# Simplex: A Manual and Software Package for Easy Nonlinear Parameter Estimation and Interpretation in Fishery Research

Anthony Mittertreiner and Jon Schnute

Department of Fisheries and Oceans
Fisheries Research Branch
Pacific Biological Station
Nanaimo, British Columbia V9R 5K6

July 1985

**Canadian Technical Report of Fisheries and Aquatic Sciences No. 1384**

Canadian Technical Report of Fisheries
and Aquatic Sciences No. 1384

July 1985

# SIMPLEX: A MANUAL AND SOFTWARE PACKAGE FOR EASY NONLINEAR PARAMETER ESTIMATION AND INTERPRETATION IN FISHERY RESEARCH

by

Anthony Mittertreiner and Jon Schnute*

Department of Fisheries and Oceans
Fisheries Research Branch
Pacific Biological Station
Nanaimo, British Columbia V9R 5K6

* Send reprint requests to this author.

Cat. No. Fs 97-6/1384E          ISSN 0706-6457

Correct citation for this publication:

Mittertreiner, A. C. and J. Schnute. 1985. SIMPLEX: A manual and software package for easy nonlinear parameter estimation and interpretation in fishery research. Can. Tech. Rep. Fish. Aquat. Sci. 1384: 90 p.

## TABLE OF CONTENTS

TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF COMPUTER LISTINGS

## ABSTRACT

Mittertreiner, A. C. and J. Schnute. 1985. SIMPLEX: A manual and software package for easy nonlinear parameter estimation and interpretation in fishery research. Can. Tech. Rep. Fish. Aquat. Sci. 1384: 90 p.

This manual describes a method for (1) fitting models in which parameters enter nonlinearly and (2) interpreting the results of the analysis to assess model validity. The manual also documents the software package SIMPLEX, a tool which easily allows the user to obtain optimal parameter estimates, produce various plots of model predictions and likelihoods, and calculate parameter covariances. SIMPLEX is particularly convenient to use when the objective function (i.e., criterion of model fit) can be expressed as a sum of similar terms, as is the case, for example, with a least squares fit. More generally, SIMPLEX can also be adapted to applications in which the objective function is analytically more complex. Typically, the user need provide only a brief section of FORTRAN code to describe the desired model; once this is done, all SIMPLEX features are immediately available for model fitting and interpretation.

The manual not only documents the SIMPLEX software, but also presents the complete theory underlying its development and use. This includes descriptions of (1) the simplex search method of function minimization, (2) the utility of various plots in assessing model fit, and (3) the numerical calculation of parameter covariances, standard deviations, and correlations. Suggestions are provided for operating the search algorithm efficiently and applying it in non-standard situations.

## RÉSUMÉ

Mittertreiner, A. C. and J. Schnute. 1985. SIMPLEX: A manual and software package for easy nonlinear parameter estimation and interpretation in fishery research. Can. Tech. Rep. Fish. Aquat. Sci. 1384: 90 p.

Le présent guide décrit une méthode permettant (1) d'ajuster des modèles dans lequel on introduit des paramètres non linéaires et (2) d'interpréter les résultats de l'analyse afin d'évaluer la validité desdits modèles. le guide donne également des détails sur le logiciel SIMPLEX, outil qui permet facilement à l'utilisateur d'obtenir la meilleure estimation possible des paramètres, de produire divers diagrammes de prédiction et de probabilité de modèles, et de calculer la covariance des paramètres. SIMPLEX est particulièrement commode quand la fonction économique (c.-à-d. le critère d'ajustement de modèles) peut être exprimé sous forme d'une somme de termes similaires, comme c'est le cas, par exemple, avec l'ajustement par la méthode des moindres carrés. De façon plus générale, SIMPLEX peut également être adapté à des cas d'utilisation où la fonction économique est analytiquement plus complexe. De façon caractéristique, l'utilisateur n'a à fournir qu'une courte section du code FORTRAN pour décrire le modèle désiré; cela fait, toutes les caractéristiques du SIMPLEX sont disponibles sur-le-champ pour l'ajustement et l'interprétation des modèles.

Le guide donne non seulement de détails sur le logiciel SIMPLEX, mais présente également toute la théorie qui a servi de base à son élaboration et à son utilisation. On y décrit notamment (1) la méthode de recherche SIMPLEX de minimalisation des fonctions (2) l'utilité de divers diagrammes dans l'évaluation de l'ajustement des modèles et (3) le calcul numérique de la covariance, de la déviation standard et de la corrélation des paramètres. On donne des suggestions pour utiliser efficacement l'algorithme de recherche et pour l'appliquer à des situations spéciales.

## FOREWORD

By Jon Schnute

This report, together with the software it documents, is the product of a long evolutionary process. It is directed at solving one of the most important technical problems in fisheries statistics (indeed, in applied statistics generally): finding a model that fits the data. At the center of this problem lies an irritating technical difficulty; model parameters must be estimated. For linear parametric models, explicit formulas can be used to calculate the estimates directly. Unfortunately, this case is the exception, rather than the rule. Realistic models typically involve parameters nonlinearly, and an iterative approach to estimation must be applied.

For the last several years I have been particularly interested in an iterative estimation method based on the so-called simplex search technique. It has several advantages: it is comprehensible without fancy mathematics, it requires a minimum of preparation to apply, it can flexibly be tailored to a wide variety of problems, and it usually works, i. e., finds the answer. It can also readily be implemented on a small computer, as I described in an earlier report (Schnute 1982).

There is more to model fitting, however, than parameter estimation alone. In the second preface to the report just mentioned, I listed several features not addressed there, such as (1) residual plots to guide model selection, (2) calculation of the covariance matrix of parameter estimates, and (3) weighted least squares. This report describes a complete revision of the earlier software, now implemented on the DEC VAX minicomputer in FORTRAN, including options for obtaining high-resolution graphics plots. The limitations cited above are resolved; indeed many new features are now available. Furthermore, as a manual for model fitting, this report is more complete than the previous one, although it still doesn't comprise a full textbook on that subject.

The Pacific Biological Station has been particularly fortunate to have Tony Mittertreiner as a summer student working on this project for the summers of 1983-84. I began by describing the software to him, discussing the possibilities for a comprehensive package on the VAX, and writing algorithmic flow charts of what should happen. We eventually also got help from Donna Sweeney in writing some of the graphics subroutines. Tony did much independent work and thinking on this problem and made many helpful suggestions. He wrote and tested most of the code described here and actively enlisted and trained users of the package. In my view this project constitutes a classic case of cooperation between the laboratory and a student, in which both learn and profit from the experience.

I invited Tony to document his work, and he did so far beyond my expectations. Essentially, he produced a rough draft of the manual here, complete with fresh insights into (and questions about) the simplex method itself. At this point, I became deeply engaged in the final phases of the project myself and wrote extensive revisions; indeed, I essentially rewrote the entire manuscript. Both Tony and I burned the midnight oil to complete everything in time for his departure at the end of August, 1984. I considered the effort worthwhile partly because I was fascinated by Tony's fresh viewpoint; he thought of describing things in a way which I had become too jaded to imagine. I hope readers will profit from the particular perspective here, which perhaps only a student can bring to a problem.

Jon Schnute
Pacific Biological Station
Nanaimo, B.C. V9R 5K6
Canada

September, 1984

# 1. INTRODUCTION

Readers of mathematical biology may not require an introduction to nonlinear parameter estimation. Topics such as growth modeling, response surface analysis, and size-frequency analysis, to name a f , rely heavily on an ability to find estimates of parameters in nonlinear equations. Unfortunately, computer algorithms for this purpose typically involve derivative calculations, which may mean that the user has to provide code for the derivative of the function being considered -- often a nontrivial task. Because of the difficulty or time involved, many practitioners resort to linear or quadratic models, which may or may not prove adequate; or users may turn to "canned" programs, thus limiting themselves to models and features provided by the package.

This SIMPLEX package has been designed to circumvent the problems outlined above. Since SIMPLEX involves a direct search method, no derivatives need be calcuated. Because the user specifies the function or model, he or she is free to experiment with variations. The package also offers plotting capabilities, a calculation of parameter covariances, and an initial search option. Hopefully these features give the user enough flexibility that nonlinear parameter estimation can be performed almost as easily as linear regression.

SIMPLEX is written in ANSI standard FORTRAN 77, in the interest of transportability. The version described here is written for the VAX/VMS 11-780; it is enhanced with some system services to aid in the minimization procedure, and the high resolution software is written with Tektronix IGL Plot 10 for a TEK 4105 color graphics terminal and a 4662 flatbed plotter. Readers interested in a copy of the software should contact the Computer Centre, Pacific Biological Station, Nanaimo, B. C., V9R 5K6, for information.

To illustrate briefly the capabilities of this package, consider some data* for freshwater mussels (Anodonta kennerylii), consisting of mean lengths (mm) at ages one to sixteen years as follows: 7.36, 14.3, 21.8, 27.6, 31.5, 35.3, 39.0, 41.1, 43.8, 45.1, 47.4, 48.9, 50.1, 51.7, 51.7, and 54.1. Suppose that one wished to fit these data to the von Bertalanffy curve (Ricker 1975, p. 221):

$$y(t) = y_\infty [1 - e^{-K(t-t_0)}] \tag{1.1}$$

where $y(t)$ is the length at age $t$ and $(y_\infty, K, t_0$ is a vector of three

*****************************************************************

*For details on the acquisition of these data, see Dr.Kim Hyatt, Lake Enrichment Project, Pacific Biological Station, Nanaimo, B.C. V9R 5K6, Canada. In this manual, the data are used only to illustrate software applications, without regard to underlying species biology. See also the acknowledgements.

parameters. If the sum of squares of residuals is used to determine the best fit, this model could be incorporated into the SIMPLEX package with only four additional lines of FORTRAN code:

```
(A)       N = 3

(B)       PRED = PARS(1)*(1 - EXP(-PARS(2)*(XO(1) - PARS(3))))

(C)       RES = YO - YP

(D)       TERM =  (YO - YP)**2
```

Here line (A) defines the number of model parameters. These are associated with the vector PARS in (B) as follows:

$$PARS(1) = y_\infty,\quad PARS(2) = K,\quad PARS(3) = t_o.$$

Thus, (B) and (1.1) define the same model, where YP is the predicted response (i. e., the length y(t)) to the variable XO(1) (i.e., the age t). The variable XO is indexed because SIMPLEX allows predictions to depend on more than one variable. Line (C) defines the residual as the difference between observed (YO) and predicted (YP) responses, and finally (D) defines a single term in the objective function as the square of a residual.

These four lines of code adequately describe the model and allow the user to estimate parameters by least squares. After setting up a suitable data file (described later) and linking in the above code, the user can start the SIMPLEX program. Starting with the initial guess

(1.2) $(y_\infty, K, t_o = (50.0, 1.00, 1.00)$

supplied by the user, SIMPLEX takes a few seconds to arrive at the optimum estimate

(1.3) $(y_\infty, K, t_o) = (57.3, 0.16, 0.15).$

The complete analysis for this problem is given in a later section.

SIMPLEX also allows the user to construct both low and high resolution plots depicting the model fit. For example, Listing 1.1 shows a low resolution plot of observations and predictions; Fig. 1.1 gives a more precise high resolution plot, along with a graph of the curve defined by the initial estimate (1.2). Model residuals can also be plotted, as shown in Listing 1.2 (low resolution) and Fig. 1.2 (high resolution). Note that the low resolution example plots the residuals themselves, while the high resolution Fig. 1.2 shows normalized residuals. In fact, either choice of residuals is available at each level of resolution.

Listing 1.1. Low resolution plot of observed (O) and predicted (P) length at age for freshwater mussels. Predictions are based on the optimal least-squares estimate (1.3) of the parameter vector.

```
L        +
E      80+
N        |
G        +
T      60+
H        |
         +                                      POPPOPPPOPPOPPPOPP
         |                            POPPOPPPOP
N      40+                     POPPOPP
         +              PPOPP
M        |          PPPO
M      20+      PPO
         +   PPO
         | POP
         P-+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--+---+--
                          5                  10                15

                                    AGE IN YEARS
```

Listing 1.2. Low resolution plot of residuals for the mussel length at age data. These are computed as the difference between observations and predictions based on the von Bertalanffy model (1.1) with the optimal parameter estimate (1.3).

```
        2.0+
           |
           +
R          |
E       1.0+
S          |
I          +
D          |
U        .0+
A          |
L          +
S          |
       -1.0+
           |
           +
           |
       -2.0+-----+-----+----+-----+-----+-----+----+-----+-----+---
                       10         20          30         40     50

                          PREDICTED LENGTH (MM)
```

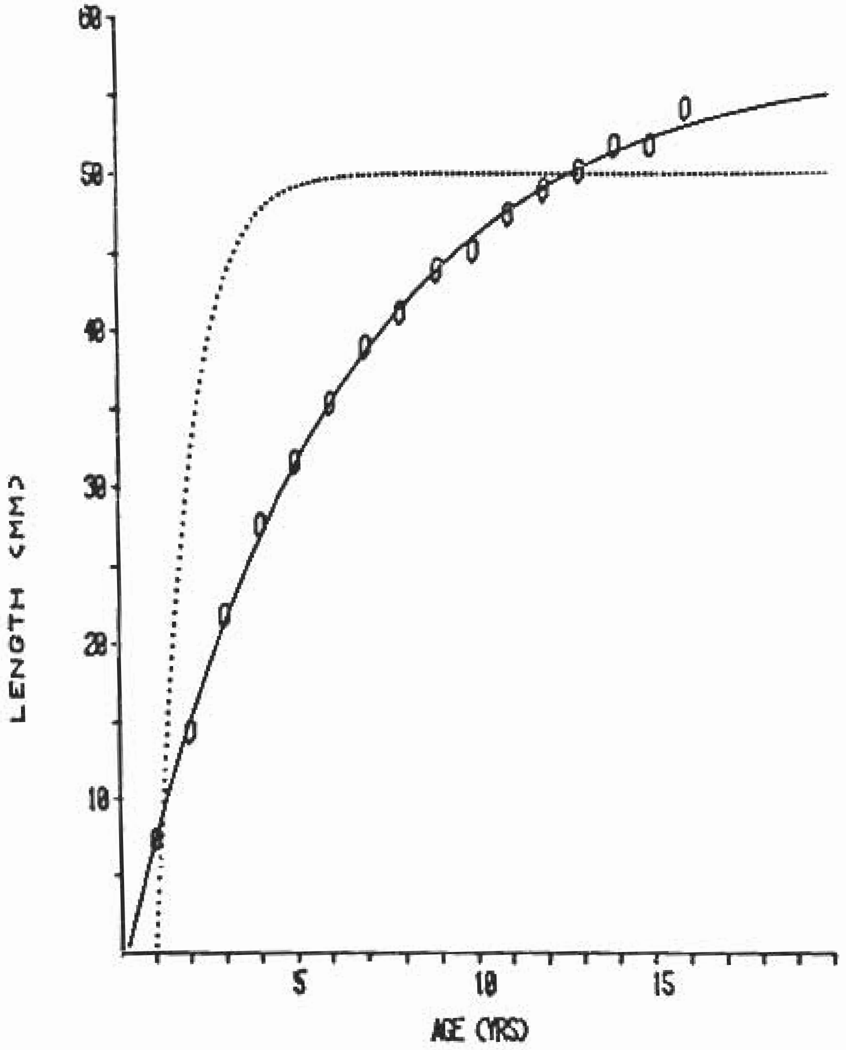


Fig. 1.1. High resolution plot of observed data (O), and two von Bertalanffy curves (1.1) for freshwater mussel data. The dotted (upper) curve corresponds to the first estimate of $(y_\infty, K, t_0) = (50.0, 1.00, 1.00)$, while the solid (lower) curve represents the final estimate $(y_\infty, K, t_0) = (57.3, 0.16, 0.15)$, optimal for the sum of squares criterion.

