UNIVERSITY OF CALIFORNIA

Los Angeles

A mathematical model of the growth and distribution of *Dendraster excentricus*

A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Engineering

by

J. Stanley Warford

1984

The dissertation of J. Stanley Warford is approved.

Jack W. Carlyle

J. Carlyle, Committee Chair

Walter J. Karplus

W. Karplus

Jim K. Omura

J. Omura

Peter Narins

P. Narins

Richard R. Vance

R. Vance

University of California, Los Angeles

1984

TABLE OF CONTENTS

FIGURES

ACKNOWLEDGMENTS

I acknowledge my advisors at UCLA, Jim Omura in Systems Science and Jack Carlyle in Computer Science, whose moral support contributed the the successful completion of my degree. Rick Vance contributed substantially to my understanding of the biological issues in this study. I also thank Steve Davis and Joe Williams who initiated the ecological project on which this dissertation is based. Steve has been a particular source of encouragement. Thanks to Ken Coley for his help in the field. The project would not have been possible without the assistance of the Department of the Navy, who allow ecological research within the Pacific Missile Test Center at Pt. Mugu. This research was funded by Pepperdine University, where this author teaches. I gratefully acknowledge the support of Ken Perrin, Natural Science Division Chair at Pepperdine for procuring that support. I thank my family for their perseverance.

VITA

October 16, 1944--Born, Bakersfield, California

1966--B.S. Mathematics, Pepperdine College

1968--M.S. Physics, Rensselaer Polytechnic Institute

1971-1975--Aerospace Engineer, Honeywell, Inc.

1975-1984--Professor of Computer Science, Pepperdine University

PUBLICATIONS

"Diffusion of Tin into Zinc", Physical Review, B, February 15, 1970

"Computer Aided Magnetic Field Calculations for Nonlinear Shielding Materials", with D. Goodrich, IEEE Transactions on Magnetics, September, 1973

PAPER PRESENTED

"Population Dynamics of Sand Dollars in Mugu Lagoon: A Mathematical Model", Third Biennial Mugu Lagoon/San Nicolas Island Ecological Research Symposium, October 20, 1983

ABSTRACT OF THE DISSERTATION

A mathematical model of the growth and distribution of *Dendraster excentricus*

by

J. Stanley Warford

Doctor of Philosophy in Engineering

University of California, Los Angeles, 1984

Professor J. Carlyle, Chair

A mathematical model is constructed to describe the growth dynamics of a low density population of *Dendraster excentricus* (common name, sand dollar) in the Pt. Mugu, California, lagoon. The model integrates the effects of recruitment, growth, and mortality into a single system based on Leslie matrix theory. The goal is to estimate the values of the controlling parameters in the ecological system from a time sequence of histograms of measured size data. A squared error fit criterion is defined over the time sequence and a nonlinear least squares method is employed to estimate the parameter values.

A computer implementation of both the model and the

minimization algorithm is presented. This system identification study indicates, on the basis of best model fit to the data, that growth of Dendraster may occur in three age specific stages with a distinct growth rate for each stage, in contrast to previous studies which indicate two growth stages for this species. Quantitative estimates for growth rates, survival probabilities, and fecundity are reported. The model is also extended to account for the effects of immigration into the system. Spatial variations at the data collection site are investigated.

The methodology employed in this study is unique in that the various components of the system--growth, recruitment, and mortality--are integrated into a single model. The identification is performed system-wide with all of the components in place. This technique allows the growth curves to be estimated directly from the time sequence of size histograms. The methodology should be of general value in biological modeling since the size of an organism is invariably easier to measure than its age.

1. Introduction

This dissertation describes a mathematical model of a biological system. It is based on data taken in an ecological research project directed by Professors Stephen D. Davis and Joseph B. Williams in the Natural Science Division at Pepperdine University, where this author also teaches.

1.1 The location

The data were collected at the Point Mugu Naval Test Center. The site is pristine since access to the area by the general public is prohibited. The objective of the research is to describe the growth dynamics of *Dendraster excentricus*, commonly known as sand dollar, in the eastern arm of Point Mugu Lagoon.

The lagoon is bounded on the north by a *Salicornia* marsh and on the south by a barrier beach as shown in Figure 1.1-1. The mouth of the lagoon is on the far west end of the eastern arm.

In the winter of 1976, nine sampling stations were

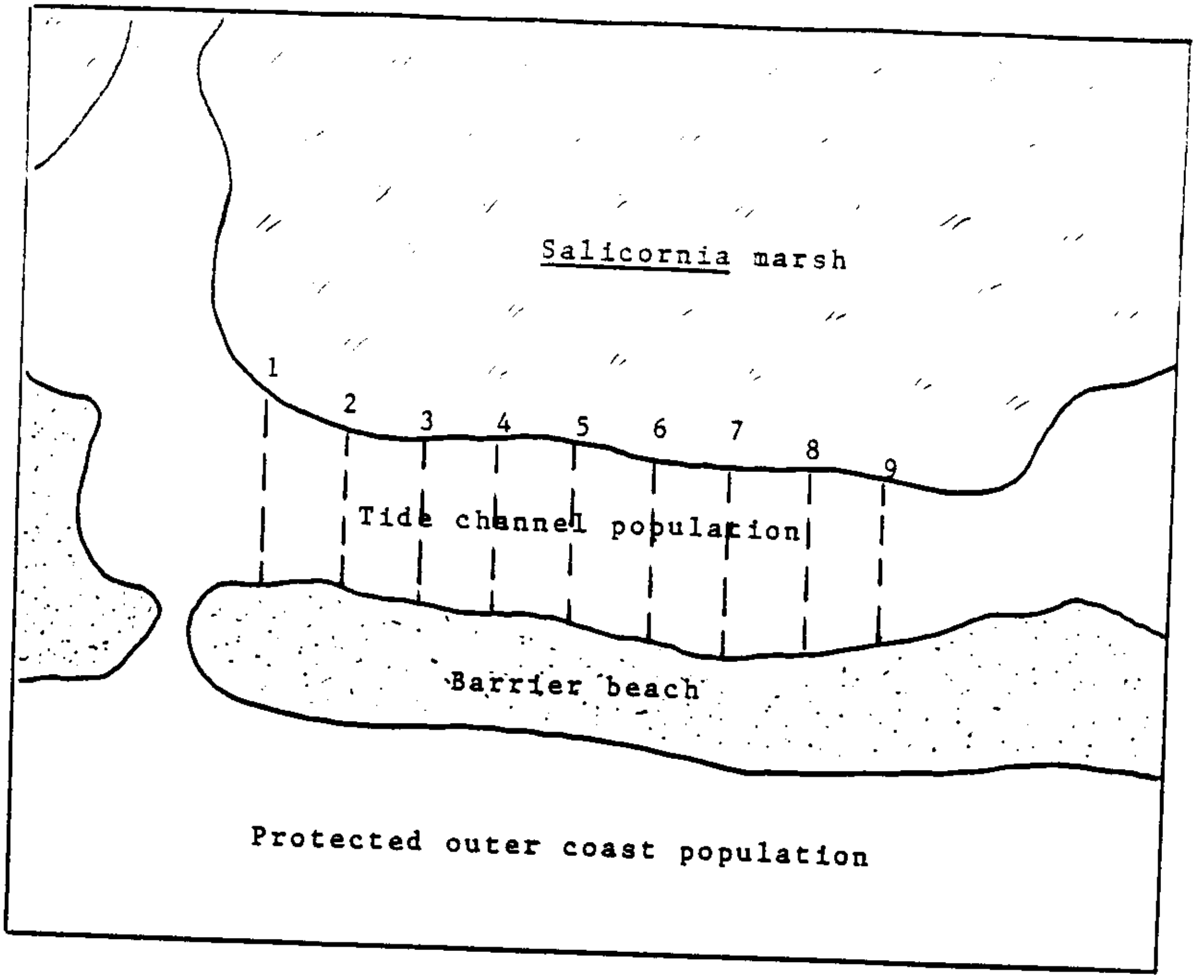


Figure 1.1-1. Pt. Mugu Lagoon.

established at 100 meter intervals along the lagoon. After some preliminary data were taken to assess the viability of the project, a systematic data gathering procedure was established beginning in April, 1977 (i.e. 4-77). Data was taken at monthly intervals from 4-77 to 1-78.

In February of 1978 Southern California received record breaking rains. As a result the entire Dendraster population was annihilated. Some sporadic data was taken the following few years, but the population was extremely low compared to previous levels.

In 1982 the population began to re-establish itself Data are now available following the established data taking procedure for 8-81, 10-81, 12-81, 2-82, and the six month period from 7-82 to 1-83. The recent severe storm in 2-83 characterized by high tides and massive debris in the lagoon has again annihilated the population.

It should be mentioned that this Dendraster population under study is probably not typical of the species. In the protected outer coast region shown in Figure 1.1-1 is a population of much greater density than that found in the lagoon. It is a more common habitat for Dendraster than is the lagoon.

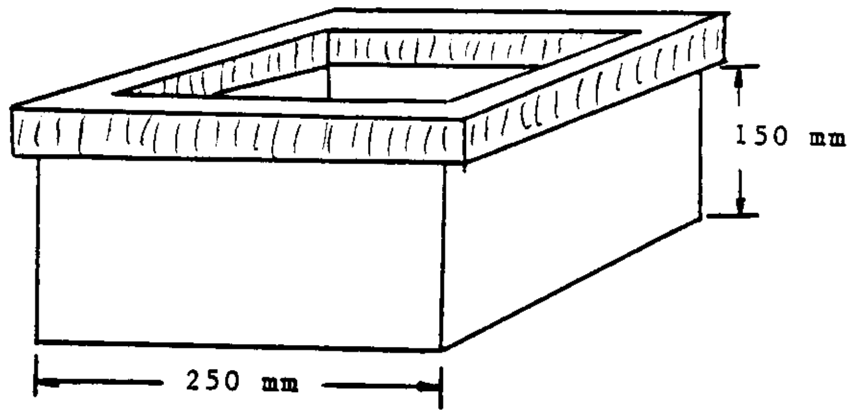
The data taken on Dendraster as reported in the literature generally falls into one of two categories: field data and laboratory data. Field data are desirable because it is a description of the "real world". But some data are difficult, if not impossible, to obtain in the field. A laboratory environment allows such data to be taken under repeatable and controlled conditions. The price to be paid, however, is the inability to duplicate exactly the environmental conditions in the field. In that sense the laboratory envioronment is not a typical one.

The data reported here can perhaps best be described as lying midway between these two extremes. The conditions in the lagoon are mild enough that an extensive, systematic data gathering procedure is possible. The level of detail in this data would be practically impossible to obtain in outer coast habitats. So it is more "real" than laboratory data since it is truly field data. But it is less "real" than it would be if taken in an outer coast population.

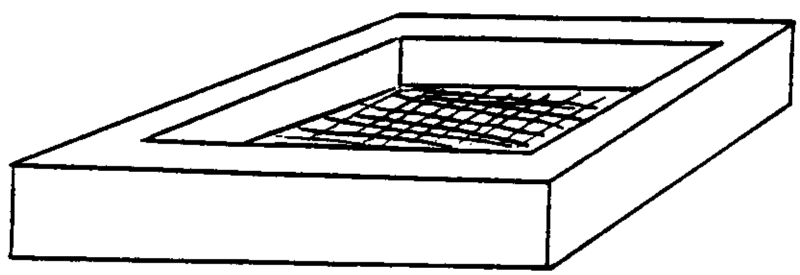
1.2 The sampling procedure

Samples were taken during low tide. Water depth was always less than about one meter to facilitate sampling.

An individual sample was taken with an open ended box with metal sides as shown in Figure 1.2-1. The sides are each 250 mm in length, forming a square of area 0.0625 m^2.



(a) Sample box.



(b) Filter box.

Figure 1.2-1. Sampling equipment.

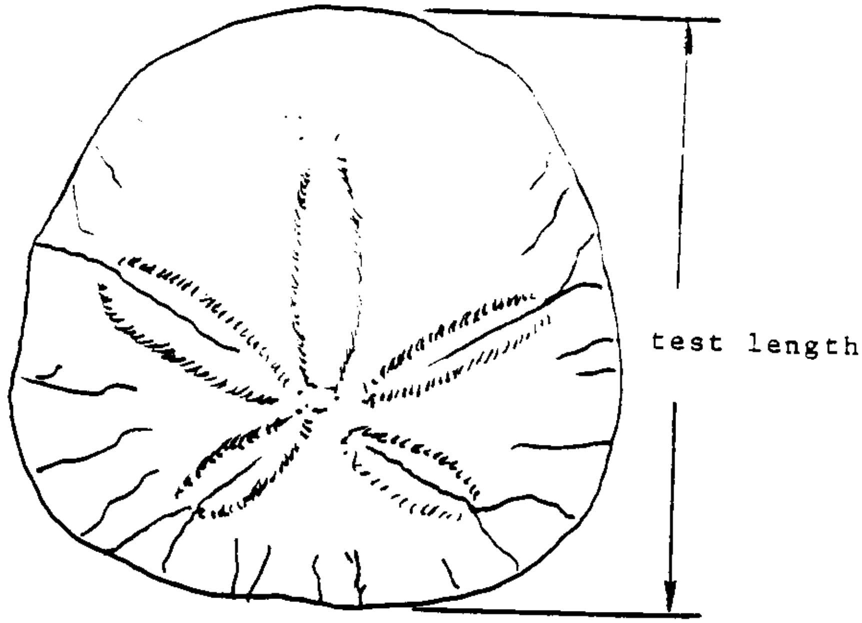
They extend below a wooden frame to a depth of 150 mm. The box is inserted into the sand. The sand is then removed to a depth of about 200 mm and sifted through a wire screen, also shown in the figure, with a mesh spacing of 2 mm.

A recording is made of each Dendraster filtered out. The Dendraster shell is called its 'test'. The test length is recorded as the diameter passing through the anal pore and the madraporite as shown in Figure 1.2-2. Measured test lengths are from 3 mm to about 75 mm. A notation is also made as to whether the individual is alive or dead.

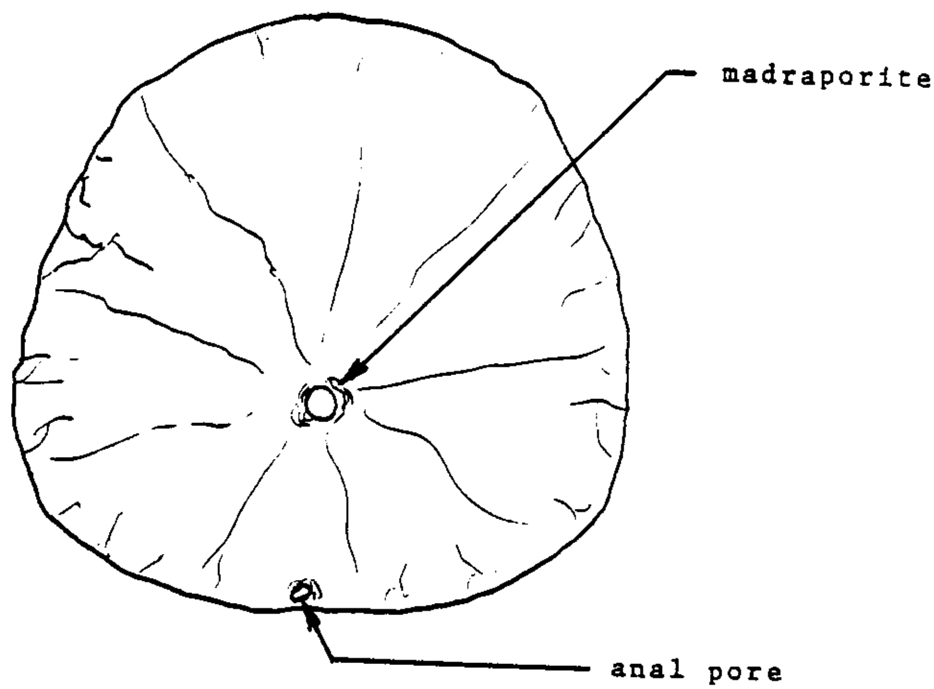
Two workers were required to sample the entire lagoon. They started at the barrier beach side of station 1. Each worker took a random digit between 1 and 9 from a table of random numbers. The random digits determined the number of paces each worker took along the barrier beach in opposite directions.

Both workers took their first individual sample at mean tide level. After recording the data and discarding the samples behind them, they took 5 paces into the lagoon toward the Salicornia marsh. At that location they took another sample. They continued sampling at 5 pace intervals until the lagoon had been crossed.

An additional termination condition that was established early in the project was that the total number



(a) Top view



(b) Bottom view

Figure 1.2-2. _Dendraster excentricus_.

of individual samples taken be at least 15. This was to insure a large enough sample size even in the case of a narrow constriction in the lagoon. It was rarely invoked.

Figure 1.2-3 summarizes the sampling procedure.

1.3 The data set

Appendix 6.1 is a listing of the raw data for the recent month of 9-82. Prevailing currents had caused the mouth of the lagoon to migrate eastward making sampling of stations 1 through 5 impossible. For comparison, the data for the pre-rain month of 7-77 is also listed.

Each line of data in the file represents a measurement of the test length of a single individual. It consists of four integers whose meaning is, in order,

* the station number
* the sample number
* the test length in millimeters
* a code indicating if the individual is dead or alive

The data are sorted by station number, and within a given station by sample number.

The sample numbers are arbitrarily assigned when the data are entered with the text editor. They do not indicate the order in which the data was taken as the

```
program Sample Lagoon
   SN := 1    {SN is the station number.}
   repeat
      N1 := random digit between 1 and 9
      N2 := another random number between 1 and 9

      Beginning at station number SN, one worker takes N1
      paces along the barrier beach in one direction, and
      another worker takes N2 paces along the barrier beach
      in the other direction.  Both workers start at the
      mean tide level.

      repeat
         Take an individual sample.
         Take 5 paces toward the Salicornia marsh.
      until (The number of individual samples taken >= 15)
      and   (The lagoon has been crossed)

      SN := SN + 1
   until (SN > 9)
end Sample Lagoon
```

Figure 1.2-3. Algorithmic description of sampling procedure.

workers crossed the lagoon toward the Salicornia marsh. A skip in the sample number indicates that no individuals were found in the samples that are not listed. For example, in the data of 9-82 the fact that the first sample number in station 6 is 13 implies that samples 1 through 12 contained no individuals. Similarly, samples 1 through 9 in station 7 were empty.

The data for the ten month pre-rain period are shown in histogram form in Figure 1.3-1. In this time sequence of histograms the first month is the upper left plot and the second month is to the right of it (not below it). This convention of displaying a time sequence of distributions is used throughout this dissertation. When the entire time sequence is shown on one page, each distribution must be rather small. To reduce clutter in the diagrams, the axes will often not be labeled.

The vertical axis is density (individuals per square meter). The horizontal axis is divided into bins of 5 mm, so that the first bin represents individuals from zero to 5 mm in length, the second bin represents individuals 5 to 10 mm, etc. The data in Figure 1.3-1 are integrated over all the stations in the lagoon.

Several interesting features are evident. First, the distribution is often bimodal. Hence, there is often a "generation gap" between the small presumably young group and the large presumably old group.

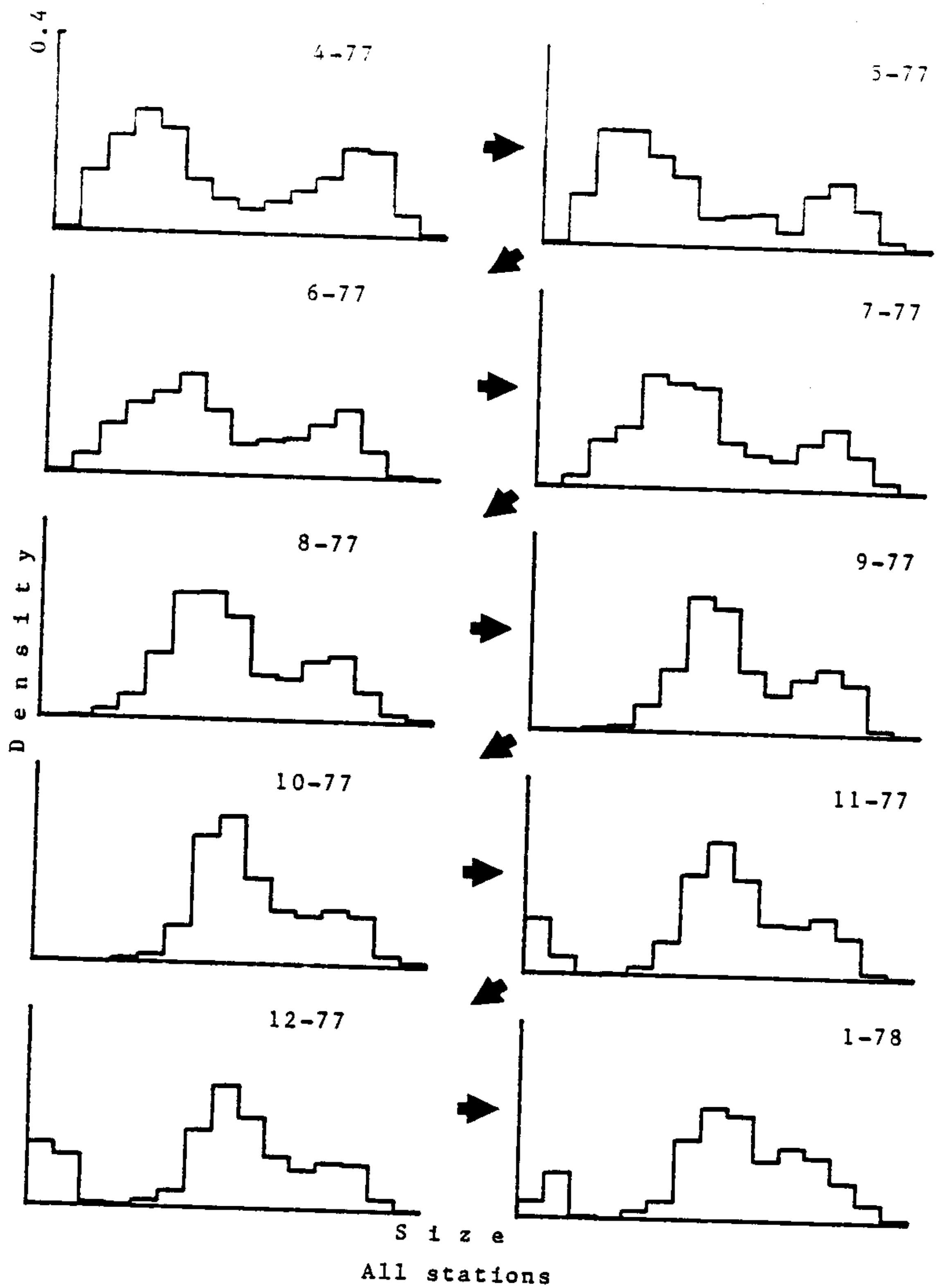


Figure 1.3-1. The 1977 size distributions.

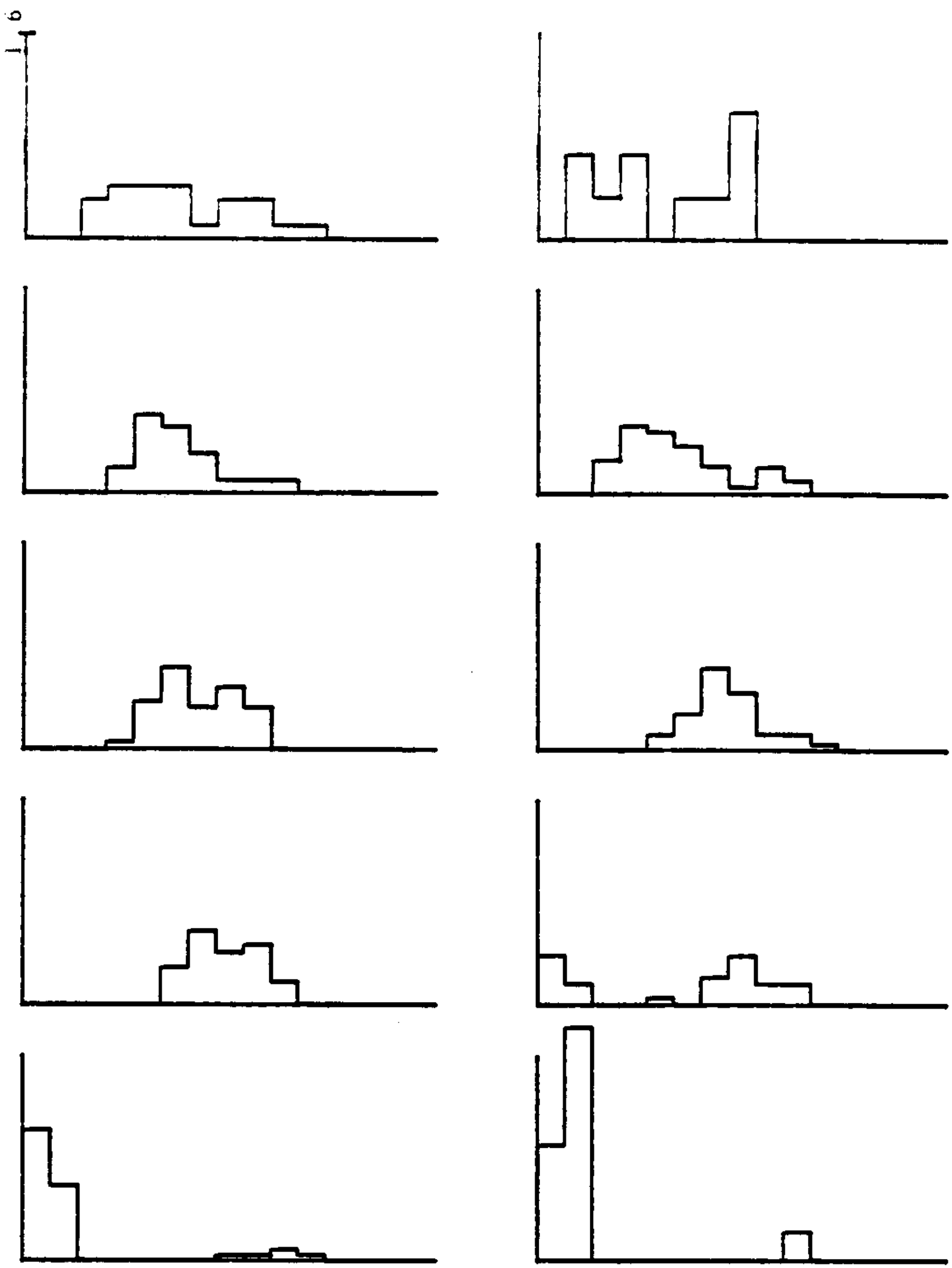
Second, with time the peaks of the distribution shift to the right. Hence, physical growth is evident in the distribution.

Third, in some months the peak on the right appears to decrease. Hence, mortality of individuals is evident in the distribution.

Fourth, in the last three months recruitment of small individuals has occured, presumably through spawning. Hence reproduction is evident in the distribution. Recruitment very likely does not involve progeny of adults in the lagoon. The long larval period guarantees that most new recruits will have been born elsewhere.

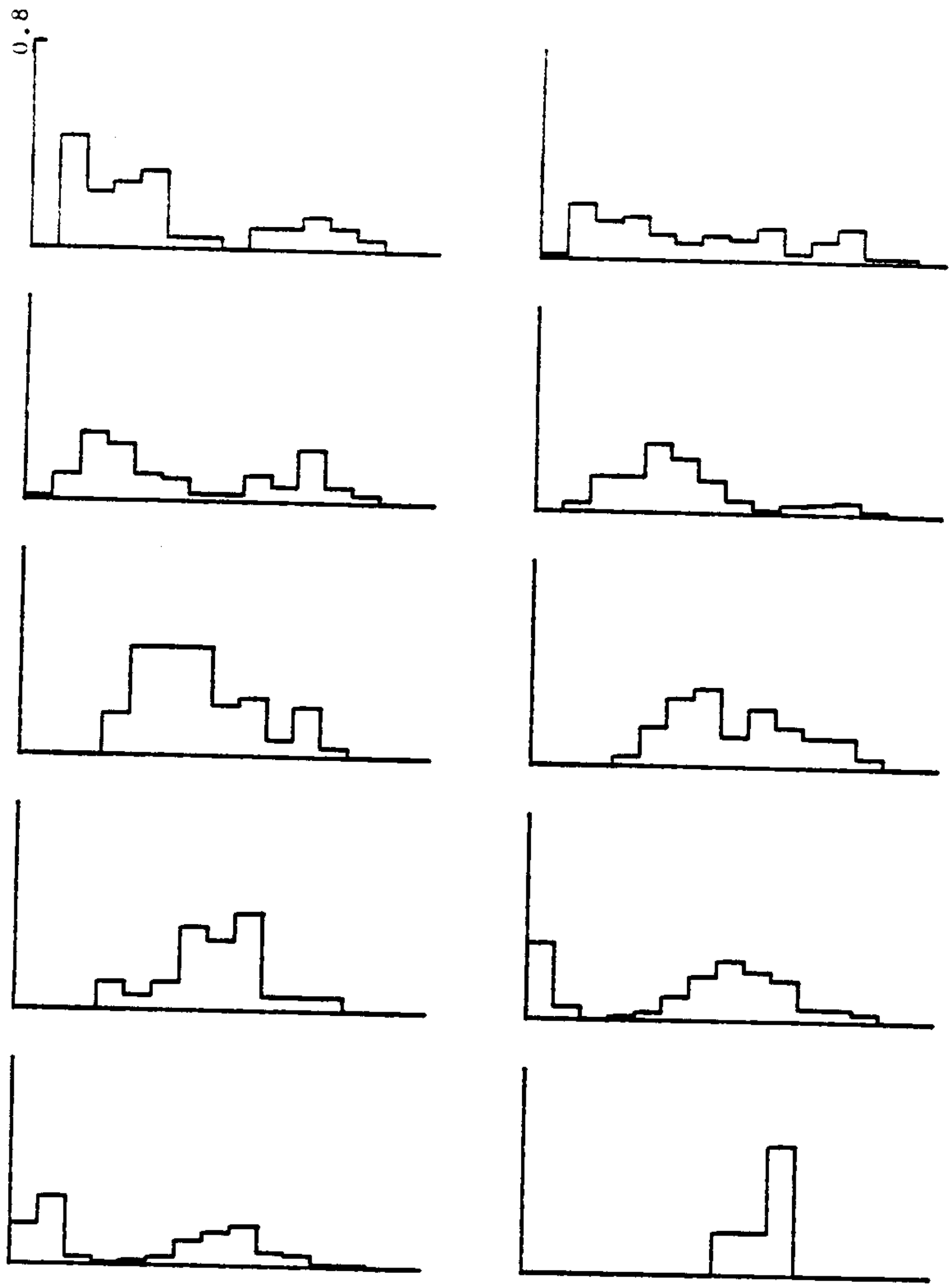
Figure 1.3-2 is a breakdown of the same data by station number. Because of the smaller sample size the pattern is more erratic than the integrated pattern for all the stations in Figure 1.3-1. Also notice the difference in scale on each figure.

The data for the recent six-month period is shown in Figure 1.3-3. The time interval between distributions is one month, except for the time interval between the penultimate and the last distribution which is two months. The same trends can be seen in the recent data, but it appears a bit more sporadic compared to the 1977 data.



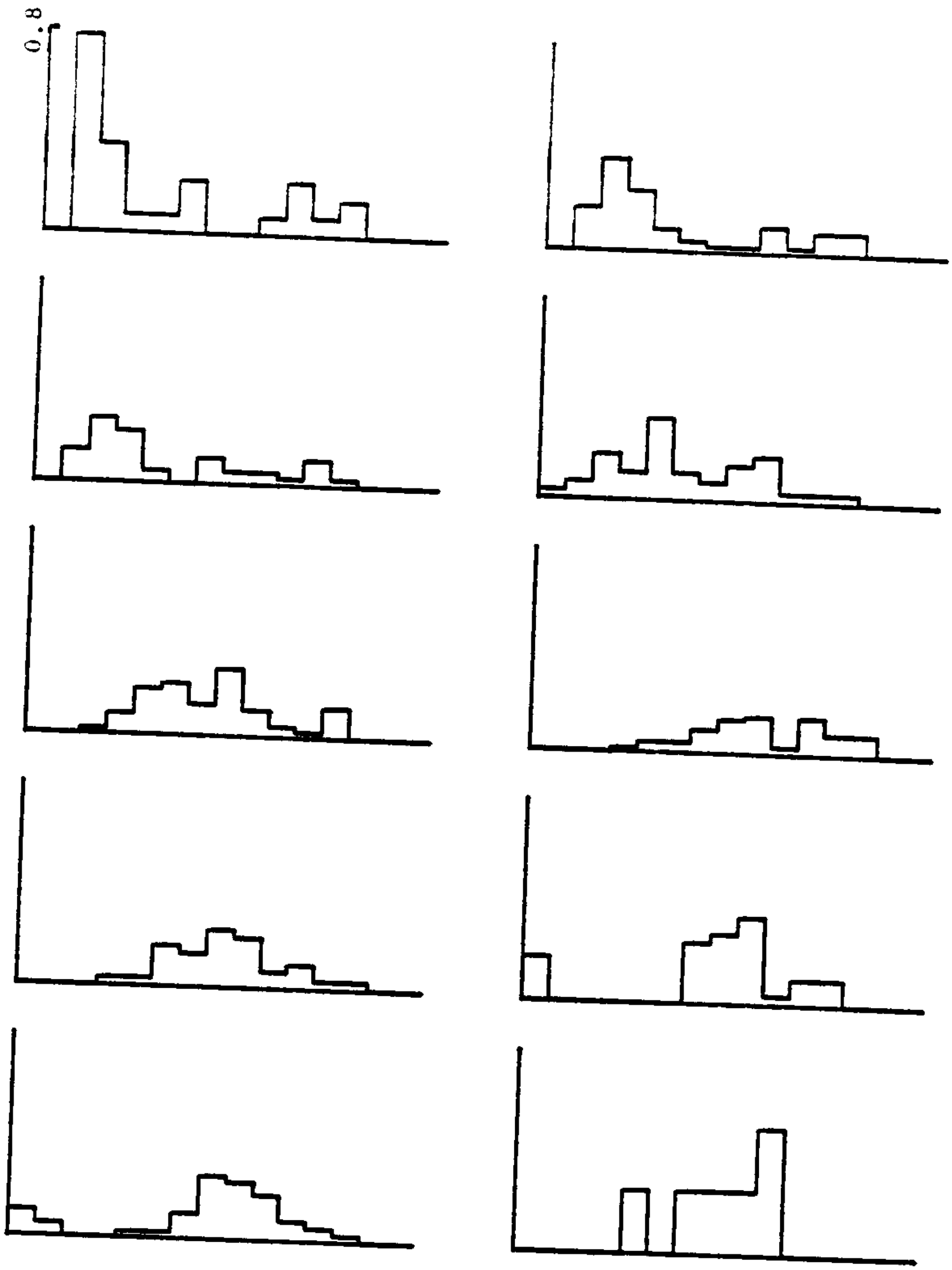
(a) Station 2.

Figure 1.3-2. The 1977 size distributions by station.



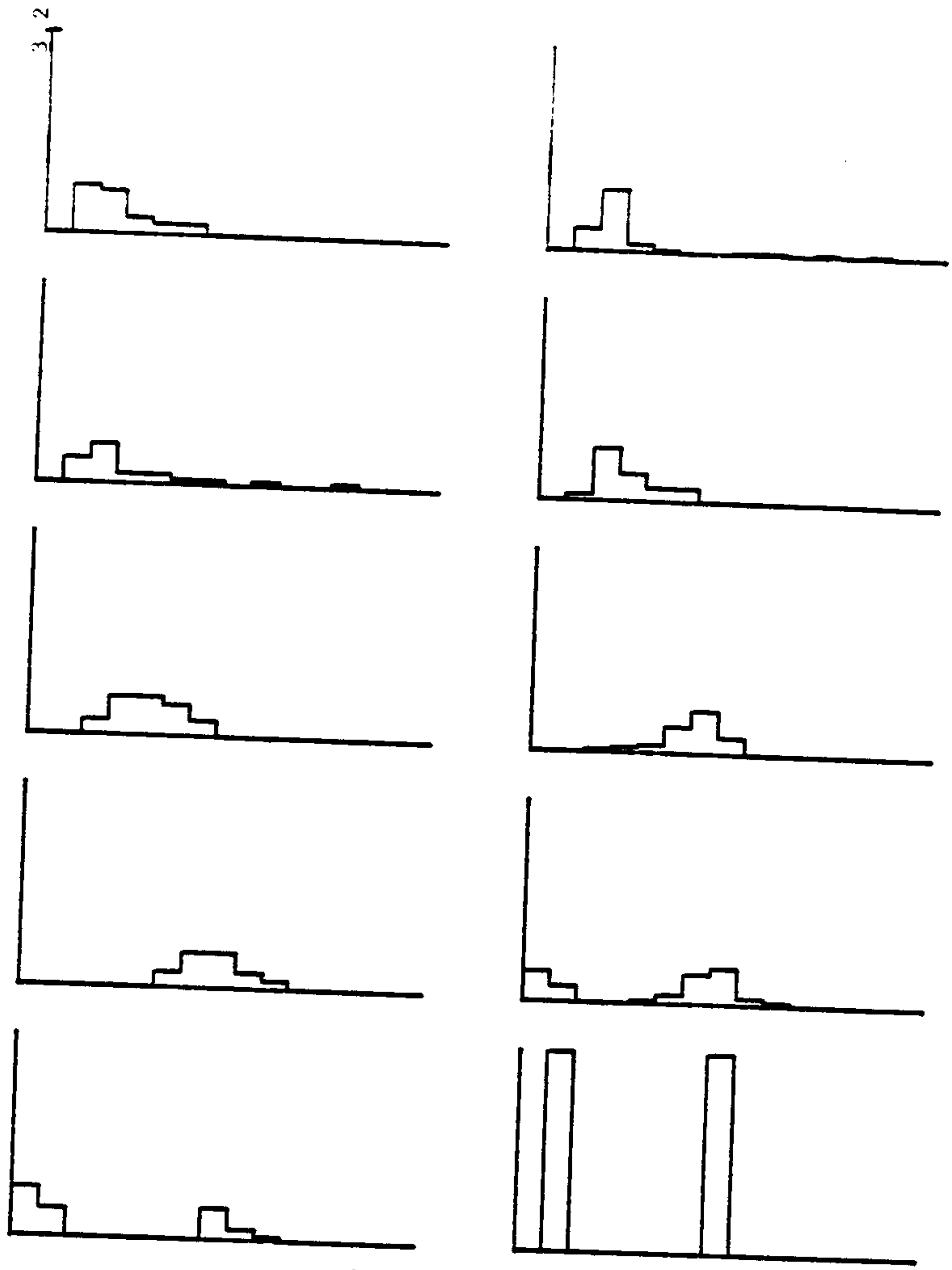
(b) Station 3.

Figure 1.3-2. (continued)



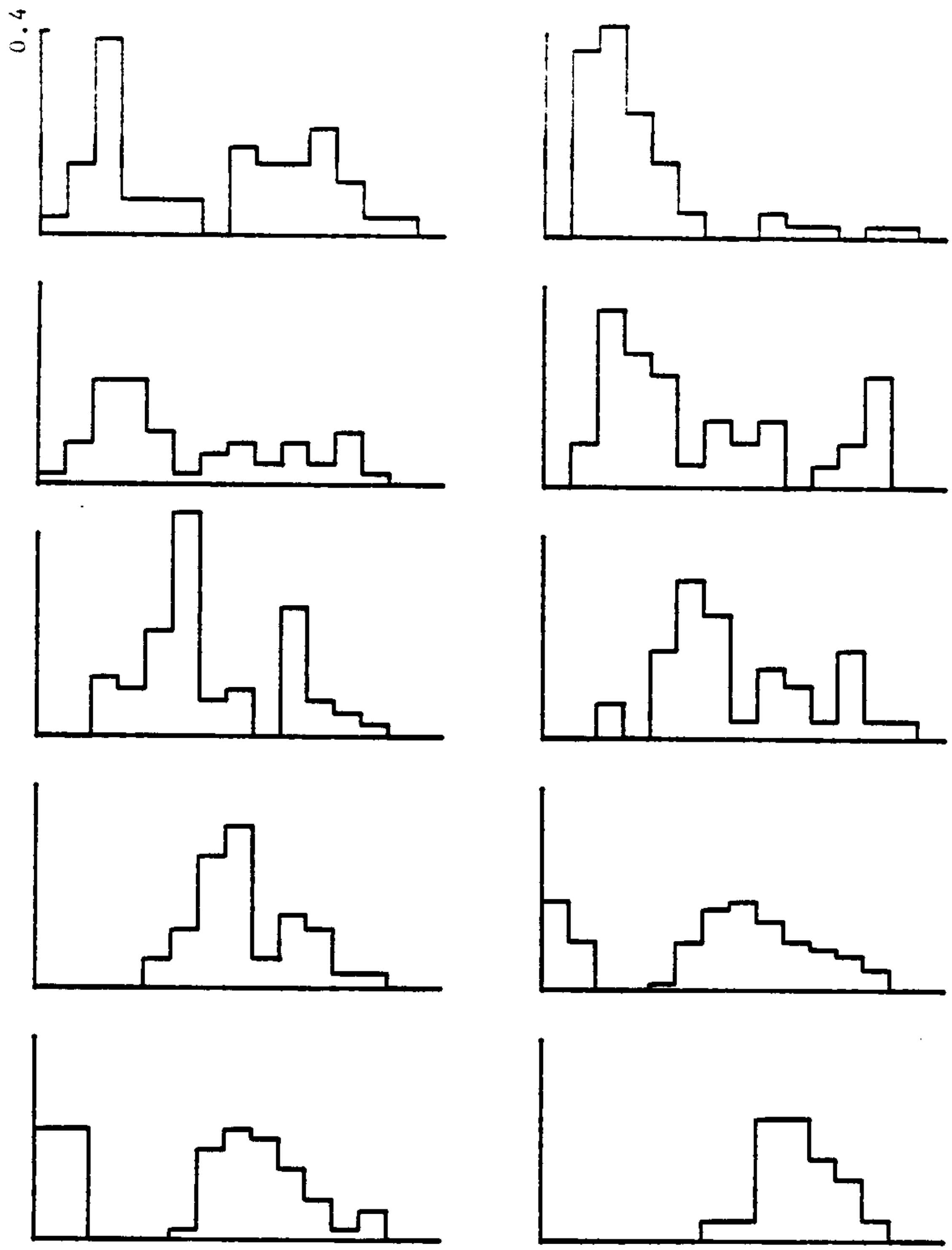
(c) Station 4.

Figure 1.3-2. (continued)



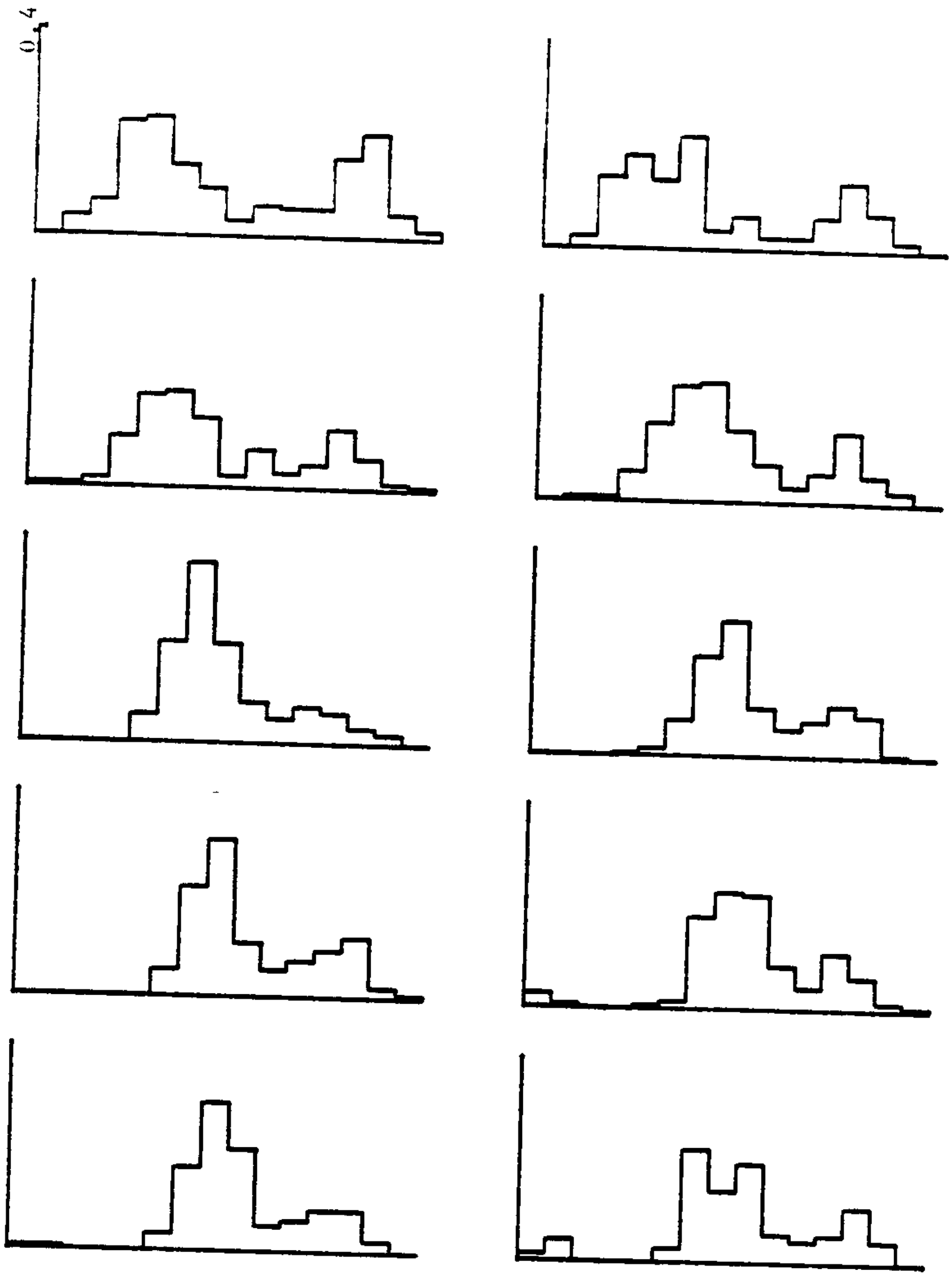
(d) Station 5.

Figure 1.3-2. (continued)



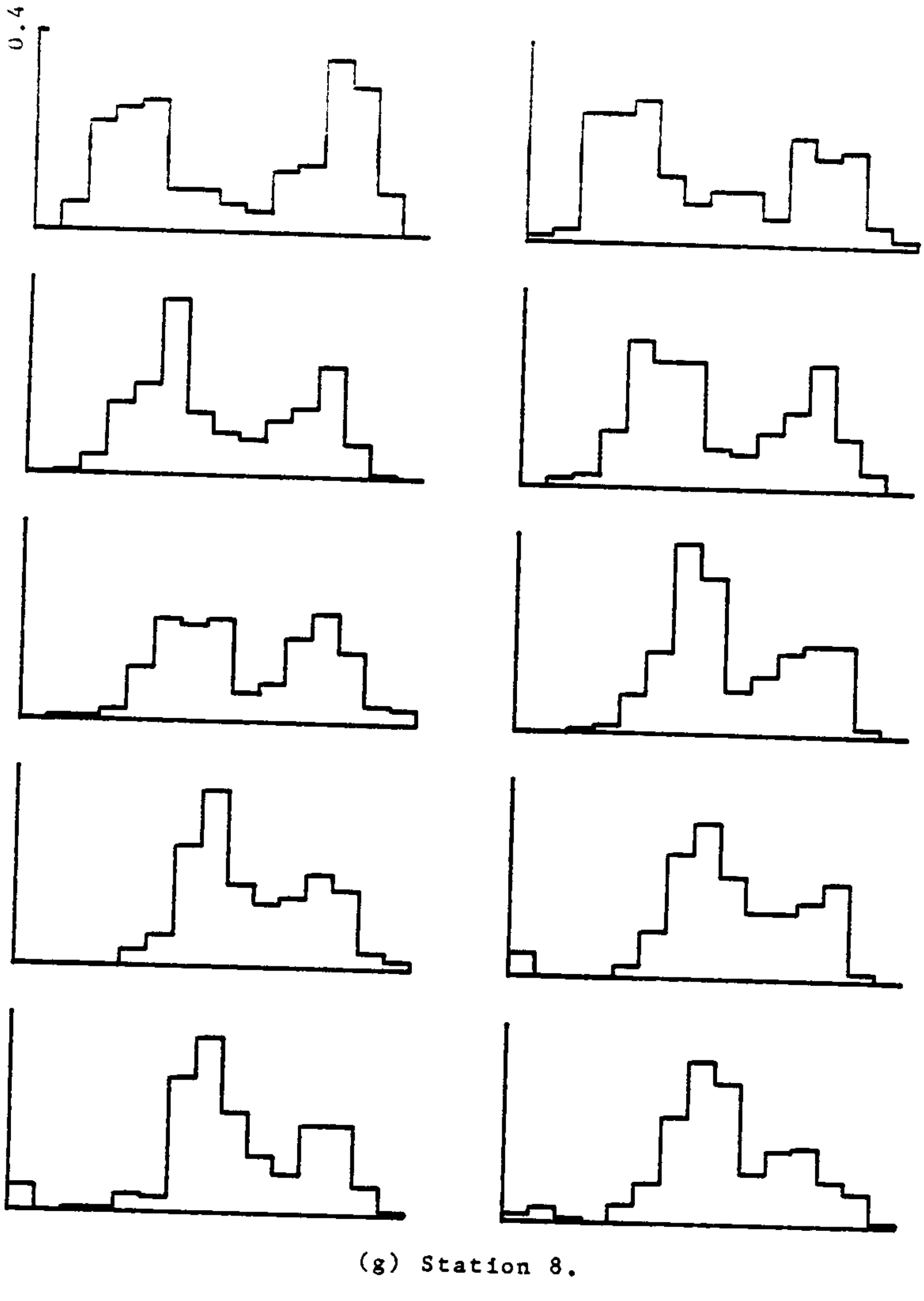
(e) Station 6.

Figure 1.3-2. (continued)



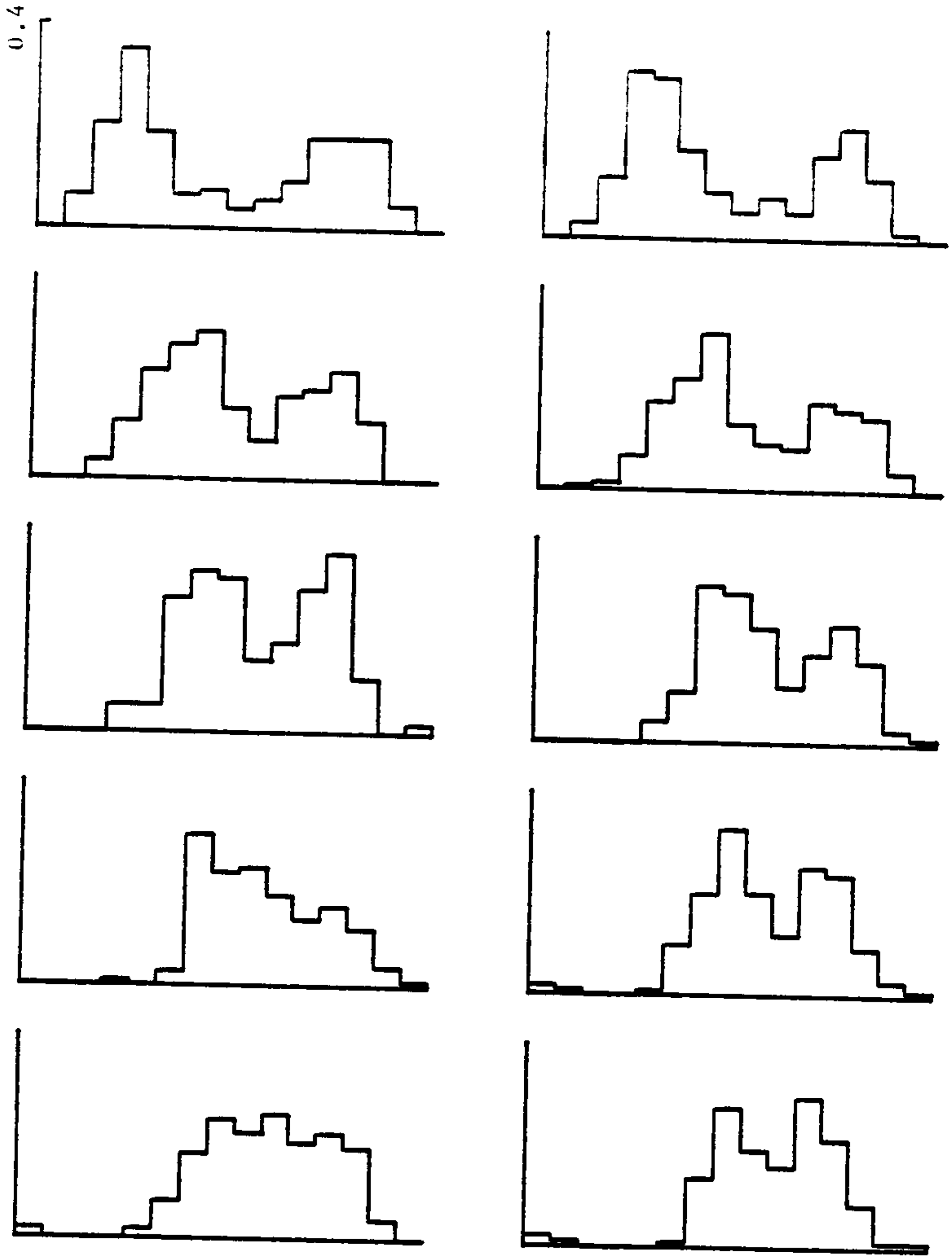
(f) Station 7.

