
     tmp_all_growth = truest_growth;
     tmp_all_hess(1,:,:) = truest_hess;
```

```
      best_params = truest_params;
      best_like = truest_like;
      best_flag = truest_flag;
      best_hess = truest_hess;
      best_growth = truest_growth;

for count_start = 1:iter_cutoff
      count_start
      rnd_start = calc_rand_start();
      tr_rnd_start = transform(rnd_start);

      [tmp_tr_results, tmp_like, tmp_flag, tmp_obs, tmp_grad, ...
              tmp_hess] = fminunc('calc_dir_like', tr_rnd_start, ...
              opt, obs, n_days, epsilon);
      tmp_param_est = untransform(tmp_tr_results);
      tmp_growth_est = calc_growth(tmp_param_est, n_days, epsilon);

      format long
      tmp_all_like = [tmp_all_like, tmp_like]
      tmp_all_params = [tmp_all_params; tmp_param_est]
      tmp_all_start = [tmp_all_start; rnd_start]
      tmp_all_growth = [tmp_all_growth, tmp_growth_est]
      % Add one to matrix dimension, put hessian there...
      tmp_all_hess(size(tmp_all_hess,1)+1,:,:) = tmp_hess;

      % Keep track of best parameter estimates (according to
      % likelihood) if like_est is the smallest element of the
      % likelihood vector
      if all(tmp_all_like >= tmp_like)
           best_params = tmp_param_est;
```

```
        best_like = tmp_like;
        best_flag = tmp_flag;
        best_hess = tmp_hess;
        best_growth = tmp_growth_est;
    end

    % Find the indices of estimates (including the best) that are
    % within tol. percent of the best estimate
    like_err = abs(tmp_all_like - best_like) ./ abs(best_like);
    good_like_idx = find( like_err < like_tol);

    save(file_test, 'obs', 'n_days', 'epsilon', 'like_tol', ...
        'true_param_v', 'truest_like', 'truest_params', ...
        'truest_growth', 'truest_hess', ...
        'best_like', 'best_params', 'best_hess', 'best_growth', ...
        'tmp_all_like', 'tmp_all_params', 'tmp_all_start', ...
        'tmp_all_growth', 'tmp_all_hess', 'good_like_idx')

end



function start_pt = calc_rand_start()

% x/(1-x) Transforms a 0-1 random variable to [0 Inf]

a = rand;
rand_a = a/(1-a);
b = rand;
rand_b = b/(1-b);
```

```
rand_d = rand;
rand_g = rand;
e = rand;
rand_e = e/(1-e);
s = rand;
rand_s = s/(1-s);

%start_pt = [rand_a, rand_b, rand_d, rand_g, rand_e, rand_s];
%start_pt = [13.191, 5.6982, 0.0089308, 0.13905, 500, 10^6];



% RV of the form 10^(x) where x is between min & max

% A_start between 10^(-5) and 10^5
a_min_exp = -5;
a_max_exp = 5;
a_exp = a_min_exp + (a_max_exp-a_min_exp).*rand;
rand2_a = 10^(a_exp);

% B_start between 10^(-5) and 10^5
b_min_exp = -5;
b_max_exp = 5;
b_exp = b_min_exp + (b_max_exp-b_min_exp).*rand;
rand2_b = 10^(b_exp);

% D_start between 10^(-5) and 10^(0)
d_min_exp = -5;
d_max_exp = 0;
d_exp = d_min_exp + (d_max_exp-d_min_exp).*rand;
```

```
rand2_d = 10^(d_exp);

% G_start between 10^(-5) and 10^(0)
g_min_exp = -5;
g_max_exp = 0;
g_exp = g_min_exp + (g_max_exp-g_min_exp).*rand;
rand2_g = 10^(g_exp);

% E_start between 10^0 and 10^5
e_min_exp = 0;
e_max_exp = 5;
e_exp = e_min_exp + (e_max_exp-e_min_exp).*rand;
rand2_e = 10^(e_exp);