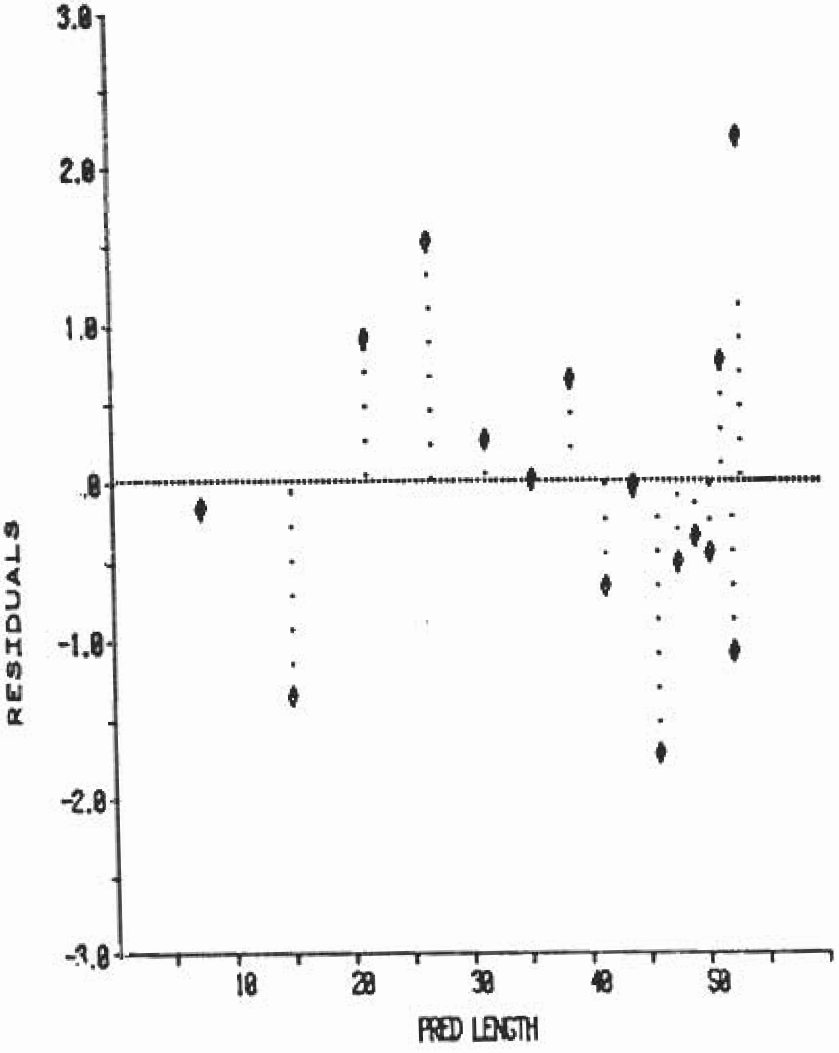


Fig. 1.2. Normalized residuals for mussle length at age data. These residuals were calculated as the difference between observations and predictions based on model (1.1) with the optimal parameter estimate (1.3).

SIMPLEX also provides the user with an option for calculating the asymptotic parameter covariance matrix, based on the matrix of second derivatives (the Hessian) of the objective function at the minimum point. This option also tabulates parameter standard deviations and correlations. Final output from the covariance calculator applied to the mussel example is shown in Listing 1.3. The method involves numerical estimates of second derivatives. Since these can be numerically sensitive to the choice of scale, three such estimates are given for each result, based on three increasingly smaller grids in parameter space.

Listing 1.3. Output from the covariance matrix calculation applied to the mussel example.

```
 POINT
  1)  57.291145      2) 0.16441514      3) 0.15506405
 STEP FOR DIFFERENCING
  1)  10.000000      2) 0.10000000      3) 0.30000001
 COVARIANCE CONSTANT  1.000000000     * 0.5 * SIGMA**-2

 GRID #          1                 2                  3
 SCALE FACTOR:  .100              .100E-01           .100E-02

                               FUNCTION VALUES
 AVERAGE F:       18.13788632      4.121189160       3.981329099
 STD DEV F:       14.82079593     0.1479410426      0.1660836190E-02
 COF VAR F:        0.81711814       0.35897659E-01   0.41715622E-03

 PAR#                          STANDARD DEVIATIONS
  1              0.6236633402     0.6521245397      0.6524315524
  2              0.5519759413E-02 0.5801261418E-02 0.5804290173E-02
  3              0.6757293868E-01 0.6949696734E-01 0.6951788838E-01

 PAR#                        COEFFICIENTS OF VARIATION
  1                0.10885859E-01   0.11382641E-01   0.11388000E-01
  2                0.33572087E-01   0.35284228E-01   0.35302650E-01
  3                0.43577437       0.44818233       0.44831725

 PAR#, PAR#                     CORRELATIONS
  1, 2            -0.92494678      -0.93158925       -0.93165596
  1, 3            -0.56895712      -0.59740172       -0.59769399
  2, 3             0.76523097       0.77986622        0.78001727

 PAR#, PAR#                     COVARIANCES
  1, 1           0.3889559619     0.4252664153      0.4256669306
  1, 2           -.3184103008E-02 -.3524337152E-02 -.3528089866E-02
  1, 3           -.2397742590E-01 -.2707465094E-01 -.2710880770E-01
  2, 2           0.3046774398E-04 0.3365463404E-04 0.3368978441E-04
  2, 3           0.2854207191E-03 0.3144187211E-03 0.3147385262E-03
  3, 3           0.4566102042E-02 0.4829828469E-02 0.4832736805E-02

                                DETERMINANTS
 COV MATRIX:     0.9569502553E-08 0.1057089487E-07 0.1058193995E-07
 COR MATRIX:       0.17684917       0.15292345        0.15268768
```

Finally, SIMPLEX allows the user to examine sections and profiles of the objective function. Here, a "section" is a graph of the function obtained by letting one parameter vary while the others remain fixed. For example, if the surface were represented by a clay model, a section could be seen by slicing with a knife and looking at the cross-section. A "profile" is a more complicated object. It is obtained by letting one parameter vary and simultaneously minimizing with respect to other parameters. Thus points on a profile represent the lowest possible value of the objective function for various values of one parameter. If we replace the idea of minimum with maximum, then everyday experience provides us with an example of a profile: the curve formed by the horizon of a mountain range. Looking in a given direction and scanning from left to right, the horizon always represents the highest elevation, without regard to the distance away. (We ignore perspective and the curvature of the earth in stating this analogy.) Statistically, a profile is extremely interesting, because it illustrates overall model sensitivity to a particular parameter. Indeed, whole statistical theories have been devised around profiles. See, for example, the theory of relative likelihood described by Kalbfleish (1971).

Figs. 1.3, 1.4, and 1.5 each show a section and profile of the sum of squares objective function for the mussel example. The parameter $y_\infty$ is allowed to vary in Fig. 1.3. The higher, narrower curve represents a section of the function along which K and $t_0$ are held fixed at their optimum values (1.3). The lower, broader curve is a profile representing the lowest possible function value for each value of $y_\infty$. Similarly, Figs. 1.2 and 1.3 represent function variation with respect to K and $t_0$, respectively.

A final feature of the profiler is that values of non-profile parameters are stored as the analysis is performed. Consequently, the user can also obtain plots showing variation in the optimal value of one parameter as another parameter changes. For example, Fig. 1.6 shows the optimal value of K (where both K and $t_0$ are allowed to vary) for each value of $y_\infty$. The figure indicates that the estimates for K and $y_\infty$ are negatively correlated: as $y_\infty$ increases, the best estimate of K decreases. Indeed, the correlation of -0.93 (grid #3) between parameters 1 and 2 in Listing 1.3 corroborates this observation.

The above examples illustrate the four main features available to the user of SIMPLEX: a minimizer, plotter, covariance calculator, and profiler. The remainder of this report is devoted to describing these features in complete detail and offering suggestions for their use in the (rather adventurous) task of building models for fisheries data.

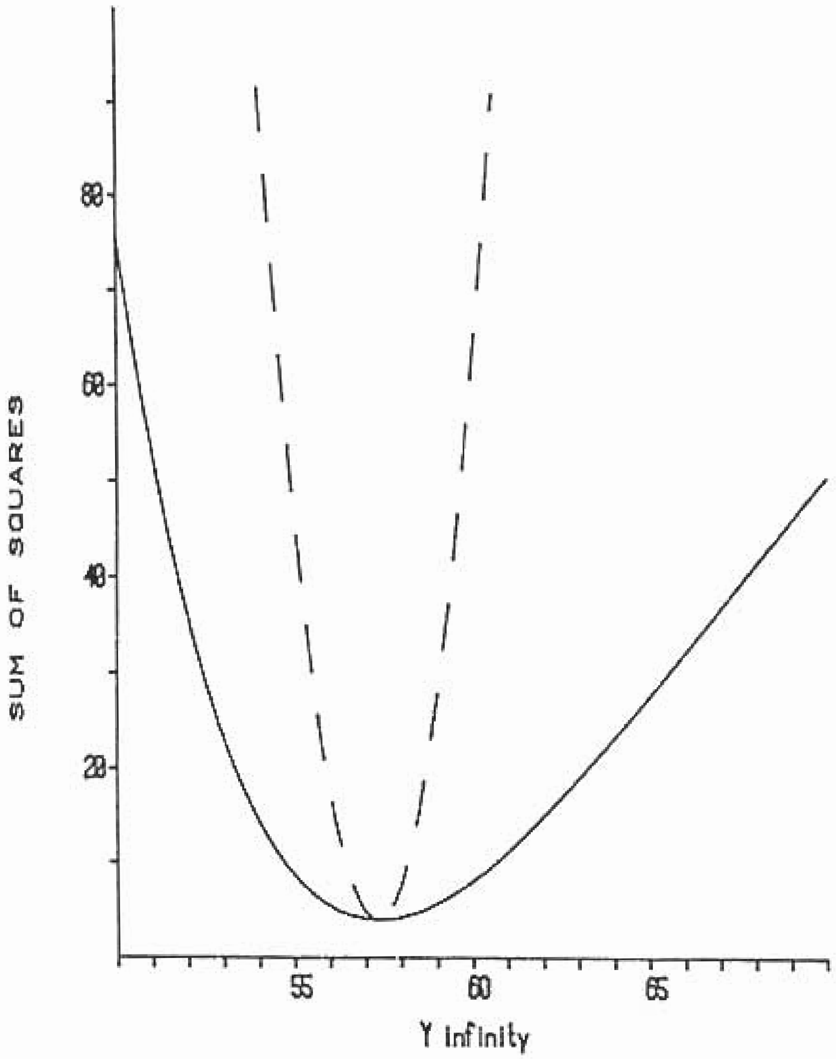


Fig. 1.3. Variations in the sum of squares function for the mussel data, as $y_\infty$ deviates from its optimal value. The solid (lower) curve is a profile, which shows function values if K and $t_0$ are optimized for each value of $y_\infty$. The dashed (upper) curve is a section, which shows function values as $y_\infty$ is varied and K and $t_0$ are held fixed at the optimal estimates (1.3).

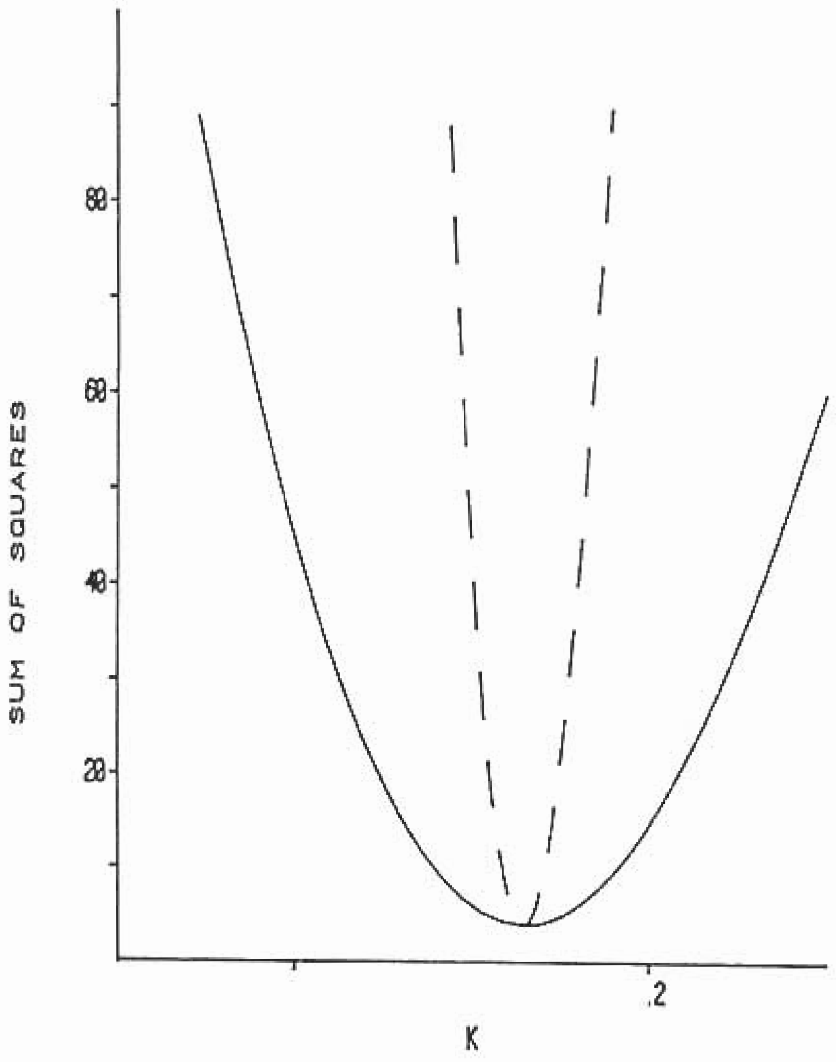


Fig. 1.4. Profile and section on K. Explanations are similar to those for Fig. 1.3.

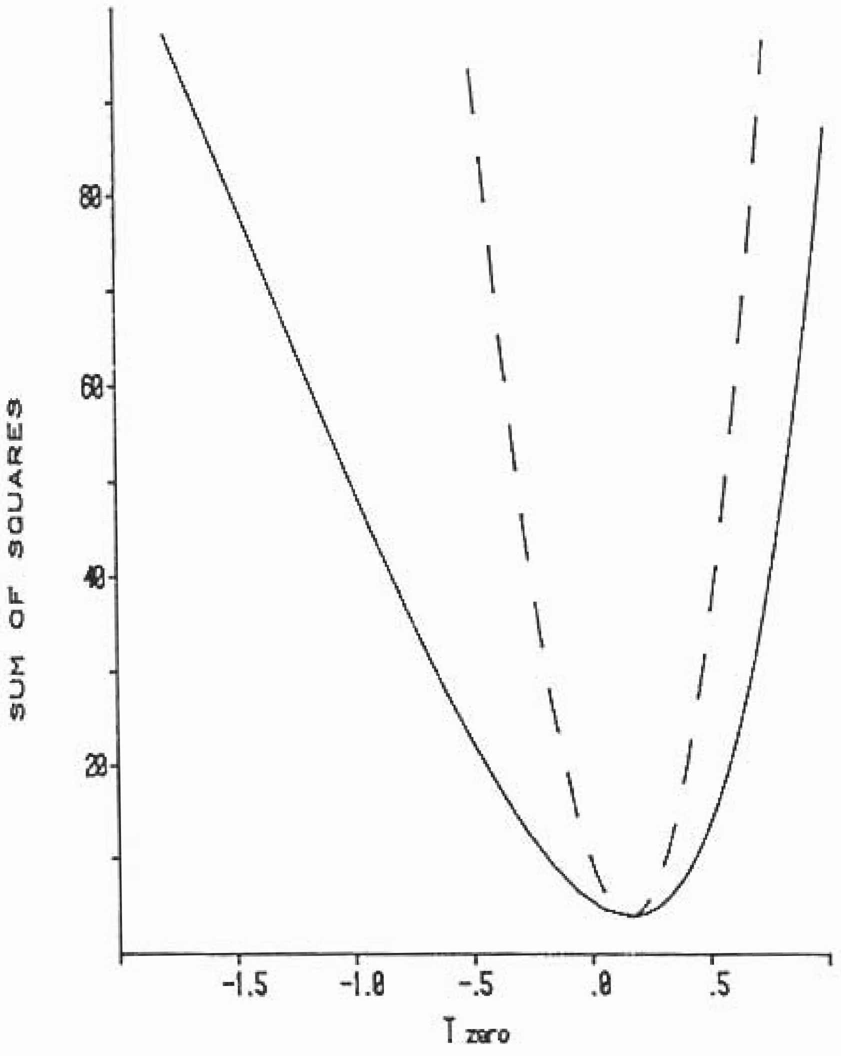


Fig. 1.5. Profile and section on $t_0$. Explanations are similar to those for Fig. 1.3.