Figure 1.3-2. (continued)



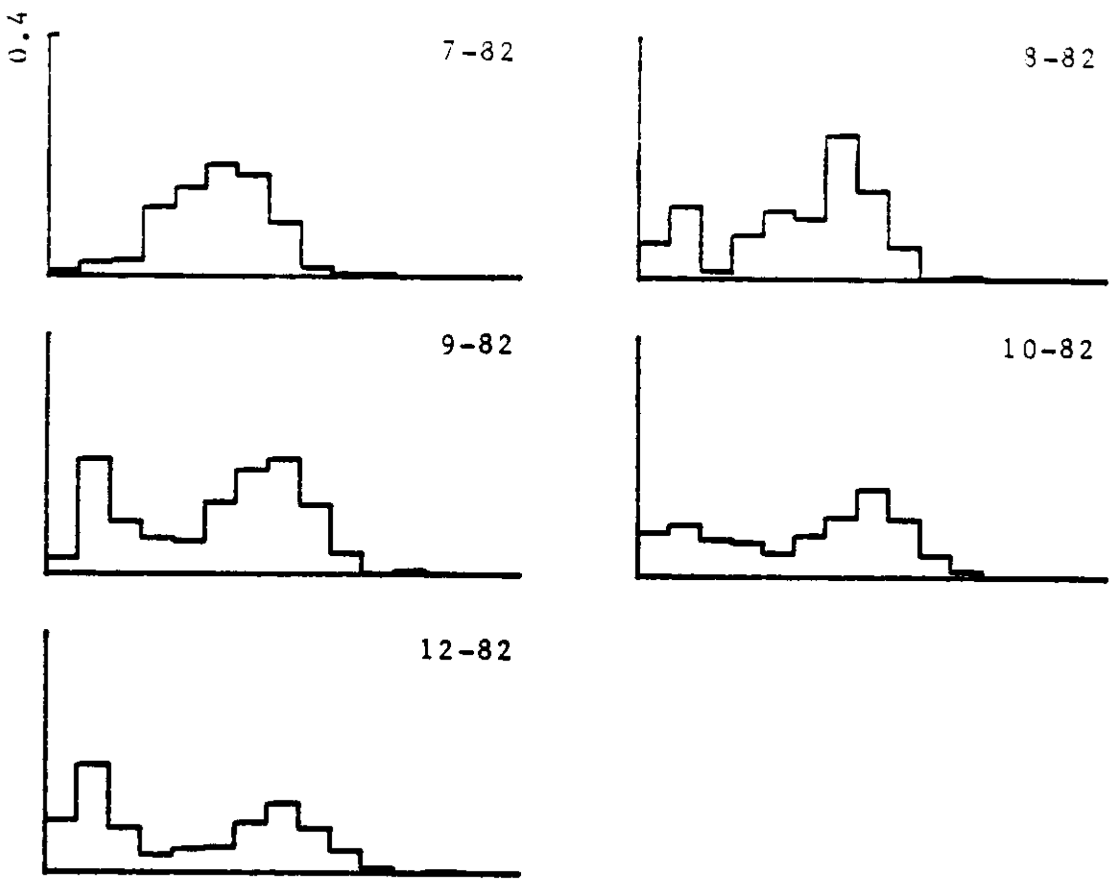
(g) Station 8.

Figure 1.3-2. (continued)



(h) Station 9.

Figure 1.3-2. (continued)



All stations

Figure 1.3-3. The 1982 size distributions.

1.4 Statement of the problem

This section is divided into two parts. The first part is a review of the previous literature on Dendraster excentricus. The second part is a statement of the problem to be solved. Section 2.4 restates the problem in more mathematical detail after the theory is developed in Sections 2.1 to 2.3.

Previous literature on Dendraster excentricus

MacGinitie and MacGinitie [MacG68] describe some of their observations of Dendraster excentricus on the North American west coast. They report densities as high as 67 individuals per square yard at Morro Bay and 468 individuals per square yard at Corona Del Mar, and state that these are maximum populations. However, they give no spatial or temporal variations in population density.

Merrill and Hobson [Merr70] observed the behavior, distribution, and biotic relationships along the Pacific coast of California and Baja California, Mexico. Their study covered the period 1963 to 1968, during which they logged over 250 hours in underwater observation.

They subjectively classified four separate habitats that the sand dollar occupies:

* bay

* tidal channel

* protected outer coast

* exposed outer coast

They found that the bahavior of the sand dollars varied between the populations in the different habitats. Where there are small currents, as in a sheltered bay, sand dollars live in shallow water, are relatively mobile, and feed on deposited material on the bottom where they lie flat. Where there are moderate currents, as in tidal channels and protected areas of the outer coast, they are more stationary, burrow partially in the sand in an inclined position, and feed primarily on suspended material. Where there are large currents in the exposed outer coast they are usually buried.

Merrill and Hobson give several density figures. In three bay populations they report the proportion of the total population as a function of depth and of disposition (i.e. whether still, moving, buried, etc) [Merr70, Table 2]. They also report the mean length as a function of depth in an outer coast population [Merr70, Figure 8], and give some maximum and average densities in samples from all four habitat types [Merr70, Table 3]. In [Merr70, Figure 9] are two size distributions as a function of depth for a population at Zuma Beach, a protected outer coast habitat. One distribution was taken during calm seas and the other was taken after a storm. Merrill and Hobson did not include a temporal study concentrating on the population growth confined to only one site.

Merrill and Hobson state that juveniles are far more widespread than adults, but age distributions are not given explicitely.

Birkeland and Chia [Birk71] studied two populations of Dendraster excentricus at Alki Point, Seattle, in different habitats. The northern Alki population lived in a smooth beach of deep sand. The southern lived in sand between cobbles over a hard clay bottom which lies 2 to 10 cm below. This difference in habitat affected the size distribution, growth rate, and abundance of each population. The northern Alki population has a lower population density, less recruitment, a lower rate of growth of young, higher growth of adults, and a larger mean adult size than the southern population.

Birkeland and Chia sampled the populations with a 1 m^2 frame place on the beach at low tide. The sand within the frame was sifted through a screen to retrieve the individuals. They considered both size structure, as measured by length of test, and age structure, as measured by growth rings. The experimental procedure for counting growth rings is somewhat involved. The sample must be dried in an oven for several days, sanded, and treated to make the growth rings visible. The growth rings are generally believed to be annual.

Size distributions given as a histogram with number of individuals in size class as a function of test length, are shown at four separate times over the span of a year [Birk71, Figure 4]. These data are for small individuals. Also shown is the change in the distribution over a 6-month period for adults as well as juveniles [Birk71, Figure 8].

Birkeland and Chia also determined the growth rate as total test length as a function of age in years [Birk71, Figure 6]. This is an interesting relationship. It is roughly linear from birth to about 4 or 5 years, at which time it abruptly flattens. The mortality rate is very high at 8 or 9 years for both age groups. Birkeland and Chia concluded that death is natural, i.e. senescence.

Timko wrote her PhD thesis here at UCLA on high density aggregation in Dendraster excentricus [Timk75]. Her data were taken mostly from Zuma Beach, a protected outer coast population, although other habitats were included for comparison.

She determined the age distribution by ring count and shows a two year trend at Zuma Beach (three measurements), a one year trend at Morro Bay (two measurements), and a one year trend at Newport Harbor (two measurements) [Timk75, Figures 1-6 to 1-8]. The measurements are annual and are given as histograms with percent of samples as a function of age class. They show essentially that the population is not stable. The distributions have a tendency to maintain their shape, shifting one year to the right with each annual measurement. Timko attributes this behavior to cannabalism of the adults on their larvae, a phenomenon she was able to demonstrate in the lab. Presumably, when the older generations begin to die out more recruitment is possible which produces a baby boom. This new generation then cannibalizes its larvae during the next cycle.

Timko determined growth rates displayed as test length as a function of age [Timk75, Figure 2-6]. The curves were very similar in shape to those of Birkeland and Chia although the parameters (slopes and intercepts) differed significantly. She also investigated the relationships between test diameter, test height, and dry weight.

Included in Timko's thesis are investigations of behavioral responses related to aggregation and inclined posture, reproductive biology, and consideration of hydrodynamic flow as it relates to feeding and diet. Also included is a density dependent age structure model which will be discussed later.

Dendraster excentricus spawn annually. They spew their eggs and sperm into the sea water where fertilization takes place. When one very ripe individual releases his reproductive material it acts as a trigger for others to release theirs.

Timko measured the spawning index (the ability to release eggs) over a one year period at approximately monthly intervals. It peaked sharply in mid-July.

The problem

The objective of this research is to model the growth and distribution of the specific population of Dendraster excentricus described in the previous sections. The problem is interesting for two reasons.

First, the quality of the raw data is high for this type of study with this particular species. The monthly determination of the size distribution of the population contains more detailed information than is available in previously published literature.

Second, the basic modeling approach is different from the approach normally used in biological models. In fact, the approach is motivated by the existence of the data. This dissertation investigates the problem from a system identification point of view. Namely, the question is: can a mathematical model be constructed with a minimum number of parameters, which can be optimally determined to produce a good fit to the field data?

The common approach in population modeling is to hypothesize a mathematical relation and then investigate its properties. Any use of field data is usually handled in one of two ways.

* Some of the parameters in the initial mathematical relation are estimated from field data.
* When the mathematical properties have been determined any trends in the result are compared with comparable trends in the field data.

In contrast to the common approach, the purpose of this model is to first integrate the effects of recruitment, growth, and mortality into a single system, and then to use the resulting mathematical model to estimate the values of the controlling parameters from the time sequence of histograms. The study includes a system identification of the parameters using the computational technique described in Chapter 3.

2. The general mathematical model

This chapter presents the general mathematical features of the models used to describe the system. It combines two distinct components into one system, namely Leslie matrix theory and bilinear growth models. Both components are common in the literature [Keyf68, Birk71, Timk75]. Their combination into a single system, however, is apparently unique to this study.

2.1 Leslie matrix theory

The Leslie matrix model [Lesl45] predicts the age structure of a population of animals after a unit period of time given

* a matrix whose elements represent age-specific fecundity and mortality, and
* the age structure at the present time.

In matrix notation the model can be written as

$$A\ \bar{a}(t) = \bar{a}(t+1)$$

In this equation

$$\tilde{a}(t) = \begin{bmatrix} a(0,t) \\ a(1,t) \\ \cdot \\ \cdot \\ \cdot \\ a(n,t) \end{bmatrix}$$

is a column vector with n+1 elements which represents the population's age structure at time t., The element a(i,t) is the number of females alive in the age group i to i+1 at time t. The column vector

$$\tilde{a}(t+1) = \begin{bmatrix} a(0,t+1) \\ a(1,t+1) \\ \cdot \\ \cdot \\ \cdot \\ a(n,t+1) \end{bmatrix}$$

represents the age structure at time t+1. The (n+1) x (n+1) matrix

$$A = \begin{bmatrix} f[0;\tilde{a}(t)] & f[1;\tilde{a}(t)] & \cdots & f[n-1;\tilde{a}(t)] & f[n;\tilde{a}(t)] \\ p[0;\tilde{a}(t)] & 0 & \cdots & 0 & 0 \\ 0 & p[1;\tilde{a}(t)] & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & p[n-1;\tilde{a}(t)] & 0 \end{bmatrix}$$

is a quantity which describes the transition of the population vector during one increment of time. The elements $f[i;\tilde{a}(t)]$, i = 0, 1, ..., n, $f[i;\tilde{a}(t)] >= 0$, describe the fecundity of the species. Specifically, $f[i;\tilde{a}(t)]$ represents the average number of daughters who will be alive at time t+1, born in the interval t to t+1 to each female who was in the age group i to i+1 at time t. The elements $p[i;\tilde{a}(t)]$, i = 0, 1, ..., n-1, where $0 < p[i;\tilde{a}(t)] <= 1$, represent the probability that a female aged between i and i+1 at time t will be alive at time t+1 in the age group i+1 to i+2.

For human populations a time interval of 5 years is typical [Hopp76] with 16 age classes (n = 15). In that case

$a(0,t)$ = number of people with ages a, $0 < a < 5$

$a(1,t)$ = number of people with ages a, $5 < a < 10$

...

$a(15,t)$ = number of people with ages a, $75 < a < 80$

Hence the total population having ages greater than 80 is ignored.

As the notation indicates, the elements of the Leslie matrix in general depend on the current population distribution $\tilde{a}(t)$. That is, the fecundities and the survival probabilities are density dependent. Phenomena like overcrowding and competition for food resources affect the functional dependency.

The theory of population growth under the assumption of constant fecundities and survival probabilities is well developed [Lesl45, 48], [Keyf68, 71]. Given the initial age distribution $\tilde{a}(0)$, the evolution of the population can be described by

$$\bar{a}(m) = A\ \bar{a}(m-1) = \ldots = A^m\ \bar{a}(0)$$

The problem is therefore to determine the characteristics of A^m as m increases.

Renewal theory can be invoked in several ways. One way is to reduce the vector equation to a scalar equation and obtain a renewal equation which recursively describes the dynamics of the birth rate of the population as a whole. It can be shown [Fell68] that for certain values of f[i] and p[i] the population is not viable and the birth rate approaches zero. If the population is viable the birth rate itself increases roughly at a constant rate.

It is also possible to apply renewal theory directly to the full Leslie model [Lesl45, 48], [Keyf68, 71]. The spectral decomposition of A can be used to analyze the powers of A. A is called "honest" if it has a unique strictly maximum positive real eigenvalue, e. In such a case, for large m the population vector grows at a geometric rate, e, but the distribution between the age classes remains fixed.

The theory of population growth with density dependent fecundities and survival probabilities has been developed primarily for nonage-specific models. In these models the Leslie matrix is 1 x 1 and the scalar element a(t) is denoted N(t) for the population as a whole. Time is usually considered a continuous independent variable, rather than a discrete one as in the Leslie matrix. The dynamics are then described by the differential equation

$$\frac{dN(t)}{dt} = f(N(t)) \ .$$

Hence, the population growth rate, dN/dt, is some function, f, of the population at time t. In the Malthusian model f is a constant times N(t), and the per capita rate of growth is density independent.

The well known logistic equation [May73] models the system by selecting f to be rN(1 - N/K), so that

$$\frac{dN(t)}{dt} = rN(t)[1 - N(t)/K],$$

where the parameter r is a measure of the intrinsic per capita growth rate, and K is called the total carrying capacity of the environment. The solution is the familiar sigmoid population growth curve [Else81], so called because it is shaped like the letter "s". When N << K the slope of N(t) is rN(t). As N(t) approaches K the slope of N(t) approaches zero. The function N(t) flattens out and N = K is the stable equilibrium population.

An interesting variation on this theme is the introduction of a time lag T built into the regulatory mechanism [May73].

$$\frac{dN(t)}{dt} = rN(t)[1 - N(t - T)/K],$$

This model would have application, for example, in a system in which herbivores graze upon vegetation, which takes time T to recover. This equation has been investigated extensively in the mathematics literature. If $rT < 1/(2\pi)$ the assymptotic solution is stable at $N = K$. But if $rT > 1/(2\pi)$ the solution is unstable and oscillates. Nicholson performed some classic laboratory experiments [Nich54] with the Australian sheep-blowfly, _Lucilia cuprina_, in which the time delay T was equal to the time for a larva to mature into an adult. His experimental data showed the oscillation which is in good agreement with the model [May73].

So the theory is well developed for the age-specific, density independent, Leslie model. It is also well developed for the nonage-specific density dependent logistic based model. The theory is not as well developed for age-specific density dependent models. Indeed, the mathematical complexity of such models precludes many general statements. Instead, most studies of such systems are done numerically. For example, Leslie has investigated the oscillations which result from considering both age-specific density dependence and time lag features in the matrix model [Lesl59].

Timko uses the Leslie matrix as the basis for several models. One model incorporates larviphagy (cannabalism of larvae by parents) by introducing the following assumptions

* Each adult is represented as the center of a foraging space.
* The size of the foraging space doubles with each year increase in age.
* Larvae settle uniformly and all those within the foraging space of an adult are eaten.

This has the result of making the fecundities density dependent. The model was investigated by iterating from an initial age vector of 10 newly settled animals and was found to have an oscillating behavior. Four age classes were used ($n = 4$), presumably to check the model in a simple case to determine gross behavior.

Another model contained 13 age classes, with the elements of A estimated from the age class data previously described. This model incorporated larviphagy by experimentally trying to determine the foraging space under laboratory conditions. An additional complication arises from the tendency of individuals to be spatially clumped together on the ocean floor. A fertilization coefficient was also taken from field data and used in the calculation of the fecundity elements of the A matrix.

This second model was tested two ways. With an initial age distribution of 100 settled females and 100 settled males the model predicted severe oscillations.

The survivorship curves were changed accordingly (the algorithm for determination of the new survivorship curve was not given by Timko) in an effort to stabilize the system. Another test of the model was to use the measured age distribution for the Zuma Beach population and compare the predicted data with the following two years' measured data. The one year prediction was roughly comparable to the data, but the two year predicion was substantially different. No overall figure of merit for closeness of fit was given.

2.2 Seasonal recruitment in the Leslie model

This section presents the practical considerations involved in constructiong the Leslie based model for computer simulation. It shows how to exploit the additional structure which seasonal recruitment imposes on the model. The second part shows how the original Leslie parameters are ralated to the parameters of the reduced dimensionality formulation, and how this formulation relates to the models of Chapter 4.

Structure of the model

This part illustrates the problem by a specific example. Suppose data is taken on a quarterly basis which implies a time increment of three months. If the maximum age of the species is four years then the dimension of Leslie matrix is 16 x 16. The matrix is shown in Figure 2.2-1 where f[i], i = 0, 1, ..., 15 are the quarterly age specific fecundities (density independent) and p[i], i = 0, 1, ..., 14 are the quarterly age specific survival probabilities (also density independent). All entries not shown are zero. The dotted lines indicate the yearly boundaries.

Seasonal recruitment to the population implies that there is only one nonzero fecundity each year. For example, if recruitment to the population occurs during the fourth quarter, then only f[3], f[7], f[11], and f[15] will be nonzero in our example. The corresponding Leslie matrix is shown in Figure 2.2-2.

Now consider the time sequence of a general population with the matrix of Figure 2.2-2. Figure 2.2-3 shows the time sequence for three quarters under the assumption of an initial distribution column vector with no zero entries.

The important point to notice in Figure 2.2-3 is that recruitment to the population occurs every quarter of the calendar. Hence it is not seasonal. It is what you might call "age-annual". The fact that only f[3], f[7], f[11],

f[0]	f[1]	f[2]	f[3]	f[4]	f[5]	f[6]	f[7]	f[8]	f[9]	f[10]	f[11]	f[12]	f[13]	f[14]	f[15]
p[0]															
	p[1]														
		p[2]													
			p[3]												
				p[4]											
					p[5]										
						p[6]									
							p[7]								
								p[8]							
									p[9]						
										p[10]					
											p[11]				
												p[12]			
													p[13]		
														p[14]	0.0

Figure 2.2-1. The general 4-year Leslie matrix with quarterly time increments.

$$
\left[\begin{array}{cccc:cccc:cccc:cccc}
0.0 & 0.0 & 0.0 & f[3] & 0.0 & 0.0 & 0.0 & f[7] & 0.0 & 0.0 & 0.0 & f[11] & 0.0 & 0.0 & 0.0 & f[15] \\
p[0] & & & & & & & & & & & & & & & \\
 & p[1] & & & & & & & & & & & & & & \\
 & & p[2] & & & & & & & & & & & & & \\ \hdashline
 & & & p[3] & & & & & & & & & & & & \\
 & & & & p[4] & & & & & & & & & & & \\
 & & & & & p[5] & & & & & & & & & & \\
 & & & & & & p[6] & & & & & & & & & \\ \hdashline
 & & & & & & & p[7] & & & & & & & & \\
 & & & & & & & & p[8] & & & & & & & \\
 & & & & & & & & & p[9] & & & & & & \\
 & & & & & & & & & & p[10] & & & & & \\ \hdashline
 & & & & & & & & & & & p[11] & & & & \\
 & & & & & & & & & & & & p[12] & & & \\
 & & & & & & & & & & & & & p[13] & & \\
 & & & & & & & & & & & & & & p[14] & 0.0
\end{array}\right]
$$

Figure 2.2-2. The 4-year Leslie matrix with seasonal recruitment during the fourth quarter.

$$
\begin{bmatrix} a[0] \\ a[1] \\ a[2] \\ a[3] \\ a[4] \\ a[5] \\ a[6] \\ a[7] \\ a[8] \\ a[9] \\ a[10] \\ a[11] \\ a[12] \\ a[13] \\ a[14] \\ a[15] \end{bmatrix}, \quad
\begin{bmatrix} f[3]\,a[3] + f[7]\,a[7] + f[11]\,a[11] + f[15]\,a[15] \\ p[0]\,a[0] \\ p[1]\,a[1] \\ p[2]\,a[2] \\ p[3]\,a[3] \\ p[4]\,a[4] \\ p[5]\,a[5] \\ p[6]\,a[6] \\ p[7]\,a[7] \\ p[8]\,a[8] \\ p[9]\,a[9] \\ p[10]\,a[10] \\ p[11]\,a[11] \\ p[12]\,a[12] \\ p[13]\,a[13] \\ p[14]\,a[14] \end{bmatrix},
$$

$t = 0 \qquad\qquad t = 1$

Figure 2.2-3. Time sequence for the matrix of Figure 2.2-2 with an arbitrary initial population.

```
f[3] p[2] a[2] + f[7] p[6] a[6] + f[11] p[10] a[10] + f[15] p[14] a[14]
p[0] (f[3] a[3] + f[7] a[7] + f[11] a[11] + f[15] a[15])
p[1] p[0] a[0]
p[2] p[1] a[1]
.......................................................
p[3] p[2] a[2]
p[4] p[3] a[3]
p[5] p[4] a[4]
p[6] p[5] a[5]
.......................................................
p[7] p[6] a[6]
p[8] p[7] a[7]
p[9] p[8] a[8]
p[10] p[9] a[9]
.......................................................
p[11] p[10] a[10]
p[12] p[11] a[11]
p[13] p[12] a[12]
p[14] p[13] a[13]
```

t = 2

Figure 2.2-3. (continued)

and f[15] are nonzero means that reproduction can only occur in the fourth quarter of an individual's age, i.e. the quarter just before her birthdate. It does not necessarily imply that recruitment will only occur in the fourth calendar quarter.

If seasonal recruitment occurs only during the fourth calendar quarter (as opposed to the age quarter) an additional assumption in the mathematical model is necessary. Namely, three fourths of the entries in the initial age distribution column vector must be zero. Figure 2.2-4 shows the appropriate initial age distribution vector and its time sequence with the Leslie matrix of Figure 2.2-2. Recruitment to the population now occurs not at $t = 1, 2, 3$ but only at $t = 4$.

Since the structure of the distribution vector is the same at $t = 4$ as it was at $t = 0$, the temporal behavior is repetitive. Recruitment to the population will only take place at time $t = k$ where $k \bmod 4 = 0$, that is, every calendar quarter.

Reducing the dimensionality

In this research, data was taken monthly. Furthermore, the age of the species can be as high as 8 years. The full Leslie matrix with the structure of

t = 0	t = 1	t = 2	t = 3
a[0]	0.0	0.0	0.0
0.0	p[0] a[0]	0.0	0.0
0.0	0.0	p[1] p[0] a[0]	0.0
0.0	0.0	0.0	p[2] p[1] p[0] a[0]
a[4]	0.0	0.0	0.0
0.0	p[4] a[4]	0.0	0.0
0.0	0.0	p[5] p[4] a[4]	0.0
0.0	0.0	0.0	p[6] p[5] p[4] a[0]
a[8]	0.0	0.0	0.0
0.0	p[8] a[8]	0.0	0.0
0.0	0.0	p[9] p[8] p[8]	0.0
0.0	0.0	0.0	p[10] p[9] p[8] a[0]
a[12]	0.0	0.0	0.0
0.0	p[12] a[12]	0.0	0.0
0.0	0.0	p[13] p[12] a[12]	0.0
0.0	0.0	0.0	p[14] p[13] p[12] a[12]

Figure 2.2-4. Time sequence for the matrix of Figure 2.2-2 with an initial population which implies seasonal recruitment during the fourth quarter.

```
┌ f[3] p[2] p[1] p[0] a[0] + f[7] p[6] p[5] p[4] a[4] + f[11] p[10] p[9] p[8] ┐
      a[8] + f[15] p[14] p[13] p[12] a[12]
  0.0
  0.0
  0.0
  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  p[3] p[2] p[1] p[0] a[0]
  0.0
  0.0
  0.0
  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  p[7] p[6] p[5] p[4] a[4]
  0.0
  0.0
  0.0
  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  p[11] p[10] p[9] p[8] a[8]
  0.0
  0.0
└ 0.0                                                                          ┘
```

t = 4

Figure 2.2-4. (continued)

Figure 2.2-2 is thus 96 x 96. There are 8 nonzero fecundities f[11], f[23], ..., f[95], and 95 nonzero survival probabilities p[0], p[1], ..., p[94]. In the Leslie matrix alone there are 8 + 95 = 103 parameters. This is clearly too large for a practical optimization effort with current computer technology.

The number of parameters can be reduced by the assumption that the survival probabilities within a given yearly age class are equal. This is a reasonable assumption. It says that, all things else being equal, an individual 4 years and 7 months old has the same probability of surviving to be 4 years and 8 months old, as an individual 4 years and 10 months old has of surviving to be 4 years and 11 months old. In other words, all individuals in the 4 year age class have equal survival probabilities to the next month.

Similarly, all 5 year olds have equal survival probabilities to the following month. But this is not necessarily the same survival probability as the one for 4 year olds.

The equivalent assumption in our quarterly example is that

$$p[0] = p[1] = p[2] = p[3]$$
$$p[4] = p[5] = p[6] = p[7]$$
$$p[8] = p[9] = p[10] = p[11]$$
$$p[12] = p[13] = p[14]$$

Considering the time increment to be annual instead of quarterly, we can now define the (primed) corresponding annual quantities to be such that the product

$$\begin{bmatrix} f'[0] & f'[1] & f'[2] & f'[3] \\ p'[0] & & & \\ & p'[1] & & \\ & & p'[2] & 0.0 \end{bmatrix} \begin{bmatrix} a[0] \\ a[4] \\ a[8] \\ a[12] \end{bmatrix}$$

represents the transition from t = 0 to t = 4 in Figure 2.2-4. Equating coefficients of a[i] in the components of the resulting distribution vector gives

$$p'[0] = p[3] \;\; p[2] \;\; p[1] \; p[0] = (p[0])^4$$
$$p'[1] = p[7] \;\; p[6] \;\; p[5] \; p[4] = (p[4])^4$$
$$p'[2] = p[11] \; p[10] \; p[9] \; p[8] = (p[8])^4$$

for the annual survival probabilities, and

$$f'[0] = f[3] \;\; p[2] \;\; p[1] \;\; p[0] \; = f[3] \;\; (p[0])^3$$
$$f'[1] = f[7] \;\; p[6] \;\; p[5] \;\; p[4] \; = f[7] \;\; (p[4])^3$$
$$f'[2] = f[11] \; p[10] \; p[9] \;\; p[8] \; = f[11] \; (p[8])^3$$
$$f'[3] = f[15] \; p[14] \; p[13] \; p[12] = f[15] \; (p[12])^3$$

for the annual fecundities.

So the annual survival probability is simply the product of the four corresponding quarterly probabilities. The annual fecundity is the product of the corresponding quarterly fecundity and the three "preceeding" survival probabilities. This result is to be expected. For example, in order for a two year old to produce offspring as reflected by f'[1] above, she must first survive through the first, second, and third quarters of her second year. These survival probabilities are precisely p[4], p[5], and p[6].

In this example the original 16 x 16 quarterly matrix with seasonal recruitment has been reduced to an equivalent 4 x 4 annual matrix. Note that the assumption of equal survival probabilities within yearly age classes is not required for the reduction. The only requirement is seasonal recruitment.

The models used in this research make the assumption of equal survival probabilities within a yearly age class. The survival probabilities and fecundities are stored in reduced dimensionality data structures whose dimensions correspond to an annual time increment. However, the data is monthly and so is the time increment in the simulations. Consequently, the values reported for survival probabilities and fecundities will usually be the monthly ones, not the annual ones. This is in spite of the fact that recruitment to the population is annual.

The way this is implemented in the program is to carry a single integer in the range 0..11 along with the reduced dimensionality data structure. The integer represents the calendar month within the year. In effect it specifies which set of rows in the full vector, corresponding to Figure 2.2-4, are nonzero. The only difference is that in Figure 2.2-4 one out of 4 rows are nonzero. In the actual vector one out of 12 rows are nonzero.

The three transitions t = 0 to 1, t = 1 to 2, and t = 2 to 3 in Figure 2.2-4 produce no recruitment to the population. But the transition t = 3 to t = 4 does. Hence, in the reduced dimensionality structure there are two separate ways to generate the population for the following month.

The program uses the calendar month integer to detect whether or not the transition is during a recruitment month. If it is a recruitment month it calculates the next age distribution from the current age distribution, the fecundity row of the Leslie matrix, and the off diagonal survival probability elements. If it is not a recruitment month it uses only the current age distribution and the survival probability elements.

A tacit assumption in the model described thus far is a closed system. That is, recruitment, when it occurs, is related to the current age vector through the fecundity elements in the Leslie matrix. This assumption is questionable.

The lagoon is open to the outer coast through its mouth on the west end. Planktonic larvae fron the protected outer coast population should be able to migrate into the system. The model should therefore perhaps be viewed as a local sample of the whole prpulation. The resulting fecundity values, even though they produce a good fit to the data, may not characterize those females in the environment of the lagoon.

In Chapter 4 we will see that the other parameters are in fact not sinsitive to the fecundity estimates. Furthermore, we will show a method whereby immigration of a more general nature can be included in the model.

2.3 The growth model

This section describes the assumptions made on the characteristics of the growth component which is incorporated into the model. Some preliminary models were constructed which included both the seasonal Leslie component and the growth component in order to observe the general bahavior of the system.

Bilinear growth, zero variance

The fundamental problem with any modeling effort is that quantities which are relatively easy to measure depend on quantities which are relatively difficult to measure. In this modelling effort the quantity which is directly measureable is the size distribution of the population. It depends on the underlying age distribution whose evolution in time depends on the elements of the Leslie matrix, namely the age-specific fecundities and survival probabilities. These quantities are extremely difficult, if not impossible, to measure directly in the field.

Furthermore, they describe the evolution of the age distribution, not the size distribution. Their effect on the size distribution depends on the size versus age relationship for the particular species.

The literature supports the hypothesis that the size versus age relationship for Dendraster falls roughly into two stages--a juvenile stage characterized by rapid growth, followed by a mature stage characterized by slow growth. Figure 2.3-1 [Birk71] and Figure 2.3-2 [Timk75] show some data taken from Dendraster populations in the Northwest and Southwest United States, respectively. It motivates the two stage growth model shown in Figure 2.3-3. Four parameters describe the growth characteristics--the slope and intercept for juveniles and the slope and intercept for mature individuals.

A model was constructed incorporating the two stage growth relationship and the Leslie matrix with constant elements. It assumed seasonal recruitment as described in

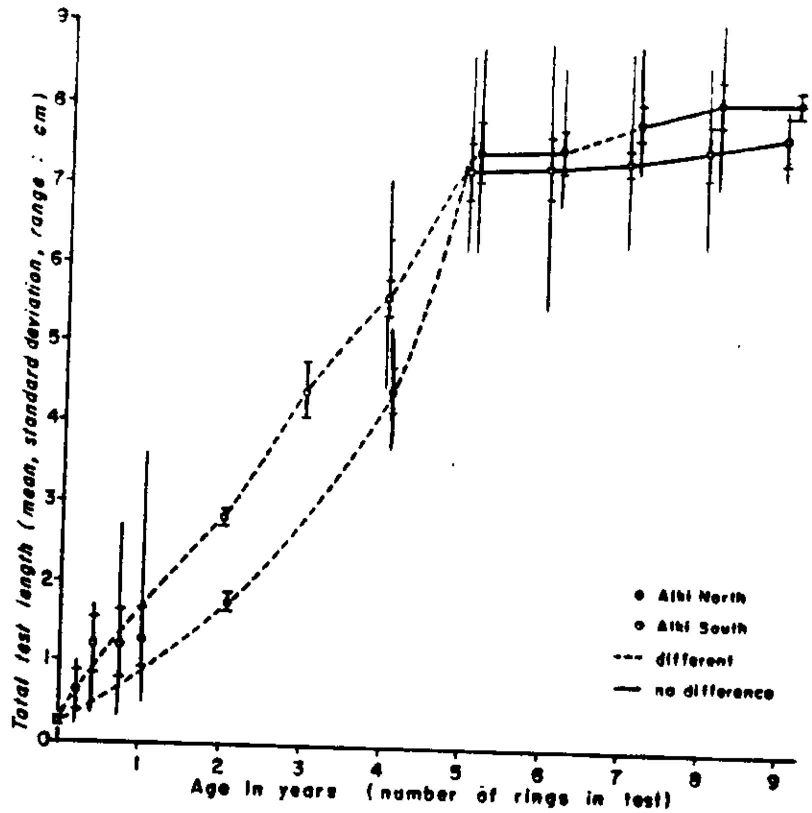


Fig. 6. Growth of *Dendraster* in two habitats: ——— mean test lengths which do not differ significantly; - - - means which do differ significantly.

Figure 2.3-1. Size versus age from [Birk71]

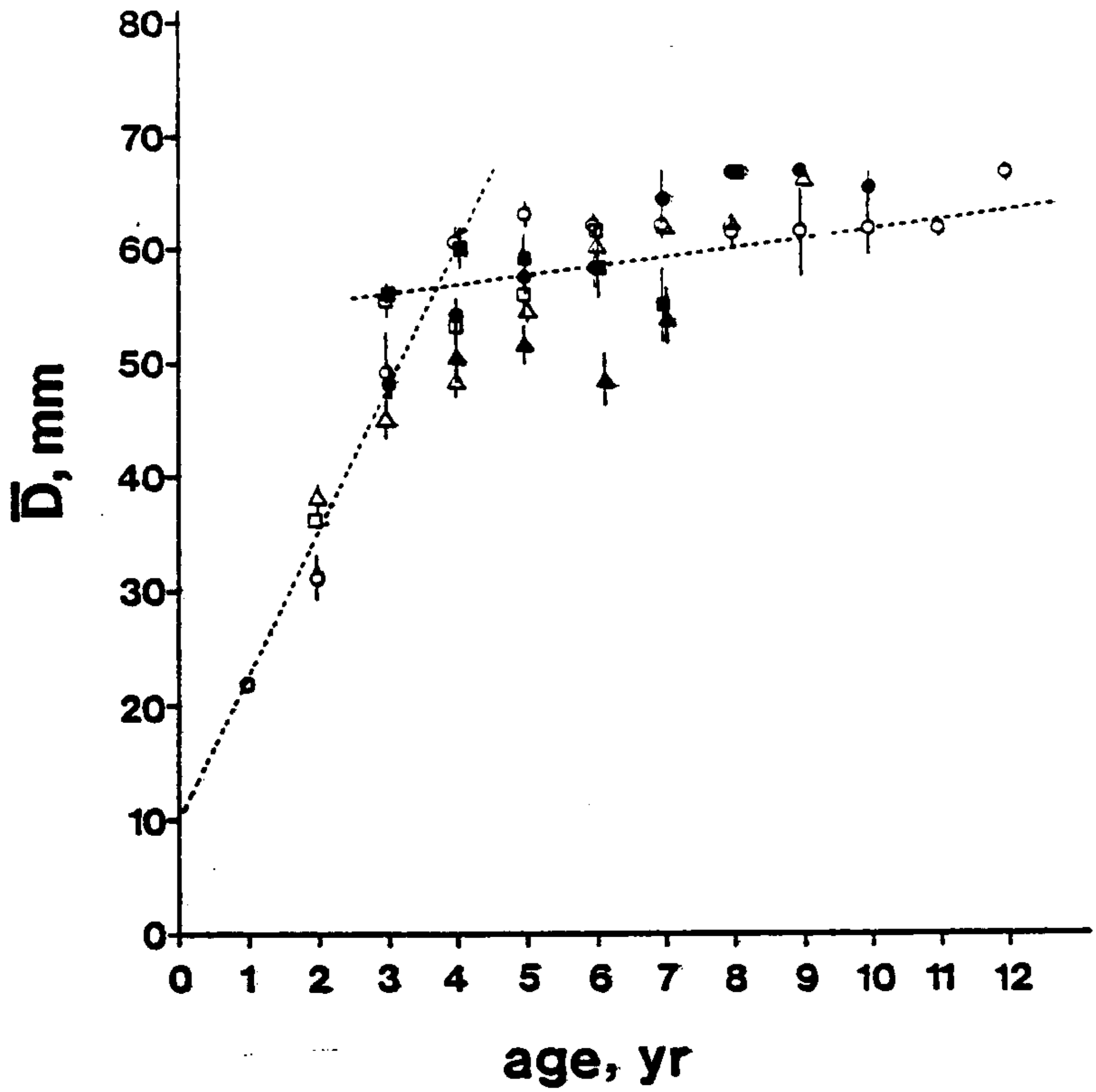


Figure 2.3-2. Size versus age from [Timk75]

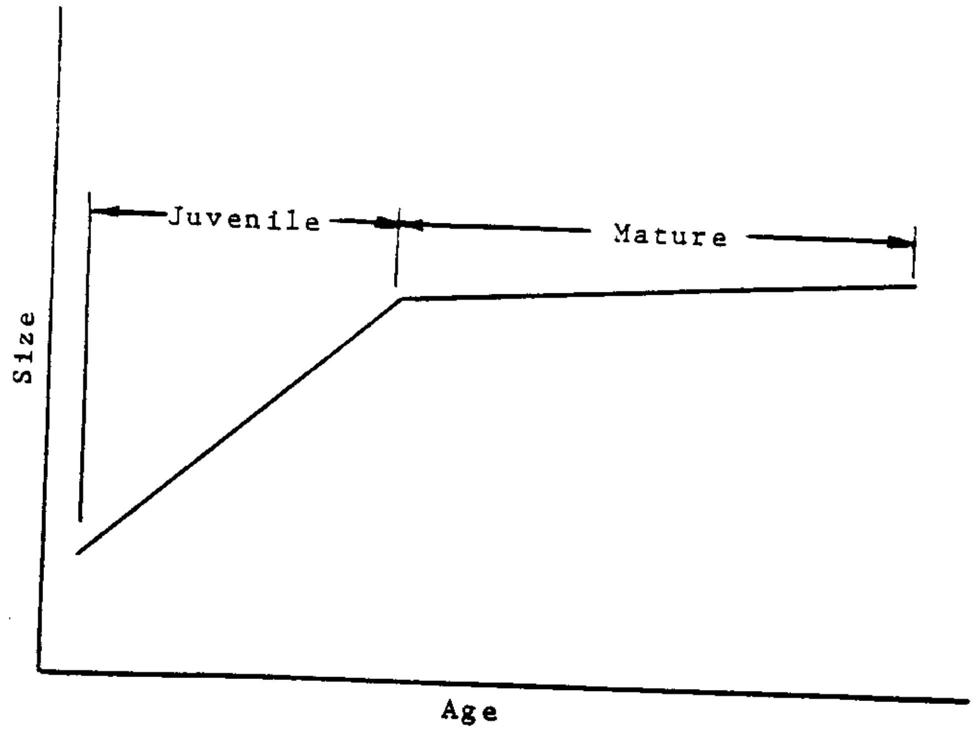


Figure 2.3-3. The bilinear growth assumption.

the previous section. The initial age distribution was chosen to be uniformly distributed juveniles as shown in Figure 2.3-4 (a). The other parameters were chosen arbitrarily just to get a feel for the behavior of the model.

Simulations of the evolution of the size distribution are shown in Figure 2.3-4 for a three year sequence and for a nine year sequence. It is interesting to compare the behavior of this system with the four features listed for the observed data in Section 1.3.

First, the observed size distribution from the field data was bimodal. This system evolves to a bimodal distribution even when started from an initial uniform distribution of juveniles.

Second, the observed data showed physical growth which was manifested by a shift of the distribution to the right. This system also shows physical growth in the same way.

Third, the observed data indicated mortality by a general decrease in the area under the distribution with time. The survival probabilities in the Leslie matrix produce the same behavior in this system.

Fourth, the observed data showed annual recruitment to the population. The seasonal recruitment modifications to the Leslie theory again produce similar behavior here.

Scale:

* Horizontal - Test length in mm. Bin size is 5 mm with first bin 0 to 5 mm. Total number of bins is 20.
* Vertical - Full scale is 8.0 for (b) and 25.0 for (c).

Model assumptions:

* Two stage linear growth curve with zero variance.
* Annual spawning during one month (month 12).
* Density independent Leslie age matrix with one year increments.

Parameter values:

* Leslie age structure matrix (25 parameters):

$$
\begin{bmatrix}
0.0 & 0.0 & 0.0 & 3.0 & 6.0 & 7.0 & 7.0 & 7.0 & 7.0 & 7.0 & 7.0 & 7.0 & 7.0 \\
0.5 & & & & & & & & & & & & \\
 & 0.6 & & & & & & & & & & & \\
 & & 0.7 & & & & & & & & & & \\
 & & & 0.7 & & & & & & & & & \\
 & & & & 0.7 & & & & & & & & \\
 & & & & & 0.9 & & & & & & & \\
 & & & & & & 0.8 & & & & & & \\
 & & & & & & & 0.7 & & & & & \\
 & & & & & & & & 0.6 & & & & \\
 & & & & & & & & & 0.5 & & & \\
 & & & & & & & & & & 0.4 & & \\
 & & & & & & & & & & & 0.3 & 0.0
\end{bmatrix}
$$

* Initial age distribution (13 parameters):

$$
\begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}
$$

* Growth factors (4 parameters):

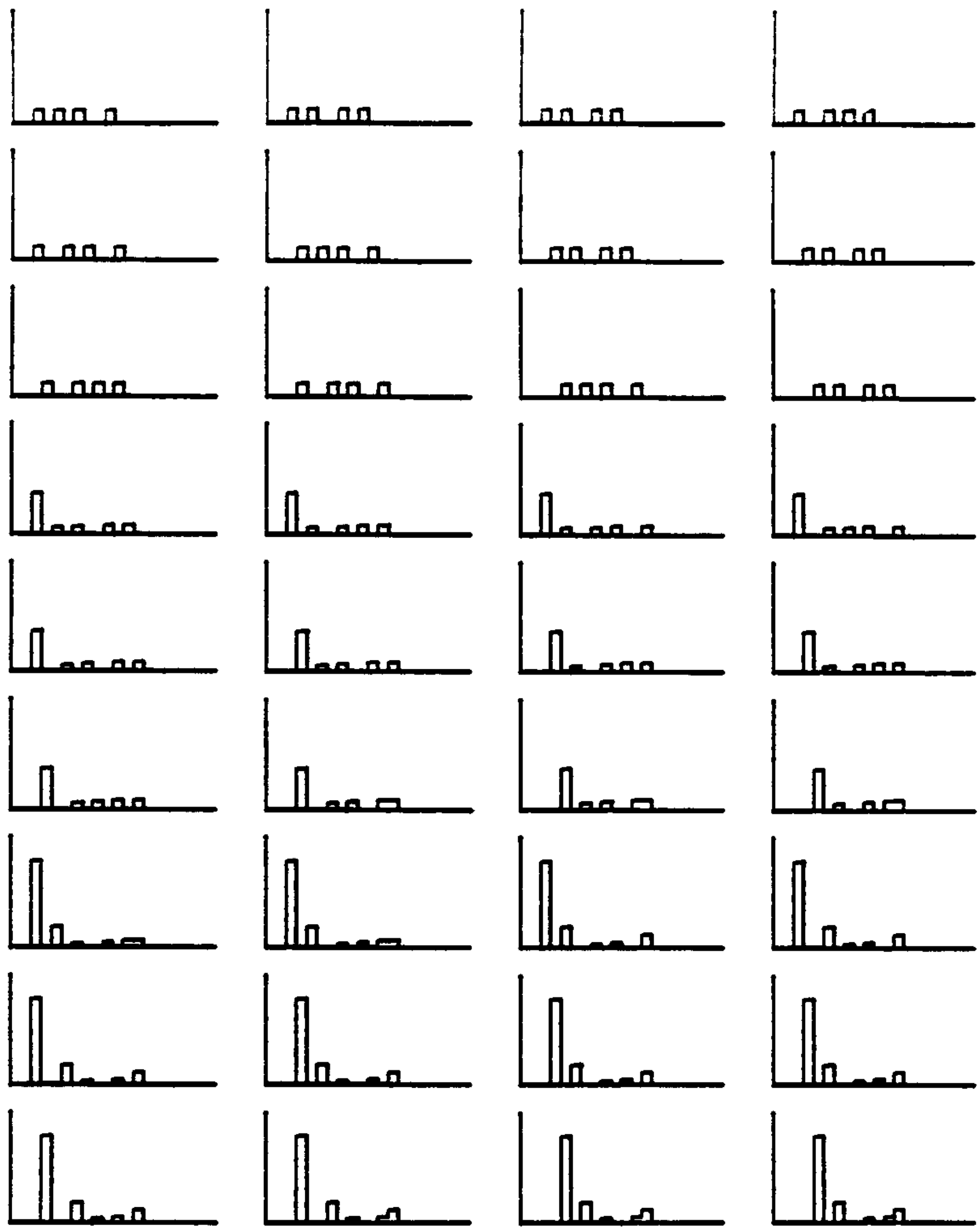
Juvenile slope	1.0 mm/month
Juvenile intercept	10.0 mm
Mature slope	0.1 mm/month
Mature intercept	55.0 mm

Total number of parameters:

* 42

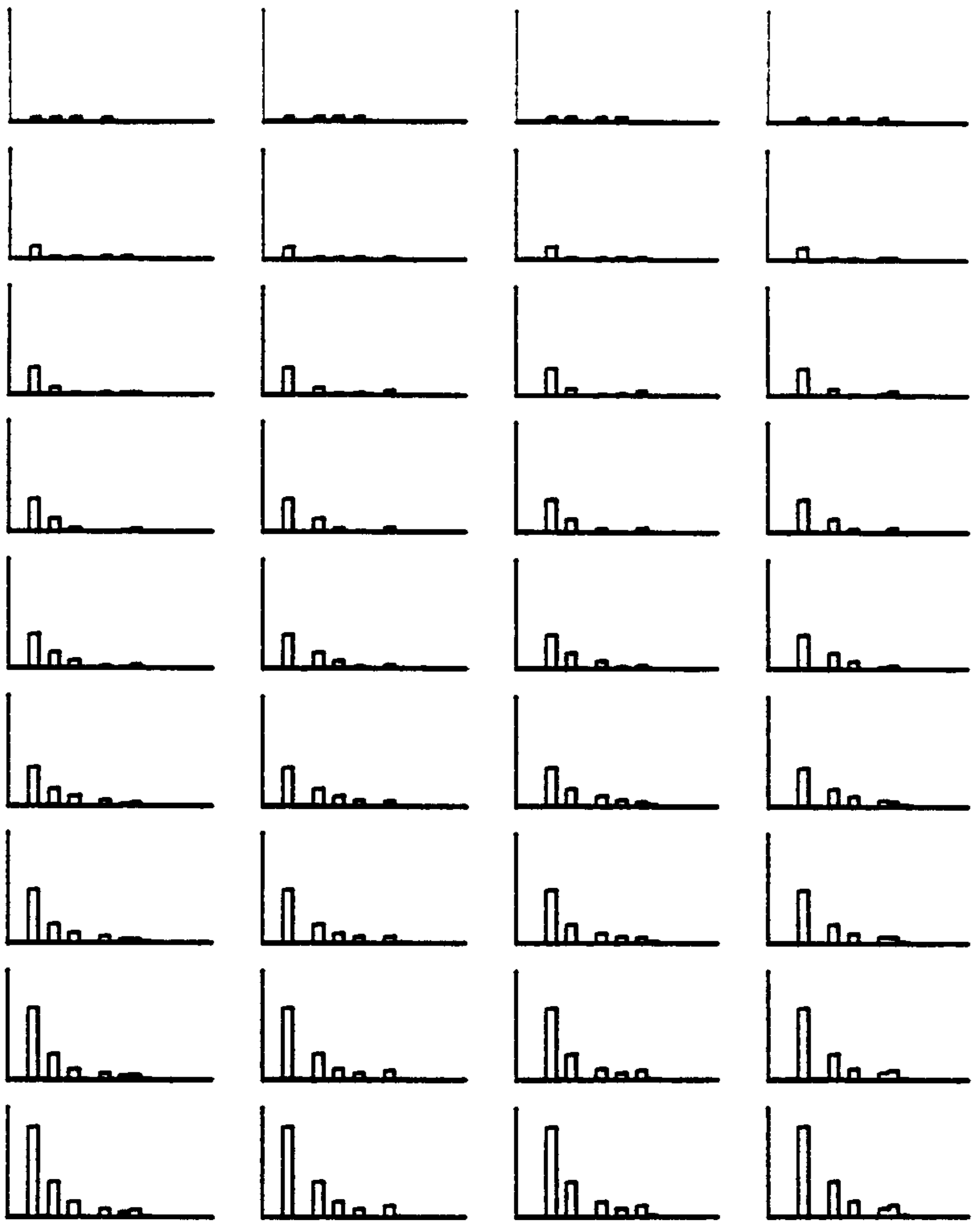
(a) Parameter values.

Figure 2.3-4. A zero variance simulation.



(b) Three year sequence at monthly intervals.

Figure 2.3-4. (continued)



(c) Nine year sequence at quarterly intervals.

Figure 2.3-4. (continued)

Two stage linear growth, finite variance

One feature of Figure 2.3-4 which looks highly unrealistic is the existence of empty bins between peaks in the histograms. They occur because the growth model assumed zero variance in the test length at a given age. The assumption of zero variance is not physically reasonable. Individuals of the same age are not all exactly the same size.

The question is how to incorporate the finite variance into the model. Perhaps the most theoretically satisfying way is to assume a normal density whose mean is given by the two stage growth curve of Figure 2.3-3, but whose variance is a constant greater than zero.

One problem with using the normal density is that it is computationally expensive. As shown in Figure 2.3-5 for the finite variance case, the number of individuals per square meter (i.e. the density of the population) in the i th bin due to the population in age class j is

$$D[i,j] = N[j] \int_{(i-1)b}^{ib} f(x;m[j],s)\, dx$$

$$= N[j] \{ F(ib;m[j],s) - F((i-1)b;m[j],s) \}$$

where

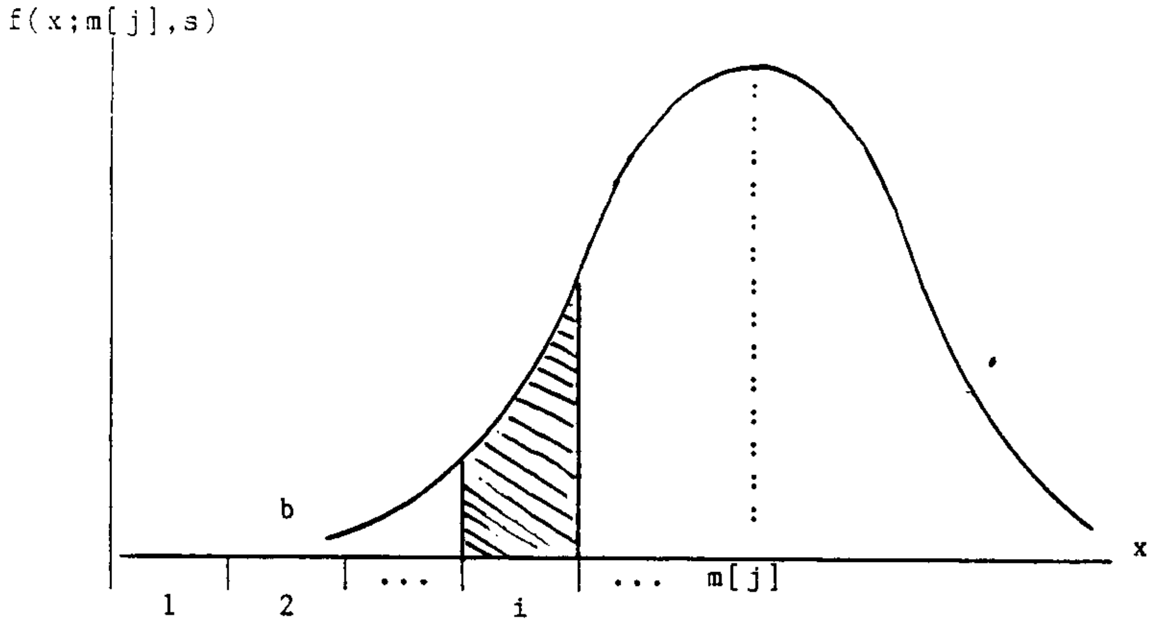


Figure 2.3-5. Normal distribution, finite variance.