% S_start between 10^1 and 10^10
s_min_exp = 1;
s_max_exp = 10;
s_exp = s_min_exp + (s_max_exp-s_min_exp).*rand;
rand2_s = 10^(s_exp);


start_pt = [rand2_a, rand2_b, rand2_d, rand2_g, rand2_e, rand2_s]
%start_pt = [13, 6, 0.15, 0.15, 400, 10^6];
```

B.12 recurse_infinite_rand

```
function recurse_infinite_rand(filename, obs, n_days, epsilon)

% Generate "infinitely many" (until stopped) starting points, and
% calculate parameter estimates while starting at each.  Later,
```

```
% we'll find the best likelihood and use the corresponding parameter
% estimate and hessian matrix.

file_test = ['recurse_inf_rand_' filename]

like_arbitrary = 10^15;
% Likelihood estimates must be within this tol. of each other.
like_tol = 1e-5;

found_best = 0;
num_iter = 0;
good_like_idx = [];

warning('off', 'optim:fminunc:SwitchingMethod');

% Set options for recursion.  May want to later make these relative.
opt = optimset('TolX', 1e-8*n_days,'maxIter', 1000, 'TolFun', ...
                1e-8*n_days, 'LargeScale', 'off');

% Goal: Find at least 3 parameter estimates whose likelihoods are
% within tol. of the best one found.  Then make sure that there are
% at least 3 estimates where all the param. estimates are within
% tol. of each other.  Save the best one (by the likelihood metric)
% and return the results (as well as the number of tries needed)

tmp_all_like = [];
tmp_all_params = [];
tmp_all_start = [];
tmp_all_growth = [];
tmp_all_hess = [];
```

```
while 1
     length(tmp_all_like)+1
     rnd_start = calc_rand_start();
     tr_rnd_start = transform(rnd_start);

     [tmp_tr_results, tmp_like, tmp_flag, tmp_obs, tmp_grad, ...
          tmp_hess] = fminunc('calc_dir_like', tr_rnd_start, ...
          opt, obs, n_days, epsilon);
     tmp_param_est = untransform(tmp_tr_results);
     tmp_growth_est = calc_growth(tmp_param_est, n_days, epsilon);

     tmp_all_like = [tmp_all_like, tmp_like];
     tmp_all_params = [tmp_all_params; tmp_param_est];
     tmp_all_start = [tmp_all_start; rnd_start];
     tmp_all_growth = [tmp_all_growth, tmp_growth_est];
     % Add one to matrix dimension, put hessian there...
     tmp_all_hess(size(tmp_all_hess,1)+1,:,:) = tmp_hess;


     % Keep track of best parameter estimates (according to
     % likelihood) if like_est is the smallest element of the
     % likelihood vector
     if all(tmp_all_like >= tmp_like)
          best_params = tmp_param_est;
          best_like = tmp_like;
          best_flag = tmp_flag;
          best_hess = tmp_hess;
          best_growth = tmp_growth_est;
     end
```

```
    % Find the indices of estimates (including the best) that are
    % within tol. percent of the best estimate
    like_err = abs(tmp_all_like - best_like) ./ abs(best_like);
    good_like_idx = find( like_err < like_tol);

    save(file_test, 'obs', 'n_days', 'epsilon', 'like_tol', ...
       'best_like', 'best_params', 'best_hess', 'best_growth', ...
       'tmp_all_like', 'tmp_all_params', 'tmp_all_start', ...
       'tmp_all_growth', 'tmp_all_hess', 'good_like_idx')
end

function start_pt = calc_rand_start()

% RV of the form 10^(x) where x is between min & max

% A_start between 10^(-5) and 10^5
a_min_exp = -5;
a_max_exp = 5;
a_exp = a_min_exp + (a_max_exp-a_min_exp).*rand;
rand2_a = 10^(a_exp);