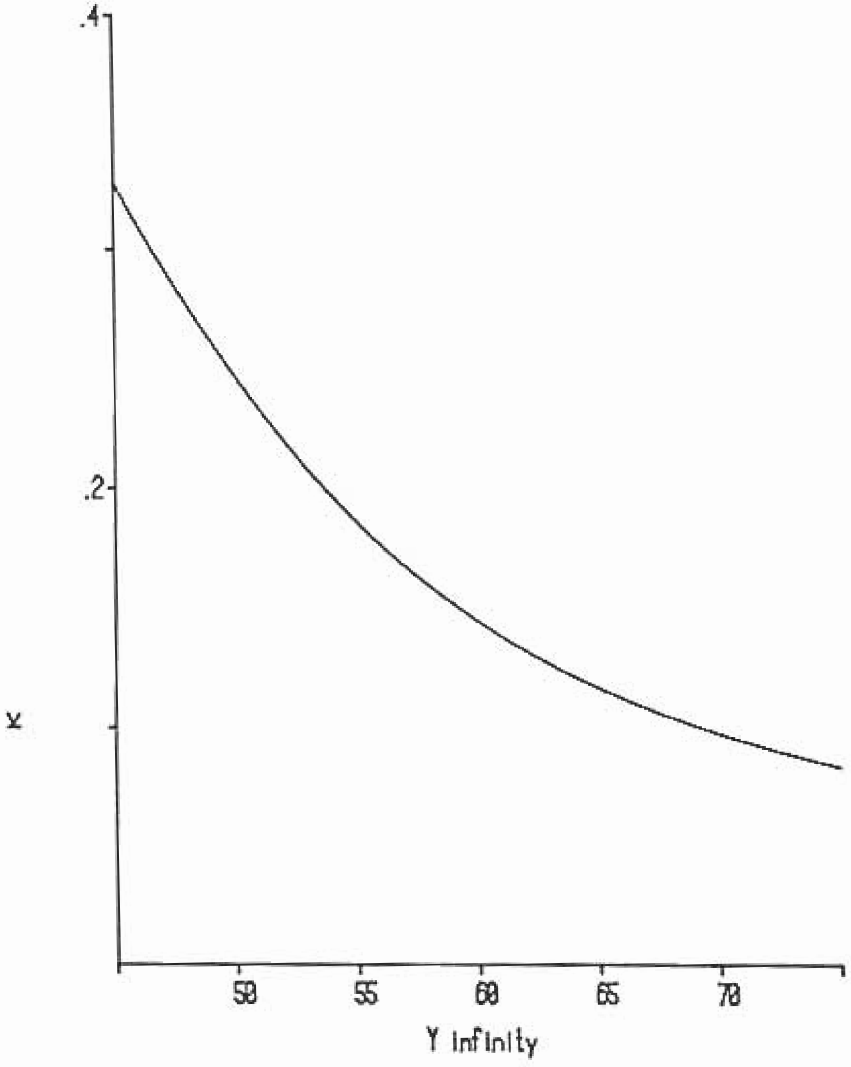


Fig. 1.6. Optimal value of K vs. a prescribed $y_\infty$. Optimal K values are found by a profile procedure in which K and $t_0$ are allowed to vary and $y_\infty$ is prescribed.

# 2. THE SIMPLEX METHOD

## 2.1. Background

The simplex search method was first conceived by Spendley, Hext, and Himsworth (1962) to optimize control variables in experiments designed to locate conditions for an optimum response. Nelder and Mead (1965) noticed that the method had potential value as a numerical technique for maximizing or minimizing a function, and they described an algorithm suitable for computers which O'Neil (1971) later formalized into a FORTRAN program. The SIMPLEX package outlined in this manual, including the minimizer, editor, profiler, and plotter, are adapted from the report by Schnute (1982), which gives a BASIC version of SIMPLEX for use with microcomputers. As Schnute explains, the problem of estimating parameters in a biological model is typically solved by minimizing a function of the parameters, such as the sum of squares of differences between observed and predicted values.

## 2.2. Simplex iteration

This section is intended to acquaint the user with the iterative simplex search procedure. A good understanding of the method will allow the user to take full advantage of the information and tools available within this package. The explanation here is based on an example, so that the main ideas can be illustrated concretely.

Consider the following function of two variables m and s:

$$F(m,s) = s^2 \exp\{[(4-m)^2 + (20-m)^2]/2s^2\} \quad , \tag{2.1}$$

where $m$ is any real number and $s>0$. It turns out that $F(m,s)$ is proportional to the reciprocal of the likelihood function for a sample of two values, 4 and 20, drawn from a normal distribution with mean $m$ and standard deviation $s$. (See Schnute 1982, p. 17, although note a typographical error in the position of parentheses in Schnute's equation (3.2).) The maximum likelihood, that is, the minimum F, should occur at the sample mean

$$m = (4 + 20)/2 = 12$$

and standard deviation

$$s = \{[(4-12)^2 + (20-12)^2]/2\}^{1/2} = 8 \ .$$

Incidentally, it is well known that the maximum likelihood estimate for $s$ in this case is biased for small samples. Here the sample size is only 2, and $s^2$ should be corrected by the factor $2/(2-1)$. (The reader may be familiar with the general correction n/(n-1), where n is the sample size.) Thus, an unbiased estimate for s would be 8 times the square root of 2. We ignore this limitation of maximum likelihood here, since our interest is strictly in the function defined in (2.1).

The problem of estimating m and s, then, boils down to finding values which minimize F(m,s). We can visualize F(m,s) as a three dimensional relief map, where m and s are the horizontal coordinates and F(m,s) is represented by the vertical height of the map. The minimum of F(m,s) would be the lowest point on the map. Our function is large for large values of s and for values of s near 0. By inspection of F in (2.1) (ignoring the known answer cited above), it is apparent that the minimum occurs between m=4 and m=20; as a first guess, suppose that the desired value of s is close to 1. These considerations suggest a map shaped like a bowl with steep sides, the bottom situated near the origin. Indeed, Fig. 2.1 shows a contour map for F based on an expanded version of the table of values listed below:

| m: | 4 | 8 | 12 | 16 | 20 |
|---|---|---|---|---|---|
| s: 1 | 4E55 | 6E34 | 6E27 | 6E34 | 4E55 |
| 2 | 3E14 | 2E9 | 4E7 | 2E9 | 3E14 |
| 3 | 1E6 | 65259 | 11030 | 65259 | 1E6 |
| 4 | 47695 | 2375 | 874 | 2375 | 47695 |
| 5 | 4183 | 613 | 323 | 613 | 4183 |
| 6 | 1260 | 332 | 213 | 332 | 1260 |
| 7 | 668 | 251 | 181 | 251 | 668 |
| 8 | 473 | 223 | 174 | 223 | 473 |
| 9 | 393 | 217 | 178 | 217 | 393 |
| 10 | 360 | 223 | 190 | 223 | 360 |
| 11 | 349 | 234 | 205 | 234 | 349 |
| 12 | 350 | 251 | 225 | 251 | 350 |
| 13 | 360 | 271 | 247 | 271 | 360 |
| 14 | 377 | 295 | 272 | 295 | 377 |
| 15 | 397 | 321 | 299 | 321 | 397 |

Clearly our initial guess of s=1 is poor, but for the sake of our example we will start off at (m,s)=(4,1) to illustrate the simplex search method. We continue as if we did not know the actual minimum location, which from the above table occurs at the point (12,8), as theory suggests.

The simplex search method begins with three arbitrary points on the map: an initial point and two others nearby. As a wild guess, try (m,s) equal to (8,1) and (4,2), in addition to the starting point (4,1). Evaluating our function at these points, we find that F(4,1) > F(8,1) > F(4,2); that is, (4,2) is the best (lowest) point, and (4,1) is the worst (highest) point. It is reasonable to look for a lower point far from (4,1) but close to (8,1) and (4,2). The method chooses the point (8,2), which is found by stepping from the highest point (4,1) to the centroid (average) of the other two points, and then taking another step of the same size in the same direction to get to the new point. This process is called the reflection of (4,1) across the two lower points. See Fig. 2.2A, where the labelled points for our example are listed below:

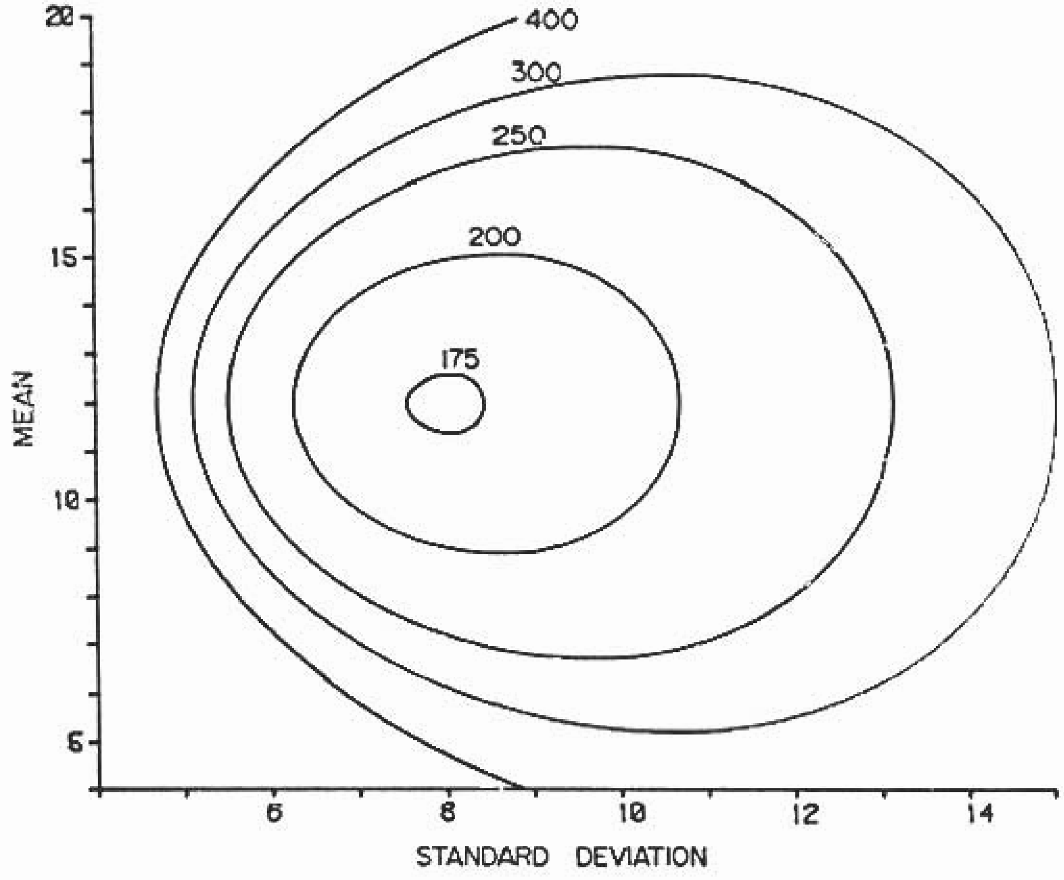


Fig. 2.1. Contour plot of the surface (2.1). This function essentially is the reciprocal of the likelihood function for two samples, 4 and 20, drawn from a normal distribution with mean m and standard deviation s. Note the minimum value found at m = 12 and s = 8, the sample mean and standard deviation, respectively.

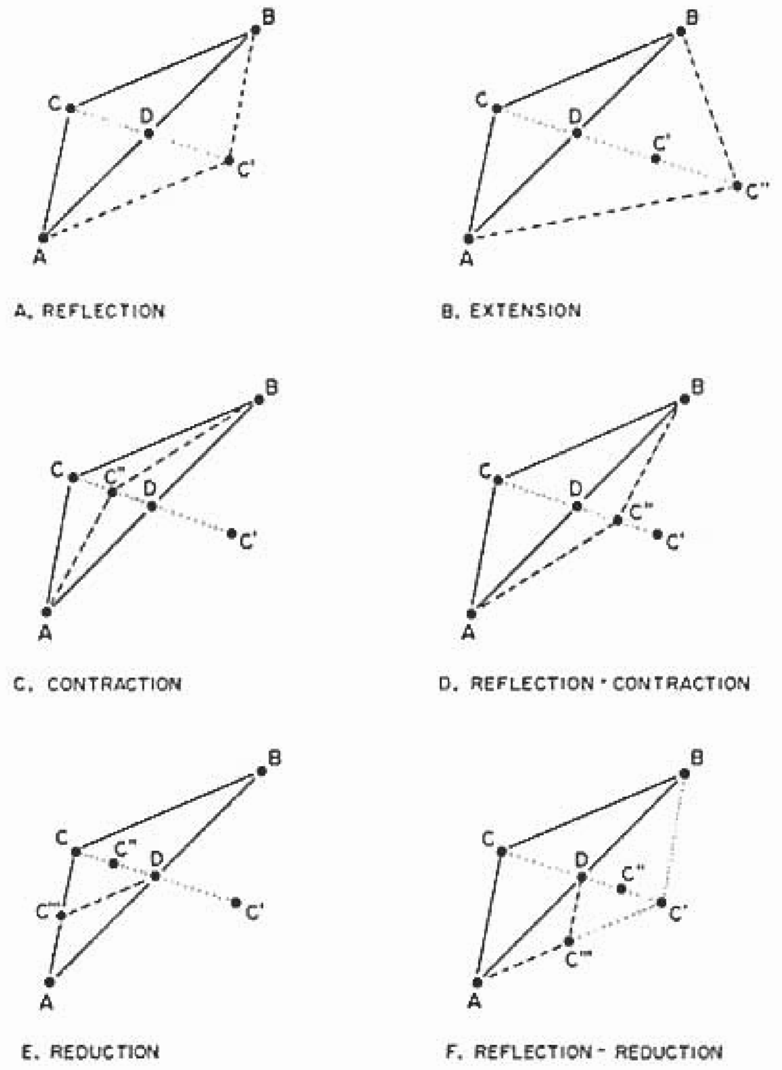


Fig. 2.2. Simplex search actions. These are the six possible actions for each simplex iteration.

| Label | Point (m,s) | Significance |
|---|---|---|
| C | (4,1) | the highest point |
| B | (8,1) | an intermediate point |
| A | (4,2) | the lowest point |
| D | (6,3) | the centroid of A and B |
| C' | (8,2) | the relection of C through D |

The new point C' gives a lower value of F than the initial guesses A, B, and C, so we are moving in the right direction. Referring to our relief map, we have initially chosen points up on the side of the bowl and then traded the highest point for a lower one. By successive movements from high value points, we eventually find the bottom of the bowl at (m,s) = (12,8). The method requires about 30 iterations (i.e., 30 different triangles) and 60 function evaluations to locate the minimum with reasonable precision. This is a simplification of what actually happens, because reflections are not always successful in finding lower points. The following paragraphs provide a more complete description of the search algorithm.

To begin this description, let us see how the method can be generalized to include functions of many variables. The search with N parameters uses N+1 points in N-dimensional space. In the example above, N is 2, and a 2-dimensional figure determined by 3 points is, of course, a triangle. Similarly, a search when N is 3 involves a 3-dimensional figure determined by 4 points, that is, a triangular pyramid. In general, a figure in N-space determined by N+1 points is called a "simplex", the term which gives the method its name. The key idea is to iterate by constructing a new simplex from the previous one in such a way that successive simplices gradually converge to the minimum point.

As the example above shows, when N=2 the three simplex points can be labeled A, B, and C where

$$F(C) > F(B) > F(A).$$

More generally, the N+1 simplex points can be labeled A, B1, B2,..., $B_{N-1}$, C, where

$$F(C) > F(B_1) > F(B_2) > \ldots > F(B_{N-1}) > F(A)$$

that is, C is the highest point, A the lowest, and the remaining N-1 points are intermediate. This suggests a general interpretation of Fig. 2.2A. The point B represents all N-1 intermediate points, the line AB represents an N-dimensional simplex with centroid D, and C' is the reflection of C through D (that is, C, D, and C' are collinear and D is the midpoint of CC'). Hereafter, we will often refer to B as a point, even though it may (when N>2) represent a collection of points. Incidentally, certain liberties are taken with the language in this and the following paragraphs. A and C are points in N-dimensional parameter space. The statement "C is higher than A" is simply an abbreviation for "C has a higher F-value than A".