N[j] = the number of individuals per square meter in age class j from the Leslie matrix

b = the histogram bin size

m[j] = the mean size for age class j from the bilinear growth curve

s = the variance

$$f(x;m,s) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-m}{s}\right)^2\right]$$

$$F(x;m,s) = \int_{-\infty}^{x} f(x;m,s)\, dx$$

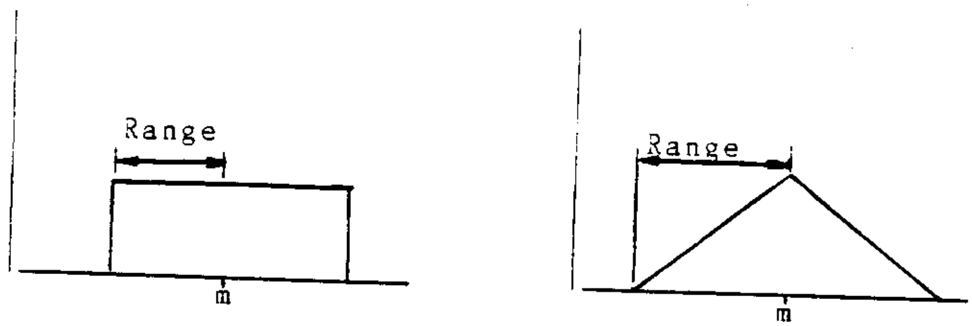
The age structure matrix gives a certain density of individuals N[j] at a specific age corresponding to age class j. The growth component of the model transforms that age to a mean size, m[j]. The total number of individuals in bin i is then obtained by summing D[i,j] over all j. Hence the computation requires evaluating the error function over all bins for all age classes for every month of the simulation, which is very time consuming.

There is also a theoretical problem with the normal distribution in this setting. The normal distribution has infinite tails. Negative size is, of course, biologically impossible, but it is implied because of this characteristic. The problem is especially acute for small individuals.

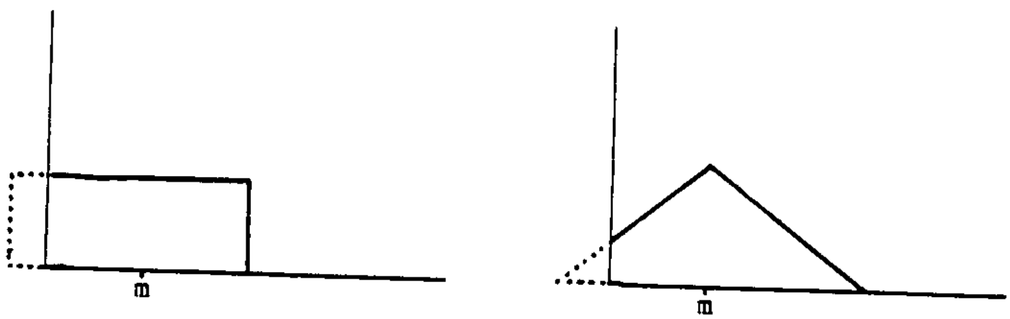
One approach to the problem of finite variance is to approximate the normal distribution by a computationally simpler one. It is known that the uniform distribution, when convolved with itself, produces a triangular distribution. Furthermore, the triangular distribution, when convolved with itself, produces a piecewise quadratic distribution. Continuing the self convolution produces a sequence of distributions which approach the normal distribution. Hence there is a sense in which the uniform distribution is an approximation to the normal distribution, and the triangular distribution is yet a better approximation.

Use of the uniform or the triangular distribution instead of the normal distribution therefore solves two problems. First, these distributions are computationally much less expensive. Second, They have finite tails and hence do not imply negative size as the normal distribution does. However, there is still a question of defining the probability distribution for small individuals when the mean of the distribution is smaller than the range.

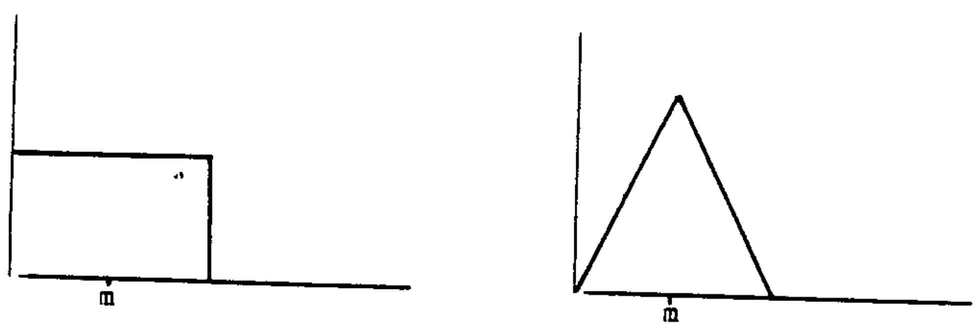
Figure 2.3-6 shows one technique for modifying the distribution for small individuals. In part (a) the range is defined as the length between the mean and the smallest value of the size for which the density is nonzero. In part (b), when the mean is less than the range, part of the distribution is lost on the negative side of the size



(a) Mean greater than range.



(b) Mean less than range.



(c) Range modified for small mean.

Figure 2.3-6. Modification of probability distribution for small individuals.

axis. If it were simply reintroduced into the clipped distribution by an appropriate scaling factor, the mean of the distribution would be changed. Specifically, it would be increased because of a shift of the area from the left tail toward the right.

Rather than change the mean, the algorithm can change the range for small individuals. If the mean is less than the range, the range is set to equal the mean as shown in part (c). So instead of increasing the mean, the variance is decreased to accomodate small individuals. Figure 2.3-7 shows the decreased variance for small individuals.

Figure 2.3-8 shows the effect that the assumption of a finite variance has on the size distribution. The size distribution at the top of the figure is the beginning of the fourth year from the simulation of Figure 2.3-4. It assumes zero variance in the growth relationship.

The distribution in the center of Figure 2.3-8 show the effect of the uniform distribution assumption. The uniform distribution f(x;m[j],s) is shown on the left. D[i,j] was calculated and summed over all age classes j for each bin i. The resulting size distribution is plotted for a standard deviation (i.e. square root of the variance) of 2.0 mm and 5.0 mm. As expected the effect is to "smear" the original distribution, lowering the peaks and raising the adjacent valleys. The gaps between the histograms are eliminated. The larger the variance the

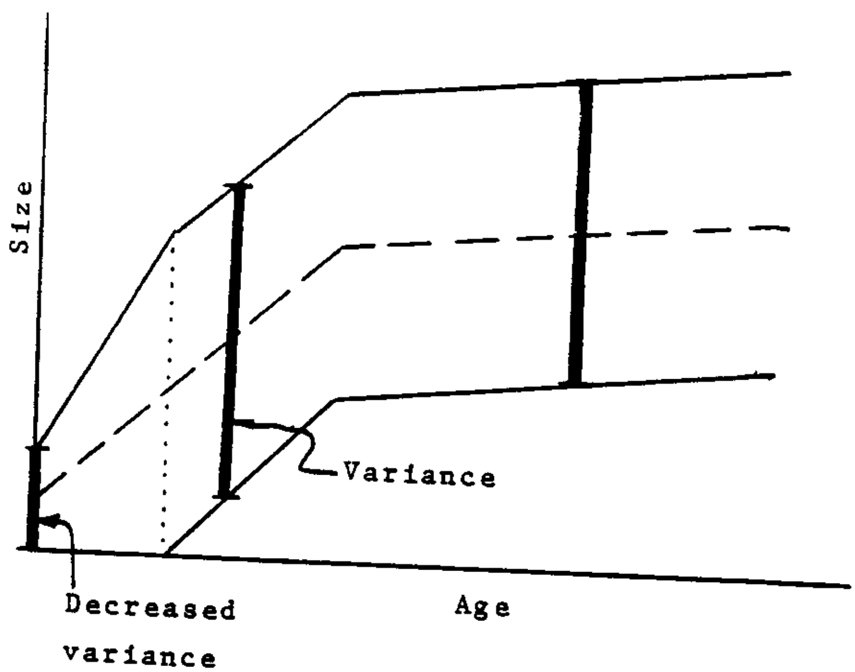


Figure 2.3-7. Decreased variance for small individuals.

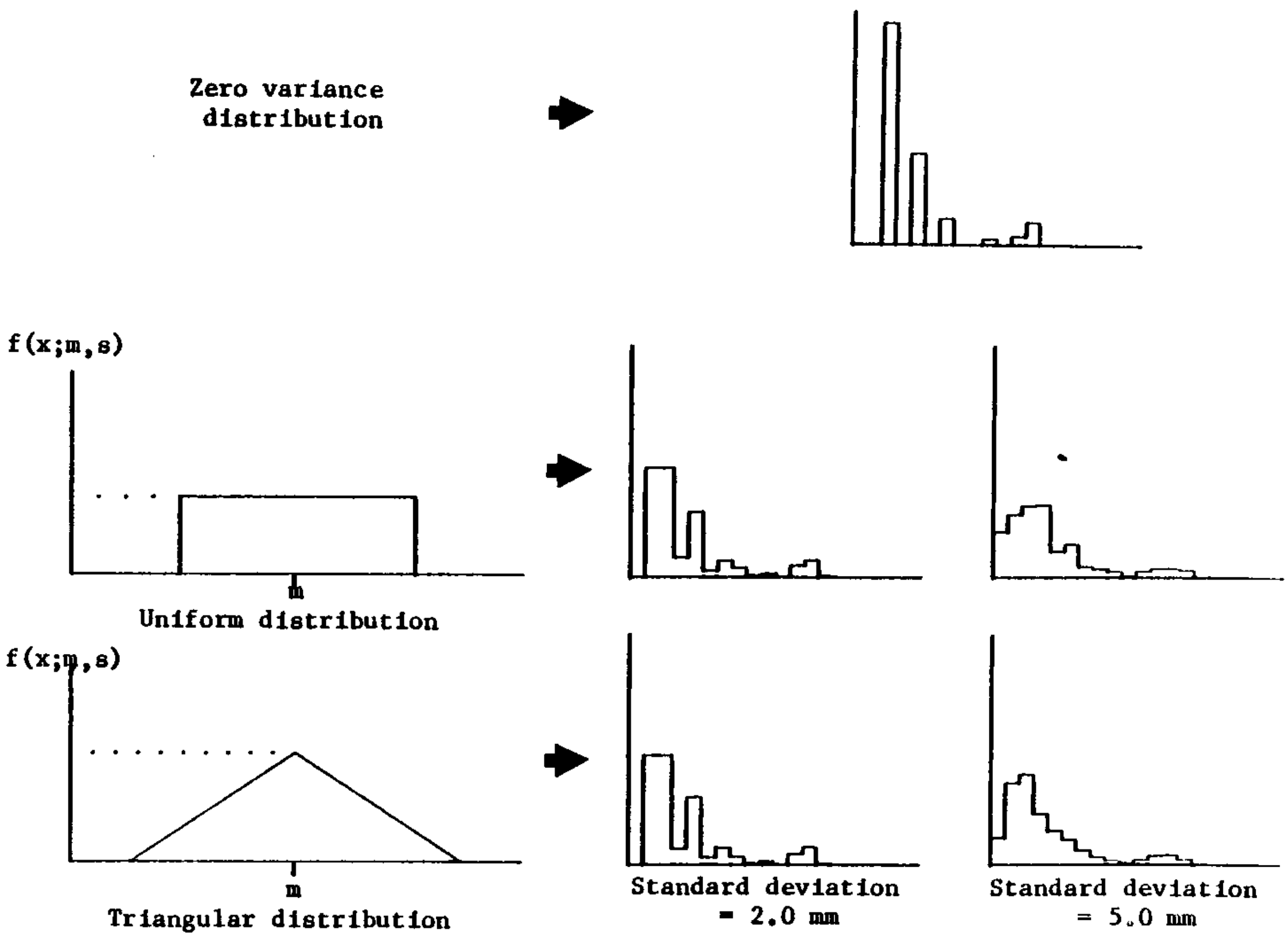


Figure 2.3-8. The effect of finite variance on the size distributions.

greater the leveling effect.

The distributions at the bottom of Figure 2.3-8 show the effect of the triangular distribution assumption. The triangular distribution is plotted with a variance equal to that of the uniform distribution directly above it. An equal variance implies a greater range for the triangular distribution. The corresponding size distributions for standard deviations of 2.0 and 5.0 mm are shown to the right.

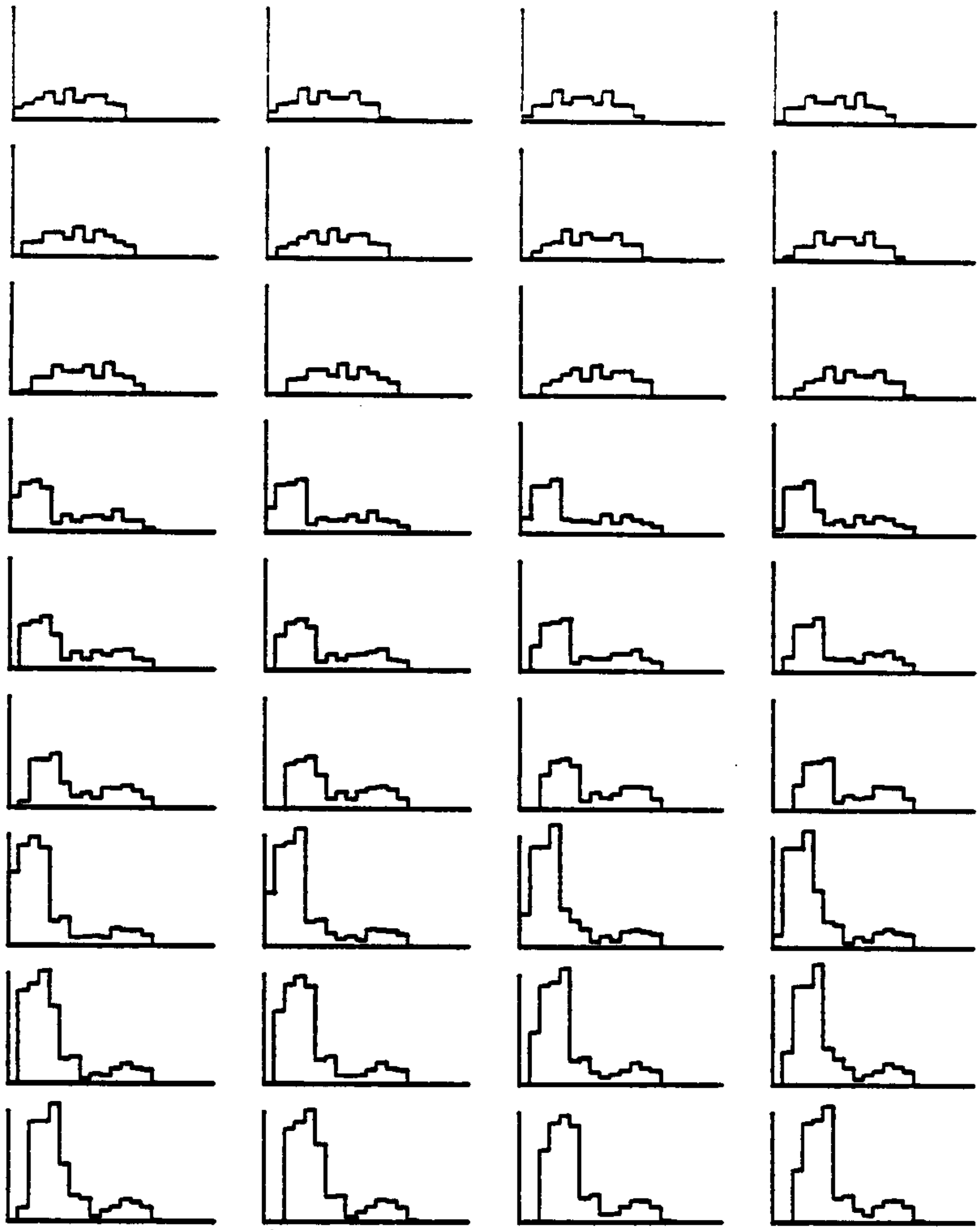
It is instructive to compare the size distributions under these two finite variance assumptions. First, when the deviation is 2.0 mm the size distribution under the uniform assumption is virtually indistinguishable from the size distribution under the triangular assumption.

Second, when the standard deviation is 5.0 mm the size distributions are similar only for large individuals. For small individuals the distribution under the triangular assumption shows a more pronounced peak than the distribution under a uniform assumption. This is explained by the fact that the range of the triangular distribution is greater than the range of the uniform distribution. The algorithm for modifying the distribution for small individuals (Figure 2.3-6) therefore comes into play to a greater extent under the triangular assumption.

The conclusion to be drawn from the above observations is that the uniform distribution is a reasonable assumption to make in simulating the temporal evolution of the system. It is computationally less expensive than the triangular distribution, but produces identical results for standard deviations on the order of 2.0 mm. For standard deviations on the order of 5.0 mm the results are different only for small individuals. Since the range for the uniform distribution is smaller than the range for the triangular distribution, the modifications for small individuals are not as severe with the uniform distribution.

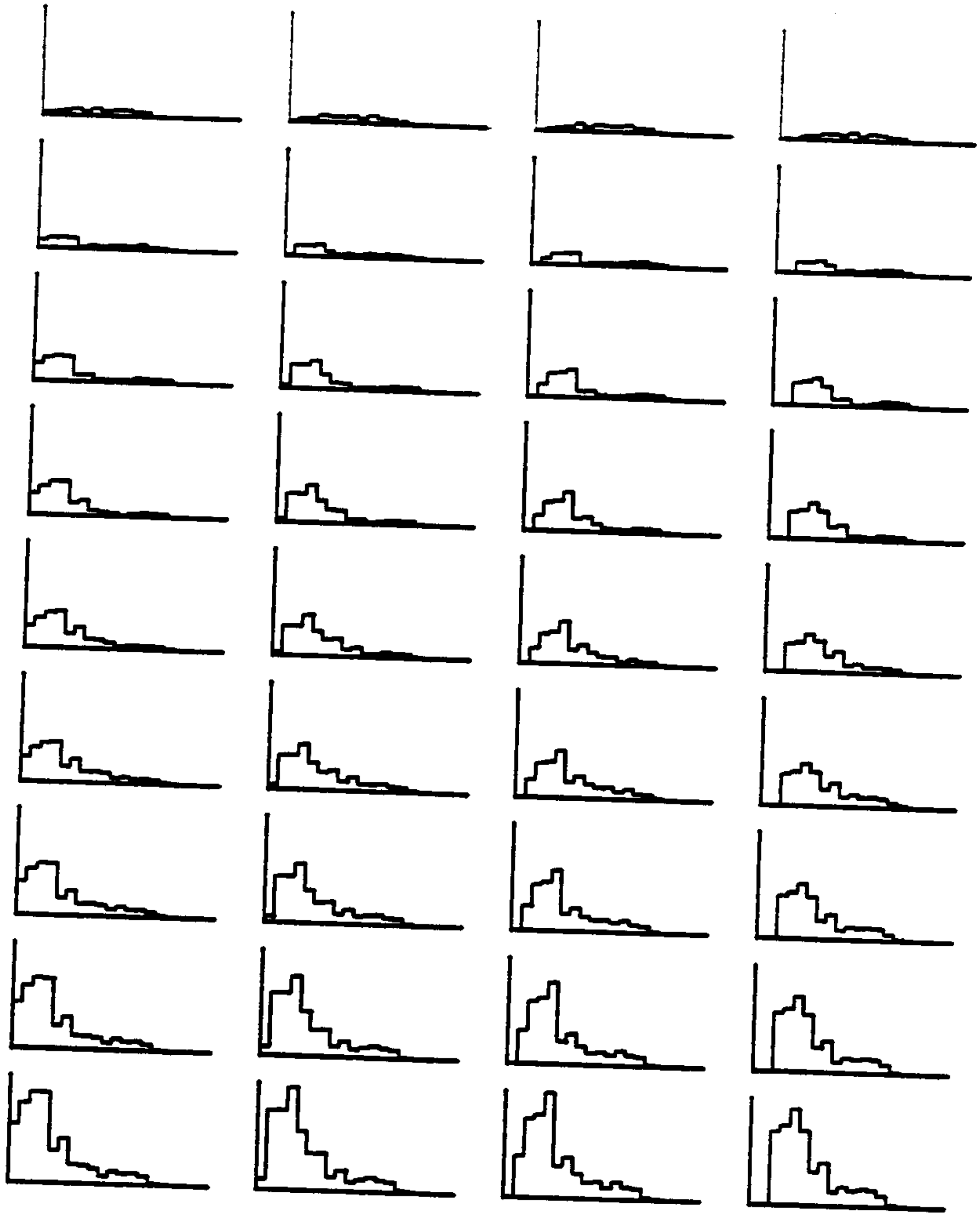
The models presented in the remainder of this dissertation all assume finite variance with the uniform distribution assumption. Figure 2.3-9 assumes the same parameter values as Figure 2.3-4 except for the uniform distribution assumption with a 5.0 mm standard deviation.

Another assumption made in the model is that the fecundity is density independent. A density dependent fecundity is not identifiable in this system because recruitment occurs only once with this field data. The value of the estimated fecundity is one data point of the density dependent fecundity relationship, at the particular density value of the system at the time of recruitment.



(a) Three year sequence at monthly intervals.

Figure 2.3-9. A finite variance simulation.



(b) Nine year sequence at quarterly intervals.

Figure 2.3-9. (continued)

2.4 Restatement of the problem

A quantitative comparison of the model with field data requires the formal definition of a scalar error function. The model produces a time series of size distributions which should match, as closly as possible, the time series of size distributions obtained from the field data. The error function is a measure of the magnitude of the deviation of the simulation from the measured data.

Several error functions are possible. The one selected in this study is the squared error criterion. For data spanning M months with N bins in each histogram the error is defined as

$$\sum_{i,j} (y[i,j] - y'[i,j])^2$$

where y[i,j] is the measured value of the population density in the i th size bin in the j th month and y'[i,j] is the simulated value. The sum on j is over all M months in which data were taken. The sum on i is over all N bins in the histogram.

Figure 2.4-1 shows the definition of the squared error. The solid lines represent field data. The dotted lines represent simulated data. The simulated data were produced with specific values for the parameters. The

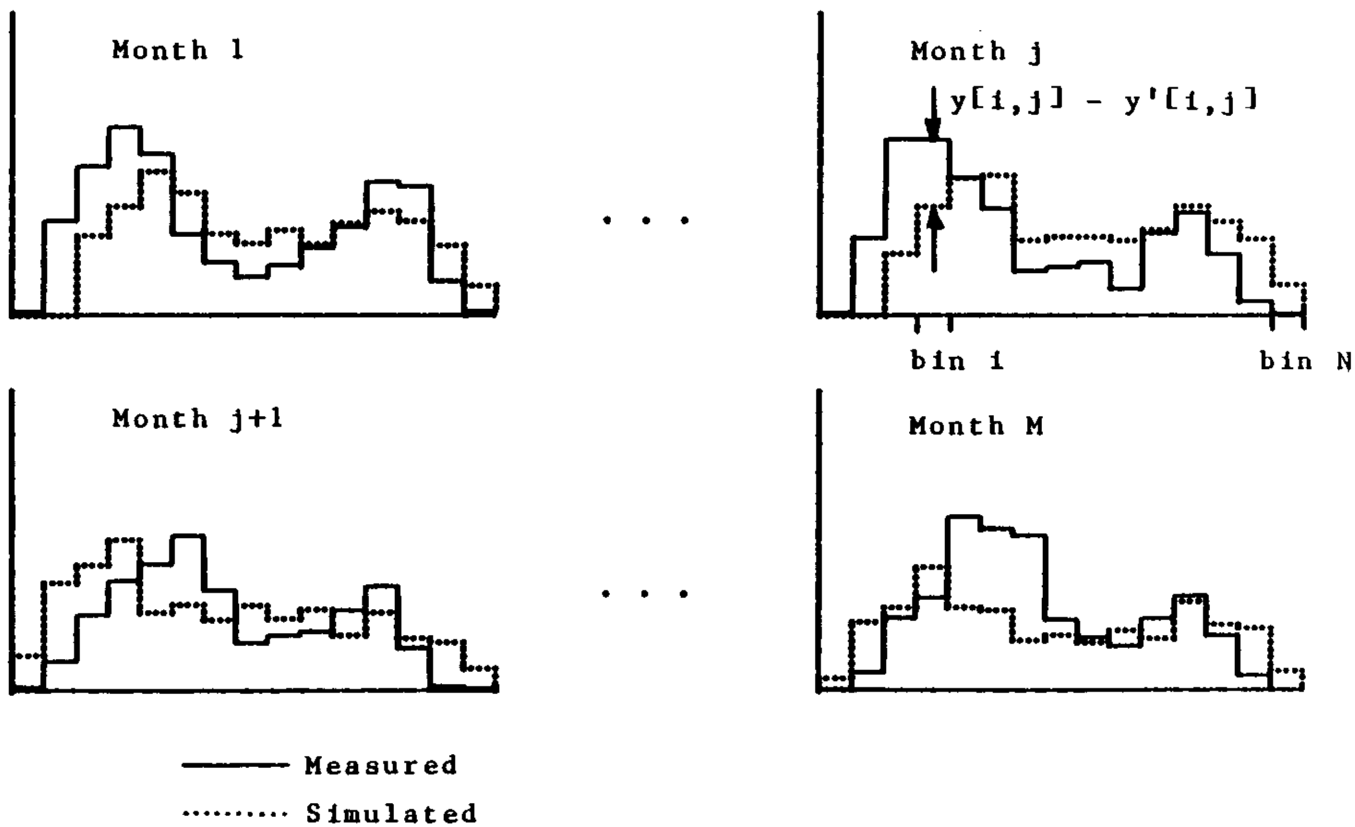


Figure 2.4-1. Definition of the squared error.

parameters included the elements in the Leslie matrix, the initial age distribution, and the slope and intercept of the juvenile and mature growth lines.

A more detailed statement of the problem is now possible. The task is

* to construct a model containing two components, the Leslie matrix and the bilinear growth assumption common in the literature,
* to minimize the squared error in the parameter space, and
* to use the values of the parameters which minimized the error as estimates of the "true" physical values in the system.

These first two chapters have essentially already presented the first step. The second and third steps characterize the system identification approach to modeling.

To show how this modeling approach contrasts with the standard methodology we compare it with Perron's work on the growth, fecundity, and mortality of the tropical marine gastropod Conus pennaceus [Perr83].

Perron marked off a 50 x 25 meter quadrat and collected samples once every two weeks by snorkeling over the study site at midtide. He measured the shell length of each snail and also marked it with a numbered tag. The mark-recapture data gave him a field measurement of the size versus age relationship. It was curve fitted to the von Bertalanffy growth equation

$$L(t) = L_0(1 - ue^{-t/T}) \ .$$

L is the shell length, t is the individual's age in years, and L_0, u, and T are parameters whose values are determined by the curve fitting routine.

Perron described the system with a 22 month time sequence of 11 bimonthly size-frequency histograms. His data were similar in many respects to those described in Section 1.3. Annual recruitment, growth, and mortality are all features of the patterns. The von Bertalanffy equation was then used to convert the size-frequency histograms into age-frequency plots by solving the growth equation for age t, given the shell length L.

A regression of the transformed data produced a single survivorship curve for the entire population

$$N(t) = e^{-vt}$$

where N(t) is proportional to the number of survivors, t is the cohort age, and v is a parameter determined by the curve fitting routine. Assuming an exponential survivorship curve is equivalent to assuming age and density independent survival probabilities p[i], since $p[i] = N(t + 1)/N(t) = e^{-v}$, a constant.

Perron also collected data on the size-specific fecundities of <u>Conus pennaceus</u> females by determining the realtionship between female size and clutch size, and by estimating the number of clutches produced per female per year. The survivorship and fecundity schedules were then combined in standard life table form for the species.

How does the modeling approach of this dissertation compare with the standard methodology exemplified by Perron's work? The objective in each case is the estimation of parameter values which characterize the biological system. The tool used to obtain the estimate in a standard analysis is regression, i.e. curve fitting. The regression is usually either linear, or a simple exponential, logarithmic, or polynomial fit so that standard software packages can be used. But the underlying statistical basis of regression is the minimization of the squared error between the data and the curve. So the standard methodology and the modeling approach of this dissertation have the common idea of estimation via minimization of the squared error.

The difference is in the application of the idea. In the standard methodology each individual component of the system is analyzed separately. Hence, in both the Perron and Timko studies a curve fit was done on the size vs age relationship, one component of the system. In the Perron study, the results of the curve fit were used to transform the size data to age data. Then another curve fit was done to obtain the value of a parameter describing the mortality component of the system.

The modeling approach of this dissertation is to combine all three components of the system - growth, fecundity, and mortality - into a single mathematical model, and then to minimize the squared error on the entire system. The minimization is done system wide, with the links between the individual components in place.

There are several interesting ramifications to this approach. One is that the growth characteristics of the species will be inferred directly from the time sequence of the size histograms. In the Perron study the growth characteristics were inferred from mark-recapture data. In the Timko study thay were inferred by the ring count method. No such ancillary data is required in this study.

Another ramification is that the validity of the model can be assessed by direct comparison with the raw field data. When a mathematical model of the entire system is constructed, it produces a time sequence of size histograms. But that is precisely the format of the raw data. As we will see in Chapter 4 when we examine specific models, a direct comparison of the simulated output with the field data can sometimes give a visual clue as to how the structure of the model can be improved.

3. The minimization algorithm

Identification of the system requires a determination of the values of the parameters which minimize the difference between the field data and the model. This chapter presents the minimization aspects of the problem. The first section explores what little nonnumerical theory is germain to the problem. The second section outlines the numerical theory behind the minimization algorithm. The third section presents the implementation of the algorithm in a program.

3.1 System identification of the Malthusian model

The system identification viewpoint is rarely taken in biological modeling. I could find no reference in the literature on Leslie matrix theory in a systems identification context. When the Leslie matrix is 1 x 1, the problem reduces to the scalar Malthusian model. In this case a logarithmic transformation of the data simplifies the system identification problem.

One of the problems with the models considered in this dissertation is that nonlinearities abound. In a linear system, even with constraints, cost minimizing algorithms are well developed. But in the models considered here, the size versus age relationship is highly nonlinear (although it is piecewise linear). So there is not much mathematical theory which can be applied to the identification problem.

It would be good to know just how much theory can be applied to the system identification problem of Leslie matrix models in the simplest case. It turns out that even in the Malthusian model where there is only one age class, that of the population as a whole, and where the field data are observations on the number in the population and not on physical size, the optimizing solution is nonlinear. This result can be stated in the following theorem.

Theorem. Suppose a population obeys the Malthusian rate equation with an initial unknown population of a_0 and an unknown rate A. If n+1 consecutive observations are made starting with an observation on the population, a, at time $t = 0$, then the identification of the two parameters a_0 and A with a least squares error criterion requires the solution of a 3n-2 degree polynomial.

Proof. Take the population at time i to be

$a_i, \quad i = 0, 1, \ldots, n$

and the two parameters to be estimated as a_0, the population at time $t = 0$, and A, the 1×1 Leslie matrix. Then the Malthusian rate equation

$$a_{i+1} = A\, a_i, \qquad i = 0, 1, \ldots, n$$

has solution

$$a_i = A^i\, a_0, \qquad i = 0, 1, \ldots, n.$$

Denote the n+1 observations by

$$r_i, \qquad i = 0, 1, \ldots, n.$$

The error in the i th observation is then $r_i - a_i$. The problem is to determine the a_0 and the A which minimize the squared error

$$\sum_{i=0}^{n} (r_i - a_i)^2 = \sum_{i=0}^{n} (r_i - A^i\, a_0)^2.$$

This is done by setting the partial derivative of the squared error with respect to each parameter to zero. Setting the partial derivative of the squared error with respect to a_0 to zero, and then solving for a_0 yields

$$a_0 = \frac{\sum_{i=0}^{n} A^i\, r_i}{\sum_{i=0}^{n} A^{2i}} .$$

Setting the partial derivative of the squared error with respect to A to zero, and then solving for a_0 yields

$$a_0 = \frac{\sum_{i=1}^{n} i\ A^{i-1}\ r_i}{\sum_{i=1}^{n} i\ A^{2i-1}} .$$

These two expressions for a_0 can be equated giving an equation involving only the rate, A.

$$\left(\sum_{i=0}^{n} A^i\ r_i\right)\left(\sum_{i=1}^{n} i\ A^{2i-1}\right) = \left(\sum_{i=0}^{n} A^{2i}\right)\left(\sum_{i=1}^{n} i\ A^{i-1}\ r_i\right)$$

Writing the i=0 term explicitly yields, for the left hand side of this equation

$$r_0 \sum_{i=1}^{n} i\ A^{2i-1} + \sum_{i,j=1}^{n} i\ A^{2i+j-1}\ r_j ,$$

and, for the right hand side

$$\sum_{i=1}^{n} i\ A^{i-1}\ r_i + \sum_{i,j=1}^{n} j\ A^{2i+j-1}\ r_j .$$

Rearranging terms then gives the equation for A as

$$\sum_{i,j=1}^{n} (i-j)\ A^{2i+j-1}\ r_j + r_0 \sum_{i=1}^{n} i\ A^{2i-1} - \sum_{i=1}^{n} i\ A^{i-1}\ r_i = 0$$

The first term in the polynomial is zero for i = j. So the maximum degree is at i = n, j = n-1. The corresponding power of A is

$$2i + j - 1 = 2n + (n - 1) - 1 = 3n - 2$$

So the estimate of A to minimize the error is the root of a 3n-2 degree polynomial. The initial population is then given by either of the expressions for a_0 after A has been determined. End proof.

For example, just three observations of the system, r_0, r_1, and r_2, requires the solution of the fourth degree polynomial

$$(-r_1)\ A^4 + (-r_2+2r_0)\ A^3 + (-2r_2+r_0)\ A + (-r_1) = 0$$

Figure 3.1-1 gives the coefficients of the powers of A for the cases n = 1, 2, 3, 4, 5, and 10.

So in a system described by the Malthusian model where just 35 consecutive observations are made on the population, identification of the system requires the solution of a 100 th degree polynomial. The standard way of simplifying this problem is to minimize the square of the difference between the logarithms of the observations and the model. Defining the cost to be minimized as

$$\sum_{i=1}^{n} (\ln r_i - \ln a_i)^2 = \sum_{i=0}^{n} (\ln r_i - \ln A^i a_0)^2$$

and then setting the partial derivative of the cost with respect to a_0 equal to zero yields

n = 1

Power of A	Coefficient of A	
1	1(r0)	
0	-1(r1)	

n = 2

Power of A	Coefficient of A	
4	1(r1)	
3	-1(r2) +	2(r0)
1	-2(r2) +	1(r0)
0	-1(r1)	

n = 3

Power of A	Coefficient of A	
7	1(r2)	
6	-1(r3) +	2(r1)
5	3(r0)	
4	-2(r3) +	1(r1)
3	-1(r2) +	2(r0)
2	-3(r3)	
1	-2(r2) +	1(r0)
0	-1(r1)	

Figure 3.1-1. Coefficients of the powers of A for system identification of the Malthusian model.

n = 4

Power of A	Coefficient of A				
10	1(r3)				
9	-1(r4)	+	2(r2)		
8	3(r1)				
7	-2(r4)	+	1(r2)	+	4(r0)
6	-1(r3)	+	2(r1)		
5	-3(r4)	+	3(r0)		
4	-2(r3)	+	1(r1)		
3	-4(r4)	+	-1(r2)	+	2(r0)
2	-3(r3)				
1	-2(r2)	+	1(r0)		
0	-1(r1)				

n = 5

Power of A	Coefficient of A				
13	1(r4)				
12	-1(r5)	+	2(r3)		
11	3(r2)				
10	-2(r5)	+	1(r3)	+	4(r1)
9	-1(r4)	+	2(r2)	+	5(r0)
8	-3(r5)	+	3(r1)		
7	-2(r4)	+	1(r2)	+	4(r0)
6	-4(r5)	+	-1(r3)	+	2(r1)
5	-3(r4)	+	3(r0)		
4	-5(r5)	+	-2(r3)	+	1(r1)
3	-4(r4)	+	-1(r2)	+	2(r0)
2	-3(r3)				
1	-2(r2)	+	1(r0)		
0	-1(r1)				

Figure 3.1-1. (continued)

n = 10

Power of A	Coefficient of A
28	1(r 9)
27	-1(r10) + 2(r 8)
26	3(r 7)
25	-2(r10) + 1(r 8) + 4(r 6)
24	-1(r 9) + 2(r 7) + 5(r 5)
23	-3(r10) + 3(r 6) + 6(r 4)
22	-2(r 9) + 1(r 7) + 4(r 5) + 7(r 3)
21	-4(r10) + -1(r 8) + 2(r 6) + 5(r 4) + 8(r 2)
20	-3(r 9) + 3(r 5) + 6(r 3) + 9(r 1)
19	-5(r10) + -2(r 8) + 1(r 6) + 4(r 4) + 7(r 2) + 10(r 0)
18	-4(r 9) + -1(r 7) + 2(r 5) + 5(r 3) + 8(r 1)
17	-6(r10) + -3(r 8) + 3(r 4) + 6(r 2) + 9(r 0)
16	-5(r 9) + -2(r 7) + 1(r 5) + 4(r 3) + 7(r 1)
15	-7(r10) + -4(r 8) + -1(r 6) + 2(r 4) + 5(r 2) + 8(r 0)
14	-6(r 9) + -3(r 7) + 3(r 3) + 6(r 1)
13	-8(r10) + -5(r 8) + -2(r 6) + 1(r 4) + 4(r 2) + 7(r 0)
12	-7(r 9) + -4(r 7) + -1(r 5) + 2(r 3) + 5(r 1)
11	-9(r10) + -6(r 8) + -3(r 6) + 3(r 2) + 6(r 0)
10	-8(r 9) + -5(r 7) + -2(r 5) + 1(r 3) + 4(r 1)
9	-10(r10) + -7(r 8) + -4(r 6) + -1(r 4) + 2(r 2) + 5(r 0)
8	-9(r 9) + -6(r 7) + -3(r 5) + 3(r 1)
7	-8(r 8) + -5(r 6) + -2(r 4) + 1(r 2) + 4(r 0)
6	-7(r 7) + -4(r 5) + -1(r 3) + 2(r 1)
5	-6(r 6) + -3(r 4) + 3(r 0)
4	-5(r 5) + -2(r 3) + 1(r 1)
3	-4(r 4) + -1(r 2) + 2(r 0)
2	-3(r 3)
1	-2(r 2) + 1(r 0)
0	-1(r 1)

Figure 3.1-1. (continued)

$$\frac{n(n+1)}{2} \ln A + n \ln a_0 = \sum_{i=0}^{n} \ln r_i$$

while setting the partial derivative of the cost with respect to A to zero yields

$$\frac{n(n+1)(2n+1)}{6} \ln A + \frac{n(n+1)}{2} \ln a_0 = \sum_{i=0}^{n} i \ln r_i$$

Solving these equations for A and a_0 yields

$$\ln A = \frac{6}{n(n-1)} \left[\frac{2}{n+1} \sum_{i=0}^{n} i \ln r_i - \sum_{i=0}^{n} \ln r_i \right]$$

$$\ln a_0 = \frac{6}{n(n-1)} \left[\frac{2n+1}{3} \sum_{i=0}^{n} \ln r_i - \sum_{i=0}^{n} i \ln r_i \right]$$

These are simply the linear regression equations for the slope and intercept, respectively, for the logarithm of the data with i as the independent variable.

The next step would be to extend this investigation of the theory to the nonscalar Leslie model. That task appears difficult at best. The structure of both A and a_0 would need to be considered. For b age classes there are b fecundities, b-1 survival probabilities, and b elements in the a_0 vector for a total of 3b-1 regression equations instead of just two. An additional difficulty in this study is the inclusion of the growth component of the model.

Because of these complications we cannot hope to find a closed form solution to the minimization problem. It must be determined numerically. The procedure used for this system, in contrast to the Malthusian model above, does not require a logarithmic transformation of the data to simplify the problem. On the contrary, such a transformation would be computationally more expensive. The algorithm minimizes the squared error directly.

3.2 Nonlinear minimization theory

This section outlines the theory behind the minimization algorithm. To state the basic problem we define the following quantities.

n = the number of parameters to be estimated, i.e. the dimensionality of the parameter space.

m = the number of data points. In this study the number of months times the number of bins in the histogram for each month, or 10 x 15 = 150.

x = a column n-vector in the parameter space.

$f_i(x)$ = the i th error. It corresponds to the error at a single bin given by $y - y'$ in Figure 2.4-1.

$F(x) = \sum_{i=1}^{m} [f_i(x)]^2$, the cost to be minimized.

x^* = the value of x which minimizes $F(x)$.

The general form of n-dimensional minimization algorithms is given in Figure 3.2-1. It is an iterative begin Minimization

Establish x_k, a current estimate of x^*.

while the conditions for convergence are not satisfied do

begin

Compute a nonzero unit n-vector q_k, the direction of the search.

Compute a positive scalar d_k, the step length, such that the correction vector is $p_k = d_k q_k$, and $F(x_k + p_k) < F(x_k)$.

Update the estimate by setting

$x_{k+1} := x_k + p_k$, and

$k := k + 1$

end

Report x_k as the parameter vector which minimizes F.

end Minimization.

Figure 3.2-1. The general minimization algorithm.

procedure. Various methods are available which differ in the way they compute p and s. In that computation, some methods assume only that F(x) is a scalar function of the vector x. However, it is generally recognized [Gill81] that if F(x) is a sum of squares of residuals, as in this modeling problem, then more efficient methods are possible which take this structure of F into account. An algorithm of that type was chosen for this problem.

The algorithm is a modification of the hybrid method for nonlinear equations proposed by Powell [Powe70]. It is a combination of the steepest descent method and the Gauss-Newton method.

The steepest descent method is based on the Taylor-series expansion about x_k,

$$F(x_k + p) \doteq F_k + g_k^T p,$$

where F_k is an abbreviation for $F(x_k)$, and g_k is an abbreviation for $g(x_k)$ which is the gradient of F at x_k,

$$g(x) = \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial F}{\partial x_n} \end{bmatrix} .$$

T signifies the transpose of a vector or a matrix. Then p_k is selected as the direction along which F decreases most rapidly local to x_k, namely, $p_k = -g_k$. Since $g_k p_k =$

$-(g_k^T g_k) = -\|g_k\|^2$ must be less than zero we are guaranteed progress at each iteration. Although global convergence for this method can be proved [Flet80], its convergence near the minimum is only linear, and its performance can be extremely inefficient.

Newton's method is based on a quadratic model of the function to be minimized. Keeping one more term in the Taylor-series expansion of F,

$$F(x_k + p) \doteq F_k + g_k p + \frac{1}{2} p^T G_k p \; .$$

G_k is an abbreviation for $G(x_k)$, the n x n symmetric Hessian matrix,

$$G(x) = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 F}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix} .$$

Then p_k is selected as the p which minimizes $g_k p + \frac{1}{2} p^T G_k p$. Specifically, p_k satisfies the linear system

$$G_k p_k = -g_k \; .$$

The p_k which is the solution to this linear equation is called the Newton direction. Newton's method has the advantage of quadratic convergence compared to the linear convergence of the steepest descent method. But it has the disadvantage of requiring the Hessian, i.e. the second deravitive of F, at each point. Also it fails for some situations where the quadratic model is a poor approximation to F outside a small neighbourhood of the current point.

The Gauss-Newton method uses the formula for p_k given by the Newton method, but takes into account the fact that F(x) has a sum of squares structure. Denote the m x n Jacobian matrix of f(x) by J(x),

$$J_{ij}(x) = \frac{\partial f_i(x)}{\partial x_j} .$$

Then the gradient and Hessian are given [Gill81] by

$$g(x) = 2J(x)^T f(x)$$

$$G(x) = 2J(x)^T J(x) + 2 \sum_{i=1}^{m} f_i(x) G_i(x)$$

where $G_i(x)$ is the Hessian of $f_i(x)$. The method approximates G(x) by the first order term, $2J(x)^T J(x)$. The linear system for determining p therefore becomes

$$(J_k^T J_k) p_k = -J_k^T f_k,$$

where the subscripts denote quantities evaluated at x_k. This method has the advantage of requiring only the Jacobian, i.e. the first derivative of F, at each point.

The method given by Powell is based on the interpolation idea of Levenberg and Marquardt [Leve44, Marq63]. At each step the predicted minimum is computed along both the steepest descent direction and the Gauss-Newton direction. If the Gauss-Newton method appears to be diverging, which is indicated by its correction being too large, then the displacement is biased towards the steepest descent direction of F(x).

The actual correction step is determined by maintaining a positive scalar step length, d, which is compared with the magnitudes of the predicted steps in the Gauss-Newton and the steepest descent directions. Let p_{GN} and p_{SD} be the predicted steps in the Gauss-Newton and in the steepest descent methods respectively. The algorithm for determining the actual step, p, at each iteration is then given in Figure 3.2-2.

This algorithm is different from the Levenberg-Marquardt algorithm in the way that the interpolation step is calculated. In this method the derivatives of $f_i(x)$ are approximated numerically, since analytical expressions for the derivatives are impossible to obtain. Under these circumstances Powell's method for determining the interpolation is computationally more efficient than the Levenberg-Marquardt method.

The maximum step length, d, is recomputed at each iteration. The idea is to maintain the step length large enough so that significant progress is made toward

Compute the maximum step size, d

<u>if</u> $\|p_{GN}\| < d$ <u>then</u>

$p := p_{GN}$

<u>else</u> <u>if</u> $\|p_{SD}\| < d$ <u>then</u>

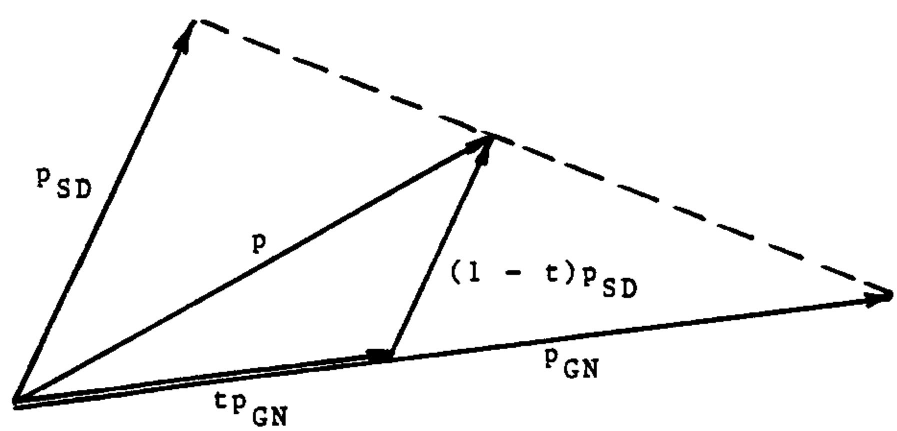
$p := tp_{GN} + (1 - t)p_{SD}$

such that $\|p\| = d$

<u>else</u>

$p := d(p_{SD}/\|p_{SD}\|)$

(a) The update algorithm.



(b) The interpolation step.

Figure 3.2-2. Powell's hybrid method.

convergence to x^*, but not so large that the search becomes unstable because of the neglect of the higher order terms in the Taylor-series expansion of $F(x_k)$.

The actual strategy for computing d follows the recommentation of Powell [Powe70]. The predicted value of $F(x + p)$ is compared with the actual value of $F(x + p)$. If the predictions are good and if $F(x + p)$ is significantly less than $F(x)$, then d increases. Otherwise d may remain unchanged or decrease by a factor of 2. The method is rather complex and includes a "damping" effect to avoid an inefficient oscillatory behavior in d.

The one major difference between the algorithm used here and the one given by Powell is that Powell's method was designed for solving simultaneous nonlinear equations. That is, in his formulation $m = n$, and the system to be solved is

$$f_i(x_1, x_2, \ldots, x_n) = 0, \qquad i = 1, 2, \ldots, n.$$

Hence the Jacobian matrix, J, is square. In particular the gradient is given by

$$g_k = 2J_k^T f_k$$

as before, but the Gauss-Newton formula for determining the correction step p_k is

$$J_k p_k = -f_k$$

which has solution

$$p_k = -J_k^{-1} f_k \ .$$

Each iteration therefore requires both J and J^{-1} for the computation of the steepest descent step and the Gauss-Newton step.

Computing the inverse of an n x n matrix requires $O(n^3)$ operations. Powell's method begins by computing J numerically with finite difference formulas and then inverting it. Subsequent iterations reduce execution time by storing the values of J and J^{-1} and applying updates to them based on the information obtained at $f_i(x_k + p_k)$. The update formulas require only $O(n^2)$ operations and hence produce huge savings in execution time. The savings come about not only because of the lower order in the number of operations, but also because of the savings of the additional n function evaluations (i.e. model simulations) which would be necessary for the finite difference determination of J. These methods are generally known as Quasi-Newton approximations [Gill81].

Denote the difference in f_i evaluated at the old and the updated points as

$$y_i = f_i(x + p) - f_i(x), \qquad i = 1, 2, \ldots, n,$$

and the inverse of the Jacobian matrix as $H = J^{-1}$. Then the formulas for the updated matricies J^* and H^* given by Powell are based on

$$J^* = J + (y - Jp)p^T / \|p\|^2$$

$$H^* = H + (p - Hy)p^T H / (p^T Hy) \; .$$

It is possible for singularities to occur with these update formulas, so sometimes only part of the full corrections are applied to force nonsingularity. The formulas are parameterized by u as

$$J^* = J + u(y - Jp)p^T / \|p\|^2$$

$$H^* = H + u \frac{(p - Hy)p^T H}{u(p^T Hy) + (1 - u)\|p\|^2} \; .$$

If the denominator in H^* is too small, as determined by

$$|(p^T Hy)| < 0.1 \|p\|^2$$

then $u = 0.8$ is used; otherwise $u = 1$. The smaller value of u was determined emperically by Powell.

In this investigation $m < n$. Hence the update method given by Powell was modified as follows. The differences are now an m-vector.

$$y_i = f_i(x + p) - f_i(x), \qquad i = 1, 2, \ldots, m.$$

The matricies J and H are stored and updated on each iteration, where H now is the n x m matrix

$$H = (J^T J)^{-1} J^T \ .$$

The update formulas for J^* and H^* look identical to the update formulas for the m = n case. The difference is in the definition of H.

For this modification to be feasible we need to show that some of the properties of the update formulas in the m = n case still apply in the m > n case. The most important property is that the update formulas preserve the relationship between p and y. Namely, since

$$(Jp)_i = \sum_{j=1}^{n} J_{ij} p_j$$

$$= \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} dx_j$$

is the predicted change in f_i, then we want the new Jacobian J^* to satisfy

$$J^* p = y$$

since y_i is the actual change in f_i. Right multiplying the unparameterized update formuls for J^* by p produces

$$J^* p = \left(J + \frac{(y - Jp)p^T}{p^{\;2}} \right) p$$

$$= Jp + y - Jp$$

$$= y \ .$$

Similarly, right multiplying the unparameterized update formula for H^* by y produces

$$H^* y = \left(H + \frac{(p - Hy)p^T H}{(p^T Hy)} \right) y$$

$$= Hy + p - Hy$$

$$= p \ .$$

So the proper relationship between p and y is maintained by the unparameterized update formulas.

A second property maintained by the update formulas is the proper relationship between J and H. With this definition of H we have

$$HJ = (J^T J)^{-1} J^T J$$

$$= I$$

the n x n identity. The parameterized version of the update formulas maintains this relationship. It can be shown by multiplying the update formulas.

$$H^* J^* = H + \left(u \frac{(p - Hy)p^T H}{u(p^T Hy) + (1 - u)\|p\|^2} \right) x$$

$$\left(J + u(y - Jp)p^T / \|p\|^2 \right)$$

$$= \ . \ . \ .$$

$$= HJ + 0$$

$$= I$$

where ... indicates some tedious but straightforward algebra which will not be reproduced here.

The modification to Powell's method is thus feasible theoretically. It also worked well when implemented with this particular model.

3.3 The implementation

This section is divided into two parts. The first part is a description of the computer programs designed to solve the problem. The second part gives some observations about software for mathematical modeling.

Program description

The mathematical models were implemented in Pascal. Appendix 6.2 is an example program listing. The bulk of the program is divided into two parts, the mathematical model of the Dendraster system and the nonlinear least squares algorithm. The mathematical model in the program listing is for the bilinear growth system described in Section 4.1.