% B_start between 10^(-5) and 10^5
b_min_exp = -5;
b_max_exp = 5;
b_exp = b_min_exp + (b_max_exp-b_min_exp).*rand;
rand2_b = 10^(b_exp);

% D_start between 10^(-5) and 10^(0)
d_min_exp = -5;
```

```
d_max_exp = 0;
d_exp = d_min_exp + (d_max_exp-d_min_exp).*rand;
rand2_d = 10^(d_exp);

% G_start between 10^(-5) and 10^(0)
g_min_exp = -5;
g_max_exp = 0;
g_exp = g_min_exp + (g_max_exp-g_min_exp).*rand;
rand2_g = 10^(g_exp);

% E_start between 10^0 and 10^5
e_min_exp = 0;
e_max_exp = 5;
e_exp = e_min_exp + (e_max_exp-e_min_exp).*rand;
rand2_e = 10^(e_exp);

% S_start between 10^1 and 10^10
s_min_exp = 1;
s_max_exp = 10;
s_exp = s_min_exp + (s_max_exp-s_min_exp).*rand;
rand2_s = 10^(s_exp);

start_pt = [rand2_a, rand2_b, rand2_d, rand2_g, rand2_e, rand2_s]
```

B.13 transform

```
function tr_vect = transform(param_vect)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% WARNING: If you change the range for a variable, you MUST change it
% in untransform as well!
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

a_var = param_vect(1);
b_var = param_vect(2);
delta_max = param_vect(3);
gamma_max = param_vect(4);
e_star = param_vect(5);
dir_shape = param_vect(6);

tr_a_var = transform_any(a_var, 0, Inf); % a_var > 0
tr_b_var = transform_any(b_var, 0, Inf); % b_var > 0
tr_delta_max = transform_any(delta_max, 0, 1); % 0 < delta_max < 1
tr_gamma_max = transform_any(gamma_max, 0, 1); % 0 < gamma_max < 1
tr_e_star = transform_any(e_star, 0, Inf); % e_star > 0
tr_dir_shape = transform_any(dir_shape, 0, Inf); % dir_shape > 0

tr_vect = [tr_a_var, tr_b_var, tr_delta_max, tr_gamma_max, ...
           tr_e_star, tr_dir_shape];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transform any non-negative range into the range (-Inf, Inf), and
% return value of parameter x in transformed range.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x_trans = transform_any(x, a, b)

if (x < a || x > b)
    error(['Error in transform_any, ' num2str(x) ...
           ' is larger than ' num2str(b) ' or smaller than ' ...
           num2str(a)]);
end
```

```
if (a < 0 || b < 0)
    error(['Error in transform_any, ' num2str(a) ' or ' ...
            num2str(b) ' is less than zero']);
end
if (a > b)
    error(['Error in transform_any, lower bound is larger than' ...
           ' upper bound']);
end

% Special case: a = 0, b = Inf
% Transform using log(x), untransform using exp(y)
if ( a == 0 && isinf(b));
    x_trans = log(x);
% Special case: a = 0, b > 0
% Transform using log(x/(b-x)), untransform using b*exp(y)/(1+exp(y))
elseif ( a == 0 && ~isinf(b) && b > 0);
    x_trans = log(x/(b-x));
% Special case: a > 0, b = Inf
% Transform using log(x-a). Untransform using exp(y) + a
elseif ( a > 0 && isinf(b));
    x_trans = log(x - a);
% General case: a > 0, b > 0
% Transform using log((x-a)/(b-x))
elseif ( a > 0 && b > 0);
    x_trans = log((x-a)/(b-x));
else
    error(['Error in transform_any: no case matches x = ' ...
           num2str(x) ', a=' num2str(a) ', b=' num2str(b)]);
end
```

B.14 untransform

```
function param_v = untransform(tr_var_vect)