With these concepts in mind, it is now possible to describe precisely the process of simplex iteration. The process always begins with a simplex ABC from which the reflected point C' is then determined. Notice, however, that the simple fact that C' is further from C than A or B does not necessarily mean that C' is a good point. Remembering our steep sided valley in the sample problem, picture C high up on one side, with A and B near the minimum. A reflection to C' is likely to take us up onto the other side of the valley, possibly to a point higher than C itself. To find out how good C' actually is, we count the number of points in the original simplex higher than C'. Call this number M. Our subsequent course of action is determined by the value of M. There are four cases to consider: $M=N+1$, $M=0$, $M=1$, and $1<M<N+1$.

Case 1: M=N+1. In this case the new point C' is lower than all points in the current simplex, that is,

$$F(C') < F(A).$$

Because reflection has led to a significant improvement, we attempt an extension to C", which involves another step in the direction from C to C' for a distance equal to DC'. This step, illustrated in Fig. 2.2B, is based on the hope that we will still be moving toward the minimum. If F(C") is less than F(C'), we accept C" as a new simplex point, otherwise we take C' as the new point. In either case, we eliminate C from the simplex. Only one extension is attempted per iteration.

Case 2: M=0. In this case the new point C' is higher than all points in the current simplex, that is,

$$F(C') > F(C).$$

The reflection step has actually been detrimental; as described earlier, we appear to have gone past the minimum. In this case we determine a new point C" by moving from C halfway to the centroid D. Such a step (Fig. 2.2C) is called contraction because it involves the conservative step of contracting the highest point of the simplex toward the centroid of the remaining lower points. If $F(C") < F(C)$ we accept the contraction, that is, we replace C with C". In case $F(C") \geq F(C)$, that is, C" does not improve C, we are forced to perform a general contraction, or reduction of the entire simplex by moving each of the N highest points half the distance towards the lowest point, as shown in Fig. 2.2E. This situtation happens rarely, because it occurs only when C" (an interior point of the simplex) is higher that all vertices of the current simplex.

Case 3: M=1. Here the reflected point C' improves the highest point C only. We might be tempted to accept C' as our new high point, but the search for the next point would then bring us right back to C, where we started. Instead we try C", the point obtained by contracting from C' to D. If C" improves C' we accept it, giving us a reflection-contraction (Fig. 2.2D). Otherwise, we are forced to reduce the entire simplex towards the low point, a process of reflection-reduction (Fig. 2.2F). Both these operations are just reflected versions of the corresponding operations performed when M=0.

Case 4: 1<M<N+1. This is the simplest case. The reflected point C' does not improve the lowest point A. On the other hand, C' improves not only C, but at least one other point as well. In this case, we accept the reflection shown in Fig. 2.2A by removing C and adding C' to the simplex. Note that C' is not the new highest point, so that reflection for the next iteration will proceed in another direction.

Appendix B gives a compact algorithmic description of the process of simplex iteration, beginning with one simplex and proceeding to the next.

## 2.3 Convergence

The simplex algorithm repeats the process of reflecting, extending, contracting, and reducing until some convergence condition is satisfied. The SIMPLEX package here uses the criterion that the difference between highest and lowest function values on the simplex must be less than a limit specified by the user. When the limit is small, this implies, of course, that all simplex points have function values fairly close together.

To understand the validity and significance of this condition for stopping, consider the action of the algorithm as it proceeds toward a minimum. While the lowest point lies outside the simplex, most actions taken are reflections and extentions, which do not decrease simplex size. The algorithm normally performs contractions and reductions only when the simplex encloses the minimum point; consequently, the simplex typically does not shrink until it is near a minimum. The conditions that (1) the simplex is small, (2) the corresponding function values are close, and (3) the simplex is near a minimum are often equivalent, so it is reasonable to test (2) only.

The preceding paragraph describes rather ideal conditions. In some problems, however, the algorithm may be distracted by narrow canyons, ridges, and multiple minima. These difficulties are discussed in greater detail later.

## 2.4. Final axial search

When the algorithm determines a simplex which passes the above convergence criterion, the low point is taken as an estimate of the minimum. At this stage an axial search is used to trap conditions of premature convergence. The search tests the proposed minimum by stepping away from it a small distance (both positive and negative) along each axis. This gives up to 2N test points; however, the axial search is stopped if a lower value is found at any such point. When this happens, it is assumed that the algorithm has converged prematurely, and the entire process is restarted from the new minimum found by the axial search.

## 2.5. Algorithm data

Before the SIMPLEX package can be run, the user must set up various operating parameters. These affect, for example, aspects of algorithm performance, such as the initial point, the initial simplex size, convergence criteria, and instructions for output. Collectively, these parameters are called algorithm data, and they are described in the following paragraphs.

(1) Initial point. As explained earlier (section 2.2), the search algorithm needs a starting point. Obviously, the closer this estimate is to the actual minimum, the faster the algorithm will converge, so a bit of thought here is recommended; however, the algorithm is remarkably robust, so the user needn't spend too much time agonizing over the choice. Once the program is running, the initial point vector contains the lowest point found so far. On completion, then, the answer lies in the initial point vector.

All initial estimates are set to a default value of 1.0 if not specified by the user.

(2) Initial step. The initial step vector is used with the initial point to define the first simplex for the search. Since the simplex requires N+1 points, it can be defined simply by taking the initial point as point #1 and then adding the step coordinates individually to obtain the remaining N points. Such a simplex might be called a "right simplex", since its sides are mutually perpendicular at the vertex determined by the initial point. (Think of a right triangle when N=2.) The size of each step coordinate should reflect, roughly, the magnitude of uncertainty in the the corresponding parameter. The step size defaults to 10% of the initial point size, and the step is altered every time the user edits the initial point value.

In our example problem, based on F(m,s) in (2.1) with the initial point (m,s)=(4,1), the initial step vector is also taken to be (4,1), giving initial SIMPLEX points of (4,1), (8,1), and (4,2).

(3) Simplex limit. As explained earlier, the simplex limit is used to test for convergence. The user can select either an absolute or relative limit. The absolute method looks only at the difference between the high and low function values on the simplex. If the user has no prior knowledge of the magnitude of the function value (a typical situation), the relative limit may be useful. Here the difference of high and low values, divided by the low, must be less than the specified value. The user must be careful with relative limits, however, because in some instances the low value might be zero, causing zero-divide errors.

The default value for the simplex limit is 0.1E-05 relative.

(4) Maximum function calls. This sets an upper limit on the number of function calls performed in a run, as a device for stopping runs which converge poorly. When this number of calls is reached, the user is given the option of quitting or continuing. The default value is N*100 + 100.

(5) Terminal output frequency. This number, say T, controls how often summary information is output to the terminal. If T=0, the user is informed only when an error condition exists or a minimum is located. If T=1, the user receives the following output every iteration:

1. number of function calls, iterations, restarts,
2. number of values improved and action taken,
3. current minimum point,
4. current maximum and minimum value.

If T>1, items 1, 3, and 4 are printed every T$^{th}$ iteration. The default value is T = 1.

(6) File output frequency. This has an effect similar to the terminal output frequency, except that information is output to a file. The default value is 0 (so no file is opened), and the default file name is SIMPOUT.DAT.

(7) Step size reduction fraction. When SIMPLEX does an axial search it finds the axial step sizes by multiplying each initial step value by the step size reduction fraction. The default value is 0.01. The user should specify this parameter with some care, because an axial search with too large a step may pass points at which the algorithm should be restarted. On the other hand, too small a step can cause multiple restarts, which make the algorithm frustratingly slow and inefficient. See section 9.3 for further discussion of this point.

# 3. PREPARATIONS

## 3.1. Two methods: MINSUM and MINFUN

Early in the development of this project, two distinct philosophies arose for incorporating the user's objective function into SIMPLEX. A desire to keep the user's work to a minimum inspired the concept of a "template", which can easily be completed with a few lines of code. Unfortunately, practical applications made it clear that a simple template was not versatile enough to cover every situation. This led to two versions of the minimizer: (1) MINSUM, which incorporates a pre-defined TEMPLATE, and (2) MINFUN, which uses a more general function routine UFUN written by the user. Essentially, MINSUM can be applied whenever the objective function can be expressed as a sum of similar terms. Otherwise MINFUN must be used. Thus, SIMPLEX refers to the composite of (1) and (2), but in practice the user is involved only with the names MINSUM and MINFUN. The idea, we thought to ourselves, was that users should always be able to have SUM FUN with SIMPLEX.

## 3.2. MINSUM and TEMPLATE

The least squares example worked in the introduction represents the typical situation in which MINSUM can be applied. The objective function is a sum of terms involving predicted and observed values of a response to various factors. Here, for example, length is the response to the factor age. Several items of information, then, must be incorporated into MINSUM. First, there are various functional relationships, such as the predicting function and the nature of each term in the objective function. Second, the data must be entered into the calculation.

### 3.2A. Creating a module from TEMPLATE

TEMPLATE consists of three distinct sections. The first is INITM, called by MINSUM to initialize various operating variables, if necessary. The second part contains the function evaluation routines PRED, RES, and TERM. The function TERM (which may depend on PRED and RES) is called many times during a minimization run. The third and final section, SUMMAR, performs any final operations needed by the user, such as summary calculations and output. TEMPLATE is shown in Listing 3.1. Its subroutines and functions are described fully in the paragraphs following. The user is responsible for completing at least PRED, RES, and TERM. INITM and SUMMAR are optional; however, even if they are not used, their skeleton forms must be left within TEMPLATE.

Listing 3.1. The module TEMPLATE for MINSUM. It includes a model initialization section (INITM), an evaluation section (PRED, RES, TERM), and a final summary section (SUMMAR). Underlined comments show the four lines of code for the model of mussel growth discussed in the introduction.

```
C*********************************************************************
      SUBROUTINE INITM (N,AUX,NAUX,X,YOBS,NVAR,NDAT)
      DOUBLE PRECISION X(20,5000)  !INDEPENDENT VARIABLES(NVAR,NDAT)
      DOUBLE PRECISION YOBS(5000)  !DEPENDENT VARIABLE VALUES (NDAT)
      DOUBLE PRECISION AUX(50)     !NAUX AUXILARY PARAMETERS
C Subroutine INITM is used to set the number of parameters N,
C initialize the vector of auxilary parameters, define the model
C (if necessary), and perform any other initializations needed
C by the user.
C EXAMPLE: N=3
      RETURN
      END
C*********************************************************************
      FUNCTION PRED (XO, NVAR, PARS, N, AUX, NAUX, ICASE, NDAT)
      DOUBLE PRECISION XO(20)          !OBSERVED POINT(NVAR)
      DOUBLE PRECISION AUX(50)         !AS DEFINED ABOVE
      DOUBLE PRECISION PARS(50)        !PARAMETERS(N)
      DOUBLE PRECISION PRED            !PREDICTED VALUE OF Y
C PRED calculates the predicted value of the passed data point.
C EXAMPLE: PRED = PARS(1) * (1 - EXP(-PARS(2)*(XO(1)-PARS(3))))
      RETURN
      END
C---------------------------------------------------------------------
      FUNCTION TERM (YO, YP, XO, NVAR, PARS, N, AUX, NAUX, ICASE, NDAT)
      DOUBLE PRECISION AUX(50),XO(20),YO,YP,PARS(50) !AS ABOVE
      DOUBLE PRECISION TERM  !TERM OF OBJECTIVE FUNCTION
      INTEGER ICASE          !DATA POINT COUNTER
C TERM calculates the term of the objective function corresponding
C to the passed data point.
C EXAMPLE: TERM = (YO-YP)**2
      RETURN
      END
C---------------------------------------------------------------------
      FUNCTION RES (YO, YP, XO, NVAR, PARS, N, AUX, NAUX, ICASE, NDAT)
      DOUBLE PRECISION AUX(50),XO(20),PARS(50)          ! AS ABOVE
      DOUBLE PRECISION YO, YP             ! OBSERVED Y, PREDICTED Y
      DOUBLE PRECISION RES                ! RESIDUAL
C RES calculates the model residual at the passed data point.
C EXAMPLE: RES = YO-YP
      RETURN
      END
C*********************************************************************
      SUBROUTINE SUMMAR (AUX,NAUX,X,YOBS,NDAT,NVAR,PARS,N,YPRED,
     *                  TERM, RESID, F, NF, NI, NR)
      DOUBLE PRECISION AUX(50),X(20,5000),YOBS(5000),PARS(50)
      DOUBLE PRECISION YPRED(5000),TERM(5000),RESID(5000), F
C SUMMAR allows the user to write information associated with
C function evaluation at the current parameter vector PARS.
      RETURN
      END
```

SUBROUTINE INITM. This is used to initialize the vector of auxiliary parameters (described below), to perform any other initializations desired by the user, and to set the number of parameters N. If N is not set here, the user will be asked to supply N at run time. When INITM is called, the data file has already been read. Consequently, the data themselves can be used to determine any required quantities, such as means, standard deviations, minima, or maxima of observed variables. Such quantities are called "auxiliary parameters". They are determined at run time, but, unlike model parameters, are not to be estimated by function minimization. Auxiliary parameters can be utilized in several ways. For example, they might also be used as flags to tell the PRED function which model to use, if the user has several available.

The data consist of a vector Y of NDAT observed responses and a matrix X of NDAT observed vectors of NVAR explanatory variables. Maximum values for NDAT and NVAR in this implementation are 5000 and 20, respectively. The software treats any particular observation as an observed response YO to the observed explanatory vector XO of dimension NVAR. The auxiliary parameter vector AUX has dimension NAUX, where the maximum value of NAUX is 50.

FUNCTION PRED. This function calculates the predicted value of the response, based on the observation vector XO, the model parameter vector PARS, and the auxiliary parameter vector AUX. PARS has dimension N, where the maximum value of N is 50.

FUNCTION TERM. This function calculates a typical term of the objective function, based on the current data point (XO,YO) and the corresponding prediction YP. The model parameters, auxiliary parameters, and the number ICASE of the current data point are also available for the calculation. Usually, TERM is a very simple function, such as (YO-YP)**2 in the least squares case.

FUNCTION RES. This function calculates the residual at the current point, based on the same data as TERM. Again, the function is usually simple. For example, in case of additive errors, RES is just YO-YP.

SUBROUTINE SUMMAR. This is called when a report on the current parameter vector and associated quantities is desired by the user. Typically, this report would be requested at the end of a successful minimization, but it can be called from MINSUM (which automatically does a function evaluation before calling SUMMAR) at any time. SUMMAR has access to all variables used in the previous routines, as well as the following:

YPRED - the vector of predicted response values (from PRED) in a function calculation at the current parameter vector PARS;
TERM - the vector of TERMs in a current function calculation;
RESID - the vector of residuals associated with the current parameter vector PARS;
F - the objective function value at the current vector PARS;
NF - the number of function calls in the most recent minimization, or 0 if no minimization has been performed;
NI - the number of iterations in the most recent minimization, or 0;
NR - the number of restarts in the most recent minimization, or 0.

SUMMAR can be used, for example, to list any pertinent information not automatically listed by MINSUM. SUMMAR can also write to external files and devices, as a means of recording data on parameter values of special interest.

## 3.2B. Preparing the data file.

The file of observed data read by MINSUM must have the following format:

Line 1: Number NVAR of explanatory variables

Line 2: FORTRAN read format for the data. If the data can be read free format, a blank line should be given. Data lines are read as

XO(1), XO(2), ... , XO(NVAR), YO

Lines 3 to the last: Data, in the order described above.

MINSUM can read up to 5000 (the maximum value of NDAT) data points. Since each point is accessed on every function evaluation, the amount of data can substantially affect program speed.

## 3.3. MINFUN and UFUN

Should the TEMPLATE method prove inadequate for the user's needs, a more general subroutine UFUN must be written. Like TEMPLATE, UFUN has three distinct sections: an initialization section, a function evaluator, and a final summary section. Listing 3.2 shows the skeletal structure of a typical UFUN. Notice that UFUN references four variables: (1) the double precision function value to be returned, (2) the parameter vector (also double precision) on the basis of which the function is calculated, (3) the number of parameters (maximum 50), and (4) the number of function calls so far.