Figure 3.3-1 is a list of all the subprograms, i.e. procedures and functions, in the program. It shows the nesting level of each subprogram by level number and by

Level number	Subprogram
1	program MinLeslie
2	procedure GetMinOptions
2	procedure GetModelParams
2	procedure GetRealData
2	procedure GetPrintOptions
3	procedure GetX
3	procedure GetF
3	procedure GetSqF
3	procedure GetItrStatus
3	procedure GetNewJ
3	procedure GetNewJInv
3	procedure GetStepLen
3	procedure GetStepType
2	procedure FromXVector
2	procedure ToXVector
2	procedure FromFVector
2	procedure ToFVector
2	procedure CalcSizeDistr
2	procedure StepMonth
2	procedure Simulate
2	procedure Evaluate
2	procedure FileFinal
2	procedure Minimize
3	procedure Invert
4	function Norm
4	procedure GaussElim
4	procedure Solve
3	procedure InitGlobalConstants
3	procedure SwapN
3	procedure SwapM
3	procedure Negate
3	function Min
3	function Max
3	function Min3
3	procedure ATransposeA
3	procedure MultNxNxM

Figure 3.3-1. The subprogram scope structure.

Level number	Subprogram
3	procedure PrintN
3	procedure PrintM
3	procedure PrintNxN
3	procedure PrintMxN
3	procedure PrintNxM
3	procedure PrintIteration
3	procedure PrintFinal
3	procedure UpdateJacobian
3	procedure CalcDirections
3	procedure CalcSteepestMin
3	procedure CalcDelta
3	procedure UpdateOrthogDir
3	procedure Dir1Update
3	procedure UpdateX
3	procedure TakeStep
3	procedure DoFirstTime
3	procedure DoComputeNewJ
3	procedure DoNormal
3	procedure DoStepLenUpdate
3	procedure DoStepDir1

Figure 3.3-1. (continued)

indentation. For example, procedure GetX is declared within procedure GetPrintOptions, and cannot be called directly by a statement in the main program, MinLeslie.

Figure 3.3-2 is a description of the subprogram calling sequence. For example, program MinLeslie calls procedures GetRealData, GetModelParams, ToXVector, GetMinOptions, GetPrintOptions, Minimize, and FileFinal. Procedure Minimize in turn calls procedures InitGlobalConstants, Evaluate, PrintIteration, etc.

The following correspondences are made between the variables declared at line 695 and the quantities discussed in the previous section. Jacobian and JInverse are J and H respectively. StpDir, NwtDir, and Delta are p_{SD}, p_{GN}, and p. NwtCoef is the parameter $\bar{t}$ in the interpolation step of Figure 3.2-2. StpCoef is a factor times (1-t) in the same figure. Rather than maintain the maximum step size, d, the program maintains its square in SqMaxStepSize.

The minimization routine begins at line 660. Because of the number of procedures defined by this routine, however, its first executable statement is at line 1608. The while loop at line 1619 is the while loop in Figure 3.2-1 of the general minimization algorithm. Its termination is controlled by the value of an enumerated variable which indicates the status of the loop. The variable, LoopStatus, is declared at line 696, and its

```
program MinLeslie
   procedure GetRealData
   procedure GetModelParams
   procedure ToXVector
   procedure GetMinOptions
   procedure GetPrintOptions
      procedure GetX
      procedure GetF
      procedure GetSqF
      procedure GetItrStatus
      procedure GetNewJ
      procedure GetNewJInv
      procedure GetStepLen
      procedure GetStepType
   procedure Minimize
      procedure InitGlobalConstants
      procedure Evaluate
         procedure FromXVector
         procedure Simulate
            procedure CalcSizeDistr
            procedure StepMonth
         procedure ToFVector
      procedure PrintIteration
      procedure DoFirstTime
      procedure DoComputeNewJ *
      procedure DoNormal *
      procedure DoStepLenUpdate *
      procedure DoStepDir1 *
   procedure FileFinal
      procedure FromXVector
      procedure FromFVector
```

* Note: An asterisk indicates the procedure contains additional procedure calls listed below.

Figure 3.3-2. The subprogram calling sequence.

```
procedure DoComputeNewJ
   procedure PrintMxN
   procedure ATransposeA
   procedure Invert
      function Norm
      procedure GaussElim
      procedure Solve
   procedure MultNxNxM
   procedure PrintNxM

procedure DoNormal
   procedure UpdateJacobian
   procedure TakeStep *

procedure DoStepLenUpdate
   function Min3
   function Min
   procedure SwapN
   procedure SwapM
   procedure Negate
   function Max
   procedure UpdateJacobian
   procedure TakeStep *

procedure DoStepDir1
   procedure SwapN
   procedure SwapM
   procedure Negate
   procedure Dir1Update
```

```
procedure TakeStep
   procedure CalcDirections
   procedure UpdateOrthogDir
   procedure UpdateX
   procedure CalcSteepestMin
   procedure CalcDelta
   procedure Dir1Update
   procedure UpdateOrthogDir
   procedure UpdateX
```

Figure 3.3-2. (continued)

type is declared at line 675. The termination conditions are those suggested by Powell [Powe70].

The program executes the update algorithm of Figure 3.2-2 in procedure TakeStep on lines 1402 to 1471.

One feature of the implementation not mentioned in the previous section is the technique of avoiding linear dependence in the directions p_k that are generated by successive iterations of the algorithm. To show the desirability of this technique, let q be any vector orthogonal to p so that $p^T q = 0$. Then, right multiplying the update formula by q gives

$$J^* q = Jq + \frac{(y - Jp)p^T q}{\|p\|^2}$$

$$= Jq .$$

So the results of applying both the old and the new Jacobian approximations to any vector that is orthogonal to p are the same.

If J happens to be updated by a set of vectors p that are linearly dependent, then there will exist a vector q such that Jq is the same for all Jacobian approximations. But the true value of Jq will probably change with the update, since the nonlinearity of the model causes the true value of J to change with x.

Special directions are therefore introduced into the correction vector if necessary, so that successive vectors p span the full space of the parameters. The variables

SpanCount and OrthogDir are maintained for this purpose. Details of the method are given by Powell [Powe70]. Procedures UpdateOrthogDir and Dir1Update contain the implementation.

Software for mathematical modeling

The overwhelming majority of software for mumerical work is still written in Fortran, one of the oldest programming languages available. This is not surprising considering that the major design goal of the language was execution efficiency [Back81].

Fortran gets its execution efficiency by mirroring the physical machine as closly as possible. For example, the flow of control constructs are simple. The GOTO statement translates directly into a machine BRANCH statement. Also the data structuring facilities are essentially restricted to arrays, which is a mirror of indexed addressing on the machine. (Or is indexed addressing a mirror of the array?)

In contrast, the algorithm described here was implemented in Pascal. This is surprising considering that one of the major design goals of the language was that it be well suited for teaching programming as a systematic discipline, with fundamental concepts clearly and naturally reflected by the language [Wirt71]. Execution efficiency was not its primary goal.

Based on my experience with this project, I believe that neither Fortran nor Pascal is an ideal language for numerical software. The reasons for coming to this conclusion are outlined in the remainder of this section, along with a suggestion for further research in numerical software.

The problem with Fortran for this project was its limited data structuring facilities. For example, consider the single variable in the main program which contains the values for all the parameters in the model. It is called ModelPerams and is declared on line 74 to be of type ModPrmType. In the type section of line 51 ModPrmType has three components:

* a component containing the initial age distribution, InitAgeDistr,
* a component containing the growth parameters, GrowthParams, and
* a componont containing the elements of the Leslie matrix, RepMat.

Each of these components are further subdivided into smaller, possibly nonhomogeneous parts. For example, InitAgeDistr contains not only an array of the initial age distribution, but also an indication of the actual calendar month and year for purposes of matching simulated data with field data.

The ability in Pascal to structure the modeling data this way had several ramifications. It made the program more self-documenting. Specifically it was easy to see which procedures operated on which components of the data structure.

But more importantly, it saved time in program modification. In the course of the study many variations on the model were constructed. Some were retained, others rejected. But each variation required program modification and testing. When the structure of a component was modified, only those procedures which operated on that component needed to be altered.

The assumption in the tradeoff is that my time is more valuable than the computer execution time. But I do not believe that execution speed is the main cause of Pascal's unsuitability for numerical work. As time goes on more efficient optimizing compilers for Pascal will be developed. And if execution speed is of primary importance the program can always be written with a minimum number of procedure calls and a maximum amount of unstructured global data. Better yet, the proverbial 10% of the program which is responsible for 90% of the execution time can be written in assembler.

The real weakness of Pascal for numerical software is identical to the weakness enunciated in a critique of the language by Kernighan [Kern81]. Here are some of his criticisms which also applied in this project.

The size of an array is part of its type. Kernighan considers this to be the "biggest single problem with Pascal". It was a problem here in that general purpose routines for matrix manipulations could not be written. To change the size of the array you must recompile with the new type (at least using the Level 0 standard without conformant array parameters [IEEE83]).

There is no separate compilation. Software development would have been more efficient if the minimization algorithm were separately compiled. Changing the model would then only involve recompiling and linking the model software.

Related program components must be kept separate. A big hindrance to readability is the fact that the first executable statement of procedure Minimize is separated by a thousand lines from the declaration of the procedure and its parameters and variables.

There are no static variables. A static variable, called an "own" variable in Algol terminology is one that is private to some routine and retains its value from one call of the routine to the next. If a Pascal procedure needs to remember a value from one call to the next, the variable must be global to the procedure. It is therefore visible to other procedures unnecessarily. An extreme example in this project is the variable containing the field data. It had to be declared and input in the main program, even though it logically belonged in procedure Evaluate where the f_i were calculated. Many variables were also declared in procedure Minimize which should have been declared at a lower level if static variables were available.

From a mathematical modeling point of view, Pascal's strength is its strong typing and data structuring facilities. But its weakness, as evidenced by the problems listed above, is that numerical routines which are portable and easy to use are difficult to design.

Fortran also has its problems with the user interface to numerical routines. Communication via COMMON is unstructured and consequently error prone. The lack of type checking in parameter lists at compile time is a familiar source of error. Morè has recently pointed out [Morè82] the desirability of "reverse communication" in nonlinear optimization software, i.e. providing the minimization routine as a subroutine which the user calls instead of as a main program for which the user supplies the subroutine. He points out that if the optimization software has a standard interface then this cannot be done in Fortran.

It would appear that two languages which are becoming commercially available are inherently better suited for mathematical modeling and numerical software than either Fortran or Pascal, namely Modula-2 and Ada. Both of these languages incorporate the data structuring and data typing features of Pascal which are important in modeling. And both are designed expressly for the purpose of establishing a library of reusable modules with a well defined user interface.

In particular, Modula-2 would solve all of the specific problems listed above. Arrays of variable length can be passed as parameters. Related program components can be physically grouped together. Static variables can be initialized and retained only in the modules which use them.

In Modula-2 separate compilation is encouraged, even to the extent that the user interface to the module can be compiled independently of the implementation of the module. Wirth calls this facility "separate compilation" in contrast to the "independent compilation" of Fortran [Wirt83]. In the separate compilation of Modula-2 the linker provides full type checking. If the interface part is modified and recompiled, and the user routine is then executed later, the linker can determine that the user's routine is working with an outdated version of the interface. The implementation part of the routine, however, can be updated and recompiled independently of the user's code.

Of course, Modula-2 and Ada are so new that virtually no numerical software written in them exists yet. Given Fortran's huge scientific and engineering applications base any numerical software will be a long time coming. But based on the experience with this project, it seems that these more recent languages would be inherently better suited for the maintenance of a library of numerical software. Further research to test this conjecture is required.

4. The specific models

This chapter presents the results of the modeling study. Each section describes the structure of a specific model, and reports the estimated values of the parameters obtained by the nonlinear least squares algorithm described in Chapter 3.

4.1 Bilinear growth

This model assumes bilinear growth as shown in Figure 2.3-3 with the additional assumption of finite variance as shown in Figure 2.3-7. The size distribution at a given age is assumed uniform. The four parameters in the growth component of the model are the slope and intercept of the juvenile growth line, and the slope and intercept of the mature growth line.

The population is divided into eight yearly age classes. Each month the population is determined by multiplying the Leslie matrix by the eight element age vector of the previous month. The initial age vector is unknown. Each element of the initial age vector is therefore a parameter in the system identification problem.

In this model the elements of the Leslie matrix have very simple properties. The survival probabilities are assumed to be both age and density independent. Furthermore, a distinct functional form for the age-specific fecundity is assumed, namely that one and two year olds do not spawn, and that the fecundity of all older females is both age and density independent. Hence, two parameters are associated with the Leslie matrix--the survival probability and the fecundity.

So there are a total of fourteen parameters in the nonlinear least squares problem.

* 2, juvenile slope and intercept
* 2, mature slope and intercept
* 8, initial age distribution
* 1, survival probability
* 1, fecundity

The field data for this part of the study was that of Figure 1.3-1, the data for the entire lagoon. With 15 5 mm bins in each month, and with 10 months, the total number of data points to fit is (15)(10) = 150. The Jacobian J of Section 3.2 is therefore a 150 x 14 matrix.

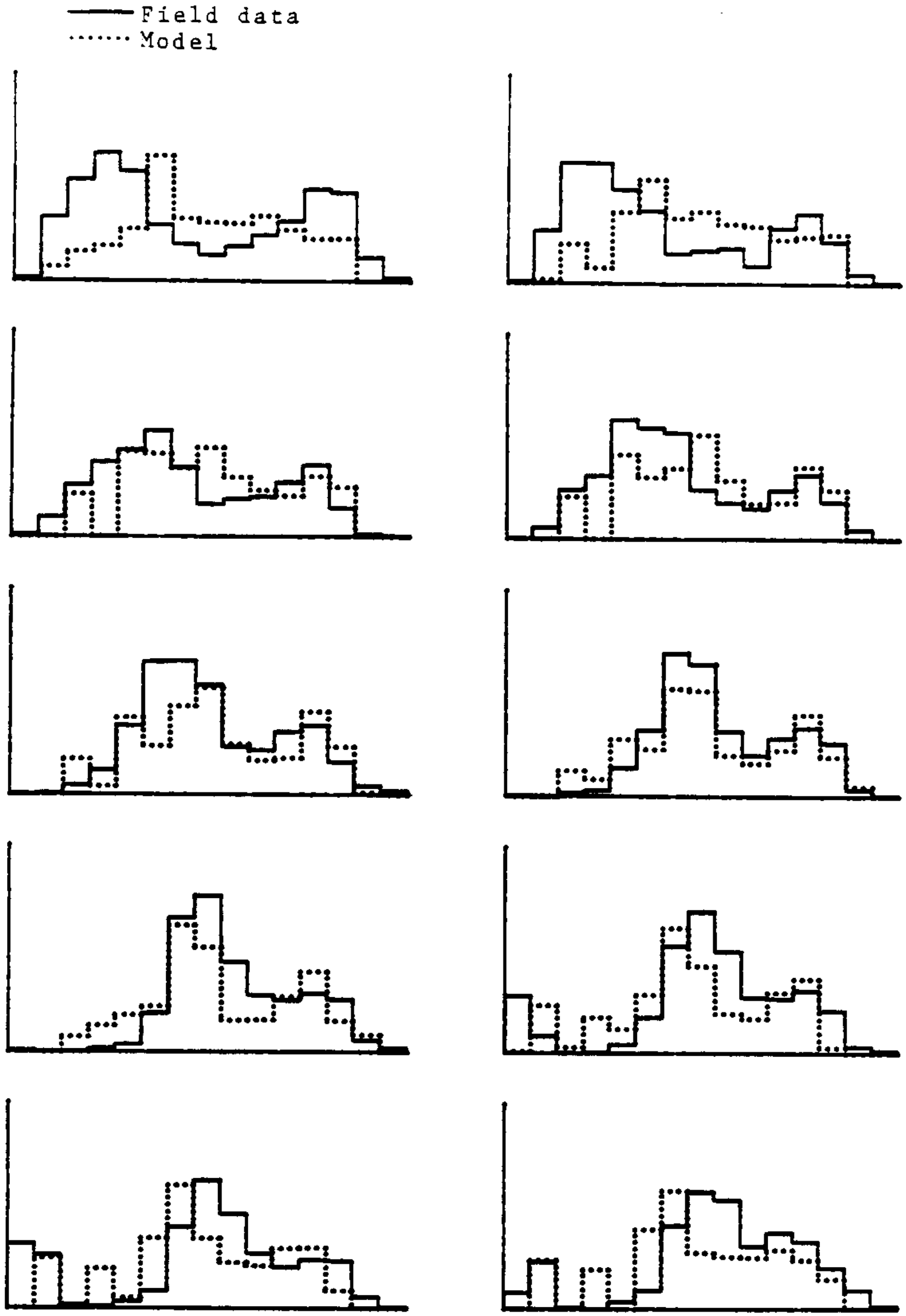
One other model parameter remains, the variance of the size for a given age. The nonlinear least squares problem was solved with this parameter fixed. Ten different computer runs were made with the standard deviation of the uniform distribution at 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, and 10.0 mm.

The results for 1.0, 3.0, 5.0, and 7.0 mm are shown in Figure 4.1-1. The field data are superimposed on the model results for ease of visual comparison. There is a marked difference between part (a) which assumed a standard deviation of only 1.0 mm, and parts (b), (c), and (d) which assumed a larger spread. The peaks in the model histogram of part (a) are sharp and have pronounced gaps between them. As expected, these gaps are filled in when a higher variance is assumed.

A cursory comparison of part (a) with the other parts would indicate that the fit for the 1.0 mm standard deviation is not as good. This is indeed the case as shown in Figure 4.1-2 which displays the squared error for all ten computer runs, along with the corresponding minimum least squares values of the four most interesting (biologically speaking) parameters. The minimum squared error was at a standard deviation of 3.0 mm. It was only slightly higher at 4.0, 5.0, 6.0, 7.0, and 8.0 mm, but it was substantially higher at 1.0, 2.0, 9.0, and 10.0 mm.

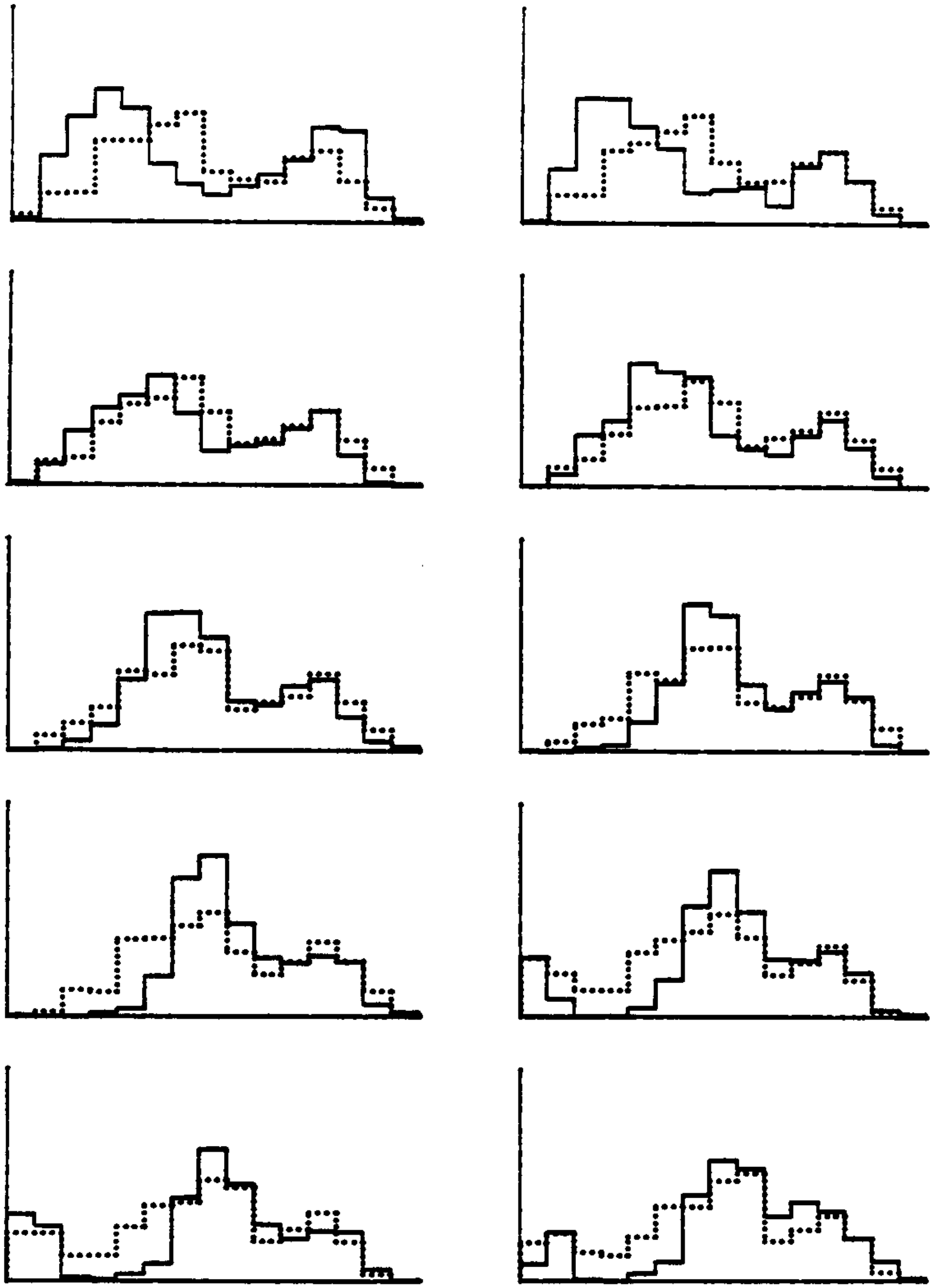
Part (a) of Figure 4.1-2 shows the survival probability for each computer run. It was very close to 0.99 in each case.

Part (b) shows the fecundity value from the Leslie



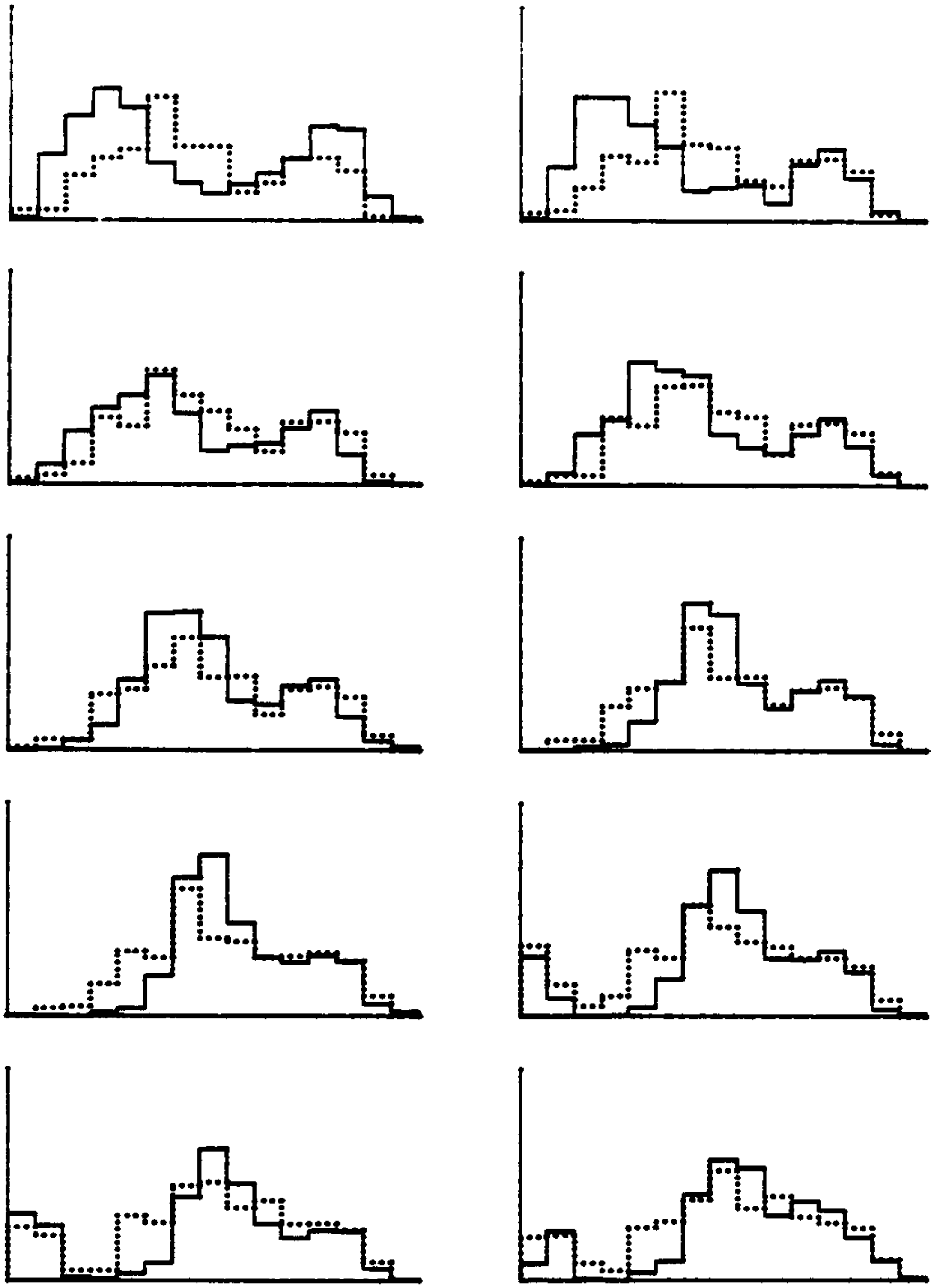
(a) Standard deviation = 1.0 mm.

Figure 4.1-1. The bilinear growth model.



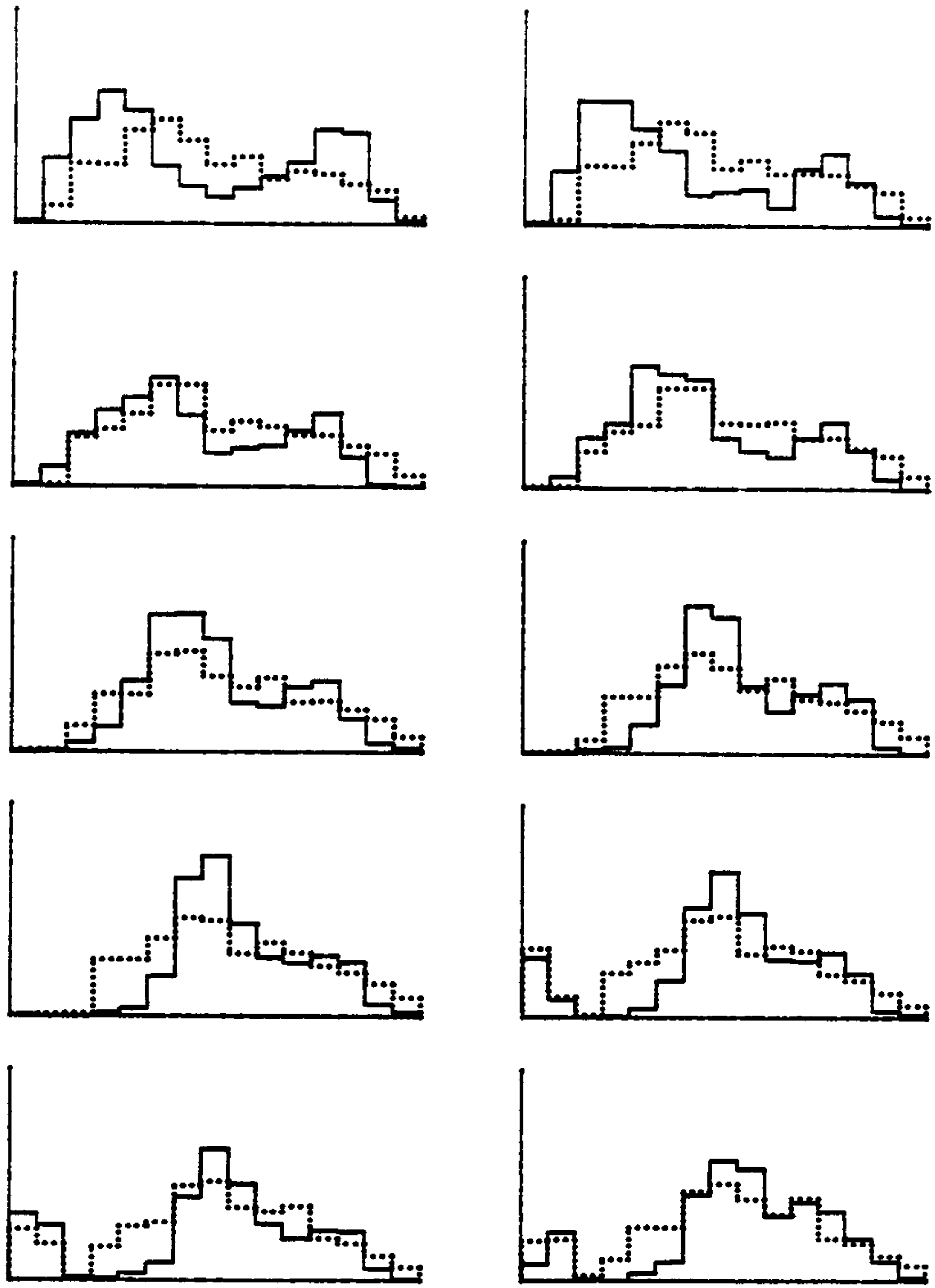
(b) Standard deviation = 3.0 mm.

Figure 4.1-1. (continued)



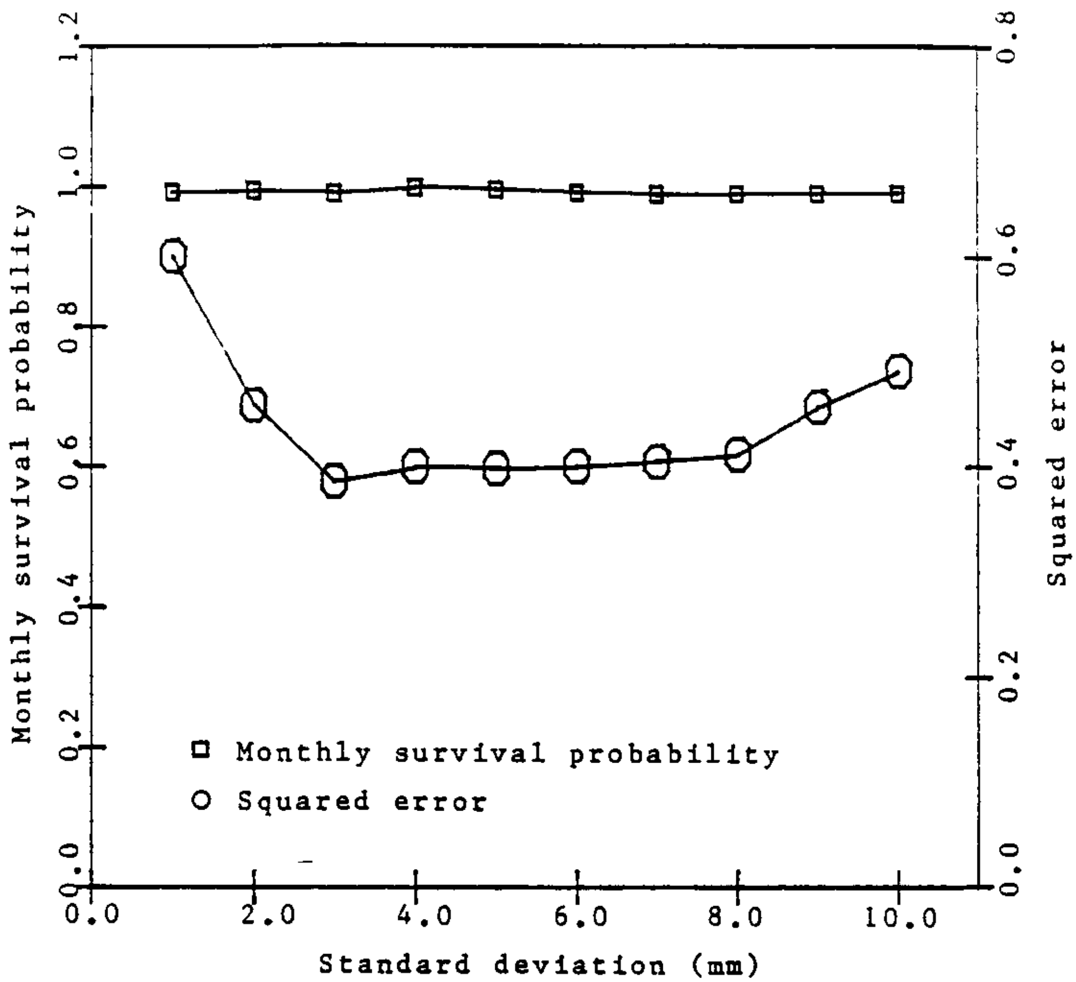
(c) Standard deviation = 5.0 mm.

Figure 4.1-1. (continued)



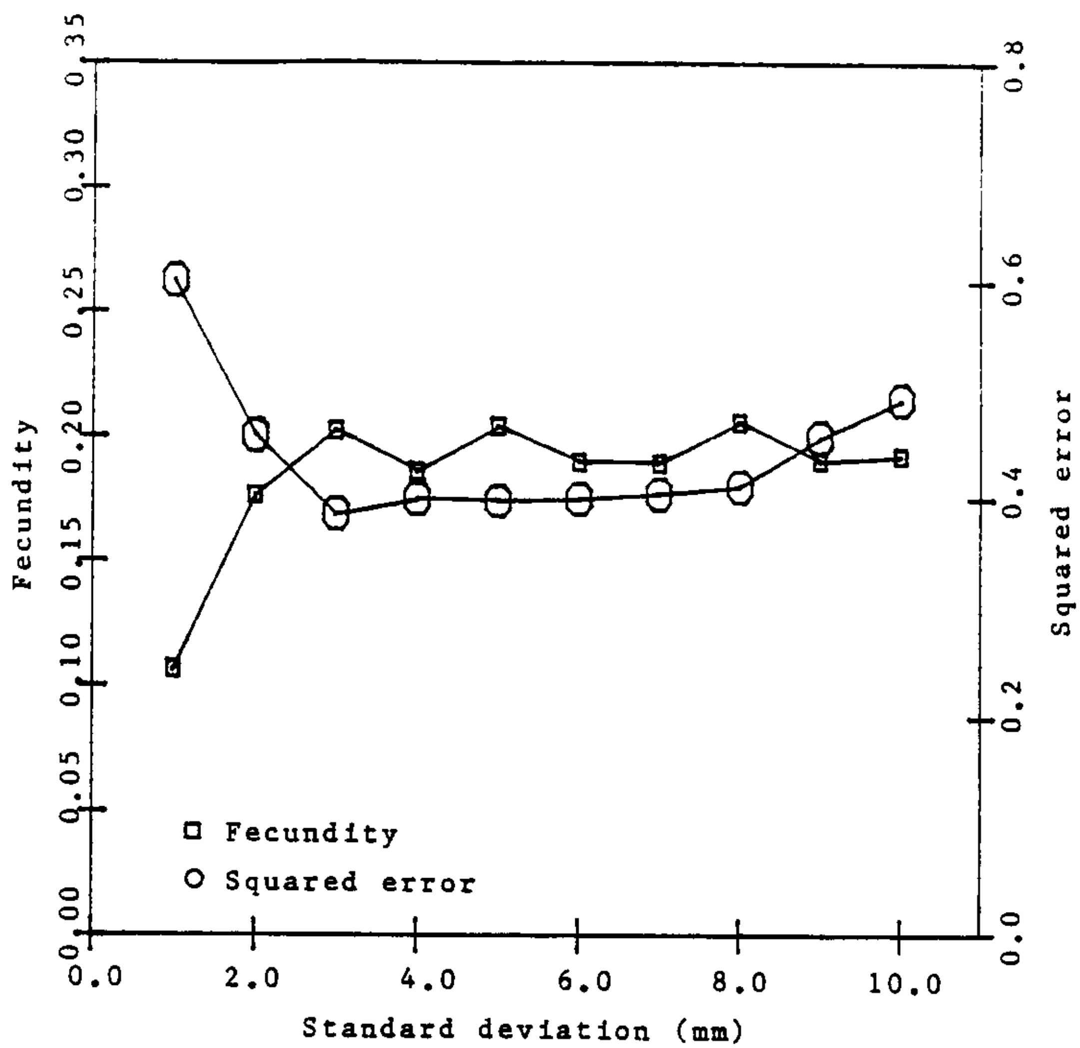
(d) Standard deviation = 7.0 mm.

Figure 4.1-1. (continued)



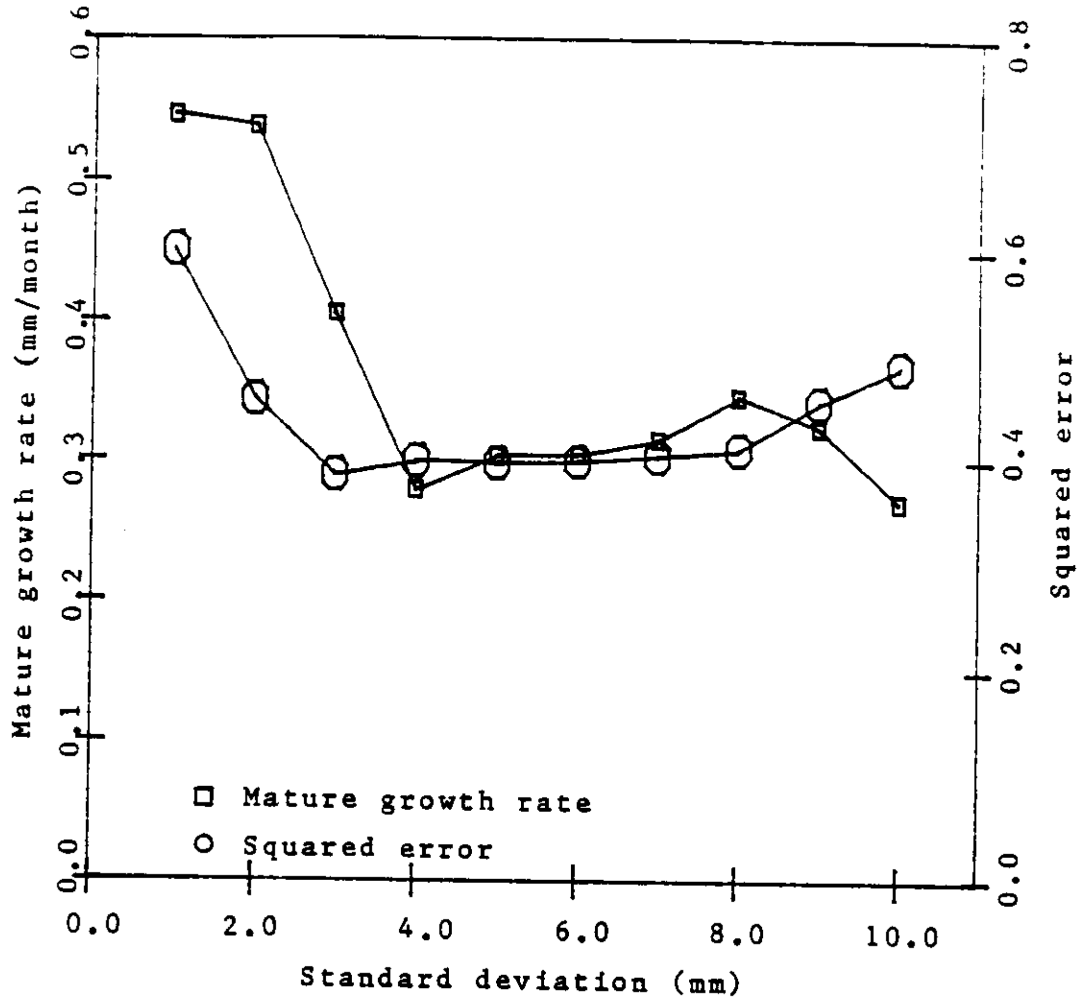
(a) Monthly survival probability.

Figure 4.1-2. The effect of growth variance on the bilinear model parameters.



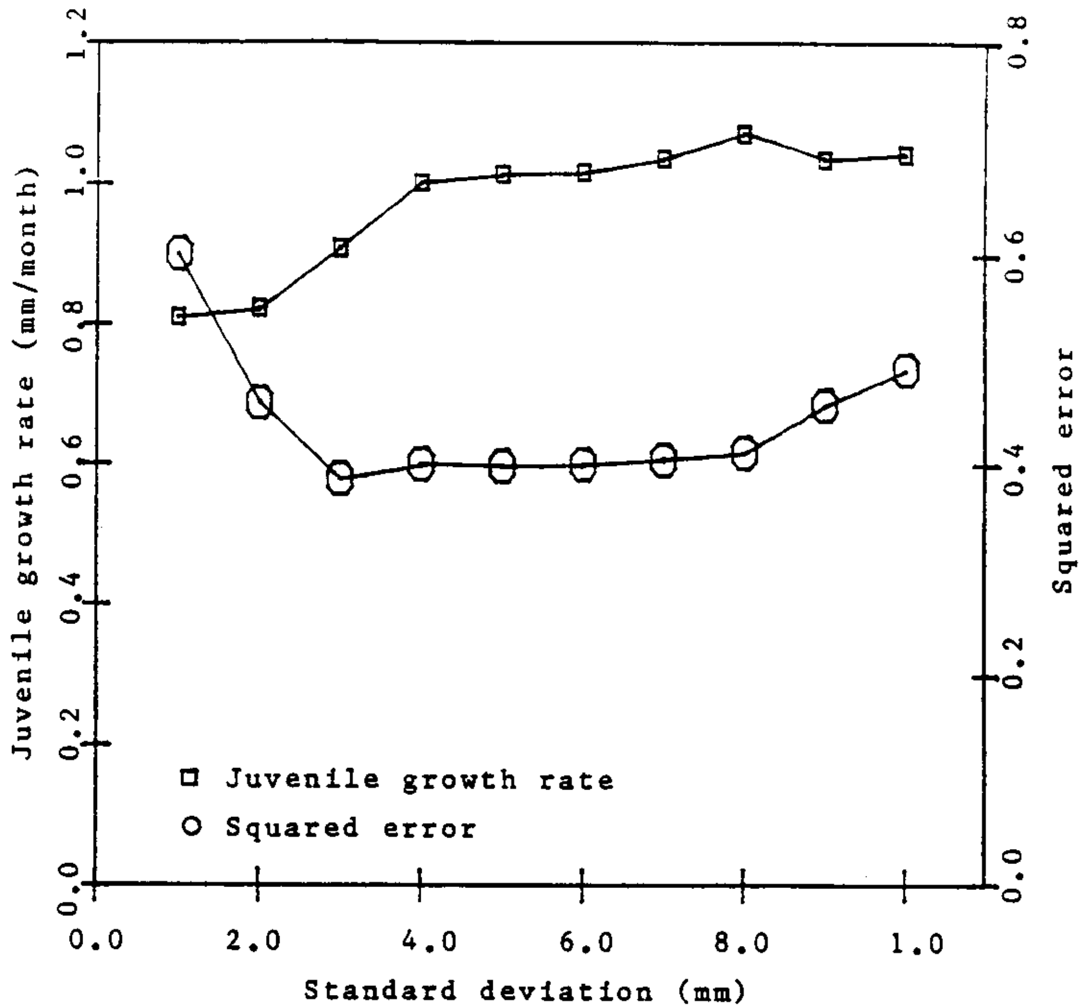
(b) Fecundity

Figure 4.1-2. (continued)



(c) Mature growth rate.

Figure 4.1-2. (continued)



(d) Juvenile growth rate.

Figure 4.1-2. (continued)

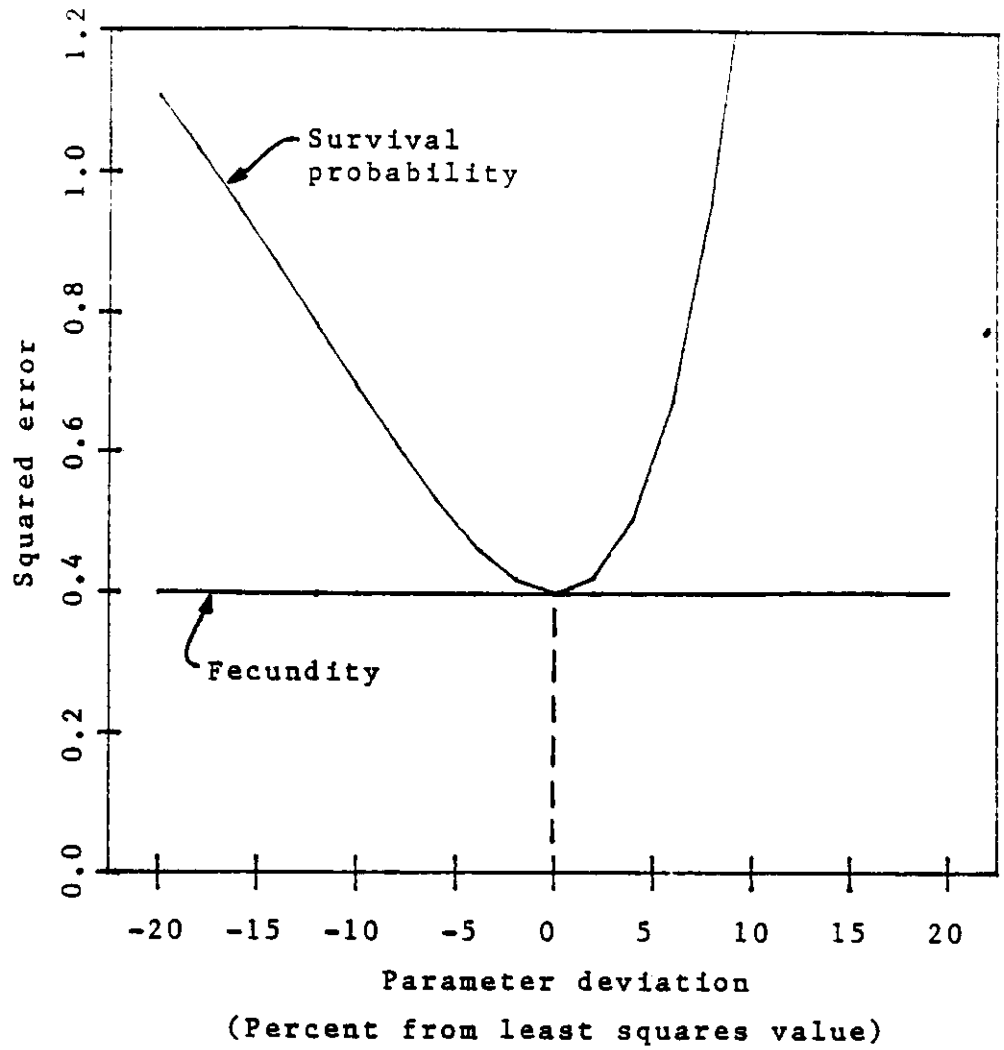
matrix. Its average value over the computer runs at 3.0, 4.0, 5.0, 6.0, 7.0, and 8.0 mm is 0.20 offspring who survive to the first age class per individual. Assuming a 50/50 ratio of male to female in the population, the fecundity is 0.40 offspring who survive to the first age class per female.

Part (c) shows the growth rate for mature individuals. Its value for the 3.0 mm computer run was significantly higher than for the 4.0, 5.0, 6.0, 7.0, and 8.0 mm runs. Its average value over the 4.0 through 8.0 mm runs is 0.31 mm/month.

Part (d) shows the growth rate for juveniles. As with the mature growth rate, its value for the 3.0 mm run was significantly different from the 4.0 through 8.0 mm runs. Its average over the latter group is 1.03 mm/month.

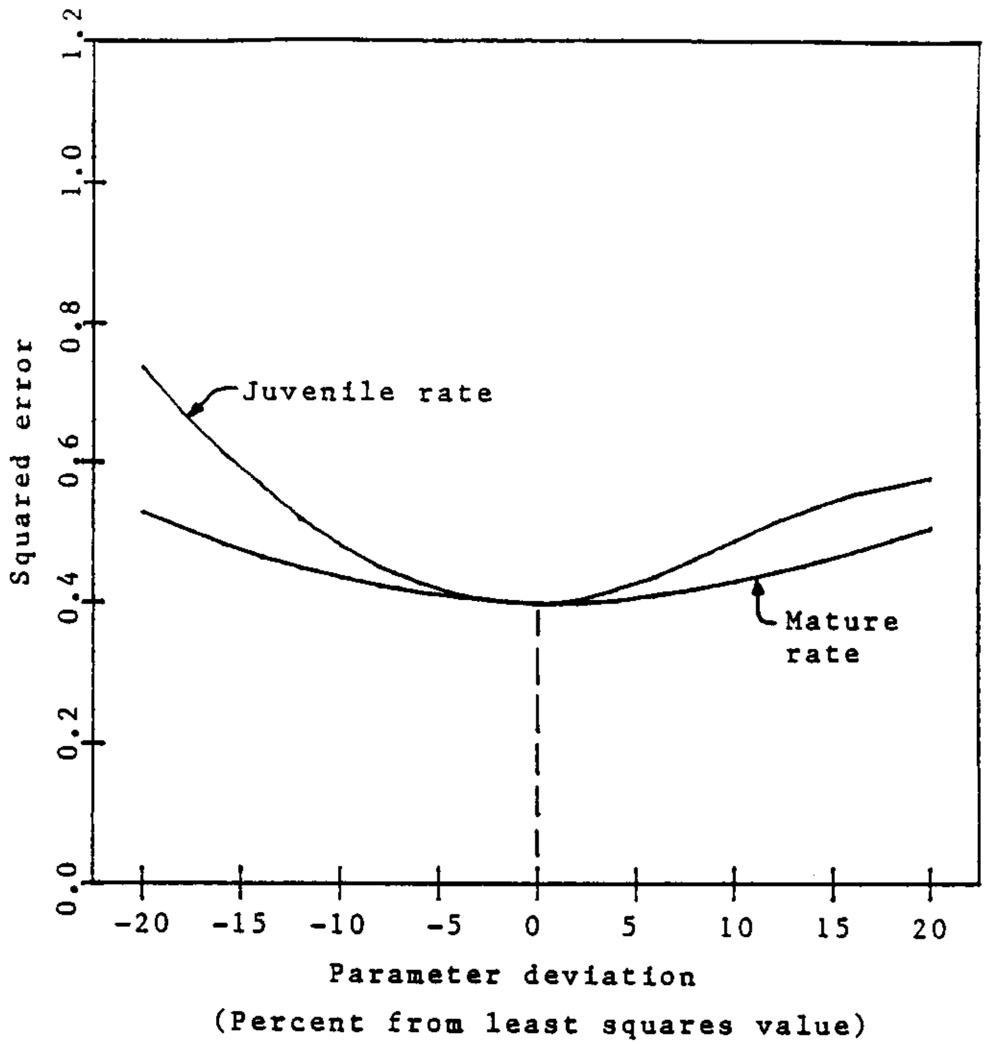
A sensitivity analysis was performed on these four parameters. It was done numerically by fixing the values of all the parameters at their least squares values, then computing the squared error when the parameter in question was varied over a range of ±20%. The sensitivity analysis was performed with the optimal values from the 5.0 mm computer run.

Figure 4.1-3 (a) shows the sensitivity analysis for the survival probability and the fecundity parameters. The model is very sensitive to changes in the survival probability. A decrease of 15% in that parameter value



(a) Leslie parameters.

Figure 4.1-3. Parameter sensitivity in the bilinear growth model.



(b) Growth parameters.

Figure 4.1-3. (continued)

will more than double the squared error. However the model appears to be very insensitive to the fecundity value.

A moment's reflection shows why. The fecundity value is responsible for the recruitment during the eighth month of the simulation. During that month and the remaining two months only the first three bins in the histogram are affected by the juveniles that are recruited into the population as a result of that fecundity value. That is a total of 9 bins out of 150 bins on which the squared error is calculated. Since only 6% of the bins are affected by the parameter value, the squared error must be relatively insensitive to its value. We will return to the question of assessing the significance of the fecundity in a later section.

Figure 4.1-3 (b) shows the sensitivity of the model to the growth rates. They lie between the two extremes of the survival probability and the fecundity of part (a), with the model being more sensitive to the juvenile growth rate than to the mature growth rate.