% Take a vector transformed using transform, and return it to
% original values

tr_a_var = tr_var_vect(1);
tr_b_var = tr_var_vect(2);
tr_delta_max = tr_var_vect(3);
tr_gamma_max = tr_var_vect(4);
tr_e_star = tr_var_vect(5);
tr_dir_shape = tr_var_vect(6);

a_var = untransform_any(tr_a_var, 0, Inf);
b_var = untransform_any(tr_b_var, 0, Inf);
delta_max = untransform_any(tr_delta_max, 0, 1);
gamma_max = untransform_any(tr_gamma_max, 0, 1);
e_star = untransform_any(tr_e_star, 0, Inf);
dir_shape = untransform_any(tr_dir_shape, 0, Inf);

param_v = [a_var, b_var, delta_max, gamma_max, e_star, dir_shape];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Transform any value from (-Inf, Inf) to a value in (a,b).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y_untrans = untransform_any(y, a, b)

if (a < 0 || b < 0)
    error(['Error in untransform_any, ' num2str(a) ' or ' ...
```

```
            num2str(b) ' is less than zero']);
end
if (a > b)
    error(['Error in untransform_any, lower bound is larger ' ...
           'than upper bound']);
end

% Special case: a = 0, b = Inf
% Transform using log(x), untransform using exp(y)
if ( a == 0 && isinf(b));
    y_untrans = exp(y);
% Special case: a = 0, b > 0
% Transform using log(x/(b-x)), untransform using b*exp(y)/(1+exp(y))
elseif ( a == 0 && ~isinf(b) && b > 0);
    y_untrans = b*exp(y)/(1+exp(y));
% Special case: a > 0, b = Inf
% Transform using log(x-a). Untransform using exp(y) + a
elseif ( a > 0 && isinf(b));
    y_untrans = exp(y)+a;
% General case: a > 0, b > 0
% Transform using log((x-a)/(b-x)), untransform using
% (b*exp(y)+a)/(1+exp(y))
elseif ( a > 0 && b > 0);
    y_untrans = (b*exp(y)+a)/(1+exp(y));
else
    error(['Error in transform_any: no case matches y = ' ...
           num2str(y) ', a=' num2str(a) ', b=' num2str(b)]);
end
```

Bibliography

[1] J. Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 44(2):139–177, 1982.

[2] J. Aitchison and J. J. Egozcue. Compositional data analysis: Where are we and where should we be heading. *Mathematical Geology*, 37(7):829–850, 2005.

[3] A. Azzalini. *Statistical Inference based on the likelihood.* Number 68 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1996.

[4] C. Calder, M. Lavine, P. Müller, and J. S. Clark. Incorporating multiple sources of stochasticity into dynamic population models. *Ecology*, 84(6):1395–1402, 2003.

[5] H. Caswell. *Matrix population models: construction, analysis, and interpretation.* Sinauer Associates, Inc., 2nd edition, 2001.

[6] M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions.* John Wiley & Sons, Inc, 2nd edition, 1993.

[7] G. K. Grunwald, A. E. Raftery, and P. Guttorp. Time series of continuous proportions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(1):103–116, 1993.

[8] The MathWorks. Matlab. 2007a, Natick, MA.

[9] R. J. Olson, A Shalapyonok, and H. M. Sosik. Bench top flowcytobot measured *Synechococcus* time series of continuous proportions. Unpublished data.

[10] R. J. Olson, A Shalapyonok, and H. M. Sosik. An automated submersible flow cytometer for analyzing pico- and nanophytoplankton: Flowcytobot. *Deep-sea Research, part 1*, 50:301–315, 2003.

[11] C. S. Reynolds. *The Ecology of Phytoplankton.* Ecology, Biodiversity and Conservation. Cambridge University Press, 2006.

[12] H. M. Sosik, R. J. Olson, M. G. Neubert, A. Shalapyonok, and A. R. Solow. Growth rates of coastal phytoplankton from time-series measurements with a submersible flow cytometer. *Limnological Oceanography*, 48(5):1756–1765, 2003.