Item (4), here labelled NF, determines which section of UFUN is called by MINFUN. UFUN itself should not tamper with NF. Before a minimization run is started, MINFUN sets NF=0; UFUN, when called, should branch to the initialization section. If variables that must be available later are declared in this section, then a SAVE statement should be included here. During minimization, MINFUN keeps track of NF>0, and UFUN should simply evaluate the function. On completion of a minimization, MINFUN sets NF=-1; in this case UFUN should branch to the final summary section. Since MINFUN saves the parameters and function value associated with the minimum, UFUN can use these values (without recalculation) in the final section.

Listing 3.3 shows a complete UFUN subroutine associated with the example (2.1) discussed in Section 2. There, we were concerned with estimating the mean m and standard deviation s for a normal distribution, based on two sample values: 4 and 10. Listing 3.3 generalizes this idea by allowing user input of the sample values, here called a and b. Thus the function (2.1) is replaced here by

$$(3.1) \qquad F(m,s) = s^2 \exp([(a-m)^2 + (b-m)^2]/2s^2) \quad .$$

As in Section 2, we are again interested only in the function F, and we ignore the small sample bias (a factor of $2^{1/2}$) in the maximum likelihood estimate for s.

Listing 3.2. A skeletal listing for a typical UFUN. The subroutine depends on four variables, declared as shown. It has three sections which are referenced depending on the value of NF.

```
      SUBROUTINE UFUN(F, PARS, N, NF)
      DOUBLE PRECISION F         ! Returned function value
      DOUBLE PRECISION PARS(*)   ! Parameter vector
      INTEGER N                  ! Number of parameters
      INTEGER NF                 ! Number of function calls so far

      IF (NF.EQ.0) THEN
C       Section 1: Initialization when NF=0; may include a SAVE.
      ELSE IF (NF.GT.0) THEN
C       Section 2: Function evaluation when NF>0.
      ELSE
C       Section 3: Final summary when NF<0.
      END IF
      RETURN
      END
```

Listing 3.3. A complete sample UFUN. This can be used to estimate the mean m and standard deviation s for a normal distribution, based on two sample values, a and b. Maximum likelihood estimates are equivalent to minimizing F in (3.1).

```
      SUBROUTINE UFUN (F, PARS, N, NF)
      DOUBLE PRECISION  F, PARS(*), A, B, EXPTRM
      INTEGER N, NF

      IF (NF.EQ.0) THEN              ! Initialization section *****
         N = 2                       ! Two parameters: m and s
         PRINT *, ' Please input values for A and B.'
         PRINT *, ' A: '                                  ! Get a
         READ (5, 10) A
         PRINT *, ' B: '                                  ! Get b
         READ (5, 10) B
 10      FORMAT (F18.0)
         SAVE
      ELSE IF (NF.GT.0) THEN        ! Evaluation section *********
          EXPTRM = .5 * (PARS(2) ** (-2)) *
                   (( A - PARS(1))**2 + (B - PARS(1))**2)
          F = PARS(2) ** 2 * EXP ( EXPTRM )

      ELSE                          ! Final summary section ******
         OPEN (UNIT = 10, FILE = 'OUTPUT.MIN', STATUS = 'NEW')
         WRITE ( 6, 20) A, B, PARS(1), PARS(2), F
         WRITE (10, 20) A, B, PARS(1), PARS(2), F
20       FORMAT (//' GIVEN SAMPLE VALUES A = ', F15.8,/,
     *           '                   AND B = ', F15.8,//,

     *           '             MEAN CALCULATED AS: ', F15.8,/
     *           ' STANDARD DEVIATION CALCULATED AS: ', F15.8,/
     *          /' F IS: ', F15.8)
         CLOSE (10)

      END IF
      RETURN
      END
```

## 3.4. Linking SIMPLEX (on the VAX 11/780 at the Pacific Biological Station)

The SIMPLEX package consists of MINSUM, MINFUN, a block of subroutines called COMMON, the TEMPLATE, and the IMSL and IGL libraries. The routines MINSUM and MINFUN, as described earlier, are the basis of this package. MINSUM requires a user-completed TEMPLATE, and MINFUN requires a user-written UFUN. The two basic link commands are:

```
$ LINK TEMPLATE,SIMPLEX:MINSUM,COMMON,SYS$SHARE:IMSLIBD/LIB,PBS$IGLOPT:/OPT
```

and

```
$ LINK UFUN,SIMPLEX:MINFUN,COMMON,SYS$SHARE:$IMSLIBD/LIB,PBS$IGLOPT:/OPT
```

The file names "TEMPLATE" and "UFUN" are not sacred, and the user may choose suitable names, as desired. For example, a version of TEMPLATE that implements a growth model might be called "GROWTH". Copies of MINSUM, MINFUN, COMMON, and TEMPLATE, along with the command files GOMINSUM and GOMINFUN described below, can be found in the directory with the logical name "SIMPLEX:". Linkages are represented in the diagram below.

```
**********                        ************
.        .                        .          .
. MINSUM .----- CALLS -----> . TEMPLATE .
.        .  !                     .          .
**********  !                     ************
            !
            !-- CALLS ------> **********                    ********
                              .        .                    . IMSL .
                              . COMMON .-- CALLS --> . AND  .
                              .        .                    . IGL  .
            !-- CALLS ------> **********                    ********
            !
**********  !                       ********
.        .  !                       .      .
. MINFUN .----- CALLS -------> . UFUN .
.        .                          .      .
**********                          ********
```

The linking procedure can be simplified somewhat with the command files GOMINSUM.COM and GOMINFUN.COM. Calling these programs with the name of an object file will result in that file being linked into the package, and the resulting executable image being given the same name as the object file. Typical calls would be:

```
$ @ SIMPLEX:GOMINFUN UFUN
```

or

```
$ @ SIMPLEX:GOMINSUM TEMPLATE
```

Note that the .OBJ extensions must not be added. The file names TEMPLATE and UFUN are not sacred here, as long as the specified files contain the required functions and subroutines with the standard names. Listings 3.4 and 3.5 show the command files GOMINSUM and GOMINFUN.

Listing 3.4. The command file GOMINSUM.COM.

```
$       SET VERIFY
$!*****************************************************************
$! USED TO LINK USER'S TEMPLATE ROUTINE INTO SIMPLEX PACKAGE WITH
$! MINSUM.  TO USE ENTER:
$!        $ @ GOMINSUM file
$! WHERE file IS THE NAME OF THE USER ROUTINE.
$!******************************************************************
$       LINK/NOMAP 'P1, SIMPLEX:COMMON, MINSUM, -
                   SYS$SHARE:IMSLIBD/LIB, PBS$IGLOPT:/OPT
$       SET NOVERIFY
$       EXIT
```

Listing 3.5. The command file GOMINFUN.COM.

```
$       SET VERIFY
$!*****************************************************************
$! USED TO LINK USER'S ROUTINE INTO SIMPLEX PACKAGE WITH MINFUN.
$! TO USE ENTER:
$!        $ @ GOMINFUN file
$! WHERE file IS THE NAME OF THE USER ROUTINE.
$!*****************************************************************
$       LINK/NOMAP 'P1, SIMPLEX:MINFUN, COMMON, -
                SYS$SHARE:IMSLIBD/LIB, PBS$IGLOPT:/OPT
$       SET NOVERIFY
$       EXIT
```

## 3.5. Preparations summary

The user has the option of using MINSUM and TEMPLATE or else MINFUN and UFUN. Steps required for the first option are listed below:

Step 1. Prepare a file of observed data in the format described above in Section 3.2B.

Step 2. Create a file "name.FOR", where "name" is a prefix chosen by the user, based on TEMPLATE. As described in Section 3.2A, this file will contain the routines INITM, PRED, TERM, RES, and SUMMAR.

Step 3. FORTRAN compile "name.FOR" to obtain "name.OBJ".

Step 4. Execute the command file GOMINSUM by entering:

```
$ @SIMPLEX:GOMINSUM name
```

This will result in a file "name.EXE", which can be RUN to obtain all the SIMPLEX options described in this report.

The steps required for the second option with MINFUN and UFUN are similar. A data file, as in Step 1, may or may not be necessary. In Step 2 "name.FOR" should contain the subroutine UFUN, as described in Section 3.3. Steps 3 and 4 still apply, except that the command file in Step 4 should be GOMINFUN, rather than GOMINSUM.

# 4. SIMPLEX OPERATION: MINIMIZING

## 4.1. System initialization

To initiate SIMPLEX operation, the user enters

```
$ RUN filename
```

where "filename" is the name of an appropriate .EXE file, created as described in Section 3.4. Once started, SIMPLEX immediately calls either INITM from MINSUM or UFUN (with NF=0) from MINFUN to perform the user's initializations. The user then provides either the name of a previously created algorithm data file, or accepts default algorithm data values. If the name of an algorithm file is entered, SIMPLEX checks to ensure that the value of N from the algorithm data file matches the value obtained from TEMPLATE or UFUN. At this point the main SIMPLEX menu is displayed. Listing 4.1 shows the menu items. These are covered in more detail in the following section.

Listing 4.1. Simplex main menu items.

```
     1)    RE-INITIALIZE COMPLETELY
     1a)   RE-INITIALIZE DATA (MINSUM ONLY)
     1b)   RE-INITIALIZE MODEL (MINSUM ONLY)
     2)    EDIT ALGORITHM DATA
     3)    LOAD NEW ALGORITHM DATA
     4)    SAVE ALGORITHM DATA
     5)    MINIMIZE
     6)    REPORT ON MINIMUM
     7)    FUNCTION CALL AT CURRENT POINT
     8)    PLOT PREDICTIONS AND RESIDUALS (MINSUM ONLY)
     9)    PLOT PROFILE
     10)   COMPUTE COVARIANCE MATRIX
     11)   SEARCH GLOBALLY FOR MINIMUM
     12)   EXIT
```

## 4.2. Menu Options

1. RE-INITIALIZE. With this option, either INITM is called from MINSUM or UFUN is called from MINFUN with NF=0. MINSUM users have the choice of (1A) reading in new data only, (1B) calling INITM only, or (1) both.

2. EDIT ALGORITHM DATA. The user may choose to edit all the algorithm data, only particular variables, or, in the case of POINT and STEP, only particular fields of these variables. When the editor is summoned, a list of the algorithm data is presented. A menu allows selection of the various

editing alternatives, as itemized in the top of Table 4.1. Algorithm data types are listed in the lower portion of Table 4.1, along with command codes for the editing functions. A value selected for editing is displayed before the user changes it; the current value can be retained simply by pressing <CR>. Ordinarily, when a value is changed, this has no effect on other values; however, there is one exception to this rule. Whenever the user edits a POINT field, the corresponding field in STEP is automatically changed to the default value, even if the user accepts the default POINT field.

Table 4.1. Codes for editing algorithm data.

| EDIT FUNCTIONS | ENTER: |
|---|---|
| Edit all values | AL |
| Edit particular value(s) | Edit code, as below |
| Help | H |
| Review current values | Return ( <CR> ) |
| Exit | Q |
| **VARIABLE TO EDIT** | **EDIT CODE** |
| INITIAL OR CURRENT POINT OF SIMPLEX | IP |
| INITIAL OR CURRENT POINT, FIELD n | IPn |
| INITIAL STEP OF SIMPLEX | IS |
| INITIAL STEP, FIELD n | ISn |
| SIMPLEX LIMIT FOR CONVERGENCE | SL |
| MAXIMUM ALLOWABLE FUNCTION CALLS | MF |
| FREQUENCY OF TERMINAL OUTPUT | TF |
| FREQUENCY OF FILE OUTPUT | FF |
| OUTPUT FILE (ACCESSED IF FF > 0) | FN |
| STEP REDUCTION FRACTION FOR RESTART | SR |

3. LOAD NEW ALGORITHM DATA. The new data may be read from an existing file. Alternatively, the user can request automatic defaults, which can then be edited.

4. SAVE ALGORITHM DATA. The current algorithm data is written to a file specified by the user.

5. MINIMIZE. The simplex search algorithm is initiated to attempt minimization of the user function. As the algorithm iterates, various conditions may arise which must be resolved by user input. These conditions are outlined later, along with suggestions for user action.

6. REPORT ON MINIMUM. SIMPLEX calls the user summary routine. If MINSUM is active, the user has an additional option of displaying the observed, predicted, and residual values, along with standard deviations of the normalized residuals.

7. EVALUATE FUNCTION. The current minimum point is output, along with its function value.

8. PLOT PREDICTIONS. This option is available with MINSUM only. The user can plot observed data, predicted curves, observed versus predicted response values, and residuals. These plots can be displayed on the terminal (low resolution), a graphics terminal (high res), or a hard copy plotter (high res). They can also be written to a file (low/high res) for later display. Sections 5 and 6 describe the various options in detail.

9. PLOT PROFILE. This option is available with both MINSUM and MINFUN. It can be used to obtain plots of function values when parameters are varied from the minimum point. Such plots can be very useful in deciding how well the model's parameters are determined by the data. As in the previous option, plots can be sent to a low or high resolution terminal, plotter, or file. Detailed information on profile plots is given in Section 7.

10. CALCULATE COVARIANCE MATRIX. Theoretical properties of the log likelihood function are exploited here to obtain numerical estimates of parameter standard deviations, correlations, and covariances at the current minimum point. Section 8 below discusses the underlying theory and the software options available.

11. GLOBAL SEARCH. Given a set of parameter ranges, the global search option calculates function values over the search range. Two types of search are available: (1) a grid search, which employs the IMSL routine ZSRCH to calculate values over an evenly spaced parameter grid, and (2) a random search, which uses a pseudo-random number generator to calculate random points distributed uniformly in parameter space. In either (1) or (2), the function value at each point is calculated and displayed, with the minimum point found being optionally retained as the current point. The user can also have the points printed to a file.

## 4.3. Minimization

### 4.3A. Interpreting the output

As SIMPLEX iterates through a minimization, it produces output consistent with the terminal output frequency and file output frequency, as described in Section 2.5 on algorithm data. Besides this iterative information, it may also print messages which indicate problems requiring user input. The two possibilities are discussed below.

(1) "MAXIMUM FUNCTIONS REACHED, CONTINUE? (<Y>,N)". This message occurs when the number of function evaluations has reached the maximum set in the algorithm data. If the user elects to quit, the entire current simplex is saved, making this an opportune time to stop and perhaps revise the algorithm data. Should the user elect to continue without revisions, the function counter (NF) is reset to 1, and minimization is resumed.

(2) "CONSECUTIVE RESTARTS INDICATE TUNING PROBLEMS". This message is informational only. A condition of having two or more consecutive restarts, or a large total number of restarts, is indicative of algorithm tuning problems. Usually, the algorithm should be halted with the control C (^C) option and the algorithm data should be revised. Section 9 discusses the problem of slow convergence further.

### 4.3B. Control C (^C)

On some systems, a user interrupt option may be implemented to allow a pause in program execution. On the VAX the interrupt signal is a control-C (^C). After receiving this signal, SIMPLEX completes the current iteration, outputs the status of the current simplex, and offers the option of continuing (with no changes) or returning to the main menu. If the user returns to the main, the current low point is kept as the new initial point, and the entire current simplex is saved. Then, if the user returns to the minimizer without changing the point values, the procedure resumes exactly where it left off.

### 4.3C. Minimum found

When SIMPLEX finds a point which passes both the simplex limit test and the axial search test, it displays the point, its function value, and the numbers of function calls, iterations and restarts required. Control is returned to the main menu, with the final point retained as the current point.

# 5. PLOTTING FACILITIES

## 5.1. Plot types

SIMPLEX has embedded within it fairly extensive plotting abilities which allow the user to plot observed and predicted data, residuals, objective function values, and parameter variations. In fact, users may at times employ SIMPLEX just to create graphs! Besides selecting between high and low resolution graphs, the user can choose from a wide range of plot types: scatter plots, histograms, line graphs, and step function graphs. These plot types are listed in Table 5.1, along with the codes used in SIMPLEX to select each type. The user can specify plotting characters: any keyboard symbol for low resolution, or one of the "markers" of Table 5.2 for high resolution. Also, for high resolution plots, the user can select the various line types and histogram panel patterns listed in Tables 5.3 and 5.4. Again, each plot type has an appropriate selection code.