Figure 4.1-4 shows the estimated growth curves for each of the ten computer runs. They were plotted from the optimal values of the slope and intercept of the juvenile and mature growth lines. The curves for the extreme values of the standard deviation lie outside the grouping of the 3.0 through 9.0 mm curves. The transition from

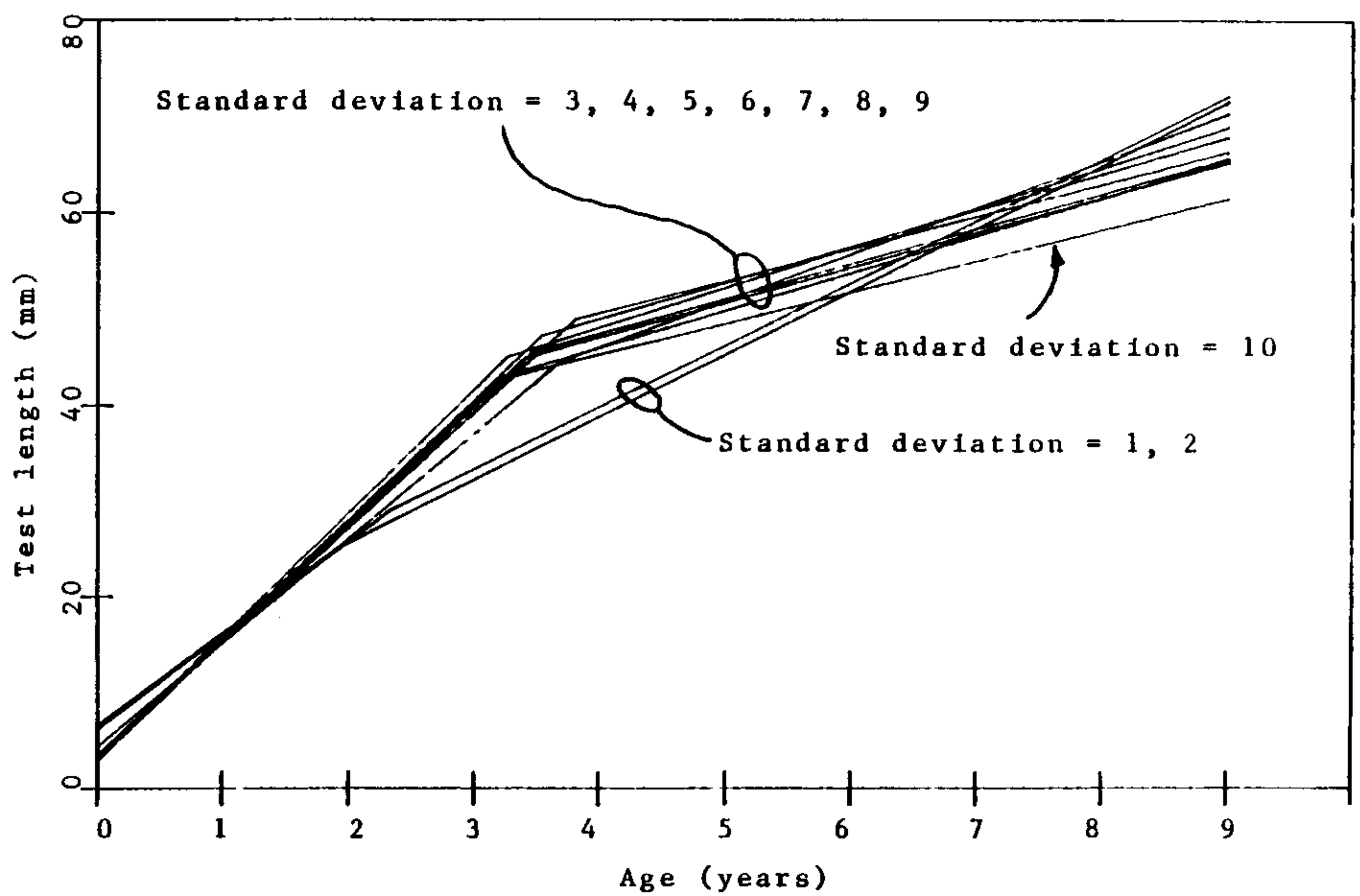


Figure 4.1-4. Optimal growth curves for the bilinear model.

juvenile growth rate to mature growth rate is between 3 and 4 years of age.

An attempt was made to find the optimal value of variance of the uniform size distribution. The variance was included as a fifteenth parameter in the minimization routine. The attempt was not successful. The problem is that there are many local minima to which the algorithm will converge depending on the specific starting values of the parameters. This is not surprising considering the range of almost equal values of the squared error as a function of standard deviation as shown in Figure 4.1-2. This is not such a critical problem in the estimation of the parameters since Figure 4.1-2 shows the estimated values to be fairly independent of the assumed standard deviation within the range of 4.0 to 7.0 mm.

4.2 Spline growth fit

Section 2.4 noted the possibility of assessing the the validity of the model by direct comparison with the field data. The spline fit growth model was motivated by such a comparison.

In figure 4.1-1, based on the bilinear growth assumption, the major peak on the right which consists of older individuals is pretty well matched with the model.

But the major peak on the left is not so well matched. For these younger individuals, the best the model can do is overestimate the number of larger individuals in the early months of the simulation, and underestimate the number of larger individuals in the later months. On the sequence of histograms the peak from the field data gradually "overtakes" the peak from the model.

So to improve the structure of the model we need a way to let the younger individuals grow at a faster rate without affecting the rate of growth of either the juveniles or the adults. The bilinear growth model contained four growth parameters, juvenile slope and intercept and mature slope and intercept. The spline fit model adds one more degree of freedom in the growth component of the model.

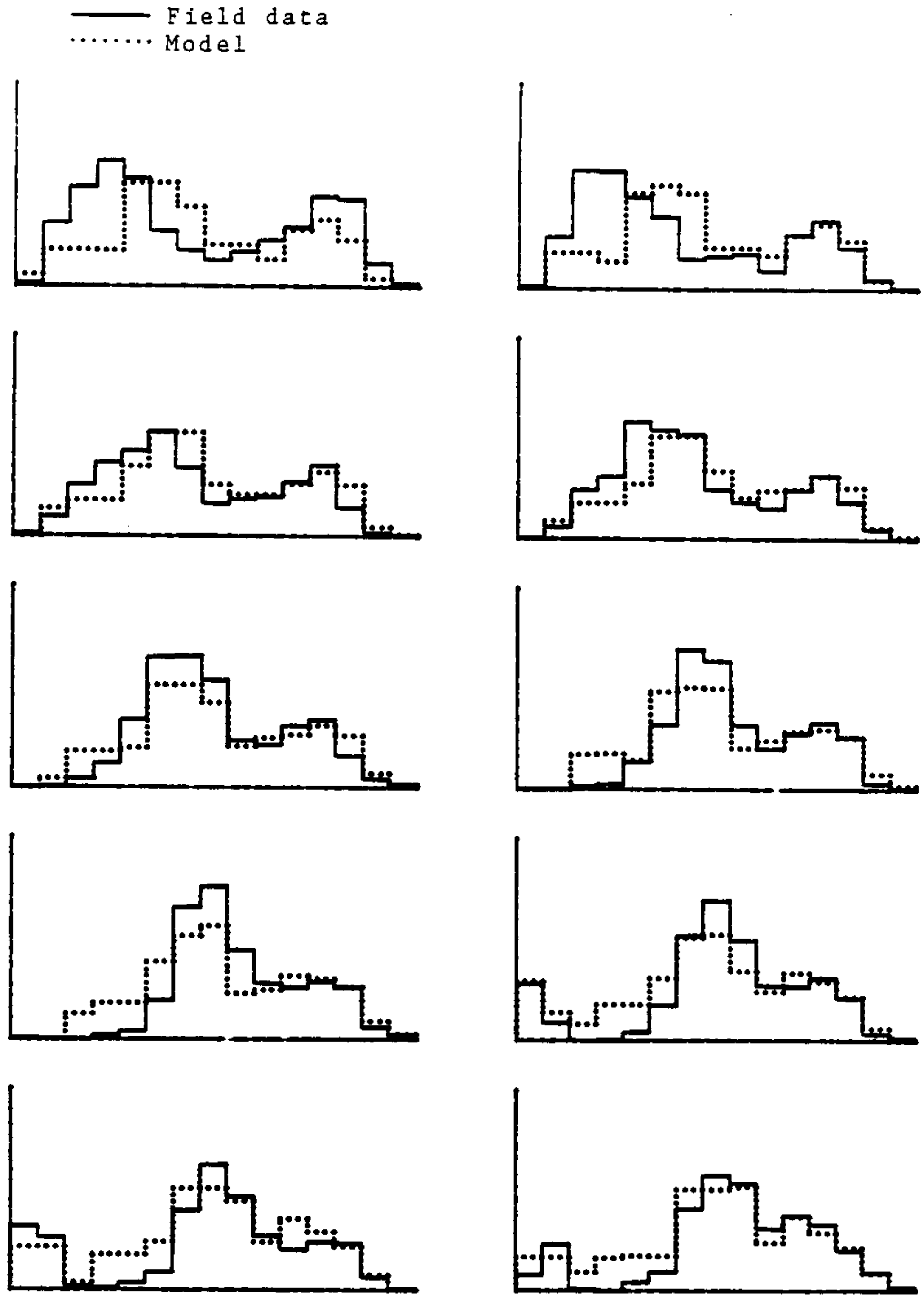
The idea is to use the method of cubic splines [Vand83] to give the size versus age relationship a curvilinear nature. Five degrees of freedom were obtained by placing five spline knots at equally spaced intervals of age from 0 to 8 years. That is, the knots were at the 0, 2, 4, 6, and 8 year points. The size value of each knot was allowed to vary as a parameter in the minimization. Sizes between the knots were then given by the interpolating cubic polynomial. Hopefully, adding a degree of freedom will produce a better overall fit of the model to the field data.

The additional assumption of finite variance as shown in Figure 2.3-7 was made. The only difference is that the growth relationship is now curvilinear instead of piecewise linear. The size distribution at a given age is again assumed uniform. All other aspects of the model are unchanged from the bilinear model.

The Jacobian J of Section 3.2 is now a 150 x 15 matrix. The nonlinear least squares problem was solved with the growth variance parameter fixed. Ten different computer runs were made with the standard deviation of the uniform distribution at 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, and 10.0 mm.

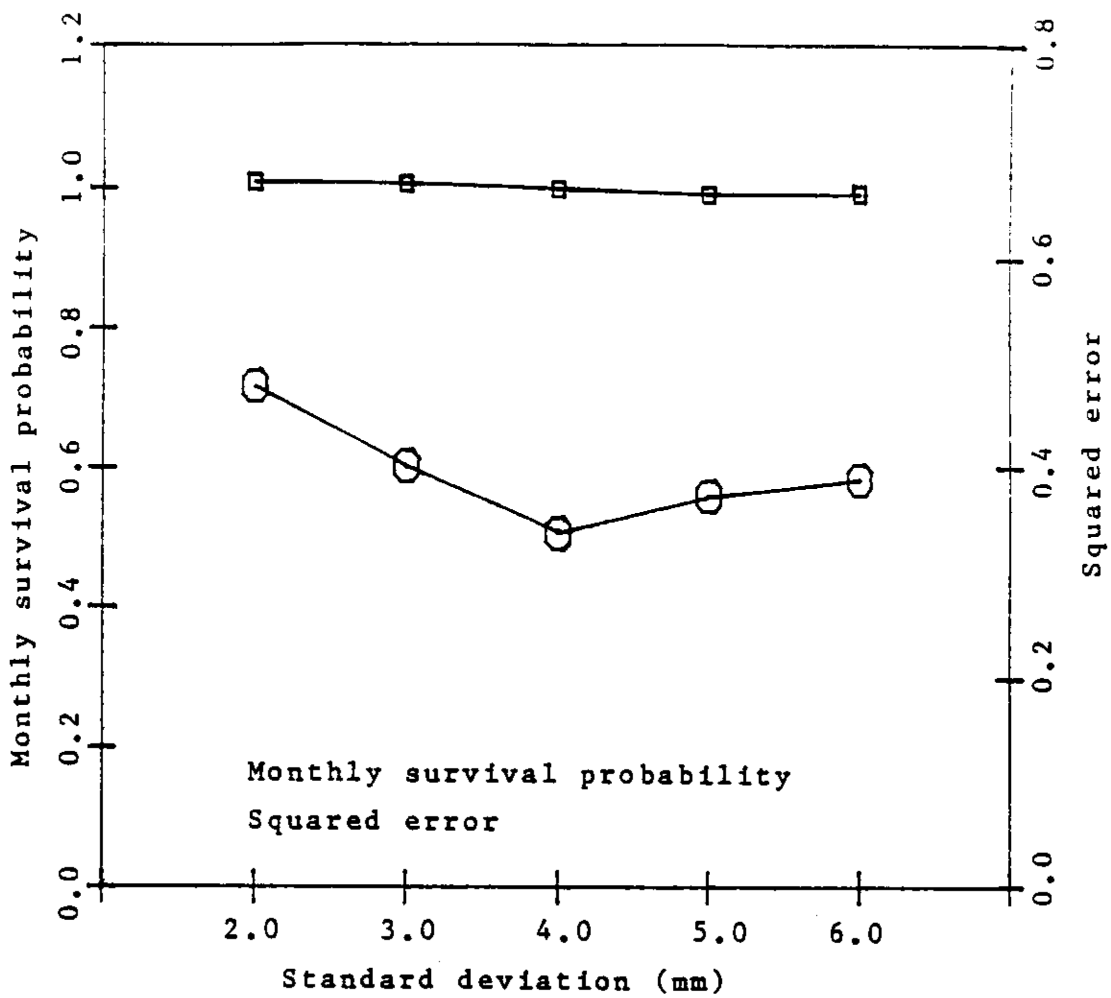
The lowest squared error occurred at a standard deviation of 4.0 mm. The results for that case are shown in Figure 4.2-1. The inclusion of an extra degree of freedom in the growth component of the model is having the desired effect. In months 5 through 10 the large peak on the left tracks the field data much more closely without affecting the good fit of the more mature individuals.

Figure 4.2-2 displays the squared error for five of the ten computer runs, along with the corresponding minimum least squares values of the survival probability and the fecundity. The scale is identical to that of Figure 4.1-2 for ease of comparison. The value of the minimum squared error is 0.34 compared with 0.40 in the binliear model.



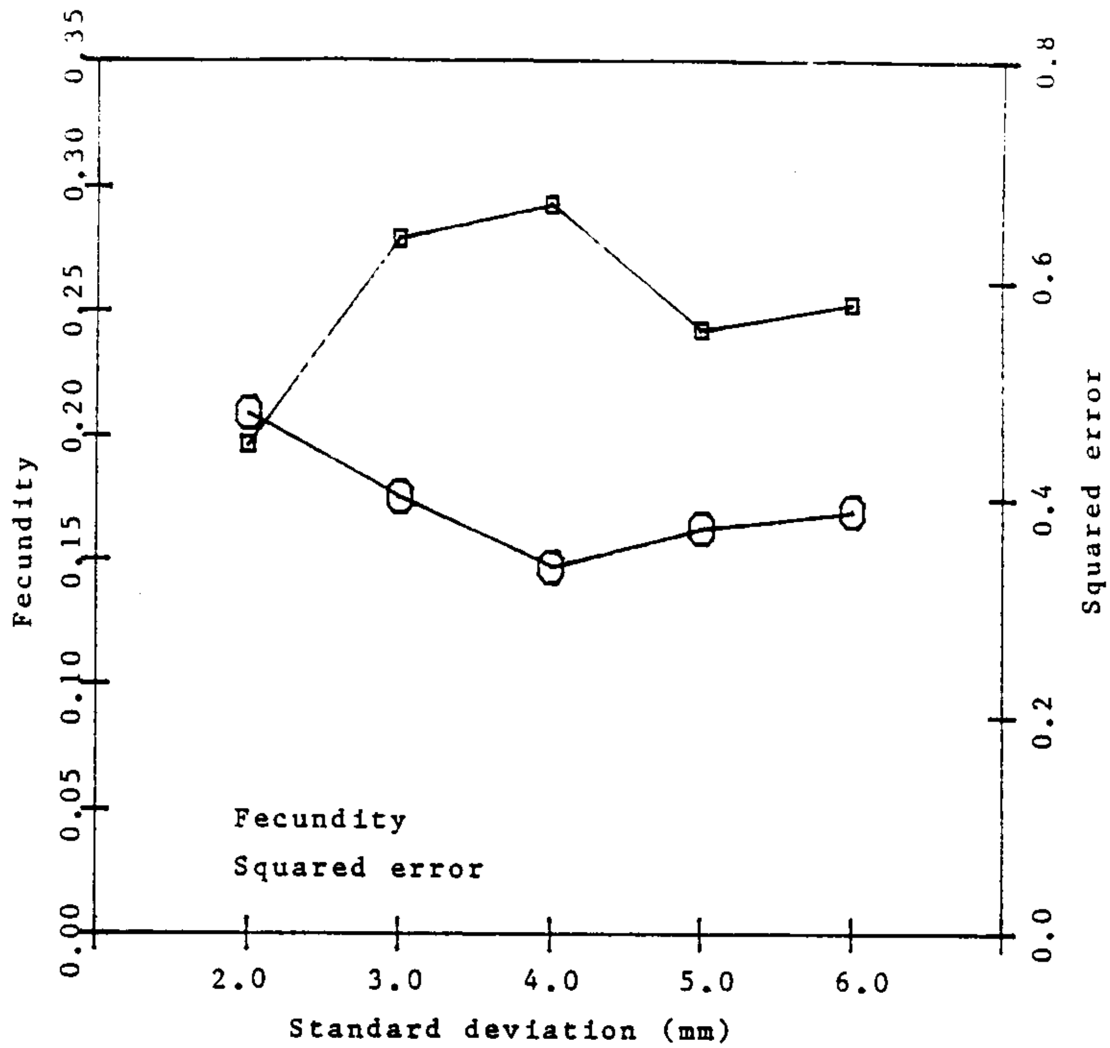
Standard deviation = 4.0 mm.

Figure 4.2-1. The spline fit growth model.



(a) Monthly survival probability

Figure 4.2-2. The effect of growth variance on the spline fit model parameters.



(b) Fecundity

Figure 4.2-2. (continued)

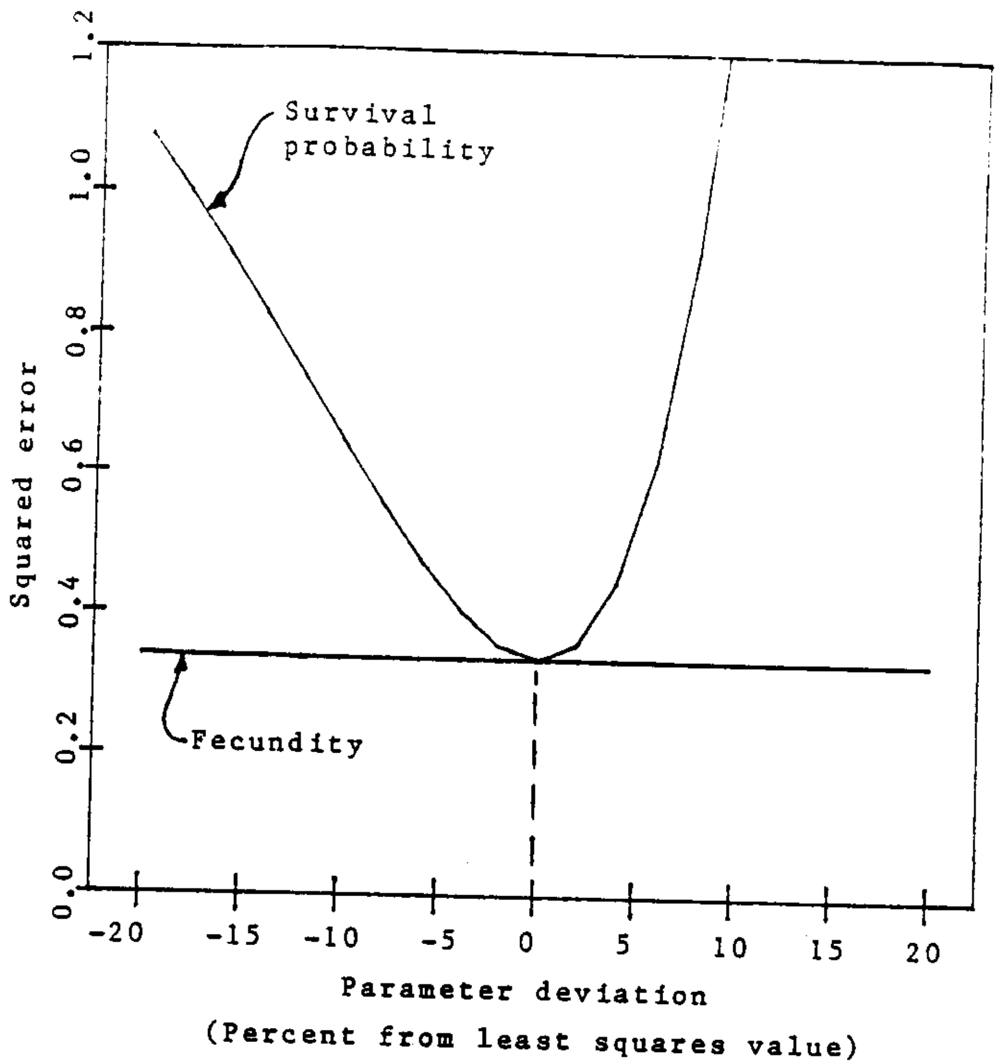
Part (a) of Figure 4.1-2 shows the survival probability. It was again very close to 0.99 in each case.

Part (b) shows the fecundity value from the Leslie matrix. Its value at 4.0 mm is 0.29 offspring who survive to the first age class per individual. Assuming a 50/50 ratio of male to female in the population, the fecundity is 0.58 offspring who survive to the first age class per female.

A sensitivity analysis was performed on the Leslie parameters and the growth parameters at 4.0 mm standard deviation. Figure 4.2-3 (a) shows the same behavior of the sensitivity to the survival probability and the fecundity as in the bilinear model.

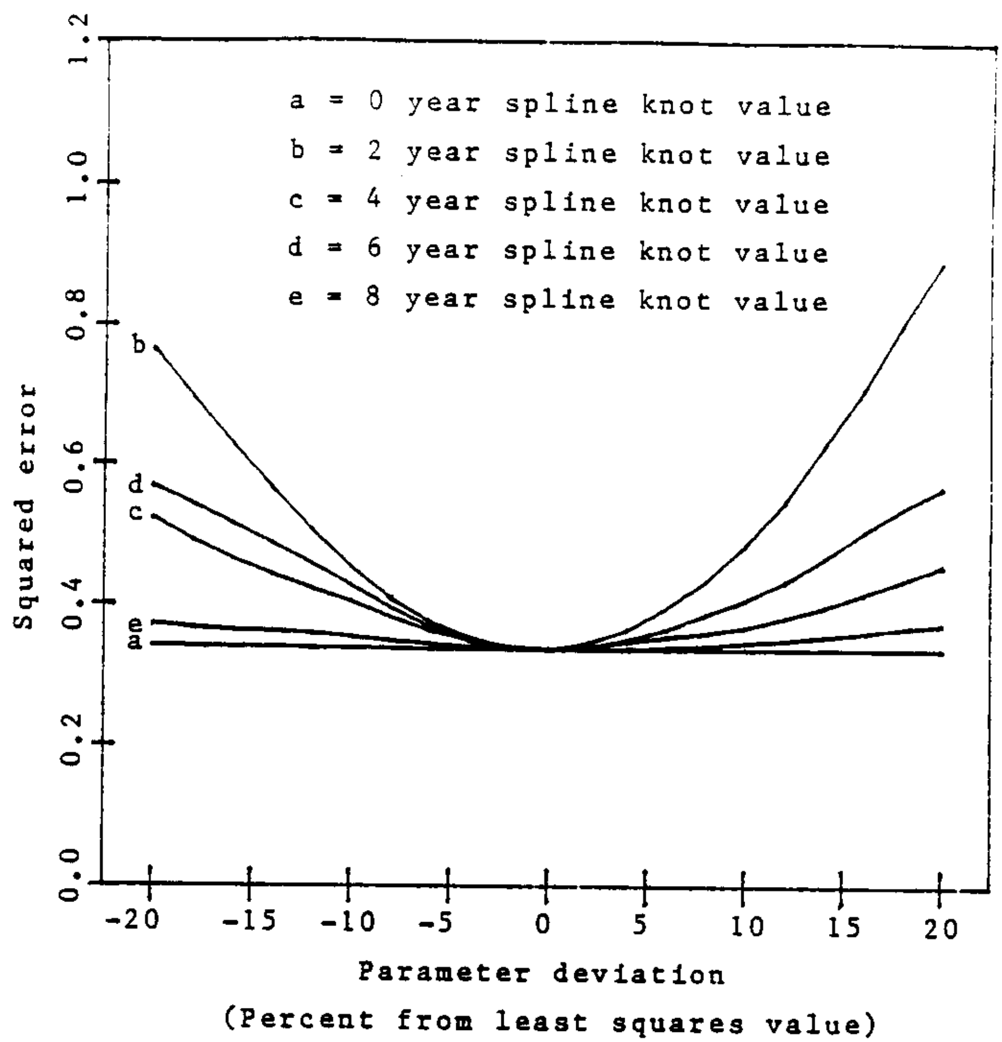
Part (b) of the figure shows the sensitivity to the spline knot values. The squared error is not as sensitive to the knot values at the endpoints (curves a and e in Figure 4.2-3 (b)) as it is to the knot values at the interior points. That is to be expected since the interior knots are in the "middle of the data" and therefore affect the fit to a greater degree than the end knots.

Figure 4.2-4 shows the estimated growth curves for five of the ten computer runs. They were plotted from the optimal values of the spline knots and show the cubic polynomials of the fit.



(a) Leslie parameters

Figure 4.2-3. Parameter sensitivity in the spline fit growth model.



(b) Growth parameters

Figure 4.2-3. (continued)

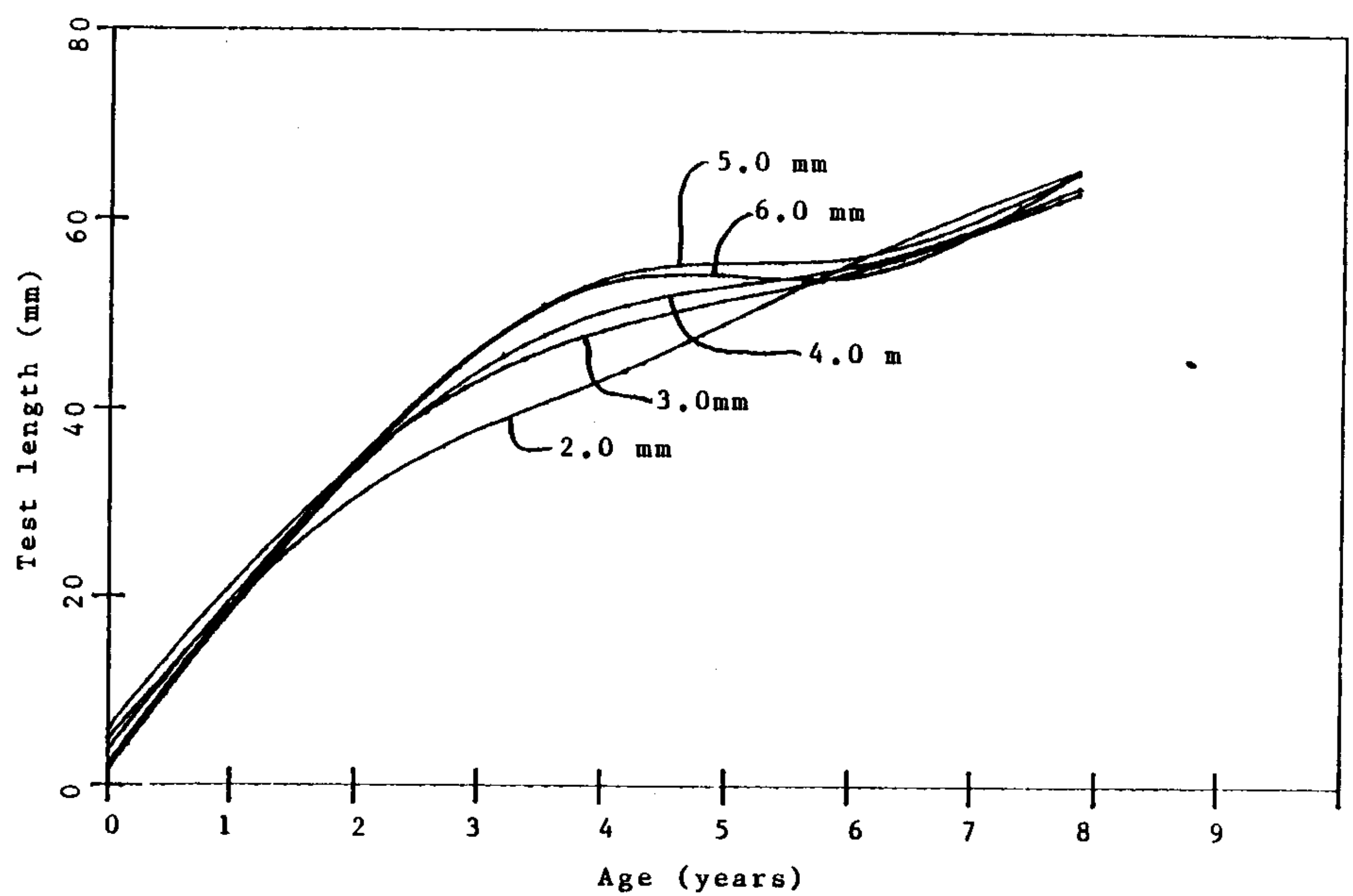


Figure 4.2-4. Optimal growth curves for the spline fit model.

The software which generated the spline fit growth curve was implemented in a general way so that n equally spaced spline knots could be placed between ages 0 and 8 years. In addition to the 5 spline case, computer runs were made with 6, 7, 8, 9, and 10 equally spaced splines.

The results were less than satisfactory for two reasons. First, one would expect the minimum squared error to decrease as the number of degrees of freedom in the model increases. Although this was generally true it did not always hold. Sometimes by adding an extra spline the minimum squared error actually increased, albeit slightly.

Another problem was the physically unrealistic shape of the estimated growth curves. As splines are added they create wiggles that are an artifact of the model.

4.3 Trilinear growth

The trilinear growth model was motivated by the problem of the unrealistic characteristics of the growth curves produced by the spline fit models.

The idea is to increase the number of degrees of freedom in the growth component, not by increasing the number of splines in a curvilinear fit, but by increasing the number of segments in a piecewise linear fit.

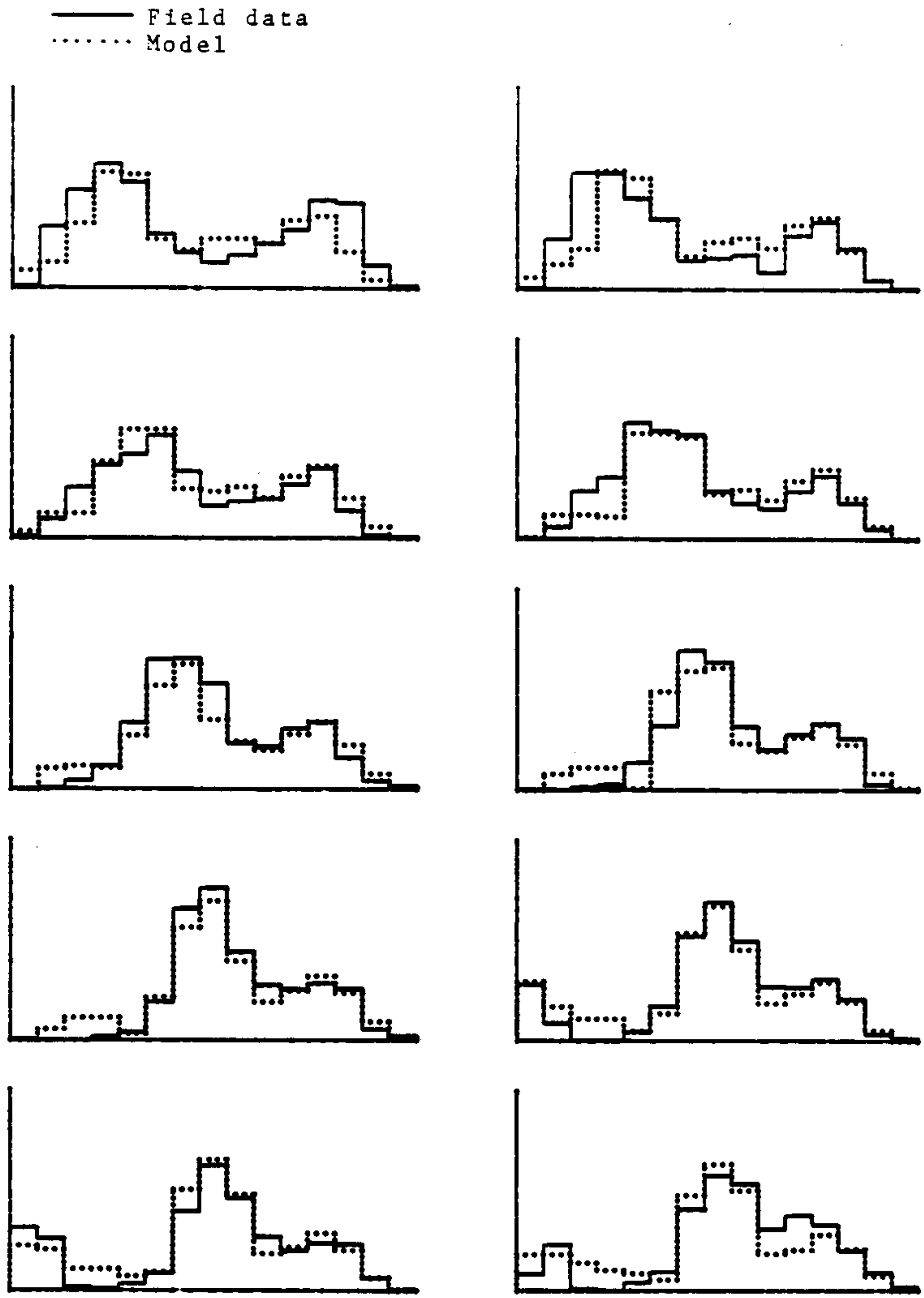
Six degrees of freedom in the growth component were obtained by assuming that growth occurs in three stages - juvenile, midlife, and mature - with a slope and intercept for the line at each stage.

The additional assumption of finite variance as shown in Figure 2.3-7 was made. The only difference is that the growth relationship is now trilinear instead of bilinear. The size distribution at a given age is again assumed uniform. All other aspects of the model are unchanged from the bilinear model.

The Jacobian J of Section 3.2 is a 150 x 16 matrix. The nonlinear least squares problem was solved with the growth variance parameter fixed. Ten different computer runs were made with the standard deviation of the uniform distribution at 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, and 10.0 mm.

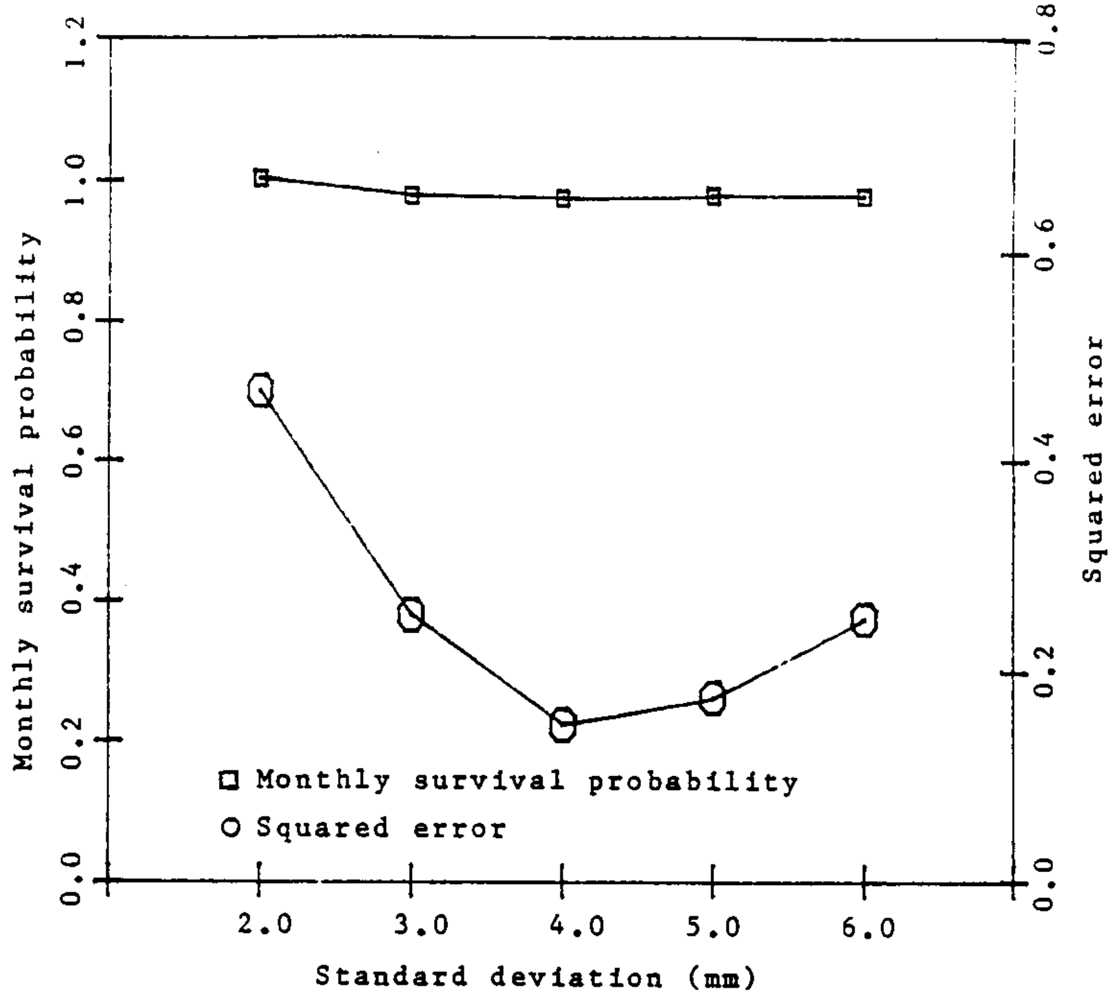
As before, the lowest squared error occurred at a standard deviation of 4.0 mm. The results for that case are shown in Figure 4.3-1. Now the model tracks the field data exceptionally well compared with the previous models. In particular, both major peaks are accounted for by the model throughout the simulation.

Figure 4.3-2 displays the squared error for five of the ten computer runs, along with the corresponding minimum least squares values of the survival probability, the fecundity, and the growth rates. The scale is



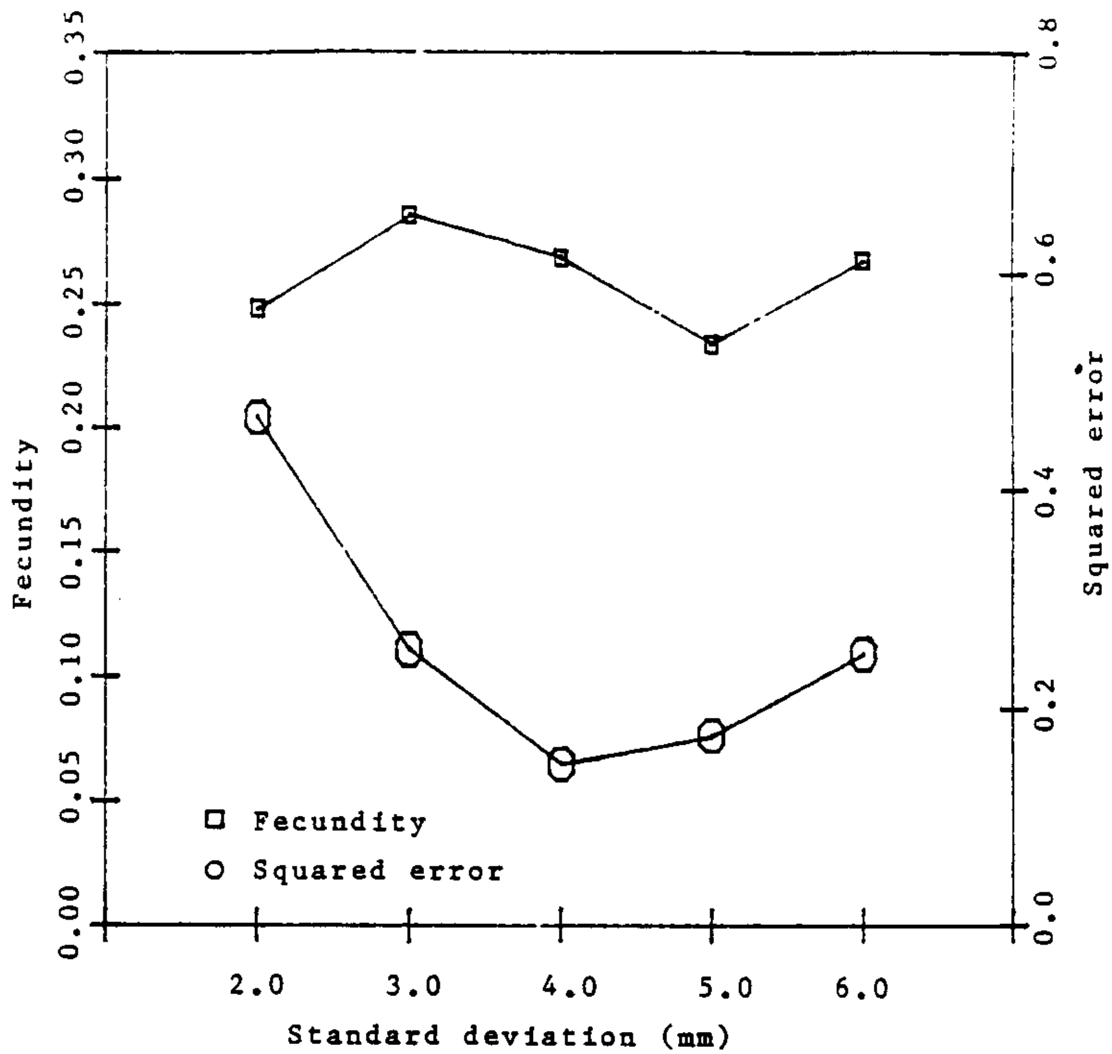
Standard deviation = 4.0 mm.

Figure 4.3-1. The trilinear growth model.



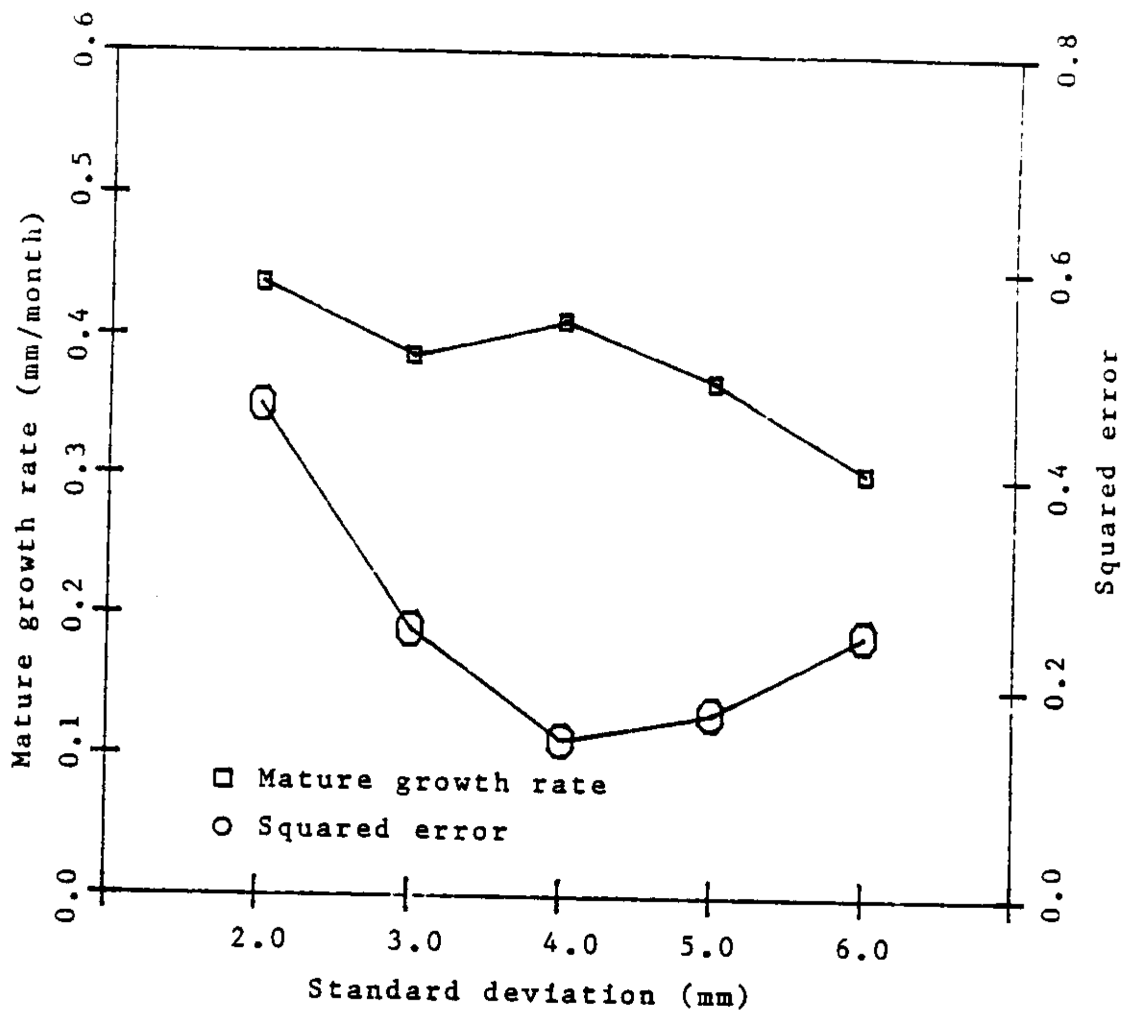
(a) Monthly survival probability

Figure 4.3-2. The effect of growth variance on the trilinear growth model parameters.



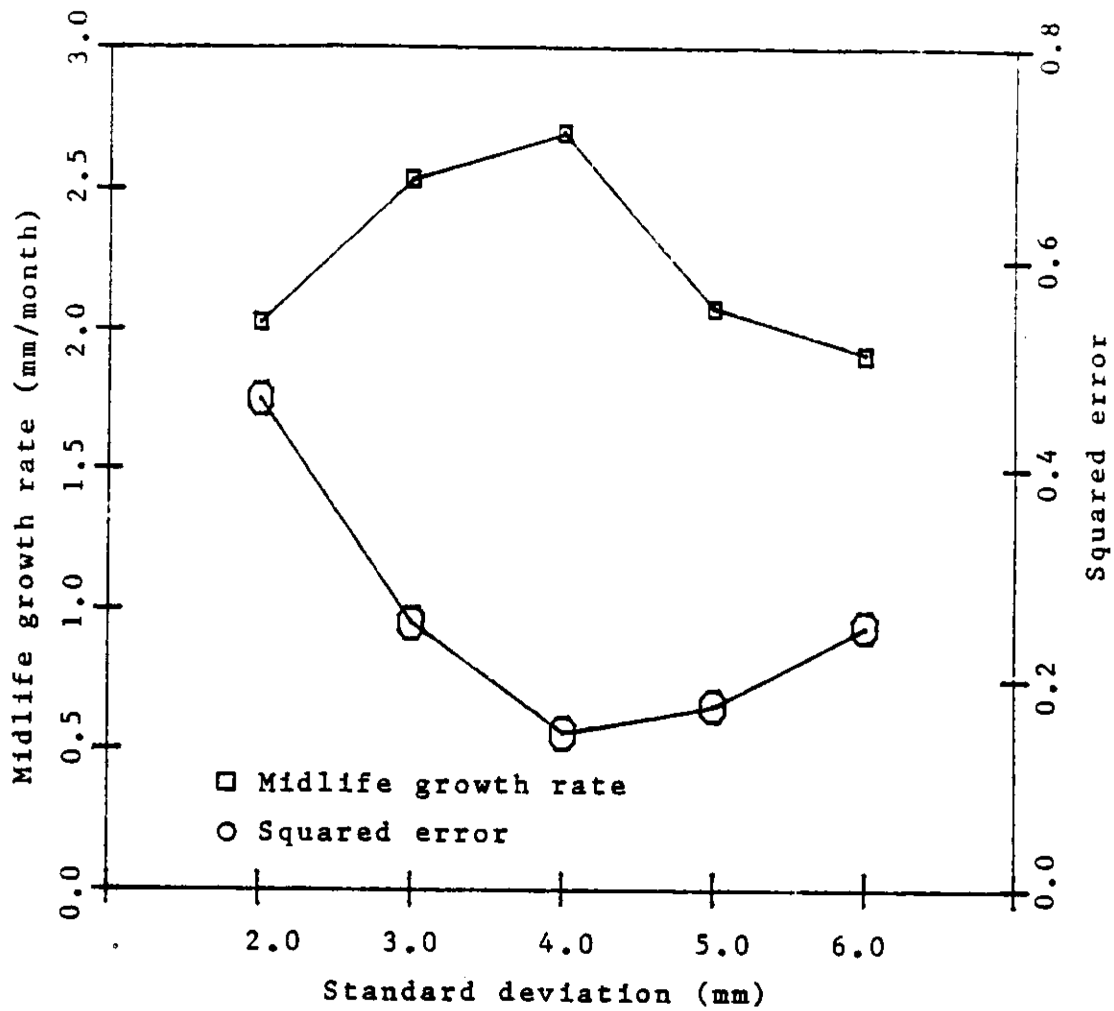
(b) Fecundity

Figure 4.3-2. (continued)



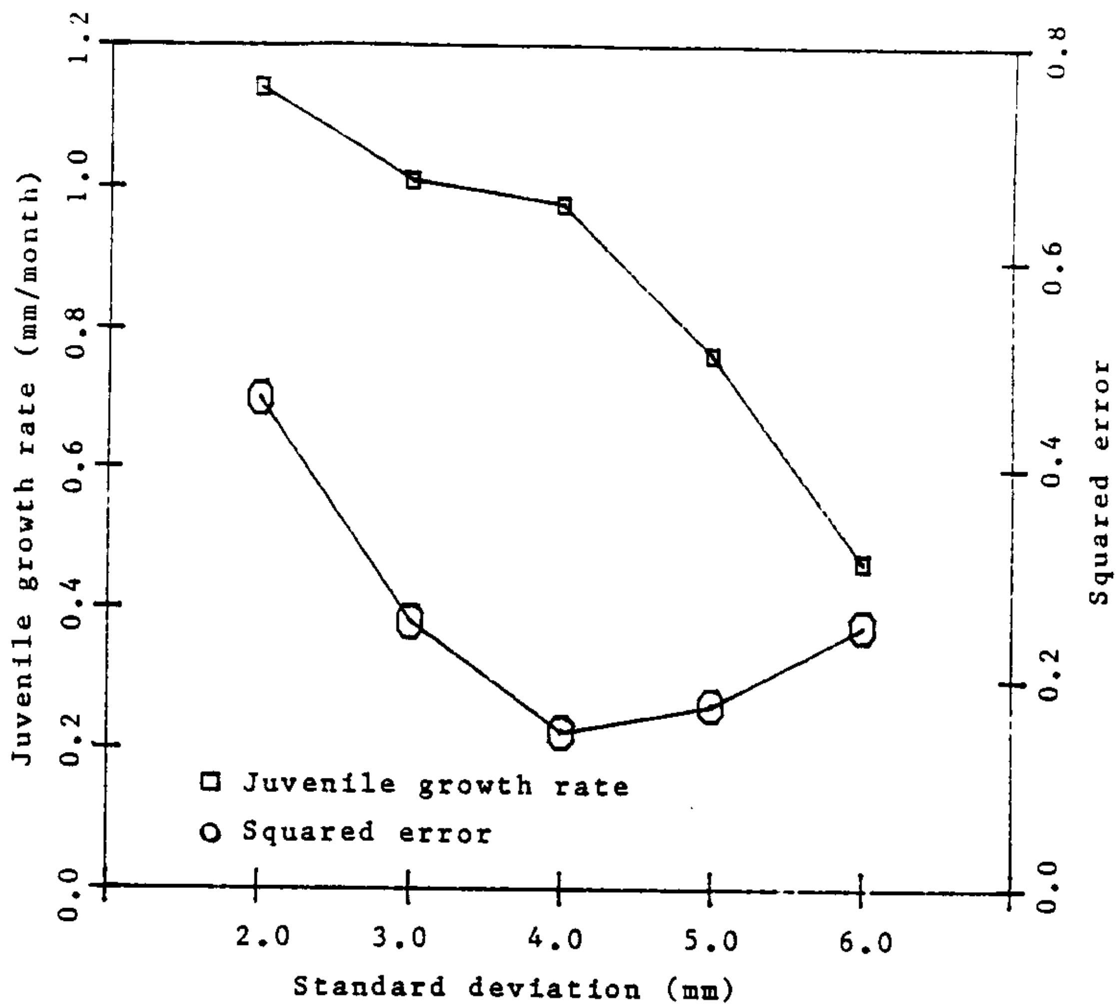
(c) Mature growth rate

Figure 4.3-2. (continued)



(d) Midlife growth rate

Figure 4.3-2. (continued)



(e) Juvenile growth rate

Figure 4.3-2. (continued)

identical to that of Figures 4.1-2 and 4.2-2 for ease of comparison. The value of the minimum squared error is 0.15 compared with 0.34 in the spline fit model and 0.40 in the bilinear model.

Part (a) of Figure 4.3-2 shows the survival probability for each computer run. Its value is 0.975 in the best (4.0 mm) fit case.

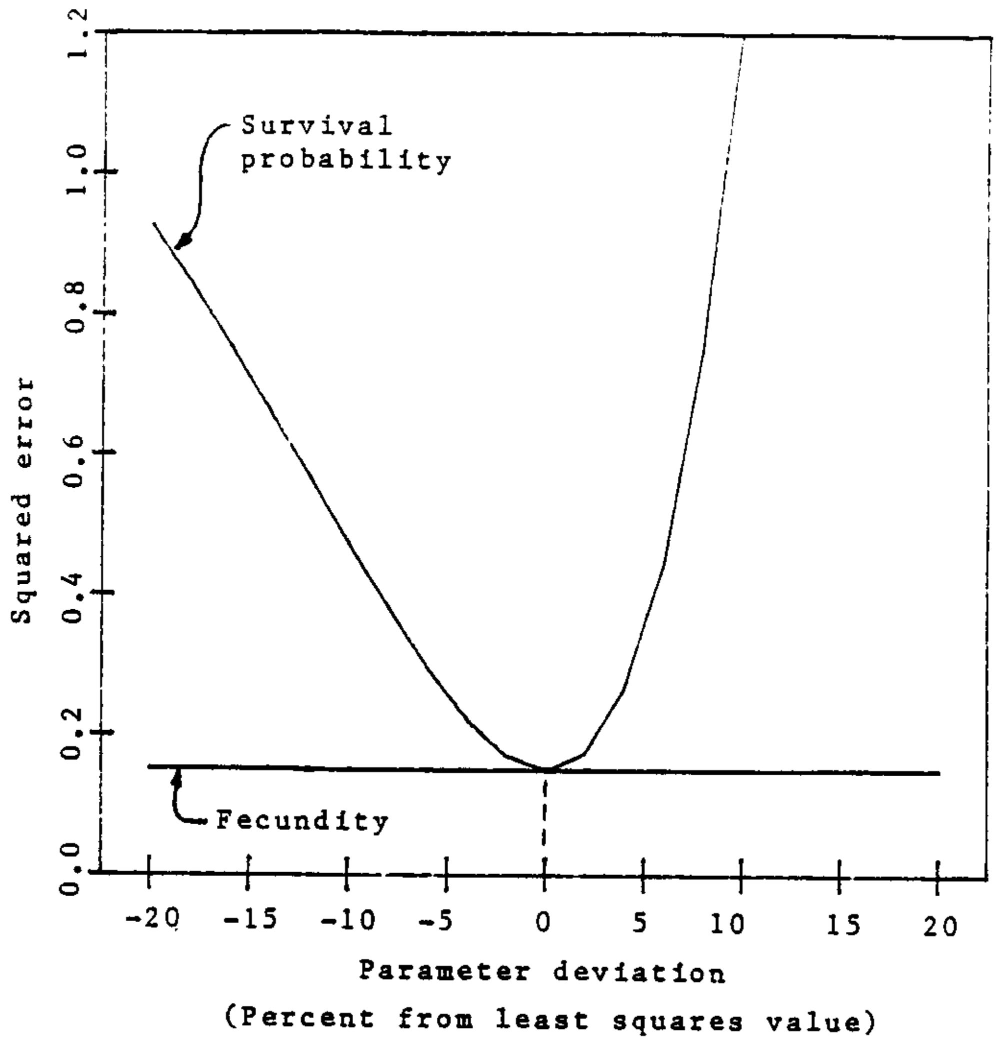
Part (b) shows the fecundity value from the Leslie matrix. Its value at 4.0 mm is 0.27 offspring who survive to the first age class per individual. Assuming a 50/50 ratio of male to female in the population, the fecundity is 0.54 offspring who survive to the first age class per female.

Part (c) shows the growth rate for mature individuals. Its value for the 4.0 mm computer run is 0.41 mm/month.

Part (d) shows the growth rate for midlife individuals. Its value for the 4.0 mm computer run is 2.70 mm/month.

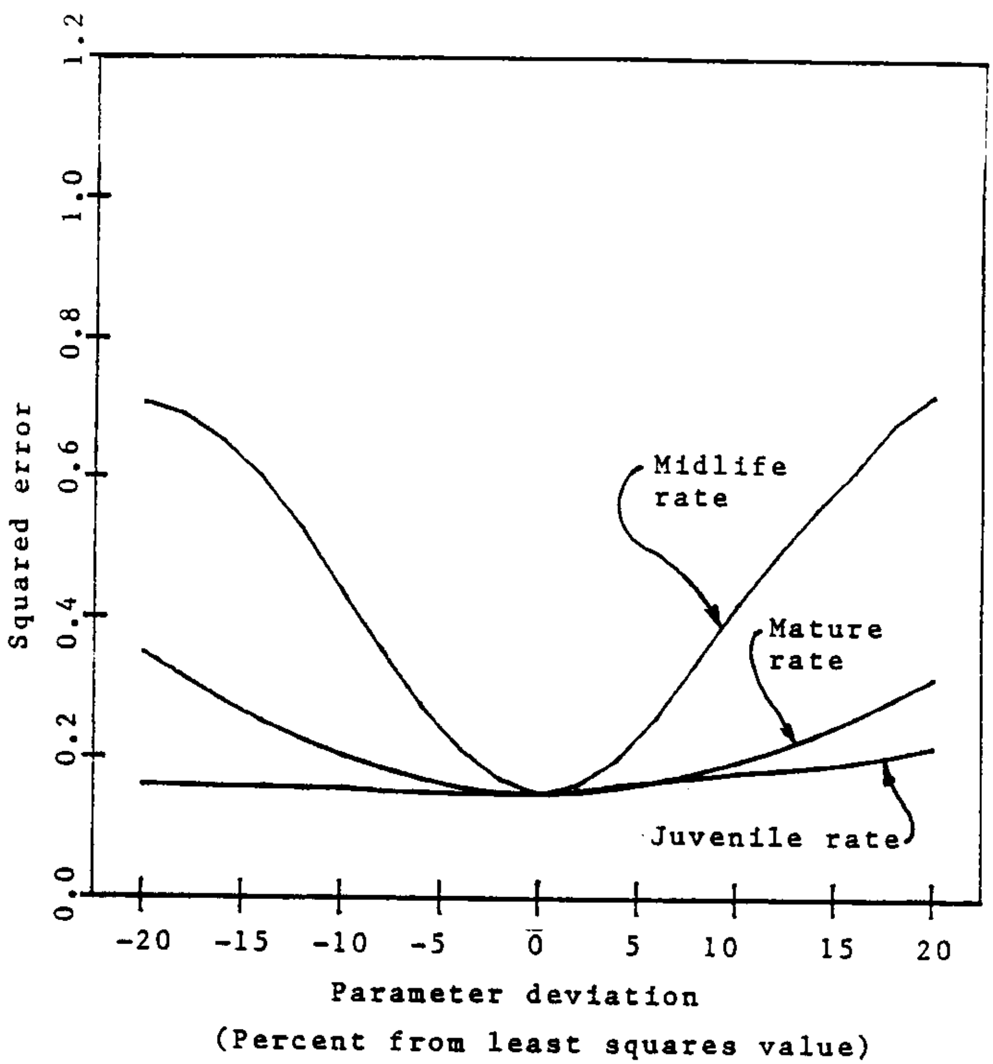
Part (e) shows the growth rate for juveniles. Its value is 0.98 mm/month, higher than mature individuals but much less that midlife individuals.

The sensitivity analysis is shown in Figure 4.3-3. The model shows the usual sensitivity to the survival probability and insensitivity to the fecundity. Also note that the model is least sensitive to the juvenile growth



(a) Leslie parameters

Figure 4.3-3. Parameter sensitivity in the trilinear growth model.



(b) Growth parameters

Figure 4.3-3. (continued)

rate compared to the other rates.

Figure 4.3-4 shows the estimated growth curves for five of the ten computer runs. The midlife stage is between 1 and 2 years.

4.4 Age specific survival probabilities

Another improvement in the structure of the model is possible by relaxing the assumption that the survival probabilities are age independent. In this variation of the model the single age independent survival probability parameter is replaced by a set of parameters which are the age specific survival probabilities.

A straightforward implementation of this idea would be to replace the age independent survival probability with the seven separate off-diagonal elements of the Leslie matrix. In keeping with the goal of using a minimum number of parameters in the model, however, we elect to replace the age independent parameter with only four parameters as shown in the table below.

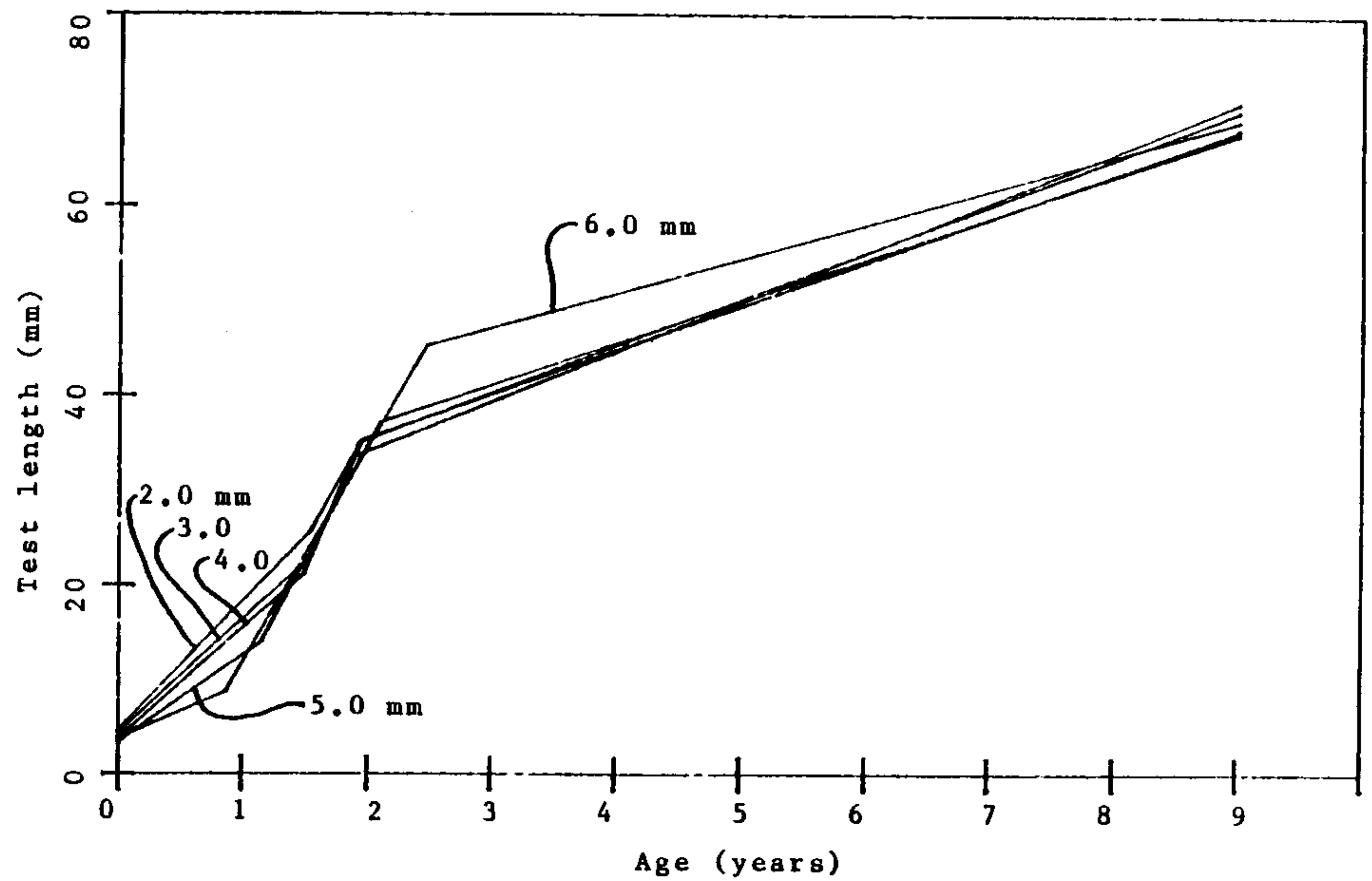


Figure 4.3-4. Optimal growth curves for the trilinear growth model.

Age class (years)	Survival probability
0	p [1]
1	p [1]
2	p [3]
3	p [3]
4	p [5]
5	p [5]
6	p [7]
7	p [7]

The assumption of finite variance as shown in Figure 2.3-7 is made. The growth relationship is assumed to be trilinear. The size distribution at a given age is again assumed uniform. The 19 parameters of the model are

* 2, juvenile slope and intercept
* 2, midlife slope and intercept
* 2, mature slope and intercept
* 8, initial age distribution
* 4, survival probabilities
* 1, fecundity

The Jacobian J of Section 3.2 is a 150 x 19 matrix. The nonlinear least squares problem was solved with the growth variance parameter fixed. Seven different computer runs were made with the standard deviation of the uniform distribution at 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, and 8.0 mm.

The lowest squared error occurred at a standard deviation of 5.0 mm. The results for that case are shown in Figure 4.4-1. A detailed visual comparison of this

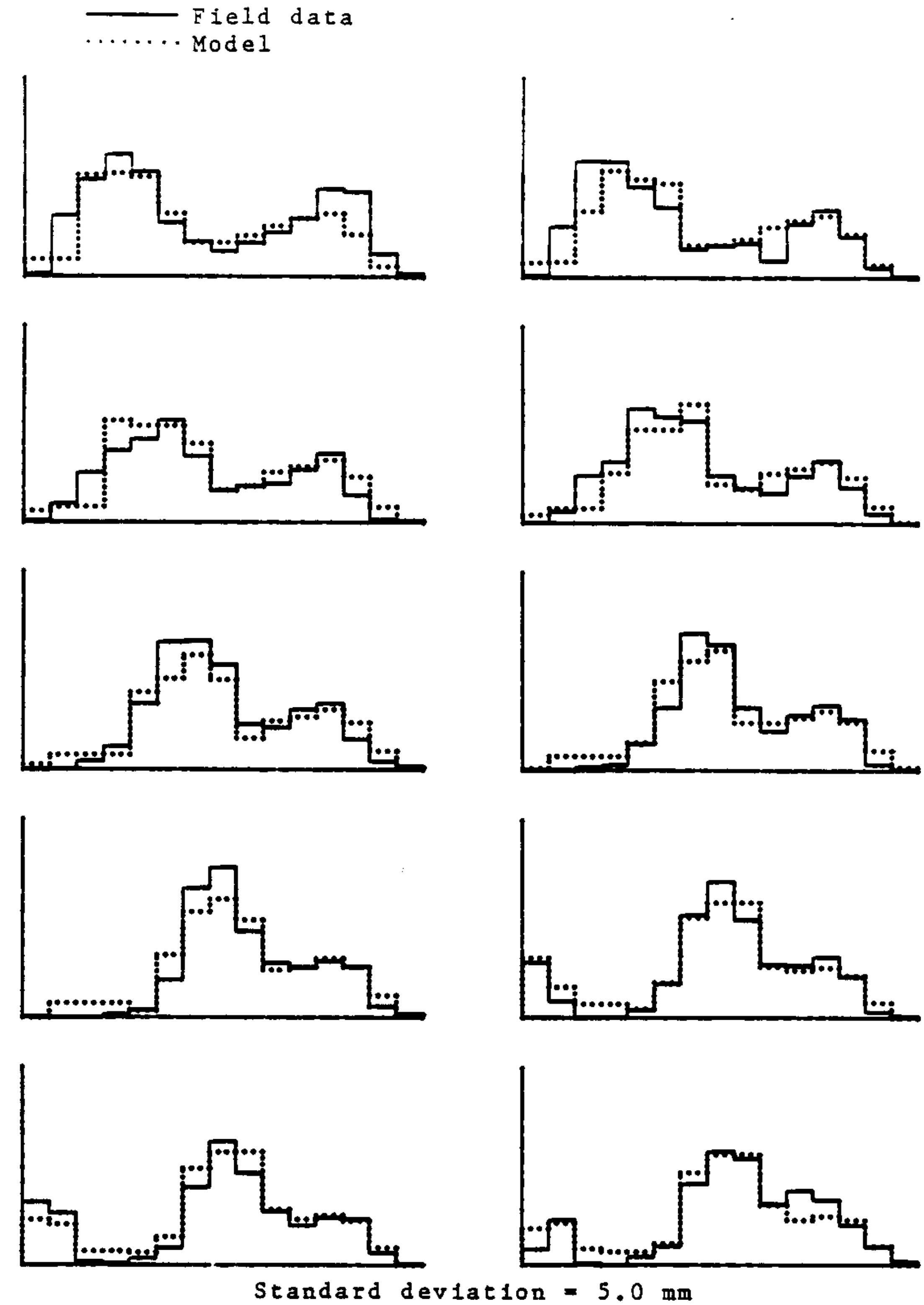


Figure 4.4-1. The age specific survival probability model.

figure with Figure 4.3-1 shows a noticeable improvement in the fit. In fact the squared error in this model is 13% less than the squared error in the age independent survival probability model of Figure 4.3-1.

Figure 4.4-2 displays the squared error for five of the seven computer runs, along with the corresponding minimum least squares values of the fecundity and the growth rates. The scale is identical to that of Figures 4.1-2, 4.2-2, and 4.3-2 for ease of comparison.

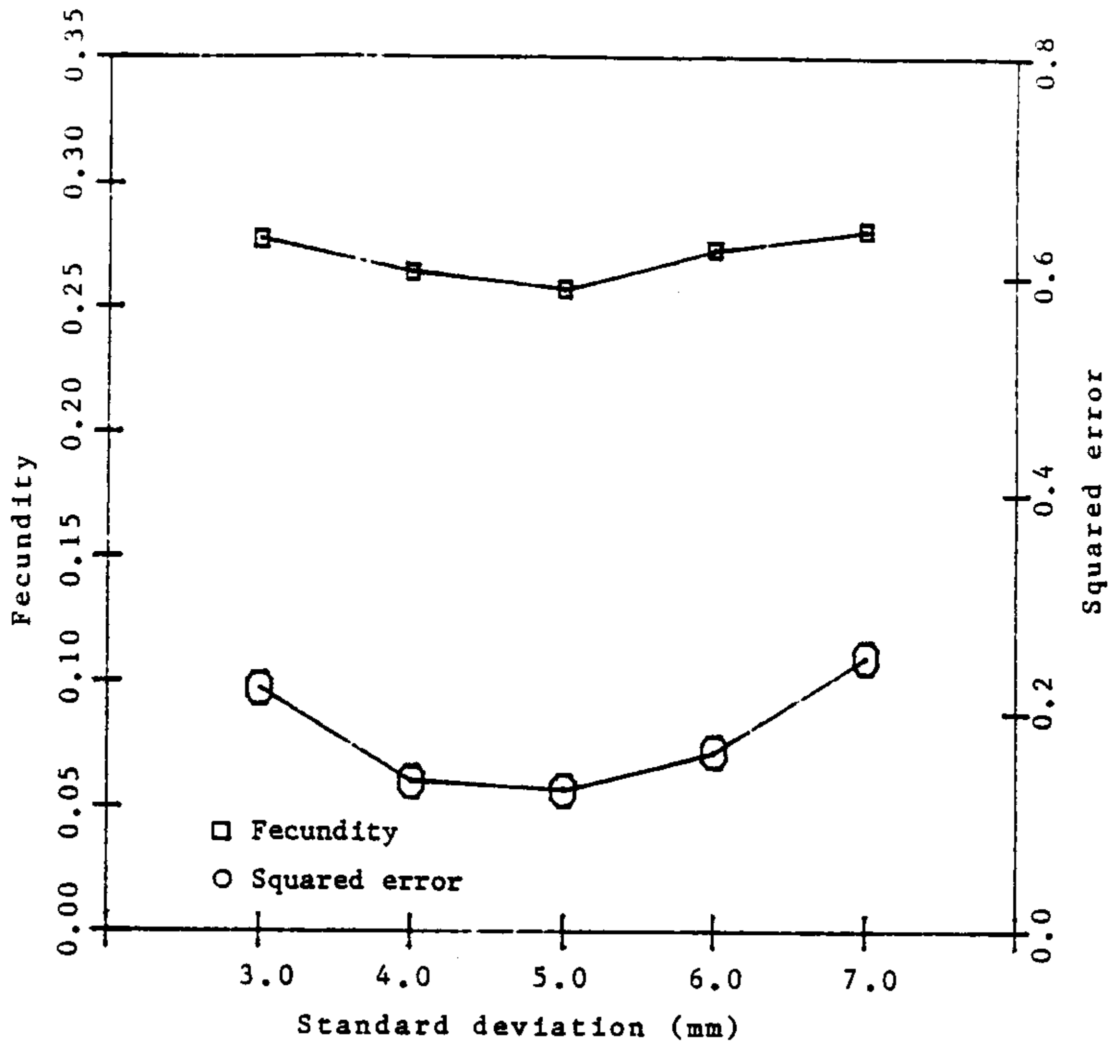
Part (a) of Figure 4.4-2 shows the fecundity value from the Leslie matrix. Its value at 5.0 mm is 0.26 offspring who survive to the first age class per individual. Assuming a 50/50 ratio of male to female in the population, the fecundity is 0.52 offspring who survive to the first age class per female.

Part (b) shows the growth rate for mature individuals. Its value for the 5.0 mm computer run is 0.38 mm/month.

Part (c) shows the growth rate for midlife individuals. Its value for the 5.0 mm computer run is 2.80 mm/month.

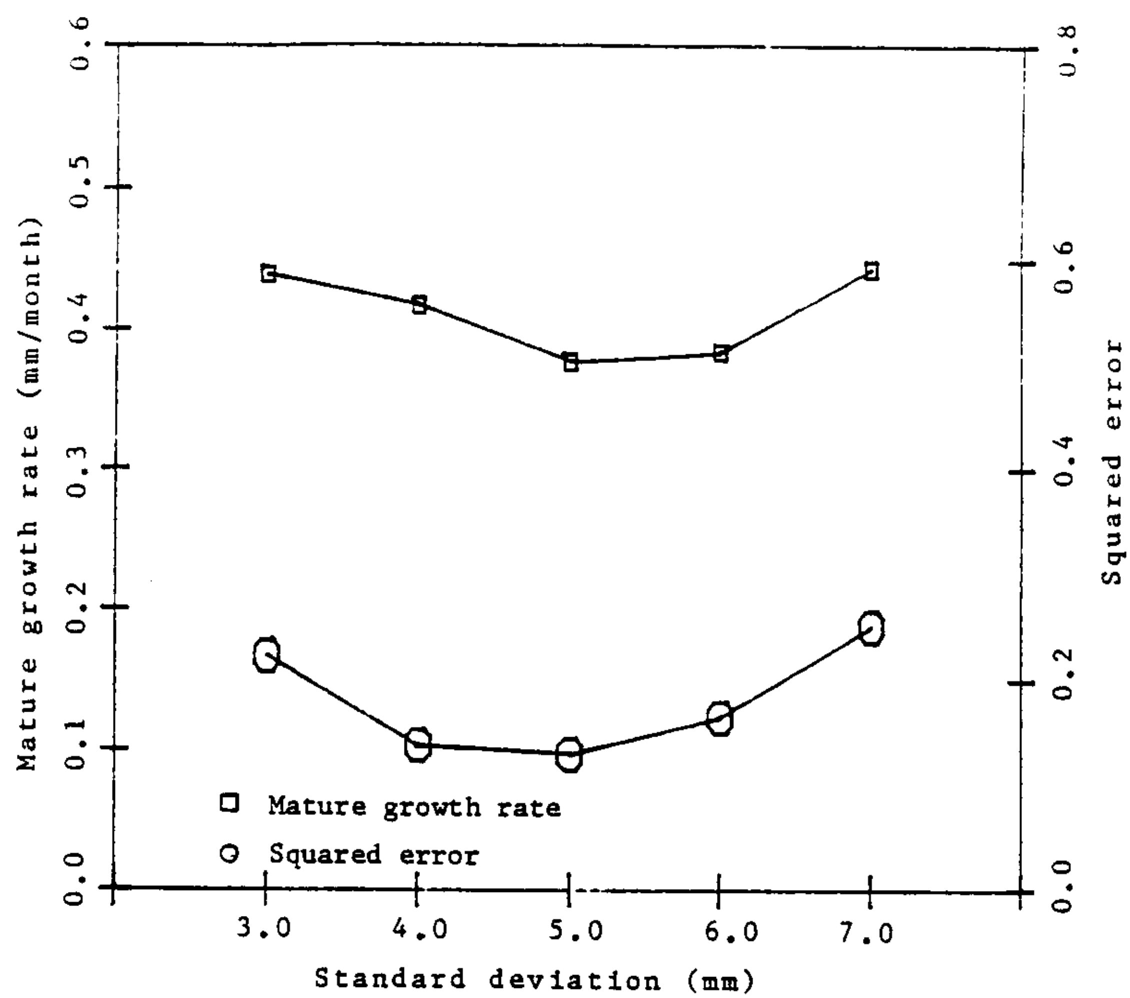
Part (d) shows the growth rate for juveniles. Its value is 0.95 mm/month, again higher than mature individuals but much less that midlife individuals.

The sensitivity analysis is shown in Figure 4.4-3. Since the effect of the single age independent survival



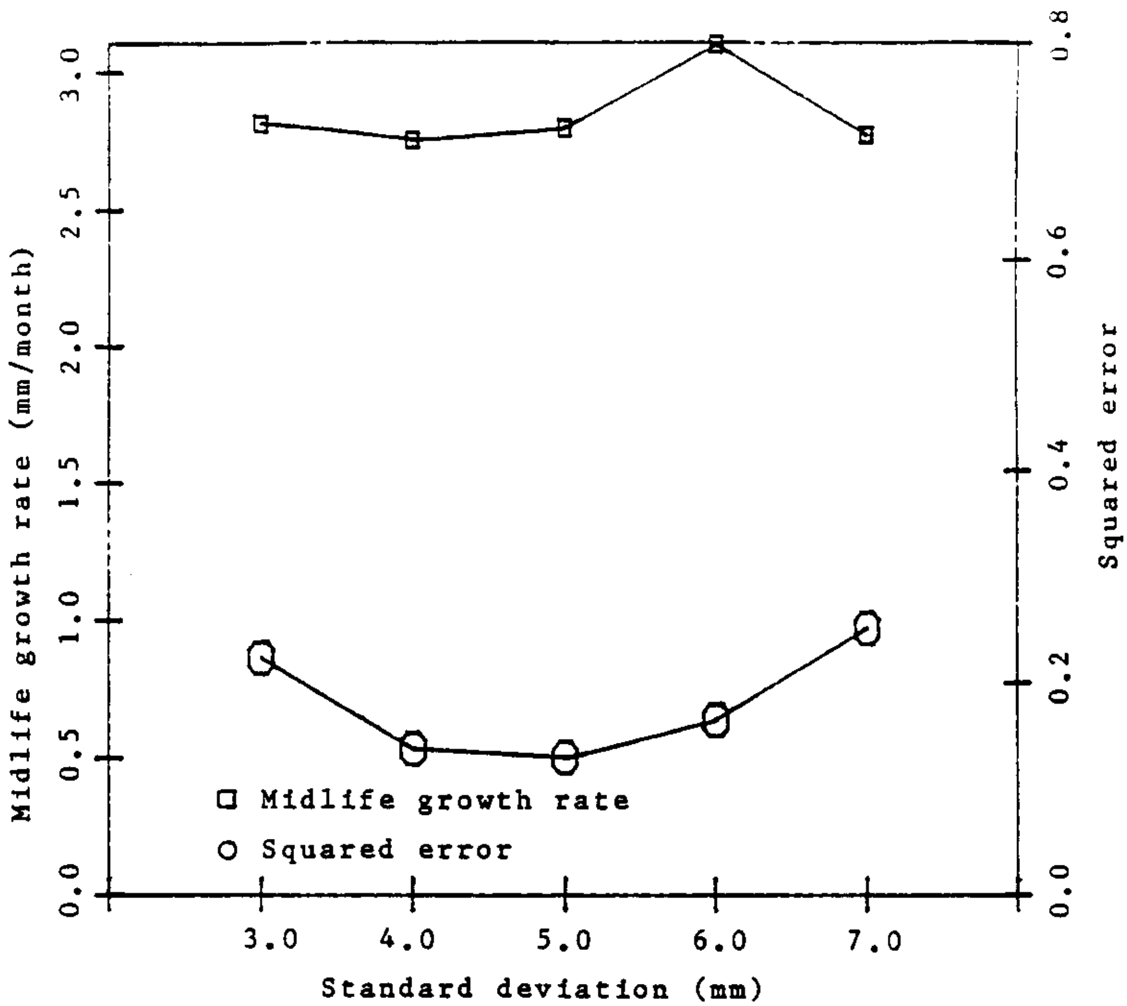
(a) Fecundity

Figure 4.4-2. The effect of growth variance on the age specific survival probability model.



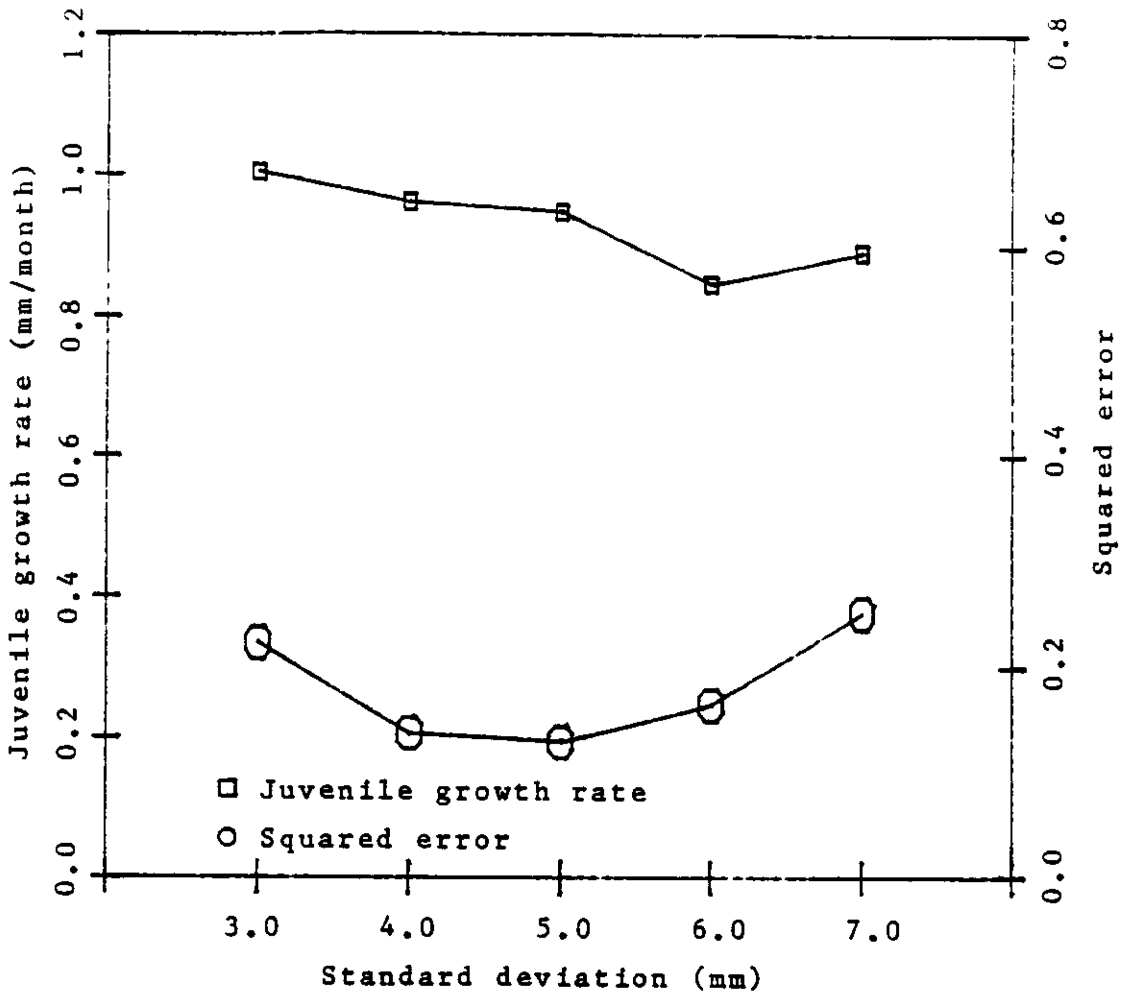
(b) Mature growth rate

Figure 4.4-2. (continued)



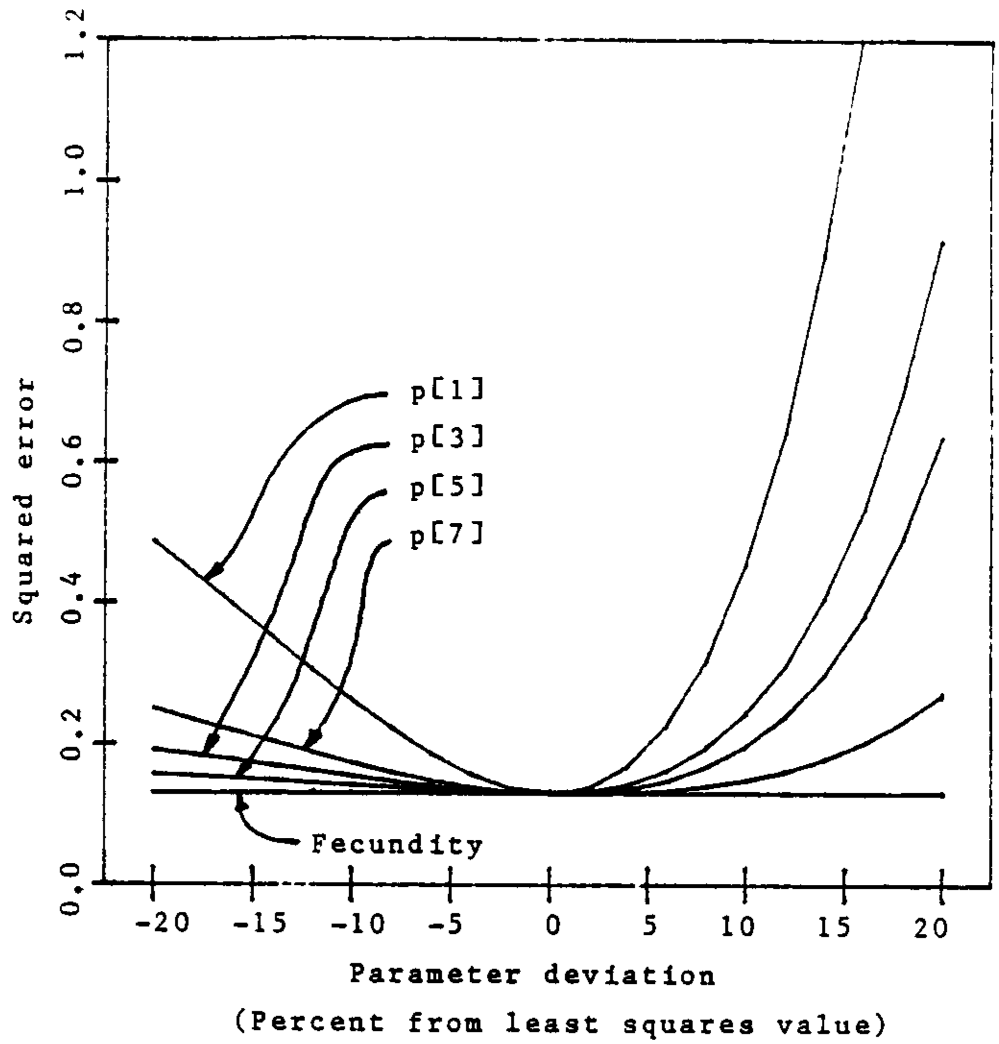
(c) Midlife growth rate

Figure 4.4-2. (continued)



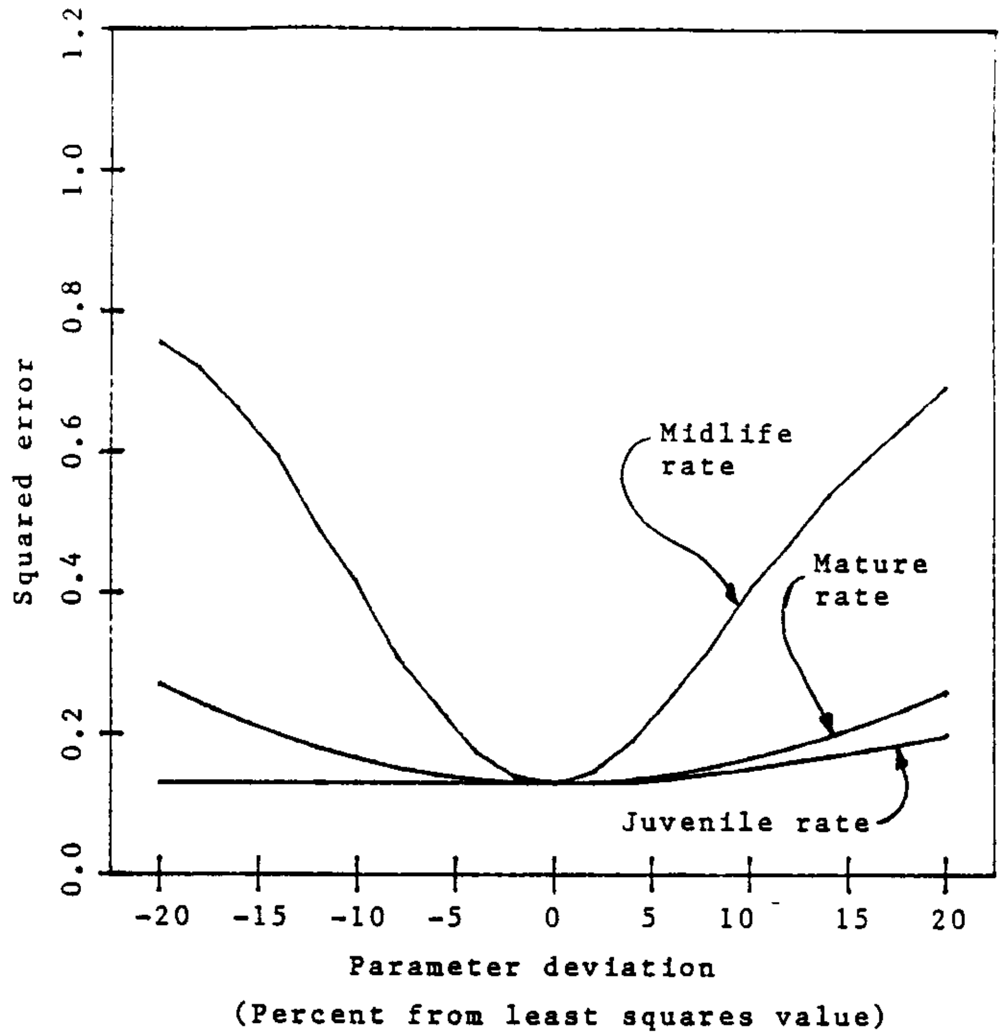
(d) Juvenile growth rate

Figure 4.4-2. (continued)



(a) Leslie parameters

Figure 4.4-3. Parameter sensitivity in the age specific survival probability model.



(b) Growth parameters

Figure 4.4-3. (continued)

probability is now shared by p[1], p[3], p[5], and p[7], the model is not as sensitive to each of these four parameters as it was to the single parameter as shown in Figure 4.3-3 (a). The model shows the usual insensitivity to the fecundity. As before, it is least sensitive to the juvenile growth rate compared to the other rates.

The sensitivity is a measure of how rapidly the squared error increases as a given parameter deviates from its estimated value. If the sensitivity is high, as it is for the midlife growth rate, we have confidence in the estimated value, since a small deviation from that value increases the squared error a great deal. But what if the sensitivity is low, as it is for the fecundity? We have seen that the reason for its low sensivity is that the fecundity only has an effect on a few of the 150 bins in the simulation. Do we therefore have less confidence in the estimation of the fecundity value?

One approach to the problem of estimating the accuracy of the computed parameter values is to use nonparametric statistics. These methods have the advantage of being free from normal distribution theory, but at the cost of a stiff computational requirement.

The method chosen here is the jackknife procedure [Efro79]. Suppose we have an estimate, $\hat{r}$, of a parameter, r, based on n observations. A common measure of accuracy of the estimate is the standard deviation,

$$s = \sqrt{E\ [(\hat{r} - r)^2]},$$

the root mean square difference of $\hat{r}$, based on the n data points, from r. The Jackknife estimate $\hat{s}$ of s involves recomputing the estimate $\hat{r}_{(i)}$ on the set of data points obtained by deleting the ith data point from the original data set. The estimate of s is then

$$\hat{s} = \sqrt{\frac{n - 1}{n} \sum_{i=1}^{n} (\hat{r}_{(i)} - \hat{r})^2}$$

The question in this study is "what constitutes a data point?". At the finest granularity, an individual bin in the 150 bins of the time sequence of histograms is a data point. But 150 computer runs to obtain an estimate of the accuracy is out of the question. Instead we choose to define a data point, for purposes of the jackknife procedure, as the histogram for a single month.

Ten additional computer runs were made, deleting, in turn, the size histogram for one month from the field data. Figure 4.4-4 shows the results. In all cases the estimate of the accuracy is less than 3% of the estimate of the parameter value. There appears to be no correlation between the sensitivity of the parameter value and the jackknife estimate of accuracy.

Figure 4.4-5 shows the estimated growth curves for five of the ten computer runs. They are remarkably

i	$\hat{r} - \hat{r}_{(i)}$							
	Fecundity	p [1]	p [3]	p [5]	p [7]	Juvenile rate	Midlife rate	Mature rate
1	0.00003	0.00601	0.00307	-0.01404	-0.00770	0.00014	-0.00034	0.00790
2	0.00046	-0.00149	0.00723	-0.00099	0.00183	0.00142	-0.01829	0.00675
3	0.00119	0.00147	0.00568	-0.00078	0.00060	0.00124	-0.00327	0.00223
4	0.00094	0.00369	-0.00499	-0.00066	-0.00142	0.00194	-0.00254	0.00177
5	0.00011	0.00346	0.00049	-0.00092	-0.00046	0.00037	0.00289	-0.00061
6	0.00012	-0.00015	0.00229	-0.00089	-0.00016	0.00054	0.00208	0.00032
7	0.00042	0.00344	0.00217	-0.00107	-0.00220	0.00143	0.00120	0.00132
8	-0.00088	-0.00003	0.00094	-0.00067	-0.00026	0.00067	-0.00064	-0.00060
9	0.00297	-0.00248	-0.00122	-0.00102	-0.00088	0.00056	-0.00038	0.00055
10	-0.00053	-0.00149	-0.00066	-0.01267	0.00138	0.00022	0.00104	-0.00139
$\hat{r}$:	0.2577	0.9795	0.9791	0.9989	0.9853	0.9484	2.7997	0.3773
$\hat{s}$:	0.0033	0.0088	0.0109	0.0181	0.0081	0.0031	0.0182	0.0104

Figure 4.4-4. Table of Jackknife data for estimation of parameter accuracy.

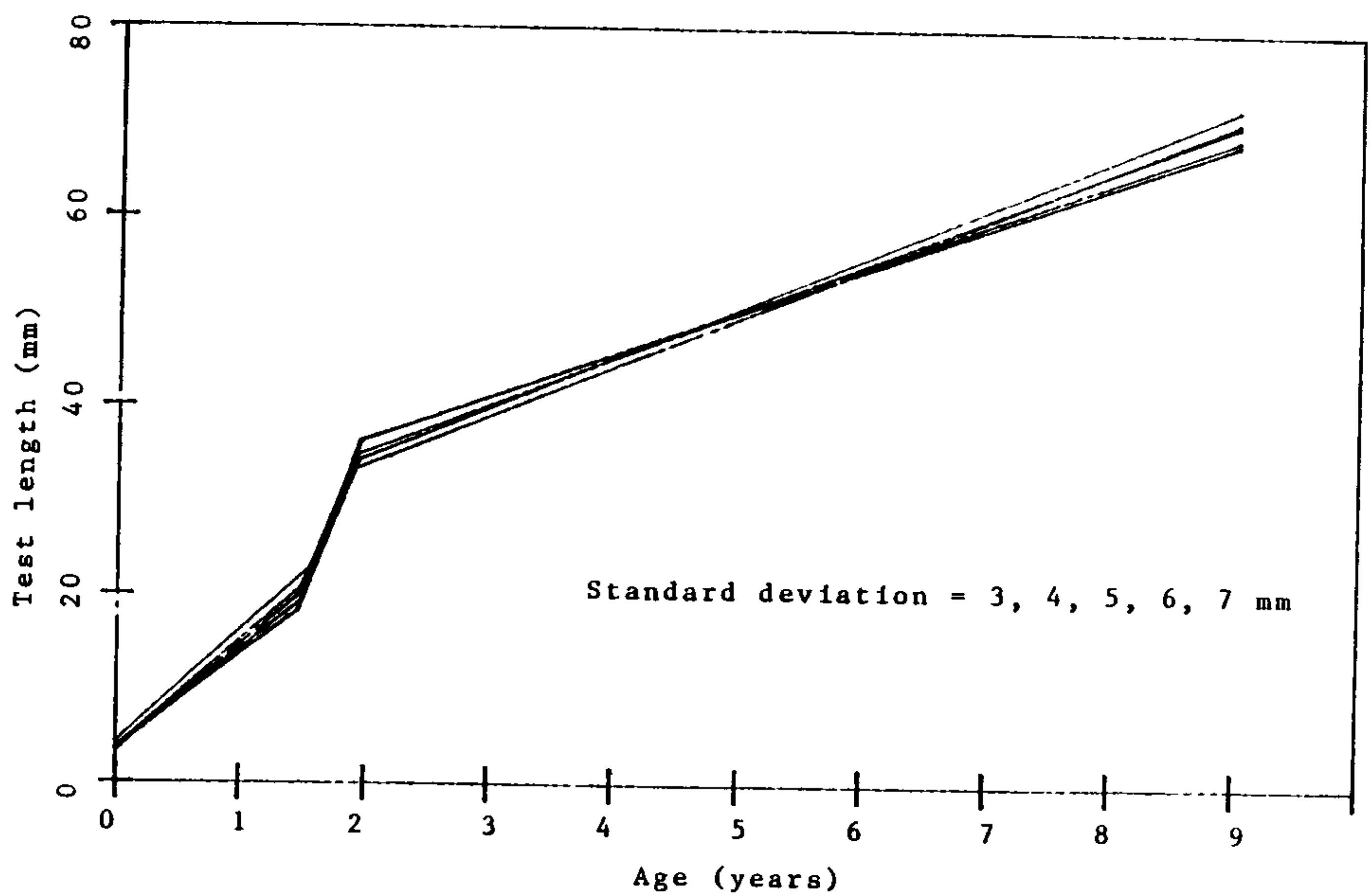


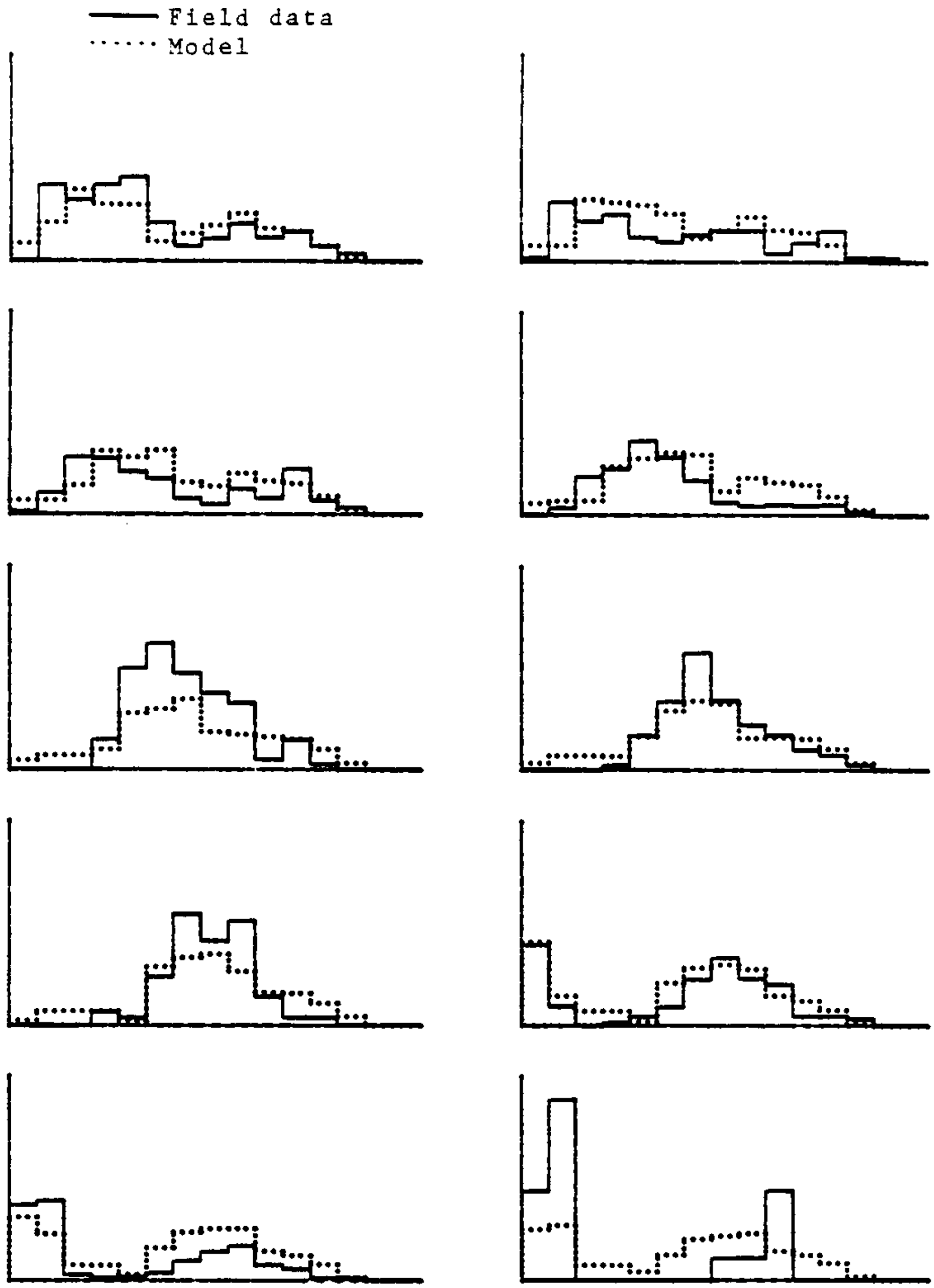
Figure 4.4-5. Optimal growth curves for the age specific survival probability model.

consistent regardless of the assumed variance of the growth distribution. The midlife stage is between 1.5 and 2 years.

4.5 Spatial variations

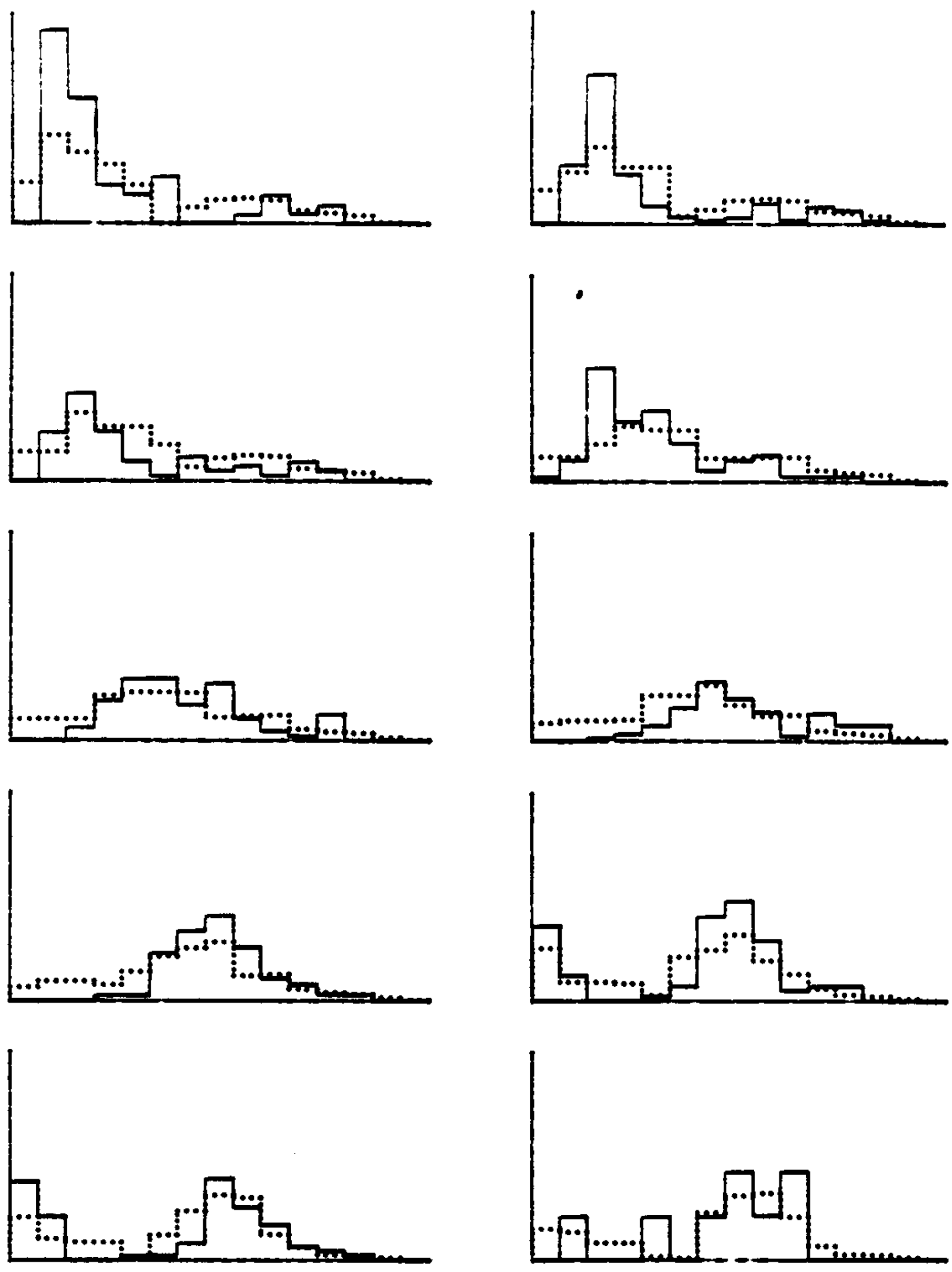
The age specific survival probability model of the previous section is successful in estimating the model parameters of the system comprising the entire lagoon. The data can also be examined by station number as shown in Figure 1.3-2. The purpose of this section is to apply the previous model to the data as a function of position in the lagoon.