One problem with low resolution graphs is that of representing coincident data points; this becomes significant for large data sets or for multiple plots on one set of axes. To avoid graphs which appear to have missing data, SIMPLEX uses a counter: if two data points are coincident the plot characters are replaced with the number "2". A third coincident point results in the number "3", and so on, from "2" to "9", then from "A" to "Z", and finally from "a" to "z". The plotter also uses a "stop" character as a limit to the counting at any one point. The default stop character is "z", so that one spot on a low resolution plot could represent up to 61 different data points. The counting concept is not applied if a space already contains a non-alphanumeric character, so the user can effectively enable and disable the counting by plotting with appropriate characters.

To create any of the low resolution plots, then, the user need only select a plot type (Table 5.1) and a plotting character. If the scatter plot option is chosen, a stop character must also be specified. Continuous curves are simulated using the scatter plot or sequential counting options with a high density of points to give the appearance of a continuous line. The sequential counting option first plots points using the count sequence outlined above; the user's plot character is used when that entire sequence has been run through.

The creation of high resolution plots involves selecting either a marker symbol, a line type, or a histogram panel pattern, depending on the plot type requested. See Tables 5.1 to 5.4 for a complete list of available options. If the connected points plot is chosen, the user may take advantage of the IGL smoothing function (which uses a cubic spline technique) to have a smoothed curve drawn through the points. The user can also indicate whether the plots should be displayed on the screen or sent to the hardcopy plotter. Unfortunately, the screen plotter recognizes different histogram panel pattern codes than the flatbed plotter; consequently, although the codes listed in Table 5.4 will result in some sort of panel on the screen plotter, the panel will not correspond to the one listed below.

Table 5.1. Options for low and high resolution plot types.

| Low Resolution | High Resolution |
|---|---|
| | 0 - Scatter Plot |
| 1 - Scatter Plot | 1 - Connected Points Plot |
| 2 - Step Function | 2 - Step Function |
| 3 - Open Histogram | 3 - Open Histogram |
| 4 - Solid Histogram | 4 - Solid Histogram |
| 5 - Vertical Bar Histogram | 5 - Vertical Bar Histogram |
| 6 - Sequential Counting Plot | 6 - Sequential Counting Plot |

Table 5.2. Options for high resolution markers (plot type 0 only).

| | |
|---|---|
| 0 - small solid square | 8 - medium X in open octagon |
| 1 - medium open square | 9 - medium + in open square |
| 2 - medium open octagon | A - medium inverted triangle |
| 3 - medium open triangle | B - medium open star |
| 4 - medium + sign | C - medium asterisk |
| 5 - small open diamond | D - large X |
| 6 - medium X sign in open square | E - up arrow |
| 7 - medium open square in X sign | F - down arrow |

Table 5.3. Options for high resolution line types (plot types 1, 2, and 3 only).

| | |
|---|---|
| 0 - solid line | 5 - large dashed dbl. dotted line |
| 1 - closely dotted line | 6 - large small dash line |
| 2 - single dotted dashed line | 7 - small dashed dbl. dotted line |
| 3 - small close dashed line | 8 - very large dashed line |
| 4 - medium spaced dashed line | 9 - medium spaced dotted line |

Table 5.4. Options for cross-hatching high resolution histogram panel patterns (plot types 4 and 5 only). These apply to the flatbed plotter. Results on the video screen may be somewhat different.

| | |
|---|---|
| 0 - close lines, 0 degrees (horizontal) | |
| 1 - close lines, 90 degrees(vertical) | |
| 2 - close lines, -45 degrees | 8,9,A-F - like 0-7 but medium spaced |
| 3 - close lines, 45 degrees | |
| 4 - close lines, -30 degrees | G-N - like 0-7 but wide spaced |
| 5 - close lines, 30 degrees | |
| 6 - close lines, -60 degrees | 0 - solid filled |
| 7 - close lines, 60 degrees | |

Listing 5.1 illustrates the various types of histograms and step functions available in low resolution. High resolution versions are similar. For example, Fig. 5.1 shows a high resolution open histogram. Finally, Listings 5.2 and 5.3, also created with the SIMPLEX plotting software, are included to illustrate one of the trickier aspects of low resolution plotting.

Listing 5.1. Histogram examples.

```
STEP FUNCTION:

!                 ****
!                 *  ******
!     *****       *       *
!     *   *********       *
!******                   *
!                         *
---------------------------
```

```
VERTICAL BAR HISTOGRAM:

!
!                  *
                   *
    *              *
    *              *
    *              *      *
    *              *      *
    *              *      *
----------------------------
```

```
OPEN HISTOGRAM:

!
!

!              ******
!   ******     *    *
!****    *     *    *
!   *    ****  *    *******
!   *    *  ****    *     *
!   *    *  *  *    *     *
---------------------------
```

```
          SOLID HISTOGRAM:

!

!            ******
!            ******
!  *****     ******
!  *****   ********
!*******   **************
!************************
!*************************
----------------------------
```

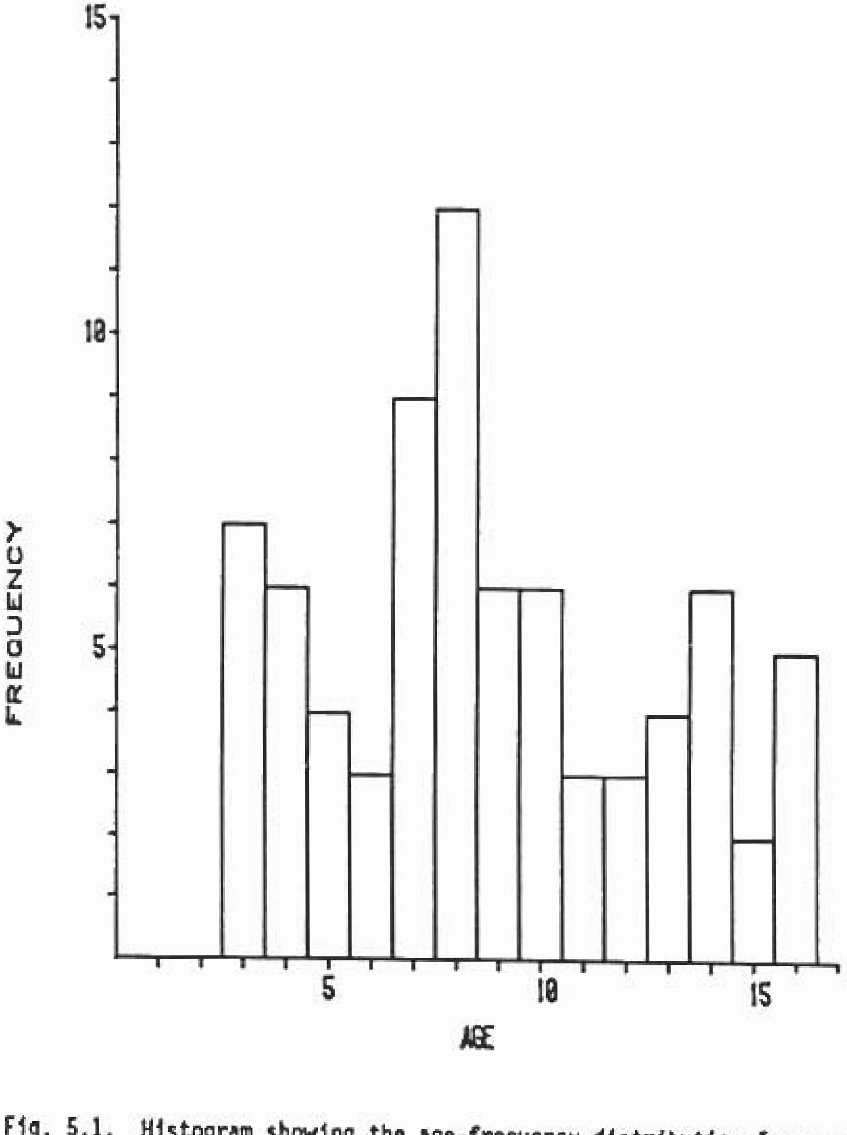


Fig. 5.1. Histogram showing the age-frequency distribution for a sample of 76 fish. The figure was created with the high resolution plotting capabilities, using the open histogram option. The frequencies of observations for ages 3 to 16 are, respectively: 7, 6, 4, 3, 9, 12, 6, 6, 3, 4, 6, 2, and 5.

Listing 5.2. Low resolution plot of observed (O) and predicted (P) length at age for freshwater mussels. Here the observed points were plotted first, then the predicted were overlayed. At any point that the two plots coincide, the automatic count feature "incremented" the O to a P, effectively masking the observed points.

```
          |
L         +
e       80+
n         |
g         +
t       60+
h         |                                                      O
          +                                   PPPPPPPPPPPPPPPPPPPP
          |                         PPPPPPPPPP
n       40+                   PPPPPP
          +              PPPPP
m         |          PPPP
m       20+       PPP
          +    PPP
          | PPP
          P-+----+--+---+--+---+--+--+--+---+--+---+--+---+---+--+
                           5                 10                15

                                AGE in Years
```

Listing 5.3. Low resolution plot of observed (O) and predicted (P) length at age for freshwater mussels. Here the observed points were plotted first, then the predicted were overlayed, but this time the "stop" character was set to "O" (the letter), so that overlayed predicted points no longer mask the O's.

```
          |
L         +
e       80+
n         |
g         +
t       60+
h         |                                                      O
          +                                    POPPPOPPOPPOPPPOPPP
          |                         POPPOPPPOP
n       40+                   OPPPOP
          +              PPOPP
m         |          PPPO
m       20+       PPO
          +    PPO
          | POP
          P-+----+--+---+--+---+--+--+--+---+--+---+--+---+---+--
                           5                 10

                                AGE in Years
```

## 5.2. Plotter hardware requirements

The high resolution plotting capabilities of the SIMPLEX package require the user to have access to a Tektronix 4105 Color Graphics Terminal and a Tek 4662 Flatbed Plotter, along with a regular VT100 or VT125 CRT terminal. With a small software modification the 4105 can be replaced with the higher resolution 4006. To use the plotting features of SIMPLEX, the user need onlylog onto the VT100, turn on the 4105 and turn on and load the 4662 with pen and paper. Normally this will allow the user to begin plotting immediately. Sometimes, however, the 4105 will have been left in a state unsuitable for our purpose.

Should the plotting terminal refuse to respond at all, first ensure that no-one (including yourself) is logged onto the 4105. This can easily be done with the command:

```
$ SHOW TERMINAL TK:
```

The terminal should have OWNER=NONE. Next check the baud rate on the terminal, by pressing the SETUP key and entering STATUS BAUD <CR>. The baud rate should be 1200. If the baud rate is incorrect, reset it using the SET BAUD 1200 command. To leave SETUP mode press the SETUP key again. Notice that while in setup mode an asterisk (*) appears as the prompt symbol.

If problems with the plotter persist, return to SETUP mode and try the FACTORY command, or leave SETUP and press SHIFT and CANCEL simultaneously. If at this point the terminal still will not respond, contact a member of the Computer Center.

# 6. PLOTTING OBSERVATIONS, ETC. (MINSUM ONLY)

The plotting software included in MINSUM provides a convenient tool for assessing how well the proposed model fits the data. Plots can involve observations, predictions, explanatory variables, and residuals. Low resolution plots, which are displayed on a regular CRT screen, are particularly useful for creating quick plots of observations and predictions, and for studying residual patterns. High resolution plots involve a bit more work, but the increased accuracy makes them more suitable for studying predicted curves. High resolution can also be important when the data set includes many data points.

Because use of the plotter generally involves a fair bit of input, and because each type of plot is prepared slightly differently, step by step instructions have been included as Appendix C. However, the plotting procedure is reasonably straightforward, and the reader may wish to skip ahead to the example in Section 12 outlining the analysis of a simple growth curve. This shows how the various plots can be used to study the MINSUM results quickly and efficiently.

## 6.1. MINSUM plotter options

Options available from the MINSUM plotter are displayed in listing 6.1. Of these, only option 3 merits lengthy description. Options 1 and 2 simply act as toggles, reversing the status of the display or resolution. The "display off" feature is especially needed when no high resolution terminal is available while creating high resolution plots. Selecting option 4 results in the most recently created plot being displayed on the appropriate device, while option 5 allows that same plot to be stored. The user will be prompted for a file name for the stored plot. Option 6 allows the user to recall saved plots; again the user will have to supply a file name. The current resolution setting does not affect the operation of options 4 through 6; i.e. high or low resolution plots are always stored and displayed at that resolution.

Listing 6.1. The MINSUM plotter main menu.

```
1) Set to high or low resolution.
2) Set display on or off.
3) Create plot.
4) Display current picture.
5) Save current picture on file.
6) Load and display saved picture.
7) Exit.
```

When option 3, "Create plot", is chosen, MINSUM displays the following list of available plots:

1) Y vs. X
2) Y vs. Y
3) X vs. X
4) RESIDUALS vs. Y
5) RESIDUALS vs. X

For each of these plots, the user is asked to supply axis ranges and labels, a title, and the codes for marker and line types, as listed in Tables 5.1 to 5.3.

The Y versus X plots are probably the most complex, in that MINSUM allows more than one plot on each set of axes. After selecting which explanatory variable X to use (if the model involves more than one X), the user chooses to plot Y-observed, Y-predicted, or exit. This choice is repeated until an exit is selected. For Y-predicted plots, the user decides whether to use the observed X-values in the plotted points or to generate X-values evenly across the range of X. Choosing the second option means that the user must specify the number of points to plot and set values for the remaining Xs (if more than one X is used). In either case, the user also specifies a plot type and character or a line code.

MINSUM also allows the user to edit the current point, so that different predicted curves can be plotted simultaneously. The high resolution plot (Fig. 1.1) in the introduction was created using this option.

Creation of a Y-observed versus Y-predicted plot is somewhat simpler. The user need only specify the variable for the horizontal axis (observed or predicted) and the ranges of the axes. For high resolution the user may also request that a 45-degree line be drawn through the plot. The X versus X plots are prepared in much the same way.

Finally, we cover the creation of residual plots. The user may elect to have residuals normalized before plotting, and may choose the horizontal variable to be Y-observed, Y-predicted, or a particular X. For high resolution plots, the horizontal line at residual zero is drawn, and the user may optionally have each residual point connected to this line with a lightly dotted vertical line.

# 7. PLOTTING PROFILES AND SECTIONS (MINSUM AND MINFUN)

## 7.1 Profiler description

An important aspect of function minimizing in general and model fitting in particular is knowing how the object function is affected by non-optimum values of the parameters. Once a minimum point is obtained, the practitioner must know how important each parameter is to the model: can it be ignored entirely or be fixed at some non-optimum value? Or must the parameter be very close to its optimum value to obtain a low level of the objective function?

The profiler is useful for obtaining such information about the parameters. Normally used near the function minimum, the profiler varies the "profile parameter" over a range of values, while minimizing the function with respect to the remaining parameters, and plots the resulting minimum values against the profile parameter. The curve so obtained, a "profile", gives a good picture of how well the profile parameter is determined by the data and the objective function. Indeed, statistical tests based on likelihood ratios can be applied to a profile to determine a (possibly asymmetric) confidence interval for the parameter. For an example, see Schnute (1983). For more general discussion, see Kalbfleisch (1971).

Because the profiler must calculate a number of minima (i.e., perform multiple simplex minimizations), it is important that the initial estimates and algorithm data be set up properly for each search. Normally, the user will only need to set up the search for the second point, as the first point is usually the minimum, previously determined by a simplex search. If the initial point is not within the profiling range, then the user will have to specify the algorithm data for a search at that point instead. In either case, the user is allowed to edit the algorithm data before minimization is carried out.

The initial estimates for points remaining after the first two are found by stepping away from the last calculated point a distance equal to the difference between the last two calculated points:

(7.1) $P3 = P2 + (P2 - P1)$,

while the step is set equal to the difference:

(7.2) $STEP = P2 - P1$.

It is hoped that in this manner the search at each remaining point will be initiated close to the actual minimum.

An interesting variation of the profile procedure is the "section", in which some or all non-profile parameters are held fixed (by setting the corresponding STEPs to zero during the EDIT phase). This shows the effect of varying only the profile parameter and typically results in a sharper curve than the profile, because the function is not being optimized at each point.

A final option of the profiler allows the user to study relationships among parameter estimates by graphing the various estimates against each other. For example, Fig. 1.6 in the introduction relates the optimal estimate of K to a prescribed value of $y_\infty$ in the growth model (1.1) applied to the mussel data. Such plots are easily obtainable, once the profiler has done its numerical work, and they provide graphic display of parameter correlations and of the "optimum fit" lines, as described below.

To understand better the meanings of "profile" and "section", consider an example from the biological literature. Schnute and McKinnell (1984, pp. 948-949, Table 2 and Fig. 9) describe a function relating total hatch rate for petrale sole (Eopsetta jordani) eggs to varying conditions of salinity ($x_1$, ppt) and temperature ($x_2$, °C). Modifying the example slightly (to illustrate a minimum, rather than maximum), let y (%) represent the percent hatch failure rate. Then y is related to $x_1$ and $x_2$ as follows:

$$(7.3) \qquad (100-y)^{\gamma} = a + bx_1^{\alpha} + cx_2^{\beta} + dx_1^{2\alpha} + ex_2^{2\beta} + fx_1^{\alpha}x_2^{\beta}$$

where $a=-44.23$, $b=8.88\times10^{-6}$, $c=5.58$, $d=-1.24\times10^{-11}$, $e=-0.122$, $f=5.65\times10^{-7}$, $\alpha=4.06$, $\beta=1.70$, $\gamma=.770$. Fig. 7.1 is a contour map representing y as a function of $x_1$ and $x_2$.

The central "*" of Fig. 7.1 gives optimum (lowest) failure rate for a salinity of 29.5 ppt and a temperature of 6.65 °C. Fig. 7.1 also shows two optimum lines and two sectioning lines. The double-dashed line (call it line #1) corresponds to temperatures optimal for various levels of salinity. The dashed-dotted line (line #2), on the other hand, indicates optimum salinity for various levels of temperature. The horizontal and vertical dotted lines (lines #3 and #4) correspond to sections defined by fixed temperature (7.5 °C) and fixed salinity (22 ppt), respectively.

These optimum and section lines can be related to Figs. 7.2 and 7.3, which show side views of the response surface. The shallow (solid) curve of Fig. 7.2 corresponds to values of y along line #1, that is, optimal (lowest) hatch failure for various levels of salinity. This is the profile for the variable $x_1$. The steeper (dashed) curve corresponds to values of y along line #3, that is, various levels of salinity and fixed temperature. This is a section with variable $x_1$. Similarly, Fig. 7.3 shows a section and profile (related to lines #2 and #4, respectively) along which temperature varies.

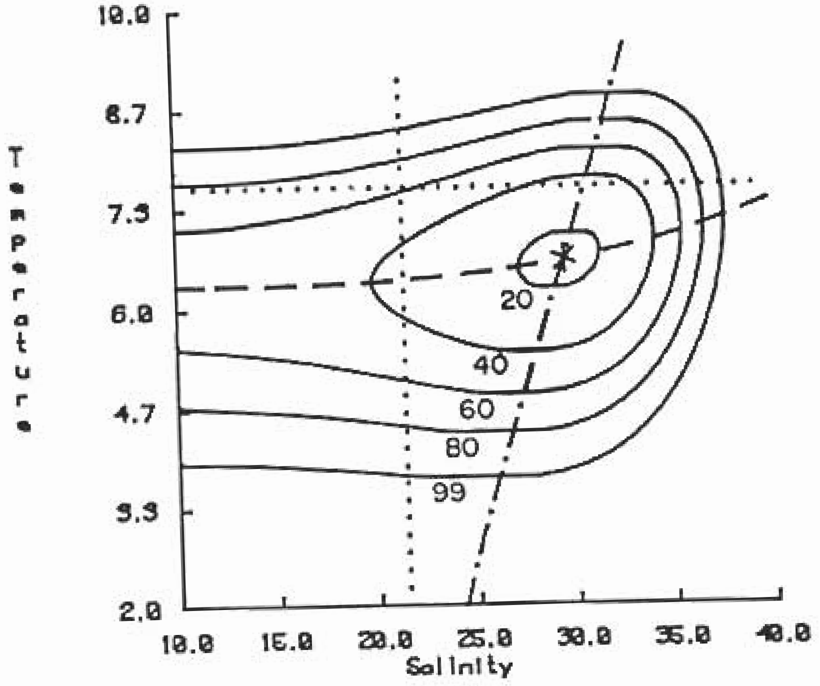


Fig. 7.1. Contours showing hatch failure rate in response to salinity and temperature for petrale sole eggs. Solid contours are lines of constant failure rate; the dashed line and dashed-dotted lines are optimal lines; and the dotted lines are sectioning lines. See the text for further discussion.

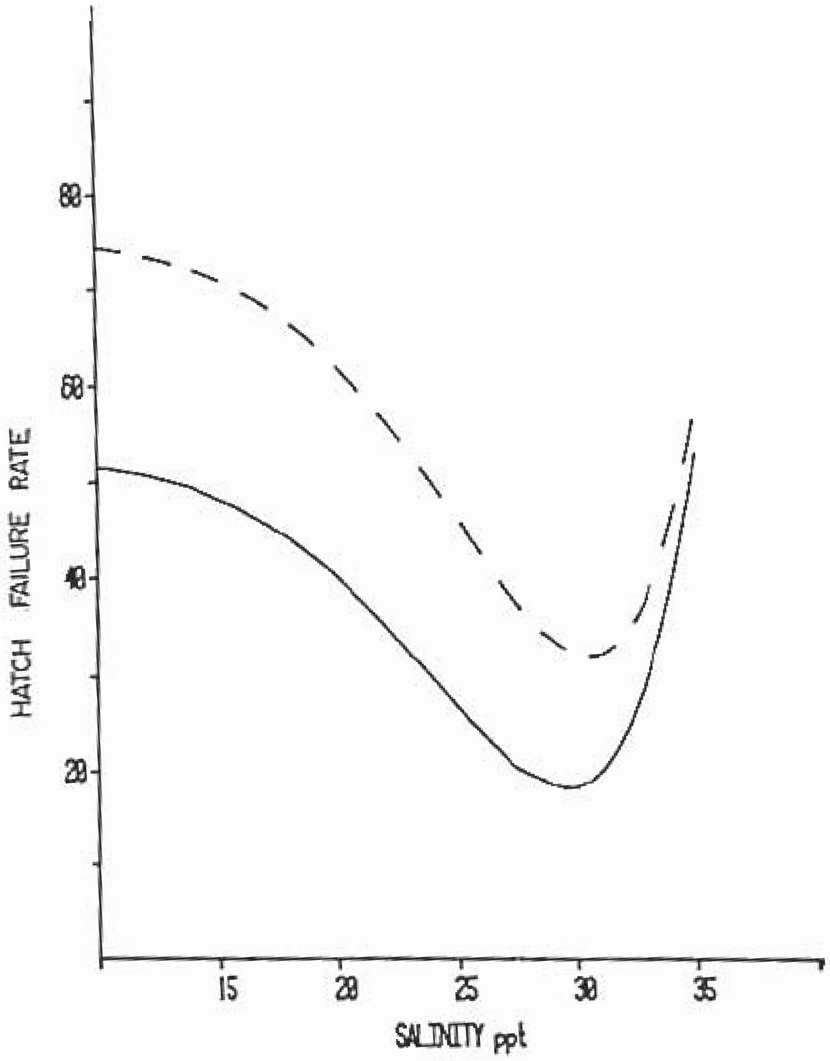


Fig. 7.2. Profile and section along salinity $x_1$ for response surface (7.3). The solid line is a profile, and the dashed line is a section. See the text for further discussion

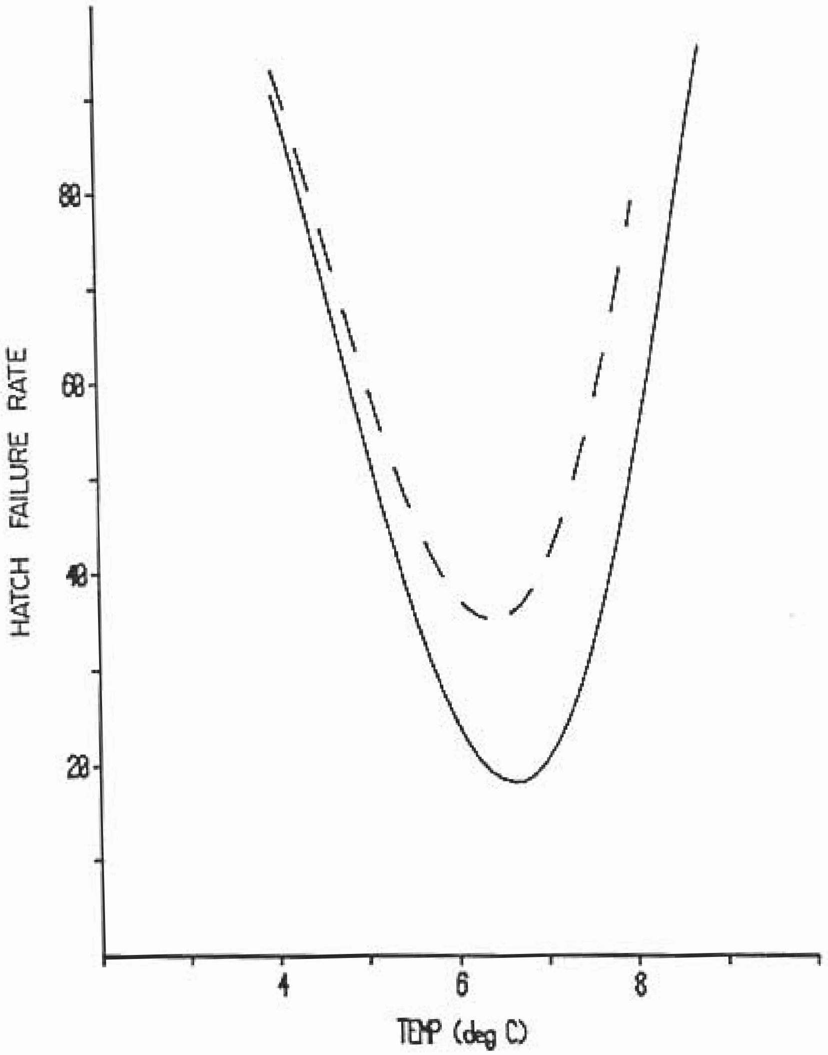


Fig. 7.3. Profile and section along temperature $x_2$ for the response surface (7.3). The solid line is a profile, and the dashed line is a section. See the text for further discussion.

## 7.2. Profiling instructions

Before calling the profiler, the user is responsible for locating a minimum point, stored automatically in the algorithm data as the initial, or current, point. This point may involve specified non-optimal values of some parameters, as long as both (1) the initial step in these parameters is set to zero and (2) the objective function has been minimized with respect to the remaining parameters. See further dicussion of this point in Section 10.2.

The profiler begins by requesting the number (i.e., index) of the profile parameter, and the current value of that parameter is displayed, along with the function value at the current point. The user is asked to give a range for the profile parameter. The value of the range endpoints determine what the next few steps will be. If the current value of the profile parameter is within the specified range, then the profiler will perform its calculation in two parts, by starting at the minimum point and working out toward each endpoint. The user is asked to supply the number of points for each range and to edit the algorithm data before calculation begins on the second point of the lower range. The profiler will calculate the remaining points on its own. Note that the two ranges need not have the same number of points.

If the current value of the profile parameter is outside the specified range, the user must perform a minimization on both the endpoint closest to the current value and the second profile point, as before. Both these minimizations are preceded by an edit of the algorithm data. The profiler then proceeds to calculate the remainder of the profile points.

During any of the above edits, neither the profile parameter value nor the associated step should be altered. (The profiler controls the profile parameter and sets its step to zero.) The user can set the simplex limit fairly high, because the profile plot is useful only to a few significant figures, and minimizations will proceed much faster. If the simplex limit is changed, remember also to reset the step size restart fraction correspondingly to avoid restarts. The user may also wish to set the terminal output frequency to 0 or to some very high value, thus avoiding huge amounts of output.

Once the edit for the second profile point is complete the remainder of the minimizations are performed automatically. If these perform poorly (e.g., restart warnings occur), the control C (^C) option can be used to stop calculations on the current point and move on to the next. Upon completion of the profiling procedure, the user may elect to create one of the various plots available or to have the profiler points and values written to a file. The plots are created by selecting one of the parameters to represent the horizontal axis and another parameter or the object function values to represent the vertical axis. These plots may be done in both high and low resolution, and can be stored on file for later retrieval.

The profile plotting procedure is reasonably simple. The user sets the ranges for the horizontal and vertical axes, provides a title and axis labels, and selects the plot type, as outlined in chapter 5. For low resolution the plot character is an asterisk (*). For high resolution the user selects a marker, line, or panel type, depending on the desired plot type.

# 8. CALCULATING COVARIANCES

## 8.1. Theory

One of the valuable features of maximum likelihood parameter estimates is that, for large samples, their distribution is known to be approximately multinormal (Kendall and Stuart, Vol. 2, 1979, p. 59). The estimates themselves are consistent; that is, they tend to the true parameter values as the sample size becomes large. Furthermore, if the objective function is taken to be the negative log likelihood, then the covariance matrix for the estimates is the inverse of the "Hessian", the matrix of second partial derivatives with respect to the parameter estimates.

It may be that the previous paragraph contains terminology unknown to the reader. Don't despair. The practical implications can be stated quite simply. Suppose that

$$X = (x_1, x_2, \ldots, x_N)$$

is the parameter vector to be estimated. Suppose also that $L(X)$ is the likelihood function for the parameters. (This is just the function that describes the probability of the observed data, given $X$.) If

(8.1) $$F(X) = -\log L(X) + C \quad ,$$

where $C$ is a constant (possibly dependent on the data, but not on the parameters), then the maximum likelihood estimate $\hat{X}$ for $X$ is the vector that minimizes $F$. The so-called Hessian for $F$ is the $N{\times}N$ matrix of second partial derivatives

(8.2) $$H = [[\partial^2 F(\hat{X})/(\partial x_i \, \partial x_j)]]$$

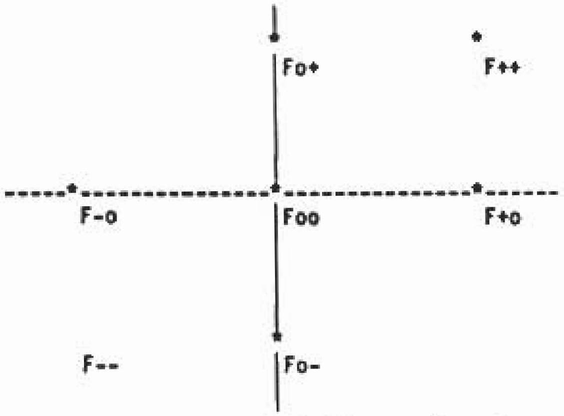
evaluated at $\hat{X}$, where $i$ and $j$ range from 1 to $N$. According to the theory, the asymptotic (that is, valid for large samples) covariance matrix $V$ for $\hat{X}$ is the inverse of $H$, that is,

(8.3) $$V = H^{-1} \quad .$$

In practical terms, if the user adopts the negative log likelihood (8.1) as an objective function, then SIMPLEX can be used to calculate the estimate $\hat{X}$. Also, the matrix $H$ in (8.2) can be used to compute the covariance matrix $V$. Although $H$ involves second derivatives which may be difficult to compute analytically, these can be approximated numerically. To describe the calculation, suppose that there are just two coordinates $x_1$ and $x_2$. If $d_1$ and $d_2$ represent small departures from the estimates $\hat{x}_1$ and $\hat{x}_2$, then define

(8.4) $\quad F_{ab} = F(\hat{x}_1+ad_1, \hat{x}_2+bd_2)$

where a and b can take the values -1, 0, and +1 (with corresponding subscripts -, o, and + on the left side of (8.4)). We are particularly interested in the values of F shown diagramatically on the $x_1x_2$-grid below:



In terms of these values, numerical approximations to the second derivatives are

(8.5a) $\quad \partial^2F/\partial x_1^2 = (F_{+o} - 2F_{oo} + F_{-o})/d_1^2 \quad .$

(8.5b) $\quad \partial^2F/(\partial x_1\partial x_2) = (F_{++}-F_{+o}-F_{o+}+ 2F_{oo}-F_{-o}-F_{o-}+F_{--})/(2d_1d_2) \quad .$

(8.5c) $\quad \partial^2F/\partial x_2^2 = (F_{o+} - 2F_{oo} + F_{o-})/d_2^2 \quad .$

Since each element of the Hessian matrix (8.2) involves only two derivatives ($x_i$ and $x_j$), the formulas (8.5) have natural extensions to the general N-dimensional case.