The data in Figure 1.3-2 shows rather severe fluctuations due to the small sample size compared with the sample size of the lagoon as a whole. To increase the sample size the stations were grouped into the pairs (2, 3), (4, 5), (6, 7), and (8, 9). The age specific survival probability model was used to fit each of the four data sets assuming a standard deviation of 0.5 mm in the size versus age relationship. Figure 4.5-1 shows the results. The vertical axis is 0.8 individuals per square meter full scale for stations (2, 3) and (4, 5), compared with 0.4 full scale for stations (6, 7), (8, 9), and for all the data presented previously for the lagoon as a whole.



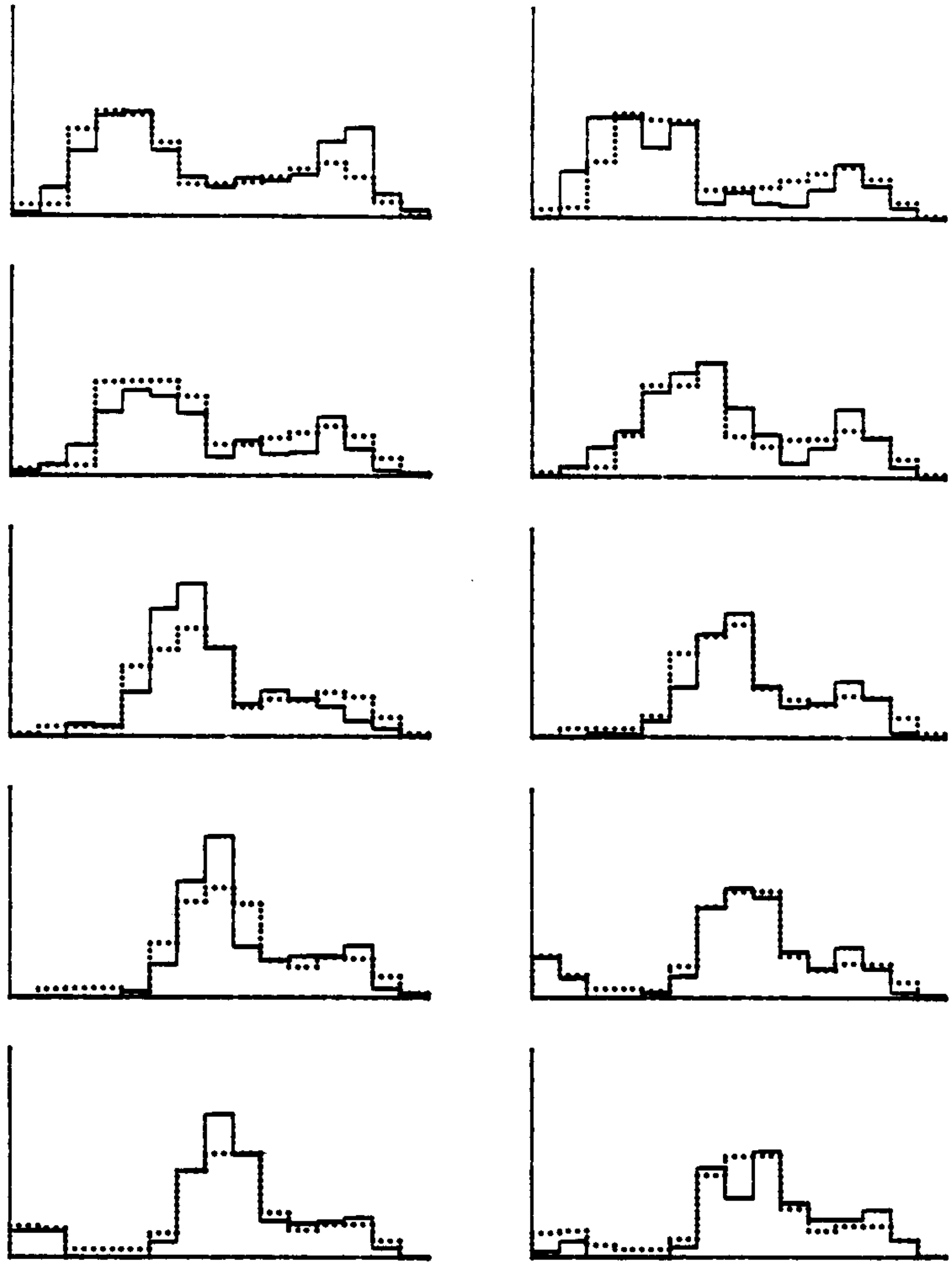
(a) Stations 2, 3

Figure 4.5-1. Spatial variations in groups of two consecutive stations.



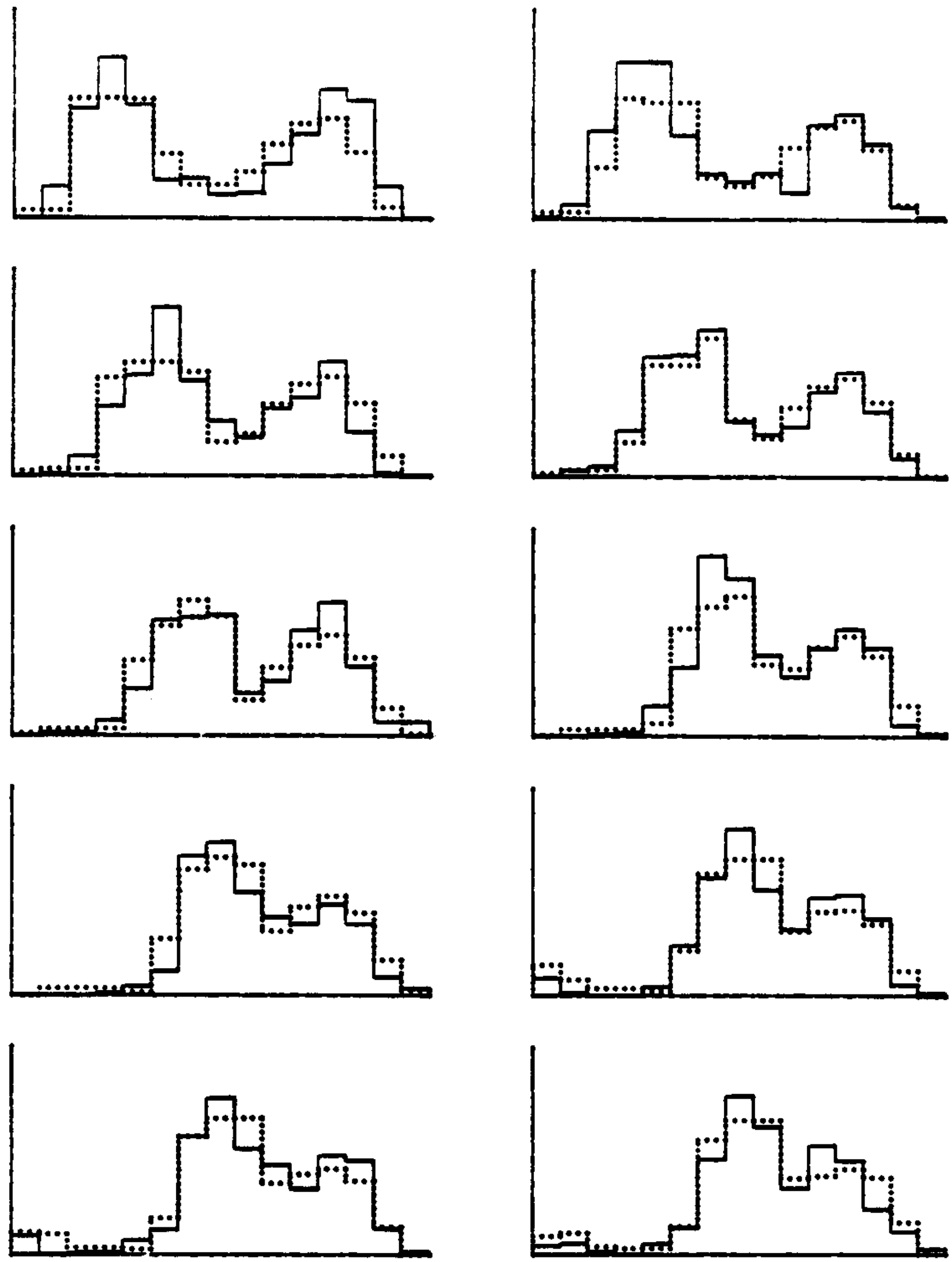
(b) Stations 4, 5

Figure 4.5-1. (continued)



(c) Stations 6, 7

Figure 4.5-1. (continued)



(d) Stations 8, 9

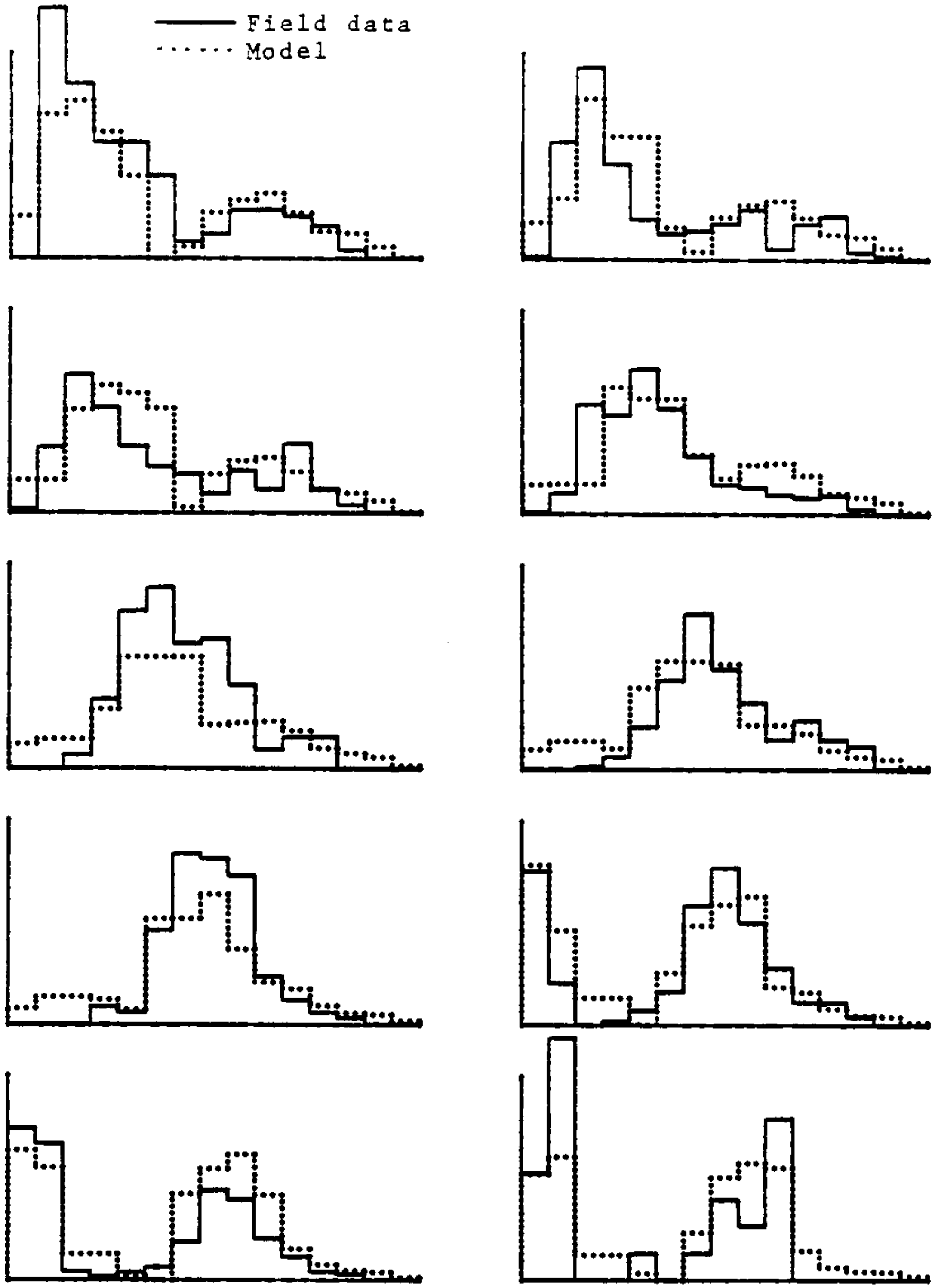
Figure 4.5-1. (continued)

A grouping of four stations was also made with (2, 3, 4, 5) and (6, 7, 8, 9). These results are shown in Figure 4.5-2. A summary of the estimated parameter values, along with the results for the entire lagoon from the previous section is given in Figure 4.5-3. The survival probabilities have been converted to their equivalent annual values. The yearly survival probability p_y is related to the corresponding monthly probability p_m by $p_y = p_m^{12}$.

The fit to the model is consistently worse for those stations on the west end of the lagoon. The squared error for the nonlinear fit for stations (2, 3, 4, 5) is more than four times the squared error for stations (6, 7, 8, 9). This contrast of the fit to the model may be an indication of the instability of the environment on the west end compared to the east end.

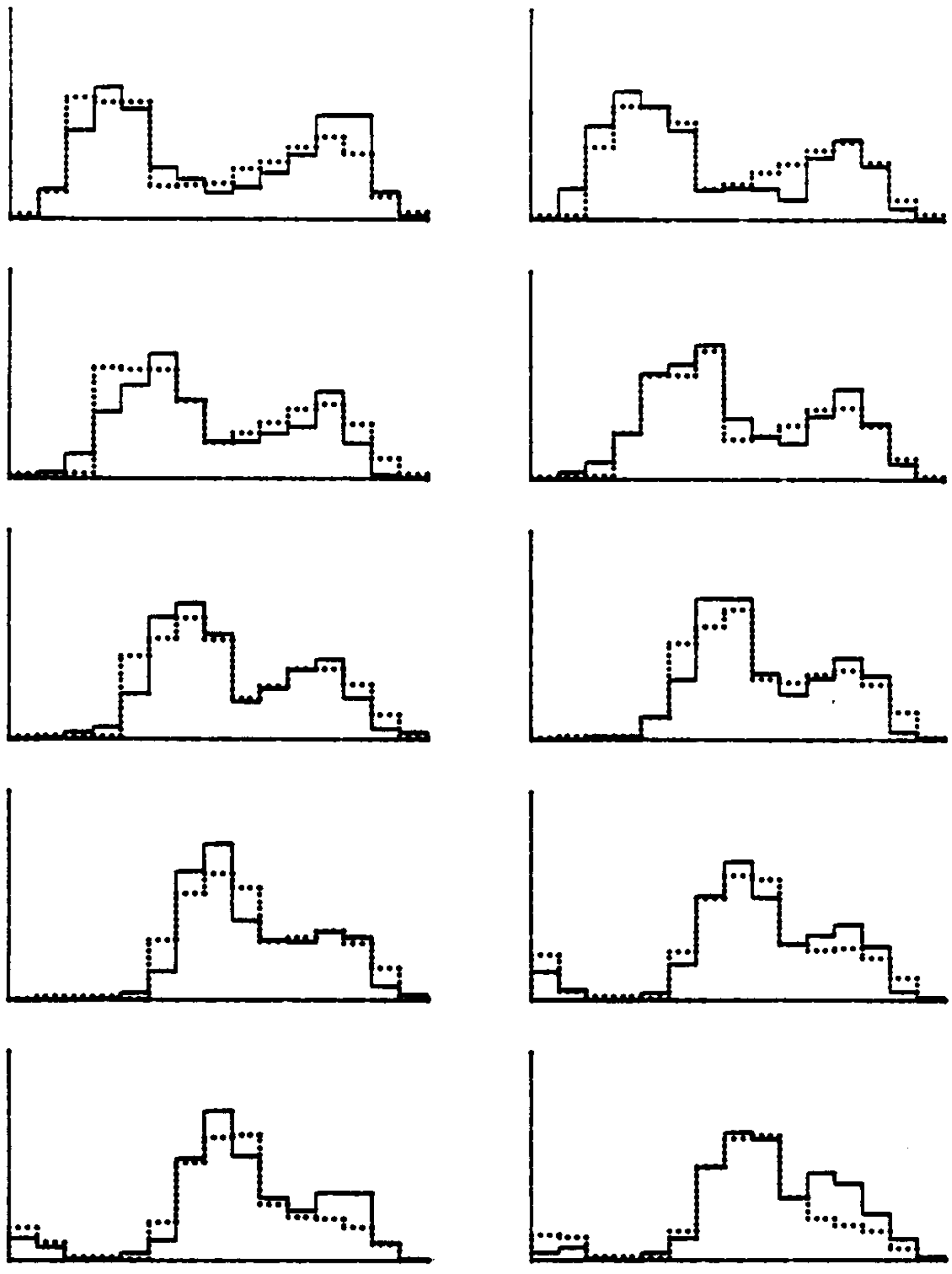
The fecundity also shows a consistent trend as a function of position in the lagoon. It is about ten times greater at the west end of the lagoon compared to the east end. This is consistent with the observation made in Section 2.2 that the planktonic larvae migrate into the lagoon from the protected outer coast population, since the mouth of the lagoon is at the west end. These fecundity values, therefore, may not characterize those females in the environment of the lagoon.

The survival probabilities show no consistent trend



(a) Stations 2, 3, 4, 5

Figure 4.5-2. Spatial variations in groups of four consecutive stations.



(b) Stations 6, 7, 8, 9

Figure 4.5-2. (continued)

Stations	Squared error	Fecun-dity	p [1]	Annualized p [3]	p [5]	p [7]	Juvenile rate	Midlife rate	Mature rate
all	0.129	0.258	0.780	0.776	0.987	0.837	0.948	2.800	0.377
2345	0.557	1.149	0.701	0.503	0.067	0.366	0.646	3.056	0.389
6789	0.125	0.138	0.621	0.911	0.999	0.004	0.834	2.952	0.504
23	1.074	0.627	0.998	0.150	0.999	0.986	0.813	2.766	0.305
45	1.003	0.371	0.553	0.999	0.432	0.415	0.744	2.664	0.353
67	0.131	0.242	0.636	0.741	0.663	0.553	0.970	2.802	0.402
89	0.166	0.088	0.779	0.921	0.999	0.987	0.918	2.848	0.372

Figure 4.5-3. Summary of spatial variation results.

in their spatial variations.

The juvenile growth rate appears to be about 20% greater at the east end compared to the west end, as does the mature growth rate. There is no such discernable trend for the midlife growth rate.

4.6 Immigration

All of the models discussed so far were applied to the 1977 data. This section describes the application of the models to the 1982 data.

Figure 4.6-1 shows the age specific survival probability model applied to the more recent data. As in Section 4.4, the model contained 19 parameters. The five months of field data are for 7-82, 8-82, 9-82, 10-82, and 12-82. There is a one month gap between the penultimate and the final month during which no data was taken. The number of data points to fit is 15 bins per month times 5 months. The Jacobian in the nonlinear least squares algorithm is 75 x 19.

In this time sequence, recruitment to the population showed up in the model during the last month. The calendar month for recruitment to the population is in fact unchanged from the previous models. Notice, however, that during the second month of the time sequence a peak

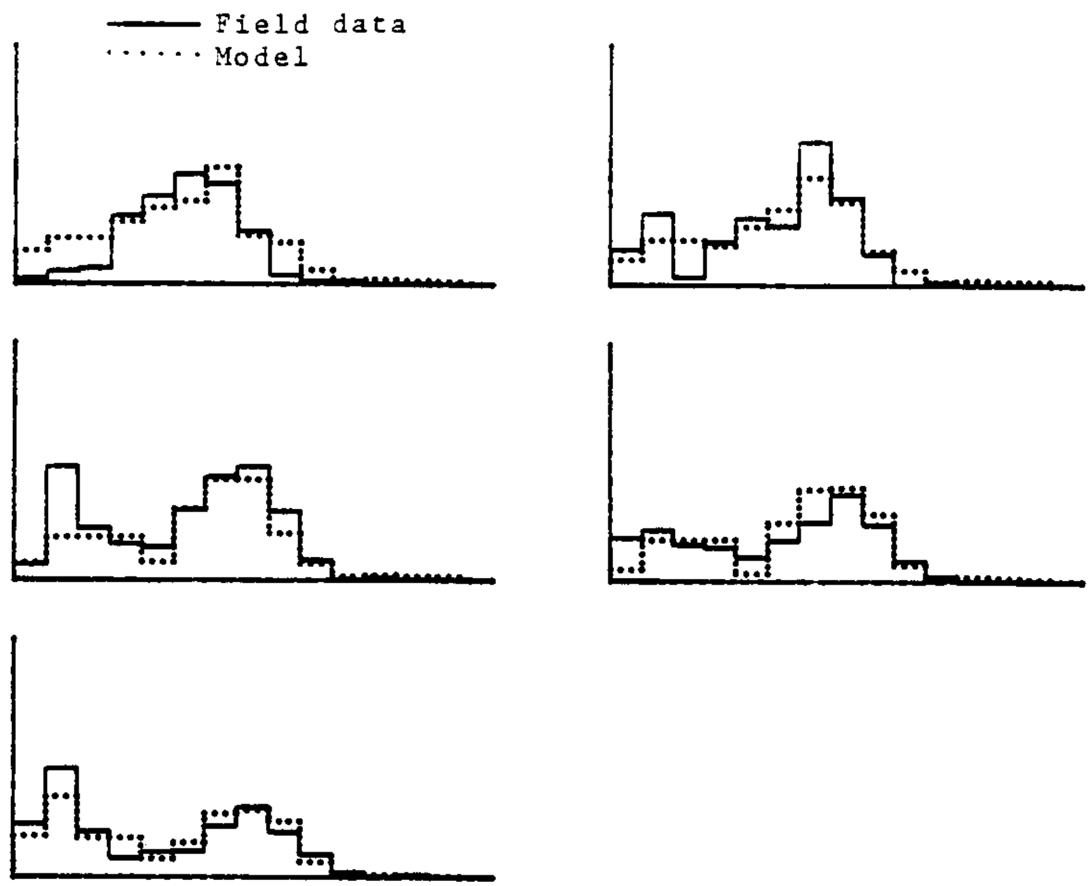


Figure 4.6-1. The model without immigration for the 1982 data.

of young individuals appears in the population. It is speculated that sometimes with the right combination of high tides and turbulent ocean conditions, waves wash over the barrier beach and carry some individuals from the high density proctected outer coast population.

A tacit assumption in all of the previous models was a closed system. Neither immigration or emmigration was considered. If there is evidence of immigration from the field data the model can be modified accordingly.

Recall from Section 2.1 that the evolution of the population is described by

$$A\ \bar{a}(t) = \bar{a}(t+1).$$

We can modify the basic model to include immigration by adding an immigration vector $\bar{I}(t)$. The modified model is now

$$A\ \bar{a}(t) + \bar{I}(t) = \bar{a}(t+1).$$

The immigration vector is conceptually easy to add to the mathematical model. But it has serious ramifications to the system identification problem. If the immigration is measureable as a separate component, it can simply be added in at the appropriate time in the simulation without changing the parameter space of the model. In this case, however, the immigration is inferred from the field data. To include it we must change the parameter space.

In the model $\bar{I}(t)$ is assumed to be zero except for the second month of the simulation. In that month the 8 components of the vector, one for each age class, are added to the population. Each one of these components becomes an additional parameter in the parameter space of the nonlinear least squares problem. There are now a total of 27 parameters.

* 2, juvenile slope and intercept
* 2, midlife slope and intercept
* 2, mature slope and intercept
* 8, initial age distribution
* 8, immigration vector during second month
* 4, survival probabilities
* 1, fecundity

The Jacobian is a 75 x 27 matrix. The resulting nonlinear least squares problem was solved as shown in Figure 4.6-2. The improvement to the fit occurs primarily in the first two months of the simulation. The following table compares the two models. The survival probabilities are annualized.

	Without immigration	With immigration
Squared error	0.0580	0.0441
Fecundity	0.802	0.882
p [1]	0.636	0.201
p [3]	0.142	0.141
p [5]	0.995	0.311
p [7]	0.862	0.806
Juvenile rate	0.722	0.789
Midlife rate	2.845	2.832
Mature rate	0.389	0.419

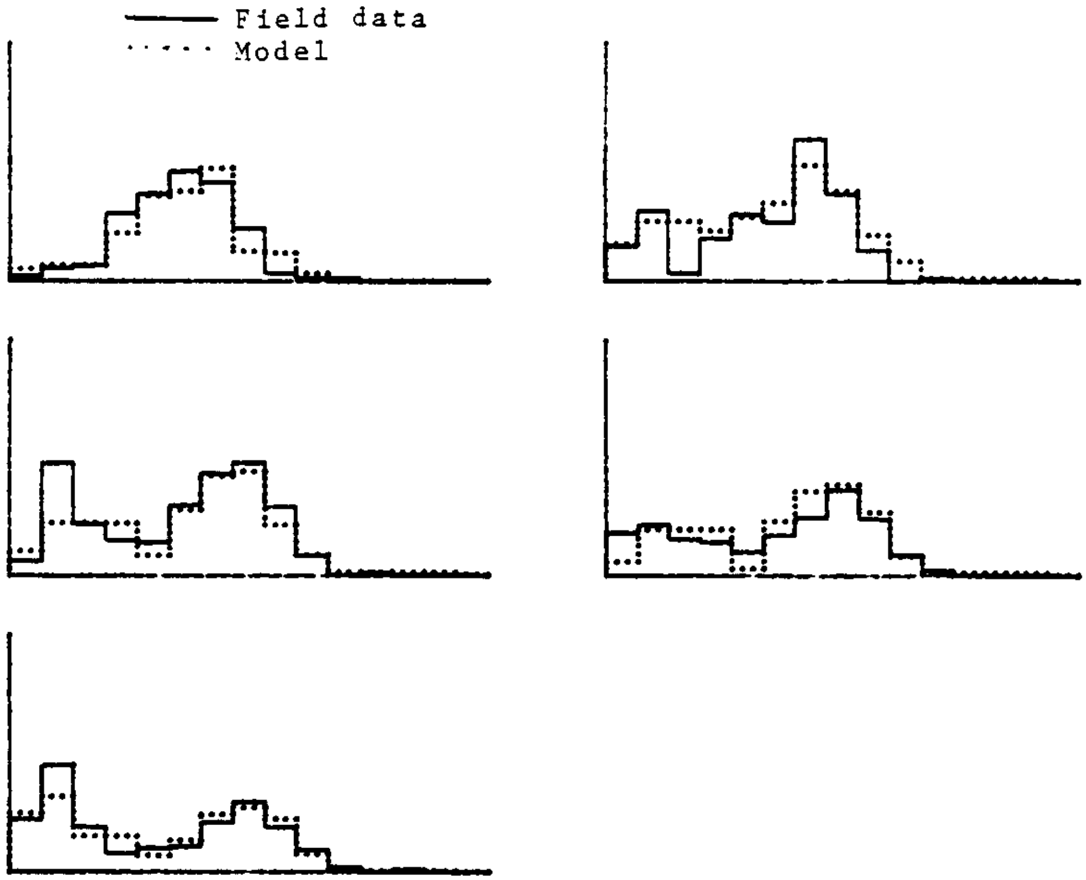


Figure 4.6-2. The model with immigration for the 1982 data.

5. Conclusions

The goal of this dissertation was to construct a mathematical model which describes the growth dynamics of the *Dendraster excentricus* population in the Pt. Mugu lagoon. The specific function of the model was to estimate the values of the controlling parameters in the ecological system from a time sequence of ten monthly size histograms. The model was in fact successful in estimating these values.

Figure 5.1-1 summarizes the estimated values. The original intent was to model the growth function as bilinear with age, as is common in the literature with this species. The modeling effort with this field data, however, indicates that the growth function is better described by a trilinear relationship. A reduction in the squared error by more than a factor of two is obtained with the trilinear assumption.

The model was also used to investigate the spatial variations of the controlling parameters along the length of the lagoon. The fecundity increased markedly from the east to the west end of the lagoon. The juvenile growth rate and the mature growth rate decreased somewhat from east to west.

	Model			
	Bilinear	Spline	Trilinear	ASSP*
Number of parameters	14	15	16	19
Squared error	0.385	0.337	0.149	0.129
Fecundity	0.202	0.293	0.269	0.258
Annualized survival probability	0.912	0.978	0.742	0.780 0.776 0.987 0.837
Juvenile rate	0.910	na	0.975	0.948
Midlife rate	na	na	2.699	2.800
Mature rate	0.405	na	0.411	0.377

* Note: Age specific survival probability model

Figure 5.1-1. A summary of the four models.

The model was also modified to take into account the possibility of immigration from outside the system. A recent time sequence of five monthly size histograms from the lagoon was successfully fit to the immigration model. The growth rates were essentially the same as those determined from the data of the ten month sequence five years earlier. The fecundity, however, was substantially greater.

Just as important as the specific parameter values estimated for this system, if not more so, is the methodology developed in this dissertation for the identification of the system. One thing we do not have in the literature is a shortage of biological and ecological models, many of them quite complex. We may even have more models and analyses of models than we have field data! What we do not have enough of, I believe, is the application of the models to the field data. Many practitioners simply do not take advantage of the wealth of theory which has been developed by the modelers.

It is a thesis of this work that one of the best uses to which a particular mathematical model of a biological system can be put, is as a tool in the estimation of physical quantities. This is the whole idea behind linear regression, a common tool for simple linear models. It should be the idea behind our more complicated nonlinear models as well.

The methodology employed in this study differed from the standard methodology in that the various components of the system - growth, recruitment, and mortality - were combined into a single system. The identification was performed system wide with all of the components in place. The contribution of this dissertation is the different way in which the theory was applied to the field data.

Two advantages accrued from this approach. First, the growth curves were estimated directly from the time sequence of the size histograms. This method of extracting the growth curves is apparently unique to this study. It should be of general value in biological modeling since the size of an organism is invariably easier to measure than its age. Hence, time sequences of size histograms are cheaper to obtain than time sequences of age distributions.

The second advantage to the system wide approach is that the model simulates the field data directly. A visual comparison of the model with the field data can often give a visual clue as to how the structure of the model can be improved. That is precisely what happened in this study with the growth curves.

There is much room for further research here. For example, this model assumed that the elements of the Leslie matrix were density independent. A density dependent fecundity is not identifiable in this system because recruitment occurs only once with this field data. The value of the estimated fecundity is one data point of the density dependent fecundity relationship, at the particular density value of the system at the time of recruitment. It would be interesting to apply the identification techniques developed here to sets of data which exhibit the cyclic property characteristic of density dependent systems.

6. Appendix

6.1 Raw data examples

The following listing is the raw data for the month 9-82.

station number	sample number	test length (mm)	0 = alive, 1 = dead
5	20	51	1
6	13	27	0
6	14	9	0
6	15	8	1
6	16	11	0
6	17	24	0
6	17	11	0
6	17	7	0
6	18	17	0
6	19	6	0
6	19	9	0
6	20	10	0
6	20	10	0
6	20	4	0
6	20	5	0
6	21	5	0
6	21	6	0
6	21	6	0
6	21	7	0
6	21	9	0
6	21	32	0
6	21	7	0
6	21	8	0
6	22	11	0
6	22	12	0
6	22	20	0
6	23	9	0
6	23	10	0
6	23	13	0
6	23	13	0
6	23	14	0
6	24	9	0
7	10	34	0
7	10	43	0
7	11	28	0
7	11	28	0
7	11	34	0
7	11	34	0
7	11	6	0
7	11	6	0
7	11	7	0
7	11	7	0
7	12	34	0
7	12	23	0
7	12	36	0
7	12	38	0
7	12	29	0
7	13	31	0
7	13	31	0
7	14	29	0
7	15	37	0
7	16	31	0
7	17	41	0
7	17	45	0
7	18	7	0
7	18	48	0
7	18	41	0
7	19	31	0
7	19	7	0
7	20	28	0
7	21	40	0
7	21	35	0
7	21	30	0
7	21	29	0
7	21	29	0
7	21	34	0
7	21	28	0
7	21	24	0
7	21	25	0
7	21	23	0
7	21	42	0
7	21	38	0
7	21	28	0
7	21	30	0
7	21	5	0
8	6	22	0
8	6	15	0
8	6	15	0
8	6	17	0
8	6	9	0
8	6	10	0
8	7	18	0
8	7	19	0
8	7	15	0
8	8	42	0
8	8	43	0
8	8	40	0
8	8	38	0
8	8	11	0
8	8	10	0
8	9	39	0
8	9	34	0

8	9	33	0
8	9	30	0
8	9	28	0
8	9	35	0
8	10	33	0
8	10	31	0
8	10	38	0
8	11	40	0
8	11	29	0
8	12	45	0
8	12	36	0
8	12	29	0
8	12	36	0
8	12	36	0
8	12	28	0
8	12	31	0
8	12	36	0
8	12	36	0
8	12	24	0
8	12	37	0
8	12	17	0
8	12	34	0
8	13	14	0
8	13	14	0
8	13	13	0
8	13	43	0
8	13	39	0
8	13	20	0
8	14	39	0
8	14	28	0
8	14	7	0
8	14	43	0
8	14	18	0
8	14	9	0
8	15	43	0
8	15	43	0
8	15	41	0
8	15	18	0
8	15	22	0
8	15	16	0
8	15	22	0
8	15	9	0
8	15	3	0
8	15	33	0
8	15	32	0
8	15	30	0
8	15	25	0
8	15	29	0
8	15	32	0
8	15	6	0
8	16	42	0
8	16	38	0
8	16	36	0
8	16	31	0
8	16	41	0
8	16	28	0
8	16	28	0
8	17	45	0
8	17	35	0
8	17	38	0
8	17	35	0
9	12	36	0
9	12	8	0
9	12	5	0
9	12	5	0
9	13	3	0
9	13	37	0
9	13	32	0
9	14	34	0
9	14	36	0
9	14	39	0
9	14	38	0
9	15	38	0
9	15	36	0
9	15	38	0
9	15	46	0
9	15	39	0
9	15	38	0
9	15	31	0
9	15	29	0
9	15	39	0
9	15	42	0
9	15	30	0
9	16	38	0
9	16	40	0
9	17	6	0
9	18	40	0
9	18	6	0
9	19	37	0
9	19	30	0
9	19	3	0
9	19	6	0
9	19	30	0
9	19	40	0
9	19	27	0
9	19	32	0
9	19	40	0
9	19	5	0
9	19	9	0
9	19	4	0
9	19	5	0
9	19	55	0
9	19	45	0
9	19	30	0
9	19	39	0

The following listing is the raw data for the month 7-77.

1	1	50	0
1	3	52	0
1	10	47	0
1	11	24	0
1	15	52	0
2	1	14	0
2	1	12	0
2	1	13	0
2	2	12	0
2	3	15	0
2	4	40	0
2	4	31	0
2	4	28	0
2	4	16	0
2	4	15	0
2	4	14	0
2	4	40	0
2	5	32	0
2	6	29	0
2	6	23	0
2	6	29	0
2	6	23	0
2	6	39	0
2	6	49	0
2	6	24	0
2	6	19	0
2	6	22	0
2	6	19	0
2	6	16	0
2	6	20	0
2	7	25	0
2	7	19	0
2	10	34	0
2	10	43	0
2	10	43	0
2	10	27	0
2	10	23	0
2	10	22	0
2	12	29	0
2	13	19	0
2	14	18	0
2	14	47	0
2	14	33	0
2	15	20	0
2	15	26	0
2	15	23	0
2	15	17	0
3	3	36	0
3	3	31	0
3	3	56	0
3	3	45	0
3	3	24	0
3	3	31	0
3	3	21	0
3	3	36	0
3	3	26	0
3	3	21	0
3	3	29	0
3	3	39	0
3	3	39	0
3	3	27	0
3	3	27	0
3	3	28	0
3	3	24	0
3	3	29	0
3	3	31	0
3	3	18	0
3	3	20	0
3	3	27	0
3	3	17	0
3	3	25	0
3	3	24	0
3	3	25	0
3	3	24	0
3	3	30	0
3	3	22	0
3	3	20	0
3	3	25	0
3	3	21	0
3	3	25	0
3	3	26	0
3	3	28	0
3	3	30	0
3	3	20	0
3	3	22	0
3	3	20	0
3	3	20	0
3	3	19	0
3	3	26	0
3	3	20	0
3	3	18	0
3	3	13	0
3	4	22	0
3	5	16	0
3	5	31	0
3	5	21	0
3	5	16	0
3	6	14	0
3	8	6	0
3	13	51	0
3	13	56	0
3	13	31	0
3	13	31	0
3	13	20	0
3	14	46	0
3	14	59	0
3	14	50	0
3	14	21	0
3	14	64	0
3	14	35	0
3	14	32	0
3	14	49	0
3	14	33	0
3	14	24	0
3	14	16	0
3	15	60	0
3	15	25	0
3	15	22	0
3	18	25	0
3	18	23	0
3	18	13	0
3	18	31	0
3	19	32	0
3	19	12	0
3	19	22	0
3	20	12	0
3	20	16	0
3	20	10	0
3	20	8	0
3	21	53	0
3	21	38	0
3	21	42	0
3	21	56	0
3	21	55	0
3	21	51	0
3	21	46	0
3	21	56	0
3	21	56	0
3	21	27	0
3	21	47	0
3	21	50	0
3	21	23	0
3	21	24	0
3	21	42	0
3	21	17	0
3	21	31	0
3	21	31	0
3	21	33	0
3	21	38	0
3	21	51	0
3	21	29	0
3	21	23	0
3	21	26	0
3	21	13	0
3	21	20	0
3	21	19	0
3	21	22	0
3	21	29	0
3	21	29	0
3	21	32	0
3	21	27	0
3	21	25	0
3	21	23	0
3	21	27	0
3	21	25	0
3	21	11	0
3	21	25	0
3	21	15	0
3	21	29	0
3	21	28	0
3	21	21	0
3	21	30	0
3	21	24	0
3	21	23	0
3	21	25	0
3	21	13	0
3	21	17	0
3	21	30	0
3	21	17	0
3	21	19	0
3	21	16	0
3	21	24	0
3	21	22	0
3	21	23	0
3	21	23	0
3	21	20	0
3	21	21	0
3	21	11	0

3	21	15	0
3	21	11	0
3	21	14	0
3	21	13	0
3	21	17	0
3	21	13	0
3	21	15	0
3	21	16	0
3	21	13	0
3	21	13	0
3	21	10	0
3	21	12	0
3	21	13	0
3	21	7	0
3	21	8	0
3	22	9	0
3	26	16	0
3	27	10	0
3	28	29	0
3	28	20	0
4	1	24	0
4	2	11	0
4	4	20	0
4	4	22	0
4	5	15	0
4	5	24	0
4	6	25	0
4	6	21	0
4	6	28	0
4	6	13	0
4	7	17	0
4	8	17	0
4	8	13	0
4	8	4	0
4	10	41	0
4	11	42	0
4	11	56	0
4	11	20	0
4	11	12	0
4	11	6	0
4	12	6	0
4	13	44	0
4	13	33	0
4	14	40	0
4	15	20	0
4	20	44	0
4	20	34	0
4	20	36	0
4	20	12	0
4	21	46	0
4	21	35	0
4	22	37	0
4	22	38	0
4	22	27	0
4	22	51	0
4	22	21	0
4	22	20	0
5	2	13	0
5	2	12	0
5	3	17	0
5	3	12	0
5	4	15	0
5	4	16	0
5	4	26	0
5	4	26	0
5	4	18	0
5	4	12	0
5	5	14	0
5	5	9	0
5	5	11	0
5	5	13	0
5	6	10	0
5	7	8	0
5	7	11	0
5	9	13	0
5	10	10	0
5	12	11	0
5	14	28	0
5	14	27	0
5	14	10	0
5	14	22	0
5	15	21	0
5	15	19	0
5	15	12	0
5	15	12	0
5	15	23	0
5	15	18	0
5	16	24	0
5	16	18	0
5	17	10	0
5	20	19	0
6	1	18	0
6	1	13	0
6	2	63	0
6	2	11	0
6	2	11	0
6	2	10	0
6	2	8	0
6	3	12	0
6	3	19	0
6	3	21	0
6	3	15	0
6	3	19	0
6	6	20	0
6	7	13	0
6	7	23	0
6	7	22	0
6	8	32	0
6	9	13	0
6	11	59	0
6	11	25	0
6	11	16	0
6	11	15	0
6	11	11	0
6	13	9	0
6	16	38	0
6	16	32	0
6	16	57	0
6	16	62	0
6	16	42	0
6	17	62	0
6	17	50	0
6	17	24	0
6	18	60	0
6	18	36	0
6	18	41	0
6	18	60	0
6	18	40	0
6	20	30	0
7	2	18	0
7	3	39	0
7	3	28	0
7	3	39	0
7	3	32	0
7	4	26	0
7	4	29	0
7	5	51	0
7	5	19	0
7	6	22	0
7	6	30	0
7	6	21	0
7	6	11	0
7	7	40	0
7	7	41	0
7	7	28	0
7	7	58	0
7	7	27	0
7	7	20	0
7	7	25	0
7	7	25	0
7	7	30	0
7	7	40	0
7	7	21	0
7	7	23	0
7	7	48	0
7	7	30	0
7	7	31	0
7	7	23	0
7	8	34	0
7	8	31	0
7	8	26	0
7	8	23	0
7	8	49	0
7	8	34	0
7	8	36	0
7	8	30	0
7	8	31	0
7	8	25	0
7	8	35	0
7	8	30	0
7	8	31	0
7	8	68	0
7	8	31	0
7	8	36	0
7	8	25	0
7	8	31	0
7	8	32	0
7	8	22	0
7	8	34	0
7	8	30	0
7	9	25	0
7	10	19	0
7	10	17	0
7	10	20	0
7	11	33	0
7	11	28	0
7	11	27	0
7	11	28	0
7	11	32	0
7	11	17	0
7	12	37	0
7	12	30	0
7	12	32	0
7	14	21	0
7	14	36	0
7	14	30	0
7	14	27	0
7	14	17	0
7	14	23	0
7	14	15	0
7	14	7	0
7	15	55	0
7	15	39	0
7	15	30	0

7	15	68	0	7	18	55	0	8	3	29	0	8	7	23	0
7	15	27	0	7	18	56	0	8	3	22	0	8	7	24	0
7	15	27	0	7	18	39	0	8	3	20	0	8	7	18	0
7	15	22	0	7	18	33	0	8	3	10	0	8	7	21	0
7	15	25	0	7	18	34	0	8	3	18	0	8	7	22	0
7	16	30	0	7	18	26	0	8	3	24	0	8	7	18	0
7	16	43	0	7	18	57	0	8	3	20	0	8	7	18	0
7	16	51	0	7	18	28	0	8	3	21	0	8	7	17	0
7	16	66	0	7	18	39	0	8	4	57	0	8	7	12	0
7	16	43	0	7	18	54	0	8	4	48	0	8	7	9	0
7	16	44	0	7	18	37	0	8	4	30	0	8	8	60	0
7	16	22	0	7	18	28	0	8	4	35	0	8	8	43	0
7	16	30	0	7	18	24	0	8	4	23	0	8	8	56	0
7	16	24	0	7	18	36	0	8	4	26	0	8	8	55	0
7	16	42	0	7	18	28	0	8	4	30	0	8	8	64	0
7	16	36	0	7	18	22	0	8	4	26	0	8	8	31	0
7	16	28	0	7	18	35	0	8	4	30	0	8	8	47	0
7	16	22	0	7	18	36	0	8	4	28	0	8	8	49	0
7	16	30	0	7	18	34	0	8	4	23	0	8	8	27	0
7	16	29	0	7	18	35	0	8	4	38	0	8	8	28	0
7	16	36	0	7	18	32	0	8	5	23	0	8	8	33	0
7	16	25	0	7	18	23	0	8	6	28	0	8	8	24	0
7	16	27	0	7	18	24	0	8	6	34	0	8	8	26	0
7	16	25	0	7	18	29	0	8	6	31	0	8	8	26	0
7	16	40	0	7	18	20	0	8	6	40	0	8	8	20	0
7	17	56	0	7	18	29	0	8	6	34	0	8	8	20	0
7	17	50	0	7	19	62	0	8	6	38	0	8	8	21	0
7	17	58	0	7	19	57	0	8	6	34	0	8	9	54	0
7	17	60	0	7	19	58	0	8	6	36	0	8	9	27	0
7	17	57	0	7	19	60	0	8	6	22	0	8	9	57	0
7	17	53	0	7	19	60	0	8	6	16	0	8	9	23	0
7	17	55	0	7	19	36	0	8	6	9	0	8	9	56	0
7	17	57	0	7	19	46	0	8	7	31	0	8	9	28	0
7	17	56	0	7	19	61	0	8	7	28	0	8	9	62	0
7	17	54	0	7	19	44	0	8	7	58	0	8	9	22	0
7	17	52	0	7	19	62	0	8	7	55	0	8	9	58	0
7	17	27	0	7	19	43	0	8	7	52	0	8	9	23	0
7	17	29	0	7	21	47	0	8	7	47	0	8	9	57	0
7	17	21	0	8	1	35	0	8	7	55	0	8	9	22	0
7	17	31	0	8	1	27	0	8	7	26	0	8	9	24	0
7	17	34	0	8	2	31	0	8	7	62	0	8	9	28	0
7	17	17	0	8	3	56	0	8	7	17	0	8	9	52	0
7	18	61	0	8	3	61	0	8	7	51	0	8	9	22	0
7	18	55	0	8	3	59	0	8	7	27	0	8	9	58	0
7	18	57	0	8	3	56	0	8	7	40	0	8	9	25	0
7	18	35	0	8	3	41	0	8	7	26	0	8	9	51	0
7	18	58	0	8	3	63	0	8	7	22	0	8	9	15	0
7	18	57	0	8	3	29	0	8	7	23	0	8	9	30	0
7	18	54	0	8	3	29	0	8	7	24	0	8	9	47	0
7	18	56	0	8	3	23	0	8	7	19	0	8	9	26	0
7	18	57	0	8	3	19	0	8	7	32	0	8	9	63	0

8	9	22	0	8	13	33	0	9	3	29	0	9	8	51	0
8	9	24	0	8	13	32	0	9	3	40	0	9	8	47	0
8	9	22	0	8	13	30	0	9	3	29	0	9	8	30	0
8	10	57	0	8	13	60	0	9	4	55	0	9	8	46	0
8	10	65	0	8	13	51	0	9	4	33	0	9	8	29	0
8	10	52	0	8	13	35	0	9	4	55	0	9	8	18	0
8	10	59	0	8	13	32	0	9	4	22	0	9	8	22	0
8	10	55	0	8	14	48	0	9	5	49	0	9	8	17	0
8	10	52	0	8	14	66	0	9	5	50	0	9	8	37	0
8	10	63	0	8	14	51	0	9	5	48	0	9	8	28	0
8	10	58	0	8	14	58	0	9	5	40	0	9	8	24	0
8	10	34	0	8	14	47	0	9	5	37	0	9	8	29	0
8	10	54	0	8	14	39	0	9	5	33	0	9	8	23	0
8	10	30	0	8	15	60	0	9	6	69	0	9	8	23	0
8	10	32	0	8	15	58	0	9	6	35	0	9	8	31	0
8	10	30	0	8	15	30	0	9	6	38	0	9	8	18	0
8	10	34	0	8	15	50	0	9	6	61	0	9	8	24	0
8	10	27	0	8	15	65	0	9	7	62	0	9	8	33	0
8	10	28	0	8	15	52	0	9	7	61	0	9	8	23	0
8	10	18	0	8	15	57	0	9	7	51	0	9	8	17	0
8	10	24	0	8	15	51	0	9	7	38	0	9	8	27	0
8	10	30	0	8	15	51	0	9	7	44	0	9	8	27	0
8	10	25	0	8	15	30	0	9	7	30	0	9	8	8	0
8	10	26	0	8	15	48	0	9	7	60	0	9	8	24	0
8	10	25	0	8	15	34	0	9	7	33	0	9	8	22	0
8	10	31	0	8	16	57	0	9	7	35	0	9	8	28	0
8	10	31	0	8	16	55	0	9	7	33	0	9	9	67	0
8	10	22	0	8	16	47	0	9	7	24	0	9	9	53	0
8	10	19	0	8	16	64	0	9	7	26	0	9	9	56	0
8	10	13	0	8	16	52	0	9	7	31	0	9	9	57	0
8	10	17	0	8	16	68	0	9	7	33	0	9	9	54	0
8	11	55	0	8	16	46	0	9	7	33	0	9	9	53	0
8	11	57	0	8	16	48	0	9	7	30	0	9	9	34	0
8	11	44	0	8	16	62	0	9	7	25	0	9	9	25	0
8	11	59	0	8	16	41	0	9	7	33	0	9	9	21	0
8	11	46	0	8	16	35	0	9	7	30	0	9	9	17	0
8	11	34	0	8	16	38	0	9	7	26	0	9	9	19	0
8	11	50	0	9	1	39	0	9	7	25	0	9	9	14	0
8	11	21	0	9	2	61	0	9	7	22	0	9	9	18	0
8	11	26	0	9	2	44	0	9	7	27	0	9	10	57	0
8	11	28	0	9	2	49	0	9	8	62	0	9	10	55	0
8	11	41	0	9	2	57	0	9	8	56	0	9	10	60	0
8	11	29	0	9	2	62	0	9	8	60	0	9	10	53	0
8	12	24	0	9	2	49	0	9	8	45	0	9	10	55	0
8	13	55	0	9	2	31	0	9	8	66	0	9	10	63	0
8	13	54	0	9	2	44	0	9	8	44	0	9	10	44	0
8	13	56	0	9	2	38	0	9	8	59	0	9	10	58	0
8	13	42	0	9	2	43	0	9	8	30	0	9	10	51	0
8	13	57	0	9	3	55	0	9	8	23	0	9	10	39	0
8	13	53	0	9	3	54	0	9	8	54	0	9	10	32	0
8	13	46	0	9	3	41	0	9	8	28	0	9	10	33	0

9	10	28	0
9	10	35	0
9	10	30	0
9	10	26	0
9	10	23	0
9	10	24	0
9	10	21	0
9	10	21	0
9	11	61	0
9	11	32	0
9	11	53	0
9	11	67	0
9	11	54	0
9	11	33	0
9	11	62	0
9	11	54	0
9	11	33	0
9	11	36	0
9	11	24	0
9	11	23	0
9	11	21	0
9	11	32	0
9	11	21	0
9	11	37	0
9	11	18	0
9	11	31	0
9	11	32	0
9	11	24	0
9	11	30	0
9	11	12	0
9	12	54	0
9	12	26	0
9	12	27	0
9	12	55	0
9	12	26	0
9	12	36	0
9	12	53	0
9	12	30	0
9	12	30	0
9	12	62	0
9	12	29	0
9	12	30	0
9	12	30	0
9	12	31	0
9	12	22	0
9	12	27	0
9	12	29	0
9	12	38	0
9	13	57	0
9	13	33	0
9	13	48	0
9	13	32	0
9	13	61	0
9	13	50	0
9	13	52	0
9	13	49	0
9	13	30	0
9	13	34	0
9	13	31	0
9	13	27	0
9	13	27	0
9	13	27	0
9	13	31	0
9	14	57	0
9	14	51	0
9	14	63	0
9	14	62	0
9	14	55	0
9	14	56	0
9	15	35	0
9	15	43	0
9	15	28	0
9	15	53	0
9	16	33	0
9	17	35	0
9	18	62	0
9	18	45	0
9	18	55	0
9	18	34	0
9	18	50	0
9	18	55	0
9	18	62	0
9	18	53	0
9	18	54	0
9	18	15	0
9	18	34	0
9	19	62	0
9	19	54	0
9	19	55	0
9	19	56	0
9	19	48	0
9	19	65	0
9	19	36	0
9	19	26	0
9	19	40	0
9	19	41	0
9	19	29	0
9	19	23	0

6.2 Program listing example

```
program MinLeslie (input, output);
   const
      NMax       = 14;  {Maximum number of parameters}
      MMax       = 150; {Max number of data points}
      MaxBin     = 15;  {Max number of histogram bins}
      BinSize    = 5.0; {In millimeters}
      MaxYear    = 8;   {Maximum age class in years}
      MaxYearM1  = 7;   {MaxYear - 1}
      MaxMonths  = 10;  {Max number of months simulated}
      FirstYear  = 77;
      ThisYear   = 82;
   type
      ArrN    = array [1..NMax] of real;
      ArrM    = array [1..MMax] of real;
      ArrNxN  = array [1..NMax, 1..NMax] of real;
      ArrMxN  = array [1..MMax, 1..NMax] of real;
      ArrNxM  = array [1..NMax, 1..MMax] of real;
      PrOptionType = (
         XTrace,
         FTrace,
         SqFTrace,
         ItrStatusTrace,
         NewJTrace,
         NewJInvTrace,
         StepLenTrace,
         StepTypeTrace);
      PrSetType   = set of PrOptionType;
      MonthRange  = 1..12;
      YearRange   = FirstYear..ThisYear;
      RepMatType  =
         record
         Fecundity    : real;
         ProbSurvive  : real
         end;
      PopType =
         record
         RecruMonth  : MonthRange;
         CurrMonth   : MonthRange;
         CurrYear    : YearRange;
         AgeDistr    : array [1..MaxYear] of real
         end;
```

```

      GrowthType =
         record
         JuvSlope        : real;
         JuvIntercept    : real;
         MatureSlope     : real;
         MatureIntercept : real;
         HalfRange       : real
         end;
      ModPrmType =
         record
         InitAgeDistr : PopType;
         GrowthParams : GrowthType;
         RepMat       : RepMatType
         end;
      SizeArray = array [1..MaxBin] of real;
      ListOfSizeDistr = array [1..MaxMonths] of
         record
         Month : MonthRange;
         Year  : YearRange;
         Size  : SizeArray
         end;
   var
      X             : ArrN; {Vector of parameter values}
      F             : ArrM; {Vector of errors}
      SqF           : real; {Squared error}
      DeltaX        : real; {Finite difference interval}
      MaxDist       : real; {Max distance to optimum}
      Acc           : real; {Desired accuracy}
      MaxCalls      : integer;
      PrintOptions  : PrSetType;
      M             : integer;
      ModelParams   : ModPrmType;
      RealData      : ListOfSizeDistr;
      SimData       : ListOfSizeDistr;
      NumMonths     : integer;
```

```
procedure GetMinOptions (
      var DeltaX   : real;
      var MaxDist  : real;
      var Acc      : real;
      var MaxCalls : integer);
   var
      Response : char;
   begin
   write ('Default minimization values? (y or n):  ');
   read (Response);
   writeln;
   if Response in ['y','Y'] then
      begin
      DeltaX := 0.001;
      MaxDist := 10.0;
      Acc := 0.001
      end
   else
      begin
      write ('Delta X:  ');
      readln (DeltaX);
      write ('Maximum distance to minimum:  ');
      readln (MaxDist);
      write ('Accuracy:  ');
      readln (Acc)
      end;
   write ('Maximum calls:  ');
   readln (MaxCalls)
   end;
```

```
procedure GetModelParams (var ModelParams :ModPrmType);
   var
      Data        : text;
      I           : integer;
      StandardDev : real;
      FileName    : string;
   begin
   writeln;
   write ('File of initial model parameters? ');
   readln (FileName);
   FileName := concat (FileName, '.TEXT');
   reset (Data, FileName);
   with ModelParams do
      begin
      with RepMat do
         begin
         read (Data, Fecundity);
         readln (Data);
         read (Data, ProbSurvive);
         end;
      with InitAgeDistr do
         begin
         read (Data, RecruMonth);
         for I := 1 to MaxYear do
            read (Data, AgeDistr [I])
         end;
      with GrowthParams do
         begin
         read (Data, JuvSlope, JuvIntercept,
         MatureSlope, MatureIntercept, StandardDev);
         HalfRange := StandardDev * sqrt (12) / 2.0
         end
      end;
      close (Data)
   end;
```

```

procedure GetRealData (
      var NumMonths : integer
      var RealData  : ListOfSizeDistr);
   var
      Data     : text;
      FileName : string;
      I, J     : integer;
   begin
   writeln;
   write ('File of real data? ');
   readln (FileName);
   FileName := concat (FileName, '.TEXT');
   reset (Data, FileName);
   read (Data, NumMonths);
   for I := 1 to NumMonths do
      with RealData [I] do
         begin
         read (Data, Month, Year);
         for J := 1 to MaxBin do
            read (Data, Size [J])
         end;
   close (Data)
   end;
```

```
procedure GetPrintOptions (var PrintOptions :PrSetType);
   var
      Response : char;

procedure GetX;
   begin
   write ('Trace X?  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      PrintOptions := PrintOptions + [XTrace]
   end;

procedure GetF;
   begin
   write ('Trace F?  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      PrintOptions := PrintOptions + [FTrace]
   end;

procedure GetSqF;
   begin
   write ('Trace error?  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      PrintOptions := PrintOptions + [SqFTrace]
   end;

procedure GetItrStatus;
   begin
   write ('Trace IterStatus?  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      PrintOptions := PrintOptions + [ItrStatusTrace]
   end;

procedure GetNewJ;
   begin
   write ('Trace new Jacobian?  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      PrintOptions := PrintOptions + [NewJTrace]
   end;
```

```
procedure GetNewJInv;
   begin
   write ('Trace new inverse Jacobian?  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      PrintOptions := PrintOptions + [NewJInvTrace]
   end;

procedure GetStepLen;
   begin
   write ('Trace maximum step length?  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      PrintOptions := PrintOptions + [StepLenTrace]
   end;

procedure GetStepType;
   begin
   write ('Trace type of step?  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      PrintOptions := PrintOptions + [StepTypeTrace]
   end;
```

```
     begin {GetPrintOptions}
     PrintOptions := [];
     write ('Trace? (y or n):  ');
     read (Response);
     writeln;
     if Response in ['y', 'Y'] then
        begin
        write ('Default trace? (y or n):  ');
        read (Response);
        writeln;
        if Response in ['y', 'Y'] then
           PrintOptions := [XTrace, SqFTrace,
           ItrStatusTrace, StepLenTrace, StepTypeTrace]
        else
           begin
           GetX;
           GetF;
           GetSqF;
           GetItrStatus;
           GetNewJ;
           GetNewJInv;
           GetStepLen;
           GetStepType
           end
        end
     end;
```

```
procedure FromXVector (
          var ModelParam : ModPrmType;
          X              : ArrN);
     {  This procedure converts the Model 5 parameters  }
     {  from their logical structure into the vector X  }
     {  for the minimization routine.  Performs  }
     {  scaling on the parameters JuvIntercept and  }
     {  MatureIntercept.  }
     var
        I : integer;
     begin
     with ModelParam do
        begin
        with InitAgeDistr do
           for I := 1 to MaxYear do
              AgeDistr [I] := abs (X [I]);
        with GrowthParams do
           begin
           JuvSlope := X [9];
           JuvIntercept := 100.0 * X [10];
           MatureSlope := X [11];
           MatureIntercept := 100.0 * X [12]
           end;
        with RepMat do
           begin
           Fecundity := abs (X [13]);
           ProbSurvive := X [14]
           end
        end
     end;
```

```

procedure ToXVector (
        ModelParam : ModPrmType;
        var X      : ArrN);
    {  This procedure converts the Model 5 parameters  }
    {  from the X vector into their logical structure  }
    {  for the simulation routine.  Performs the  }
    {  inverse scaling of FromXVector.  }
    var
        I : integer;
    begin
    with ModelParam do
        begin
        with InitAgeDistr do
            for I := 1 to MaxYear do
                X [I] := AgeDistr [I];
        with GrowthParams do
            begin
            X [9] := JuvSlope;
            X [10] := 0.01 * JuvIntercept;
            X [11] := MatureSlope;
            X [12] := 0.01 * MatureIntercept
            end;
        with RepMat do
            begin
            X [13] := Fecundity;
            X [14] := ProbSurvive
            end
        end
    end;
```

```
procedure FromFVector (
      var SimData : ListOfSizeDistr;
      RealData    : ListOfSizeDistr;
      F           : ArrM);
   {  This procedure recovers the simulated data,  }
   {  SimData, from RealData and the error vector F.  }
   var
      I, J, K : integer;
   begin
   K := 1;
   for I := 1 to NumMonths do
      for J := 1 to MaxBin do
         begin
         SimData [I].Size [J] :=
         F [K] + RealData [I].Size [J];
         K := K + 1
         end
   end;

procedure ToFVector (
      var SimData  : ListOfSizeDistr;
      var RealData : ListOfSizeDistr;
      var F        : ArrM);
   {  This procedure calculates the error vector,  }
   {  F, as the difference between the real data  }
   {  and the simulated data.  }
   {  SimData and RealData are called by reference  }
   {  for purposes of efficiency.  }
   var
      I, J, K : integer;
   begin
   K := 1;
   for I := 1 to NumMonths do
      for J := 1 to MaxBin do
         begin
         F [K] :=
         SimData [I].Size [J] - RealData [I].Size [J];
         K := K + 1
         end
   end;
```

```
procedure CalcSizeDistr (
     CurrentAgeDist : PopType;
     GrowthParams   : GrowthType;
     var SizeDistr  : SizeArray);
  {  This procedure calculates the size distribution  }
  {  from the current age distribution and the  }
  {  growth parameters.  }
  var
     I            : integer;
     BinNum       : integer;
     BNum         : integer;
     MonthlyAge   : integer;
     LowBin       : integer;
     LBin         : integer;
     HighBin      : integer;
     HBin         : integer;
     InverseRange : real;
     BinFraction  : real;
     HRange       : real;
     DeltaMonth   : integer;
     JuvSize      : real;
     MatureSize   : real;
     Size         : real;
```

```
begin
for I := 1 to MaxBin do
   SizeDistr [I] := 0.0;
with CurrentAgeDistr, GrowthParams do
   begin
   if CurrMonth < RecruMonth then
      DeltaMonth := CurrMonth - RecruMonth + 12
   else
      DeltaMonth := CurrMonth - RecruMonth;
   for I := 1 to MaxYear do
      begin
      MonthlyAge := DeltaMonth + 12 * (I - 1);
      JuvSize :=
      JuvSlope * MonthlyAge + JuvIntercept;
      MatureSize :=
      MatureSlope * MonthlyAge + MatureIntercept;
      {WRITELN ('MonthlyAge = ', MonthlyAge);}
      {WRITE ('JuvSize = ', JuvSize :10:5);}
      {WRITELN (' MatureSize = ', MatureSize :10:5);}
      if MatureSize > JuvSize then
         Size := JuvSize
      else
         Size := MatureSize;
      if Size > HalfRange then
         HRange := HalfRange
      else
         HRange := Size;
      {WRITE ('Size = ', Size :10:3);}
      {WRITELN (' HRange = ', HRange :10:3);}
      InverseRange := 1.0 / (2.0 * HRange);
      BinFraction := BinSize * InverseRange;
      LowBin :=
      trunc ((Size - HRange) / BinSize) + 1;
      HighBin :=
      trunc ((Size + HRange) / BinSize) + 1;
      {WRITE ('LowBin = ', LowBin);}
      {WRITELN (' HighBin = ', HighBin);}
```

```

          if LowBin = HighBin then
             begin
             if LowBin > MaxBin then
                LowBin := MaxBin;
             SizeDistr [LowBin] :=
             SizeDistr [LowBin] + AgeDistr [I];
             end
          else
             begin
             if LowBin > MaxBin then
                LBin := MaxBin
             else
                LBin := LowBin;
             {WRITE ('LBin = ', LBin);}
             SizeDistr [LBin] := SizeDistr [LBin] +
             AgeDistr [I] * InverseRange *
             (LowBin * BinSize - Size + HRange);
             {WRITELN (' SizeDistr [LBin] = ',}
             {SizeDistr [LBin] :10:3);}
             for BinNum :=
                (LowBin + 1) to (HighBin - 1) do
                begin
                if BinNum > MaxBin then
                   BNum := MaxBin
                else
                   BNum := BinNum;
                {WRITE ('BNum = ', BNum);}
                SizeDistr [BNum] := SizeDistr [BNum] +
                AgeDistr [I] * BinFraction;
                {WRITELN (' SizeDistr [BNum] = ',}
                {SizeDistr [BNum] :10:3)}
                end;
             if HighBin > MaxBin then
                HBin := MaxBin
             else
                HBin := HighBin;
             {WRITE ('HBin = ', HBin);}
             SizeDistr [HBin] := SizeDistr [HBin] +
             AgeDistr [I] * InverseRange *
             (Size + HRange - (HighBin - 1) * BinSize);
             {WRITELN (' SizeDistr [HBIN] = ',}
             {SizeDistr [HBIN] :10:3)}
             end
          end {for}
       end {with};
    end;
```

```
procedure StepMonth (
      CurrentAgeDistr   : PopType;
      RepMat            : RepMatType;
      var NextAgeDistr  : PopType);
   {  *** Model 5 ***  }
   {  This procedure steps the simulation through  }
   {  one month.  If it is a recruitment month it  }
   {  calculates the next age distribution from  }
   {  the current age distribution, the fecundity  }
   {  row of the Leslie matrix, and the off diagonal  }
   {  uniform survival probability.  If it is not  }
   {  a recruitment month it uses only the diagonal  }
   {  survival probabilities.  }
   var
      Temp  : real;
      I     : integer;
   begin
   NextAgeDistr := CurrentAgeDistr;
   if CurrentAgeDistr.CurrMonth = 12 then
      begin
      NextAgeDistr.CurrMonth := 1;
      NextAgeDistr.CurrYear :=
      CurrentAgeDistr.CurrYear + 1
      end
   else
      NextAgeDistr.CurrMonth :=
      CurrentAgeDistr.CurrMonth + 1;
   with RepMat do
      begin
      if NextAgeDistr.CurrMonth =
         NextAgeDistr.RecruMonth then
         begin
         Temp := 0.0;
         for I := 3 to MaxYear do
            Temp := Temp +
            Fecundity * CurrentAgeDistr.AgeDistr [I];
         NextAgeDistr.AgeDistr [1] := Temp;
         for I := 2 to MaxYear do
            NextAgeDistr.AgeDistr [I] :=
            CurrentAgeDistr.AgeDistr [I-1] * ProbSurvive
         end
      else
         for I := 1 to MaxYear do
            NextAgeDistr.AgeDistr [I] :=
            CurrentAgeDistr.AgeDistr [I] * ProbSurvive
      end
   end;
```

```
procedure Simulate (
        NumMonths   : integer;
        RealData    : ListOfSizeDistr;
        ModelParams : ModPrmType;
        var SimData : ListOfSizeDistr);
   {  This procedure simulates the Dendraster system  }
   {  through NumMonths months.  It calculates  }
   {  SimData from the model parameters.  RealData  }
   {  is only used to supply the month and year  }
   {  the real data was acquired.  Values of the  }
   {  real data are not used.  }
   var
      CurrentAgeDistr : PopType;
      NextAgeDistr    : PopType;
      I               : integer;
   begin
   writeln;
   CurrentAgeDistr := ModelParams.InitAgeDistr;
   I := 1;
   SimData [I].Month := RealData [I].Month;
   SimData [I].Year  := RealData [I].Year;
   CalcSizeDistr (CurrentAgeDistr,
   ModelParams.GrowthParams, SimData [I].Size);
   I := 2;
   while I <= NumMonths do
      begin
      StepMonth (CurrentAgeDistr,
      ModelParams.RepMat, NextAgeDistr);
      CurrentAgeDistr := NextAgeDistr;
      if (CurrentAgeDistr.CurrMonth =
         RealData [I].Month) and
         (CurrentAgeDistr.CurrYear =
         RealData [I].Year) then
         begin
         SimData [I].Month := RealData [I].Month;
         SimData [I].Year  := RealData [I].Year;
         CalcSizeDistr (CurrentAgeDistr,
         ModelParams.GrowthParams, SimData [I].Size);
         I := I + 1
         end
      end
   end;
```

```
procedure Evaluate (var F : ArrM; X : ArrN);
   {  This procedure is called by the minimization  }
   {  procedure.  It calculates the error vector F  }
   {  from the vector of parameters, X.  }
   begin
   FromXVector (ModelParams, X);
   Simulate (NumMonths, RealData, ModelParams, SimData);
   ToFVector (SimData, RealData, F)
   end;
```

```

procedure FileFinal (X : ArrN; F : ArrM; SqF : real);
   var
      Data         : text;
      Response     : char;
      Title        : string;
      FileName     : string;
      I, J         : integer;
      StandardDev : real;
   {  This procedure files the final results for  }
   {  plotting and documentation.  }
   begin
   write ('File final results? (y or n):  ');
   read (Response);
   writeln;
   if Response in ['Y', 'y'] then
      begin
      FromFVector (SimData, RealData, F);
      FromXVector (ModelParams, X);
      writeln ('Title for documentation?  ');
      readln (Title);
      write ('Name of output file?  ');
      readln (FileName);
      FileName := concat (FileName, '.TEXT');
      rewrite (Data, FileName);
      writeln (Data, NumMonths);
      for I := 1 to NumMonths do
         with SimData [I] do
            begin
            writeln (Data, Month :4, Year :4);
            for J := 1 to MaxBin do
               begin
               write (Data, Size [J] :13);
               if (J mod 6 = 0) then
                  writeln (Data)
               end;
            writeln (Data)
            end;
      writeln (Data);
      writeln (Data, 'Squared error = ', SqF :15);
      writeln (Data, Title);
```

```
      with ModelParams do
         begin
         with RepMat do
            begin
            writeln (Data, Fecundity :13:5);
            writeln (Data, ProbSurvive :13:5)
            end;
         with InitAgeDistr do
            begin
            writeln (Data, RecruMonth);
            for I := 1 to MaxYear do
               begin
               write (Data, AgeDistr [I] :13:5);
               if (I mod 6 = 0) then
                  writeln (Data)
               end;
            writeln (Data)
            end;
         with GrowthParams do
            begin
            StandardDev := HalfRange * 2.0 / sqrt (12);
            writeln (Data, JuvSlope :13:5);
            writeln (Data, JuvIntercept :13:5);
            writeln (Data, MatureSlope :13:5);
            writeln (Data, MatureIntercept :13:5);
            writeln (Data, StandardDev :13:5)
            end
         end;
      close (Data, lock)
      end {if}
   end;
```

```
procedure Minimize (
      N, M      : integer;
      var X     : ArrN; {Vector of parameter values}
      var F     : ArrM; {Vector of errors}
      var SqF   : real; {Squared error}
      XStepSize : real; {Finite difference interval}
      MaxDist   : real; {Maximum distance to optimum}
      Accuracy     : real;
      MaxCalls     : integer;
      PrintOptions : PrSetType);
   {  This procedure minimizes the two-norm of  }
   {  the vector of errors, F, over the parameter  }
   {  space X.  }
   type
      ArrNInt = array [1..NMax] of integer;
      LoopStsType = (
         Continue,
         MinPredicted,
         ToleranceMet,
         HighResiduals,
         TestNumCalls,
         TooManyCalls);
      IterStsType = (
         FirstTime,
         ComputeNewJ,
         Normal,
         StepLenUpdate,
         StepDirl,
         MinNear);
```

```
    {  In the following mnemonics Sq denotes the  }
    {  square of a quantity, either the square of  }
    {  a scalar or the two-norm of a vector.  Dot  }
    {  denotes the dot or scalar product of two  }
    {  vectors.  }
    var
       LoopStatus    : LoopStsType; {main loop status}
       IterStatus    : IterStsType; {Type of iteration}
       Jacobian      : ArrMxN;
       JInverse      : ArrNxM;  {H, Inverse of Jacobian}
       OldX          : ArrN;    {Old parameter values}
       OldF          : ArrM;    {Old error values}
       EstF          : ArrM;    {Estimate of F}
       OldSqF        : real;    {Square of OldF}
       EstSqF        : real;    {Square of EstF}
       SpanCount     : ArrNInt; {C vector}
       OrthogDir     : ArrNxN;  {D vector of directions}
       NumCalls      : integer; {Number of simulations}
       StuckHighCount : integer;
       SqXStepSize    : real; {Square of XStepSize}
       SqMaxDist      : real; {Square of MaxDist}
       SqMaxStepSize  : real;
       StepIncrFactor : real; {Step increment factor}
       StepIndex     : integer;
       StpDir        : ArrN; {Steepest descent direction}
       NwtDir        : ArrN; {Gauss-Newton direction}
       Delta         : ArrN; {Update to X vector}
       SqStpDir      : real; {Square of StpDir}
       SqNwtDir      : real; {Square of NwtDir}
       NwtDotStp     : real; {NwtDir * StpDir}
       SqDelta       : real; {Square of Delta}
       SqDelta4th    : real; {SqDelta / 4.0}
       DeltaDotD1    : real; {Delta * OrthogDir [1]}
       NwtCoef       : real; {Coefficient of NwtDir}
       StpCoef       : real; {Coefficient of StpDir}
       TwoMu         : real;
       SqMuStpDir    : real;
       J             : integer;
       {  Global constants  }
       NxNIdentity         : ArrNxN;
       InitSpanCountVector : ArrNInt;
```

```
procedure Invert (var A :ArrNxN; N :integer);
   {  This procedure inverts the N x N matrix A  }
   {  using Gaussian elimination with partial  }
   {  pivoting.  If the matrix A is nearly singular  }
   {  it sets A equal to the identity matrix and  }
   {  issues a warning.  If the matrix is ill-  }
   {  conditioned it only issues awarning.  }
   const
      Epsilon   = 1.0E-15;
      CondLimit = 1.0E6;
   var
      I, J, K       : integer;
      Sub           : ArrNInt;
      Index         : integer;
      LargestCoeff  : real;
      Temp          : real;
      Pivot         : real;
      B, C          : ArrNxN;
      NormC         : real;
      CondNum       : real;
      Singular      : boolean;

function Norm (var A :ArrNxN; N :integer) :real;
   {  Computes the maximum row-sum norm of a matrix.  }
   var
      I, J   :integer;
      RowSum :real;
      Temp   :real;
   begin
   Temp := 0.0;
   for I := 1 to N do
      begin
      RowSum := 0.0;
      for J := 1 to N do
         RowSum := RowSum + abs (A [I, J]);
      if Temp < RowSum then
         Temp := RowSum
      end;
   Norm := Temp
   end;
```

```
procedure GaussElim;
   {  Gaussian elimination with partial pivoting.  }
   {  Sub is a permutation vector to keep track of  }
   {  the row exchanges in the pivot.  }
   begin
   for I := 1 to N do
      Sub [I] := I;
   K := 1;
   Singular := false;
   while (K <= N - 1) and not Singular do
      begin
      LargestCoeff := 0.0;
      for I := K to N do
         begin
         Temp := abs (C [Sub [I], K]);
         if LargestCoeff < Temp then
            begin
            LargestCoeff := Temp;
            Index := I
            end
         end;
      if LargestCoeff = 0.0 then
         Singular := true
      else
         begin
         J := Sub [K];
         Sub [K] := Sub [Index];
         Sub [Index] := J;
         Pivot := C [Sub [K], K];
         if abs (Pivot) < Epsilon then
            Singular := true
         else
            begin
            for I := K + 1 to N do
               begin
               C [Sub [I], K] := -C [Sub [I], K] /
               Pivot;
               for J := K + 1 to N do
                  C [Sub [I], J] := C [Sub [I], J] +
                  C [Sub [I], K] * C [Sub [K], J]
               end;
            K := K + 1
            end
         end
      end;
      if abs (C [Sub[N], N]) < Epsilon then
         Singular := true
   end;
```

```
procedure Solve;
   {  Solve the upper triangular system with the  }
   {  columns of the identity matrix.  }
   begin
   B := NxNIdentity;
   for J := 1 to N do
      begin
      for K := 1 to N - 1 do
         for I := K + 1 to N do
            B [Sub [I], J] := B [Sub [I], J] +
            C [Sub [I], K] * B [Sub [K], J];
      A [N, J] := B [Sub [N], J] / C [Sub [N], N];
      for K := N - 1 downto 1 do
         begin
         A [K, J] := B [Sub [K], J];
         for I := K + 1 to N do
            A [K, J] := A [K, J] -
            C [Sub [K], I] * A [I, J];
         A [K, J] := A [K, J] / C [Sub [K], K]
         end
      end
   end;

   begin {Invert}
   C := A;
   NormC := Norm (C, N);
   GaussElim;
   if not Singular then
      begin
      Solve;
      CondNum := NormC * Norm (A, N);
      if CondNum > CondLimit then
         begin
         writeln (
         'Ill-condition detected in procedure Invert.');
         writeln ('Condition number = ', CondNum)
         end
      end
   else
      begin
      writeln (
      'Singularity detected in procedure Invert.');
      A := NxNIdentity
      end
   end {Invert};
```

```
procedure InitGlobalConstants (N :integer);
   {  This procedure initializes the global  }
   {  constants.  }
   var
      I, J : integer;
   begin
   for I := 1 to N do
      begin
      for J := 1 to N do
         NxNIdentity [I, J] := 0.0;
      NxNIdentity [I, I] := 1.0;
      InitSpanCountVector [I] := N - I + 1
      end
   end;

procedure SwapN (var A, B :ArrN);
   var
      Temp :ArrN;
   begin
   Temp := A;
   A := B;
   B := Temp
   end;

procedure SwapM (var A, B :ArrM);
   var
      Temp :ArrM;
   begin
   Temp := A;
   A := B;
   B := Temp
   end;

procedure Negate (var A :ArrM; M :integer);
   var
      I :integer;
   begin
   for I := 1 to M do
      A [I] := -A [I]
   end;
```

```
function Min (A, B :real) :real;
   begin
   if A < B then
      Min := A
   else
      Min := B
   end;

function Max (A, B :real) :real;
   begin
   if A > B then
      Max := A
   else
      Max := B
   end;

function Min3 (A, B, C :real) :real;
   begin
   if A < B then
      B := A;
   if B < C then
      Min3 := B
   else
      Min3 := C
   end;
```

```

procedure ATransposeA (
      var A :ArrMxN;
      var B :ArrNxN;
      N, M :integer);
   {  This procedure multiplies the transpose of  }
   {  the MxN matrix A  by itself, giving the square  }
   {  NxN matrix B.  }
   {  A is called by reference for efficiency.  }
   var
      I, J, K : integer;
      Temp    : real;
   begin
   for I := 1 to N do
      for J := 1 to N do
         begin
         Temp := 0.0;
         for K := 1 to M do
            Temp := Temp + A [K, I] * A [K, J];
         B [I, J] := Temp
         end
   end;

procedure MultNxNxM (
      var A : ArrNxN;
      var B : ArrMxN;
      var C : ArrNxM;
      N, M  : integer);
   {  This procedure multiplies the NxN matrix A  }
   {  by the transpose of the MxN matrix B,  }
   {  giving the NxM matrix C.  A and B  }
   {  are called by reference for efficiency.  }
   var
      I, J, K : integer;
      Temp    : real;
   begin
   for I := 1 to N do
      for J := 1 to M do
         begin
         Temp := 0.0;
         for K := 1 to N do
            Temp := Temp + A [I, K] * B [J, K];
         C [I, J] := Temp
         end
   end;
```

```

procedure PrintN (var A :ArrN; N :integer);
   var
      I : integer;
   begin
   for I := 1 to N do
      begin
      if I mod 5 = 1 then
         begin
         writeln;
         write (I :4)
         end;
      write (A [I] :12)
      end;
   writeln
   end;

procedure PrintM (var A :ArrM; M :integer);
   var
      I : integer;
   begin
   for I := 1 to M do
      begin
      if I mod 5 = 1 then
         begin
         writeln;
         write (I :4)
         end;
      write (A [I] :12)
      end;
   writeln
   end;

procedure PrintNxN (var A :ArrNxN; N :integer);
   var
      I, J : integer;
   begin
   for I := 1 to N do
      begin
      for J := 1 to N do
         begin
         if J mod 5 = 1 then
            begin
            writeln;
            write (I :4, J :4)
            end;
         write (A [I, J] :12)
         end;
      writeln
      end
   end;
```

```
procedure PrintMxN (var A :ArrMxN; N, M :integer);
   var
      I, J : integer;
   begin
   for I := 1 to M do
      begin
      for J := 1 to N do
         begin
         if J mod 5 = 1 then
            begin
            writeln;
            write (I :4, J :4)
            end;
         write (A [I, J] :12)
         end;
      writeln
      end
   end;

procedure PrintNxM (var A :ArrNxM; N, M :integer);
   var
      I, J : integer;
   begin
   for I := 1 to N do
      begin
      for J := 1 to M do
         begin
         if J mod 5 = 1 then
            begin
            writeln;
            write (I :4, J :4)
            end;
         write (A [I, J] :12)
         end;
      writeln
      end
   end;
```

```
procedure PrintIteration (
     NumCalls     : integer;
     var X        : ArrN;
     var F        : ArrM;
     SqF          : real;
     N, M         : integer;
     PrintOptions : PrSetType);
  begin
  if PrintOptions <> [] then
     begin
     writeln;
     writeln ('Call number:  ', NumCalls :6);
     if XTrace in PrintOptions then
        begin
        write ('X values:');
        PrintN (X, N)
        end;
     if FTrace in PrintOptions then
        begin
        write ('F values:');
        PrintM (F, M)
        end;
     if SqFTrace in PrintOptions then
        writeln ('Squared error:  ', SqF :12);
     if StepLenTrace in PrintOptions then
        if SqMaxStepSize < 0.0 then
           writeln (
           'Maximum step length not yet computed')
        else
           writeln ('Maximum step length:  ',
           sqrt (SqMaxStepSize) :12)
     end
  end;

procedure PrintFinal (
     NumCalls : integer;
     var X    : ArrN;
     var F    : ArrM;
     SqF      : real;
     N, M     : integer);
  begin
  writeln;
  writeln ('Call number:  ', NumCalls :6);
  write ('X values:');
  PrintN (X, N);
  {write ('F values:');}
  {PrintM (F, M);}
  writeln ('Squared error:  ', SqF :12)
  end;
```

```
procedure UpdateJacobian;
   var
      Delta        : ArrN; {X - OldX}
      Gamma        : ArrM; {F - OldF}
      DelMinusHGam : ArrN; {Delta - H * Gamma}
      GamMinusJDel : ArrM; {Gamma - Jacobian * Delta}
      DelTH        : ArrM; {Delta transpose * H}
      DelTHGam     : real; {DelTH * Gamma}
      Temp         : real;
      SqDelta      : real;
      Alpha        : real;
      JFactor      : real;
      DeltaCoef    : real;
      HFactor      : real;
      HCoef        : real;
      I, J         : integer;
   begin
   SqDelta := 0.0;
   for I := 1 to N do
      begin
      Delta [I] := X [I] - OldX [I];
      SqDelta := SqDelta + sqr (Delta [I]);
      end;
   for J := 1 to M do
      Gamma [J] := F [J] - OldF [J];
   for I := 1 to N do
      begin
      Temp := Delta [I];
      for J := 1 to M do
         Temp := Temp - JInverse [I, J] * Gamma [J];
      DelMinusHGam [I] := Temp
      end;
   for J := 1 to M do
      begin
      Temp := Gamma [J];
      for I := 1 to N do
         Temp := Temp - Jacobian [J, I] * Delta [I];
      GamMinusJDel [J] := Temp
      end;
```

```
   DelTHGam := 0.0;
   for J := 1 to M do
      begin
      Temp := 0.0;
      for I := 1 to N do
         Temp := Temp + Delta [I] * JInverse [I, J];
      DelTHGam := DelTHGam + Temp * Gamma [J];
      DelTH [J] := Temp
      end;
   if abs (DelTHGam) >= 0.1 * SqDelta then
      Alpha := 1.0
   else
      Alpha := 0.8;
   JFactor := Alpha / SqDelta;
   HFactor := Alpha /
   (Alpha * DelTHGam + (1.0 - Alpha) * SqDelta);
   for I := 1 to N do
      begin
      HCoef := HFactor * DelMinusHGam [I];
      for J := 1 to M do
         JInverse [I, J] :=
         JInverse [I, J] + HCoef * DelTH [J]
      end;
   for J := 1 to M do
      begin
      DeltaCoef := JFactor * GamMinusJDel [J];
      for I := 1 to N do
         Jacobian [J, I] :=
         Jacobian [J, I] + DeltaCoef * Delta [I];
      end
   end;
```

```
procedure CalcDirections;
   {  This procedure calculates the Newton  }
   {  direction and the steepest descent direction.  }
   var
      TempN : real;
      TempS : real;
      I, J  : integer;
   begin
   SqNwtDir := 0.0;
   SqStpDir := 0.0;
   NwtDotStp := 0.0;
   for I := 1 to N do
      begin
      TempN := 0.0;
      TempS := 0.0;
      for J := 1 to M do
         begin
         TempN := TempN - JInverse [I, J] * OldF [J];
         TempS := TempS - OldF [J] * Jacobian [J, I]
         end;
      SqNwtDir := SqNwtDir + sqr (TempN);
      SqStpDir := SqStpDir + sqr (TempS);
      NwtDotStp := NwtDotStp + TempN * TempS;
      NwtDir [I] := TempN;
      StpDir [I] := TempS
      end
   end;
```

```
procedure CalcSteepestMin;
   {  This procedure predicts the displacement to the  }
   {  the minimum along the steepest descent direction.  }
   var
      Temp : real;
      I, J : integer;
   begin
   TwoMu := 0.0;
   for I := 1 to M do
      begin
      Temp := 0.0;
      for J := 1 to N do
         Temp := Temp + Jacobian [I, J] * StpDir [J];
      TwoMu := TwoMu + sqr (Temp)
      end;
   TwoMu := SqStpDir / TwoMu;
   SqMuStpDir := sqr (TwoMu) * SqStpDir
   end;
```

```
procedure CalcDelta;
   var
      I : integer;
   begin
   SqDelta := 0.0;
   DeltaDotD1 := 0.0;
   for I := 1 to N do
      begin
      Delta [I] :=
      StpCoef * StpDir [I] + NwtCoef * NwtDir [I];
      SqDelta := SqDelta + sqr (Delta [I]);
      DeltaDotD1 :=
      DeltaDotD1 + OrthogDir [1, I] * Delta [I]
      end;
   SqDelta4th := 0.25 * SqDelta
   end;
```

```

procedure UpdateOrthogDir;
   {  This procedure expresses the new direction  }
   {  in terms of those of the direction matrix,  }
   {  and updates the counts.  }
   var
      DeltaDotDir   : ArrN;
      SqDeltaDotDir : real;
      Sigma         : ArrN;
      TempDir       : ArrN;
      SqAlpha       : real;
      Temp          : real;
      S, W          : real;
      I, J, K       : integer;
   begin
   for I := 1 to N do
      begin
      Temp := 0.0;
      for J := 1 to N do
         Temp := Temp + Delta [J] * OrthogDir [I, J];
      DeltaDotDir [I] := Temp;
      end;
   {  Assert:  IterStatus = Normal  }
   SqDeltaDotDir := 0.0;
   K := N;
   for I := 1 to N - 1 do
      case IterStatus of
      Normal:
         begin
         SqDeltaDotDir :=
         SqDeltaDotDir + sqr (DeltaDotDir [I]);
         if SqDeltaDotDir < SqDelta4th then
            SpanCount [I] := SpanCount [I] + 1
         else
            begin
            IterStatus := StepLenUpdate;
            K := I;
            SpanCount [I] := SpanCount [I + 1] + 1
            end
         end;
      StepLenUpdate:
         SpanCount [I] := SpanCount [I + 1] + 1
      end {case};
   SpanCount [N] := 1;
   IterStatus := StepLenUpdate;
```

```

   {  Make K the first direction  }
   if K > 1 then
      begin
      Temp := DeltaDotDir [K];
      TempDir := OrthogDir [K];
      for I := K downto 2 do
         begin
         DeltaDotDir [I] := DeltaDotDir [I - 1];
         OrthogDir [I] := OrthogDir [I - 1]
         end;
      DeltaDotDir [1] := Temp;
      OrthogDir [1] := TempDir
      end;
   for I := 1 to N do
      Sigma [I] := 0.0;
   SqAlpha := sqr (DeltaDotDir [1]);
   for I := 2 to N do
      begin
      S := sqrt (
      SqAlpha * (SqAlpha + sqr (DeltaDotDir [I])));
      W := SqAlpha / S;
      S := DeltaDotDir [I] / S;
      for J := 1 to N do
         begin
         Sigma [J] := Sigma [J] +
         DeltaDotDir [I - 1] * OrthogDir [I - 1, J];
         OrthogDir [I - 1, J] :=
         W * OrthogDir [I, J] - S * Sigma [J]
         end;
      SqAlpha := SqAlpha + sqr (DeltaDotDir [I])
      end;
   Temp := 1.0 / sqrt (SqDelta);
   for I := 1 to N do
      OrthogDir [N, I] := Delta [I] * Temp
   end;
```

```
procedure Dir1Update;
   {  This procedure updates X when Delta is too  }
   {  independent of OrthogDir [1],  i.e. when  }
   {  they are separated by more than "60 degrees".  }
   {  It does not use Delta to update X.  Instead  }
   {  it uses a multiple of OrthogDir [1]  to  }
   {  update X.  }
   var
      TempDir : ArrN;
      I       : integer;
   begin
   for I := 1 to N do
      X [I] :=
      OldX [I] + XStepSize * OrthogDir [1, I];
   TempDir := OrthogDir [1];
   for I := 1 to N - 1 do
      begin
      OrthogDir [I] := OrthogDir [I + 1];
      SpanCount [I] := SpanCount [I + 1] + 1
      end;
   OrthogDir [N] := TempDir;
   SpanCount [N] := 1
   end;
```

```
procedure UpdateX;
   {  This procedure updates the vector X by Delta  }
   {  and estimates the next residual vector F.  }
   var
      I, J : integer;
   begin
   EstSqF := 0.0;
   for I := 1 to M do
      begin
      EstF [I] := OldF [I];
      for J := 1 to N do
         EstF [I] :=
         EstF [I] + Jacobian [I, J] * Delta [J];
      EstSqF := EstSqF + sqr (EstF [I])
      end;
   for I := 1 to N do
      X [I] := OldX [I] + Delta [I]
   end;
```

```
procedure TakeStep;
   {  This procedure decides which type of step  }
   {  to take.  }
   begin
   CalcDirections;
   if sqr (OldSqF) > 4.0 * SqMaxDist * SqStpDir then
      if IterStatus = ComputeNewJ then
         IterStatus := MinNear
      else
         begin
         StuckHighCount := 0;
         X := OldX;
         StepIndex := 1;
         X [StepIndex] := X [StepIndex] + XStepSize;
         IterStatus := ComputeNewJ
         end
   else
      begin
      IterStatus := Normal;
      {  SqMaxStepSize was initialized to -1.0.  }
      if (SqMaxStepSize > 0.0) and
         (SqMaxStepSize > SqNwtDir) then
         begin
         {  Take the step in the Gauss-Newton  }
         {  direction.  }
         Delta := NwtDir;
         SqDelta := SqNwtDir;
         SqDelta4th := 0.25 * SqDelta;
         SqMaxStepSize := Max (SqNwtDir, SqXStepSize);
         StepIncrFactor := 1.0;
         if SqNwtDir >= SqXStepSize then
            begin
            if StepTypeTrace in PrintOptions then
               writeln ('Gauss-Newton step');
            UpdateOrthogDir
            end
         else
            begin
            if StepTypeTrace in PrintOptions then
               writeln ('Direction 1 step');
            IterStatus := StepDir1
            end;
         UpdateX
         end
```

```

    else
       begin
       CalcSteepestMin;
       if SqMaxStepSize <= 0.0 then
          SqMaxStepSize := Max (SqXStepSize,
          Min (SqMaxDist, SqMuStpDir));
       if SqMuStpDir > SqMaxStepSize then
          begin
          {  Take the step in the steepest  }
          {  descent direction.  }
          if StepTypeTrace in PrintOptions then
             writeln ('Steepest descent step');
          NwtCoef := 0.0;
          StpCoef :=
          TwoMu * sqrt (SqMaxStepSize / SqMuStpDir)
          end
       else
          begin
          {  Interpolate between steepest descent  }
          {  direction and Newton direction.  }
          NwtDotStp := NwtDotStp * TwoMu;
          {WRITELN ('NwtDotStp = ', NwtDotStp);}
          {WRITELN ('SqNwtDir = ', SqNwtDir);}
          {WRITELN ('SqMuStpDir = ', SqMuStpDir);}
          NwtCoef := (SqMaxStepSize - SqMuStpDir) /
          (NwtDotStp - SqMuStpDir + sqrt (
          sqr (NwtDotStp - SqMaxStepSize) +
          (SqNwtDir - SqMaxStepSize) *
          (SqMaxStepSize - SqMuStpDir)));
          StpCoef := TwoMu * (1.0 - NwtCoef);
          if StepTypeTrace in PrintOptions then
             writeln (
             'Interpolation step--Newton direction =',
             NwtCoef * 100.0 :6:1, '%')
          end;
       CalcDelta;
       if (SpanCount [1] >= 2 * N) and
          (sqr (DeltaDotD1) < SqDelta4th) then
          Dir1Update
       else
          begin
          UpdateOrthogDir;
          UpdateX
          end
       end
    end
end; {TakeStep}
```

```
procedure DoFirstTime;
   {  This is the first iteration.  }
   begin
   if ItrStatusTrace in PrintOptions then
      writeln ('IterStatus = FirstTime');
   OldSqF := SqF;
   OldX := X;
   OldF := F;
   StepIndex := 1;
   X [StepIndex] := X [StepIndex] + XStepSize;
   IterStatus := ComputeNewJ
   end;
```

```
procedure DoComputeNewJ;
   {  This iteration is for computing a fresh  }
   {  Jacobian with finite differences.  }
   var
      TempNxN : ArrNxN;
      J       : integer;
   begin
   if ItrStatusTrace in PrintOptions then
      writeln ('IterStatus = ComputeNewJ');
   for J := 1 to M do
      Jacobian [J, StepIndex] :=
      (F [J] - OldF [J]) / XStepSize;
   if StepIndex < N then
      begin
      X [StepIndex] := OldX [StepIndex];
      StepIndex := StepIndex + 1;
      X [StepIndex] := X [StepIndex] + XStepSize
      end
   else
      begin
      if NewJTrace in PrintOptions then
         begin
         write ('New Jacobian:');
         PrintMxN (Jacobian, N, M)
         end;
      ATransposeA (Jacobian, TempNxN, N, M);
      Invert (TempNxN, N);
      MultNxNxM (TempNxN, Jacobian, JInverse, N, M);
      if NewJInvTrace in PrintOptions then
         begin
         write ('New inverse Jacobian:');
         PrintNxM (JInverse, N, M)
         end;
      OrthogDir := NxNIdentity;
      SpanCount := InitSpanCountVector;
      TakeStep
      end
   end;
```

```
procedure DoNormal;
   {  This iteration is a normal one. }
   begin
   if ItrStatusTrace in PrintOptions then
      writeln ('IterStatus = Normal');
   UpdateJacobian;
   TakeStep
   end;
```

```
procedure DoStepLenUpdate;
   {  This iteration updates the step length. }
   var
      Diff      : real;
      SqLambda  : real;
      SqMu      : real;
      TempAbs   : real;
      TempSqr   : real;
      I         : integer;
   begin
   if ItrStatusTrace in PrintOptions then
      writeln ('IterStatus = StepLenUpdate');
   Diff := 0.9 * OldSqF + 0.1 * EstSqF - SqF;
   if Diff >= 0 then
      begin
      {  Increase step length  }
      TempAbs := 0.0;
      TempSqr := 0.0;
      for I := 1 to M do
         begin
         TempAbs := TempAbs +
         abs (F [I] * (F [I] - EstF [I]));
         TempSqr := TempSqr +
         sqr (F [I] - EstF [I])
         end;
      SqLambda := 1.0 + Diff /
      (TempAbs + sqrt (sqr (TempAbs) + Diff * TempSqr));
      SqMu := Min3 (4.0, StepIncrFactor, SqLambda);
      SqMaxStepSize :=
      Min (SqMu * SqMaxStepSize, SqMaxDist);
      StepIncrFactor := SqLambda / SqMu;
      OldSqF := SqF;
      SwapN (X, OldX);
      SwapM (F, OldF);
      Negate (EstF, M)
      end
```

```
   else
      begin
      {  Decrease step length  }
      SqMaxStepSize :=
      Max (0.25 * SqMaxStepSize , SqXStepSize);
      StepIncrFactor := 1.0;
      if SqF < OldSqF then
         begin
         OldSqF := SqF;
         SwapN (X, OldX);
         SwapM (F, OldF);
         Negate (EstF, M)
         end
      end;
   UpdateJacobian;
   TakeStep
   end {DoStepLenUpdate};
```

```
procedure DoStepDirl;
   {  This iteration updates X in the direction  }
   {  of OrthogDir [1].  }
   begin
   if ItrStatusTrace in PrintOptions then
      writeln ('IterStatus = StepDirl');
   if SqF < OldSqF then
      begin
      OldSqF := SqF;
      SwapN (X, OldX);
      SwapM (F, OldF);
      Negate (EstF, M)
      end;
   DirlUpdate;
   IterStatus := Normal
   end;
```

```
     begin {Minimize}
     InitGlobalConstants (N);
     NumCalls := 0;
     StuckHighCount := N + 4;
     OldSqF := 0.0;
     SqMaxStepSize := -1.0;
     SqXStepSize := sqr (XStepSize);
     SqMaxDist := sqr (MaxDist);
     StepIncrFactor := 1.0;
     IterStatus := FirstTime;
     LoopStatus := Continue;
     while LoopStatus = Continue do
        begin
        if IterStatus = MinNear then
           LoopStatus := MinPredicted
        else
           begin
           Evaluate (F, X);
           NumCalls := NumCalls + 1;
           SqF := 0.0;
           for J := 1 to M do
              SqF := SqF + sqr (F [J]);
           if SqF <= Accuracy then
              LoopStatus := ToleranceMet
           else if IterStatus in
              [FirstTime, ComputeNewJ, Normal] then
              LoopStatus := TestNumCalls
           {  Assert:  }
           {  IterStatus in [StepLenUpdate, StepDirl]  }
           else if SqF < OldSqF then
              begin
              StuckHighCount := N + 4;
              LoopStatus := TestNumCalls
              end
           else if SqMaxStepSize < SqXStepSize then
              LoopStatus := TestNumCalls
           else
              begin
              StuckHighCount := StuckHighCount - 1;
              if StuckHighCount <= 0 then
                 LoopStatus := HighResiduals
              else
                 LoopStatus := TestNumCalls
              end
           end;
```

```

      if LoopStatus = TestNumCalls then
         if NumCalls > MaxCalls then
            LoopStatus := TooManyCalls
         else
            begin
            PrintIteration (
            NumCalls, X, F, SqF, N, M, PrintOptions);
            case IterStatus of
            FirstTime:
               DoFirstTime;
            ComputeNewJ:
               DoComputeNewJ;
            Normal:
               DoNormal;
            StepLenUpdate:
               DoStepLenUpdate;
            StepDir1:
               DoStepDir1
            end {case};
            LoopStatus := Continue
            end
      end {while};
```

```

case LoopStatus of
MinPredicted:
   begin
   writeln ('Minimum predicted:');
   X := OldX;
   F := OldF;
   SqF := OldSqF
   end;
ToleranceMet:
   writeln (
   'Error is less than specified tolerance:');
HighResiduals:
   begin
   if StuckHighCount = 0 then
      writeln (
      'Successive evaluations failed to reduce ',
      'error:')
   else {StuckHighCount < 0}
      writeln (
      'Successive evaluations with a new Jacobian ',
      'failed to decrease error:');
   X := OldX;
   F := OldF;
   SqF := OldSqF
   end;
TooManyCalls:
   begin
   writeln ('Call limit exceeded:');
   if SqF >= OldSqF then
      begin
      X := OldX;
      F := OldF;
      SqF := OldSqF
      end
   end
end {case};
PrintFinal (NumCalls, X, F, SqF, N, M);
end {Minimize};
```

```
{  The main program. }
begin {MinLeslie}
writeln ('Model 5 fit to data.');
GetRealData (NumMonths, RealData);
GetModelParams (ModelParams);
ModelParams.InitAgeDistr.CurrMonth :=
RealData [1].Month;
ModelParams.InitAgeDistr.CurrYear :=
RealData [1].Year;
ToXVector (ModelParams, X);
GetMinOptions (DeltaX, MaxDist, Acc, MaxCalls);
GetPrintOptions (PrintOptions);
M := NumMonths * MaxBin;
Minimize (14, M, X, F, SqF,
DeltaX, MaxDist, Acc, MaxCalls, PrintOptions);
FileFinal (X, F, SqF)
end.
```

End of Compilation.

7. Bibliography

[Birk71] Birkeland, C. and F. Chia, "Recruitment risk, growth, age and predation in two populations of sand dollars, Dendraster excentricus", Journal of experimental marine biology and ecology, vol.6, 1971.

[Efro79] Efron, B., "Computers and the theory of statistics: thinking the unthinkable," SIAM Review, vol. 21, no. 4, October, 1979.

[Else81] Elseth, G.D., and K.D. Baumgardner, Population Biology, D. Van Nostrand, New York, 1981.

[Fell68] Feller, W., An Introduction to Probability Theory and its Applications, J. Wiley, New York, 1968.

[Flet80] Fletcher, R. Practical Methods of Optimization, vol. 1, J. Wiley, New York, 1980.

[Gill81] Gill, P.E., W. Murray, and M.H.Wright, Practical Optimization, Academic Press, New York, 1981.

[IEEE83] IEEE Standard Pascal Computer Programming Language, The Institute of Electrical and Electronic Engineers, Inc., New York, 1983.

[Kern81] Kernighan, B.W., "Why Pascal is not my favorite programming language," Computer Science Technical Report, No. 100, Bell Labs, Murray Hill, N.J., 1981.

[Keyf68] Keyfitz, N., An Introduction to the Mathematics of Population, Addison-Wesley, Reading, Mass., 1968.

[Keyf71] Keyfitz, N., and W. Flieger, Population: Facts and Methods of Demography, W.H. Freeman, San Francisco, 1971.

[Lesl45] Leslie, P.H., "On the use of matrices in certain population mathematics," Biometrika, 33, 1945.

[Lesl48] Leslie, P.H., "Some further notes on the use of matrices in population mathematics," Biometrika, 35, 1948.

[Lesl59] Leslie, P.H. "The properties of a certain lag type of population growth and the influence of an extreme random factor on a number of such species," Physiol. Zool., 32, 1959.

[Leve44] Levenberg, K., "A method for the solution of certain problems in least squares," Quarterly of Applied Mathematics, 2, 1944.

[MacG68] MacGinitie, G.E. and N. MacGinitie, Natural History of Marine Animals, 2nd edition, McGraw-Hill, 1968.

[Marq63] Marquardt, D., "An algorithm for least-squares estimation of nonlinear parameters," SIAM Journal of Applied Mathematics, 1963.

[May73] May, R.M., Stability and Complexity in Model Ecosystems, Princeton University Press, Princeton, 1973.

[Merr70] Merrill, R.J. and E.S. Hobson, "Field observations of Dendraster excentricus, a sand dollar of Western North America," The American Midland Naturalist, 83(2), 1970.

[Morè82] Morè, J.J., "Notes on optimization software," in Nonlinear Optimization 1981, (M.J.D. Powell, ed.), Academic Press, New York, 1982.

[Nich54] Nicholson, A.J., "An outline of the dynamics of animal populations," Austrilian Journal of Zoology, 2, 1954.

[Perr83] Perron, F.E., "Growth, fecundity, and mortality of Conus pennaceus in Hawaii," Ecology, 64(1), 1983.

[Powe70] Powell, M.J.D, "A hybrid method for nonlinear equations," and "A Fortran subroutine for solving systems of nonlinear algebraic equations," in Numerical Methods for Nonlinear Algebraic Equations, (P. Rabinowitz, ed.), Gordon and Breach, London, 1970.

[Timk75] Timko, P.L., "High density aggregation in Dendraster excentricus: Analysis of strategies and benefits concerning growth, age structure, feeding, hydrodynamics, and reproduction," PhD dissertation, Univ. of California, Los Angeles, 1975.

[Vand83] Vandergraft, J.S., Introduction to Numerical Computations, 2nd edition, Academic Press, New York, 1983.

[Wirt71] Wirth, N., "The programming language Pascal," Acta Informatica, 1, Springer-Verlag, New York, 1971.

[Wirt83] Wirth, N., Programming in Modula-2, 2nd edition, Springer-Verlag, New York, 1983.