Schnute (1983) investigates a practical fisheries problem by letting F(X) be the negative log likelihood and computing $\hat{X}$ and the covariance matrix of $\hat{X}$ as described above. Schnute and Fournier (1980) similarly use a function essentially equal to twice the negative log likelihood, so it is convenient to build general software which allows the user to include a multiplicative constant at run time. Furthermore, in some problems the objective function is S(X), a sum of squares of residuals. In particular, the objective function takes this form when the residuals are presumed normal with variance $\sigma^2$, where $\sigma$ is an extra parameter in addition to the parameter vector X. In this case, the negative log likelihood is

(8.6) $\quad -\log L(X) = n \log(2\pi\sigma) + S(X)/(2\sigma^2)$

where n is the number of observations. An estimate of $\sigma^2$, adjusted for small sample bias, turns out to be

(8.7) $\quad \hat{\sigma}^2 = S(\hat{X})/(n-N)$

If we regard this estimate to be the correct value of $\sigma^2$, then

(8.8) $\quad F(X) = (n-N)\, S(X)/[2\, S(\hat{X})]$

represents the negative log likelihood, except for a constant. It follows from (8.8) that the sum of squares function S(X) must be adjusted by the factor $(n-N)/[2\, S(\hat{X})]$ if the Hessian of S(X) is to be used in computing the covariance matrix of the parameter estimates.

Incidentally, it may happen that the user function is not the actual sum of squares required in (8.6), but a multiple of S. The objective function A in Schnute and McKinnell (1984, p. 946, eq. 5.4; see also eq. 5.5) provides a practical example. Technically, then, the objective function should be multiplied by a constant to obtain S; however, this will have no effect on (8.8) because the constant will cancel between S(X) and $S(\hat{X})$. Consequently, the covariance matrix calculation based on (8.8) will be correct in any case. Only the expression (8.7) for $\sigma^2$ is affected by the multiplicative constant.

## 8.2. Software operation

In accordance with the above theory, the covariance calculator can be used to investigate relationships among parameter estimates after a minimum point has been found. The calculator begins by computing the covariance matrix V in (8.3) and, by the usual formulas, computes standard deviations, coefficients of variation, and correlations of the estimates. Output also includes the correlation and covariance matrix determinants.

To set up the covariance calculation, the user must supply three grids of varying sizes, on which the Hessian is calculated, as described in the previous section. The base grid is specified using the current point and step ($(\hat{x}_1,\hat{x}_2)$ and $(d_1,d_2)$, respectively, in (8.4)). This grid is scaled by three user specified factors (analogous to (a,b) in (8.4)) to obtain three grids for actual use. The user is allowed to edit both the point and step values, and is then asked to revise the default scaling factors of 0.10, 0.010, and 0.001, if needed. The user will probably not want to change these values unless a previous analysis with the defaults gave poor results.

For reasons explained at the end of section 8.1, the user may also select an adjustment factor for the objective function, called the covariance constant. If MINFUN with a UFUN is being used to define the user's function, the user will receive the following choices:

1) Constant = 1
2) User chosen constant

The user should enter whatever constant would be needed to make the objective function be the negative log likelihood. The default value in this case is choice 1. If MINSUM with a TEMPLATE is used the user will also receive the choices:

3) Constant = 0.5 * SIGMA ** (-2)
4) Constant = user chosen constant * 0.5 * SIGMA ** (-2)

In accordance (8.7)-(8.8), the default here is choice 3; the constant is automatically calculated for the user. Choice 4 allows for the possibility that the objective function may actually be a constant times the required sum of squares. As explained in the last paragraph of section 8.1, this effects the estimate of $\sigma^2$ only.

When the covariance calculations are completed, the user may elect to have the results printed to the screen or a file. The user may also elect to run the final summary routine (SUMMAR for TEMPLATE or NF=-1 for UFUN) before returning to the main SIMPLEX menu.

the absolute value of their correlations

If at least two of the grids give approximately the same results, and if all the standard deviations are positive and less than one, then the user can assume the analysis was successful. Conflicting results can occur because the difference calculations based on (8.5) are sensitive to the choice of scale; this is why we use three grids. Negative standard deviations (statistical nonsense) can arise when the grid is so small that the actual minimum point is not encompassed. (Remember that the "minimimum found" is only a numerical estimate of the true minimum.) To remedy this situation, either determine the minimum more precisely or apply slightly larger grid scale factors.

The calculator also supports investigation of the covariance matrix of a subset of the parameters. This would begin by fixing some parmeters at prescribed values, with the corresponding initial steps set to zero. The remaining parameters would then be estimated by minimization. If the calculator is entered at this point, it will automatically report a covariance matrix, standard deviations, etc., only on the parameters allowed to vary.

# 9. TROUBLE SHOOTING THE SIMPLEX SEARCH

Hopefully, this SIMPLEX package will enable users to build nonlinear parameteric models with relative ease and confidence, but please remember:

WE NEVER PROMISED YOU A ROSE GARDEN!

Although we've tried to keep things simple, the fact is that the user should have some understanding of the processes involved. We hope that this manual explains things clearly enough to allow you, dear reader, to apply the software knowledgeably, and thus avoid a certain amount of grief and frustration. This section deals with some of the hazards and pitfalls involved in non-linear estimation.

## 9.1. Multiple minima

Multiple minima represent an obvious difficulty in model building, because the user may accept a local minimum point as the global minimum. The occurrence of this sort of error can be reduced by initial and final searches using the wide search option. If the searches indicate that other "pockets" exist the user should investigate them before accepting the proposed minimum. The plot feature is also quite useful in this regard; the predicted line should fit the observed data, and any pattern (as opposed to a uniform random distribution) in the residual plot is indicative of an incorrect fit or an inappropriate model.

Essentially, multiple minima indicate some ambiguity in the model itself, and the user can often benefit from trying to understand biologically why this ambiguity exists. To determine the best parameter estimates, first eliminate any minima with unrealistic parameter values. Then look at the remaining minima and attempt to eliminate some by applying knowledge outside of the mathematical model. If problems with multiple minima persist, it may be time to sit down and take a hard look at the model and the data, with an eye towards revising the model. Again, the plotting features can be useful for studying the data and residuals.

## 9.2. Overflows

Of all the problems leading to program crashes, perhaps the most common is an overflow in the math library during a function evaluation. Such a crash can be particularly frustrating if the user has just waded through a long search or profile calculation.

Users of MINSUM will find that they are partially protected from overflows because MINSUM will stop summing TERMs when the sum of squares reaches 10**30 (overflows occur at about 10**37 in double precision). Users writing their own UFUN can include a check for overflows themselves. Even if a check is introduced into the program, problems can still occur, and the best way to avoid overflows is to ensure that the program is always dealing with reasonable values. This means thinking about ranges for profiles and wide searches to ensure that an exponential of a large number will not occur, or that divisors do not get too close to zero.

## 9.3. Slow convergence

As the user becomes familiar with SIMPLEX, he or she will notice that SIMPLEX sometimes finds minima fairly quickly, while at other times the search algorithm seems to wander, making very little progress. The theory behind the search method often suggests possible reasons for poor performance. In most cases, proper manipulation of the algorithm data can improve performance tremendously. The "simplex limit" and "step reduction fraction" are particularly important data items.

To understand the mechanism leading to slow convergence, recall the two convergence checks: the simplex limit test and the axial search. Notice that the first test is based on the variation of the function values, while the second test relies on varying the parameters. Since we are generally more interested in the parameter values, it might seem that we should rely completely on the axial search test and omit the function value test; however, the latter is easier and faster to perform.

Regardless of the priority of the two tests, they must be performed on comparable scales; that is, variations in function values contemplated in the simplex limit test should be consistent with parameter variations contemplated in the axial search. If the simplex limit test admits a simplex on which the parameters are varying by ten percent, and the axial search uses the parameters varied by one percent, multiple restarts will occur because the axial search will trap points admitted by the simplex limit test. If we are actually interested in parameter variations of the order of one percent, then the action to take here is to lower the simplex limit; that is, bring it in line with step reduction fraction. Conversely, if the simplex iterates for a long time with no significant improvement in the parameters, it may be time to raise the value of the simplex limit to allow the search to converge.

# 10. ADDITIONAL POSSIBILITIES

## 10.1. Chi-square and weighted least squares

An important feature of the TEMPLATE concept is that it is not limited to the sum of squares objective function, used in the example discussed in the introduction. For example, the $\chi^2$ statistic can be used by letting

```
(10.1)    TERM = (YO - YP)**2 / YP
```

where YO and YP are interpreted as observed and expected frequencies, respectively. Thus, in this case, the TEMPLATE function PRED would be a predicted frequency based on the parameter vector PARS. Similarly, following Schnute and Fournier (1980), one might let

```
(10.2)    TERM = YO * LOG(YO/YP)  ,
```

to obtain the negative log likelihood for observed and predicted frequencies YO and YP.

Since the function TERM can depend not only on observed and predicted values YO and YP but also on the observed XOs, the auxiliary parameters AUX, and the model parameters PARS, MINSUM can also handle weighted least squares. Consider the definition

```
(10.3)    TERM = XO(AUX(1)) * (YO - YP)**2   .
```

Here, one of the explanatory variables in the data is actually used as a weighting factor within a sum of squares objective function. The auxiliary parameter AUX(1) is used to select which particular explanatory variable is so used. If one wished to consider two possible weighting schemes, both could be included in the observed data, and INITM could include code allowing the user to select either weighting scheme by setting AUX(1) accordingly.

## 10.2. Holding some parameters fixed

It is often of great interest to study the model with certain parameters fixed at prescribed values. SIMPLEX allows the user to do this extremely easily. Simply include the prescribed values in the initial point, and set the corresponding initial steps to zero. This forces the initial simplex and all subsequent simplices to lie in a hyperplane with the preset parameters held constant. The final minimum reflects optimal values for the free parameters, given the fixed values of the remaining ones.

## 10.3. Imposing constraints

At times the simplex search must be told to avoid parameter values that don't make biological sense - like a negative age or a salmon smolt weighing sixteen pounds. Normally the objective function topology will cause the simplex to avoid unreasonable parameter values, but, if this is not the case, the user can easily force the issue by adding a penalty function. For example, UFUN might include the code

```
        IF (PARS(1).GT.LIMIT) THEN
             F = F + LARGE_CONSTANT * (PARS(1)-LIMIT) ** 2
        ENDIF
```

The important thing here is that the objective function F is continuous (the added penalty goes to zero as the parameter approaches the limit), but it need not be differentiable. In MINSUM, the ICASE parameter in the TERM routine gives the user option of adding the penalty only once, at the end of the objective function calculation when ICASE=NDAT.

## 10.4. Eliminating linear parameters

In fisheries research, one often encounters models in which some parameters enter linearly and can be estimated by linear regression, given values of the remaining parameters. The earlier example (7.3), repeated here for the reader's convenience:

$$(10.4) \qquad (100-y)^{\gamma} = a + bx_1^{\alpha} + cx_2^{\beta} + dx_1^{2\alpha} + ex_2^{2\beta} + fx_1^{\alpha}x_2^{\beta} \quad ,$$

has this feature. Here, if α, β, and γ are given, then a, b, c, d, e, and f can be estimated by linear regression. If the regression itself is included in UFUN, then the simplex search needs to deal with only the three parameters α, β, and γ, rather than the full set of nine parameters. Schnute and McKinnell (1984) discuss this technique fully for (10.4) and a more general class of models, and show that it leads to efficient and robust searches for optimal parameter estimates.

# 11. VARIATIONS OF THE SIMPLEX SEARCH

Numerous articles and reports have been written since Nelder and Mead's (1965) paper, suggesting various methods for improving the performance of the simplex search algorithm. Users with programming experience might be interested in trying some of the following suggestions to see if they can improve algorithm performance in their application. These suggestions were not implemented in this package for various reasons, usually because they provided no consistent improvement (or even hindered progress) in the cases we tried. The user, however, may find that certain problems lend themselves to these alterations.

## 11.1. Adjustable action coefficients

Nelder and Mead (1965) contemplated adjustable coefficients to control the sizes of the reflections, expansions, and contractions. This SIMPLEX package uses a reflection coefficient ALPHA = 1, a contraction coefficient BETA = 0.5, and an extension coefficient GAMMA = 2. These values are suggested by Nelder and Mead(1965) and Nash(1979). Walmsley (1981), however, suggests that BETA=0.5 gives too drastic a contraction, and he proposes BETA=0.75, so the contracted point would move only one quarter the distance towards the centroid. In the limited testing done for this package, we found that for each problem the choice of starting point greatly affected which BETA was superior, but overall the difference was insignificant, so we kept BETA at 0.5.

## 11.2 Multiple extensions

Walmsley also advocates continuing the extension process as long as lower function values are obtained. Unfortunately, the second extension is often not accepted, resulting in a wasted function evaluation. Also, when it is accepted, the updated simplex is distorted (long and narrow) and requires several iterations to reshape itself. We found no benefit, and some loss, from this modification.

## 11.3 Multi-point reflections

Evans and Craig (1978) published an interesting paper proposing a modified Nelder-Mead algorithm in which all the "worst" points of the simplex are reflected through the centroid of the "best" points. The two groups are identified by determining which grouping of high and low points gives the greatest difference in mean function values. In our experience, this method finds the minimum with much fewer iterations than does the Nelder-Mead method, but with about the same number of function evaluations. The code for this algorithm can be found in COMMO2.FOR, which can simply replace COMMON.FOR.

Users are warned that fiddling with the simplex algorithm can become quite fascinating; don't lose sight of the original problem!

# 12. WORKED EXAMPLE

This sample run is designed to give the reader an overview of SIMPLEX operation. It is based on the example discussed in the introduction: fitting length at age data for freshwater mussels (Anodonta kennerylii) to a von Bertalanffy curve (1.1), using the sum of squares of residuals as the objective function. The data consist of mean lengths at each of 16 ages. Although the software allows for weighted least squares and many other objective functions (Section 10), such possibilities are not explored here. The main point of this example is to illustrate SIMPLEX operation, not the analytical process of deciding on the best choice of model and/or objective function.

Because the objective function is the sum of similar terms, we can use MINSUM with TEMPLATE. The sample template shown in Listing 3.1 suits our purposes, so it is compiled and linked with MINSUM, COMMON, IGL and IMSL, as described in Section 3.4. For example, suppose that the file VONB.FOR contains the FORTRAN code in Listing 3.1. Then the commands

```
$ FORTRAN VONB

$ @SIMPLEX:GOMINSUM VONB
```

would result in the executable file VONB.EXE, which is capable of performing all MINSUM functions applied to the von Bertalanffy curve (1.1) and a sum of squares objective function.

We must also prepare the data file, beginning with the number of independent variables (one, namely age) and a FORTRAN format specification for MINSUM to read the data. Note that we could also specify a free format read with a blank line. The data file then has the following form:

```
1
(F4.0,T7,F10.0)
   1   7.36
   2  14.33
   3  21.86
   4  27.61
   5  31.59
   6  35.38
   7  39.02
   8  41.19
   9  43.89
  10  45.08
  11  47.41
  12  48.95
  13  50.14
  14  51.79
  15  51.77
  16  54.16
```

Both programs and data are now ready to go. The remainder of this section consists of an annotated example run. The program prompts and user responses appear in the left column, with user responses denoted by a # symbol in column one. Notes and comments appear in the right hand column.

```
    COMPUTER INPUT AND OUTPUT                          COMMENTS
    -------------------------                          --------

$ RUN VONB

                FUNCTION MINIMIZER
                ------------------

 NAME OF DATA FILE?                                Enter name of data
#MUSSELS.DAT                                       file prepared for
                                                   this run
 ALGORITHM DATA FILE?
 ENTER FILE NAME OR <CR> TO
 ACCEPT DEFAULTS.                                  First run so
#<CR>                                              accept defaults

 ALGORITHM DATA SET TO DEFAULT VALUES.


                     SIMPLEX MENU                  Main menu for SIMPLEX
                     ------------

   1) Re-initialize
  1a) New data only
  1b) Model and parameters only.
   2) Edit, 3) Load, 4) Save Alg data.
   5) MINIMIZE. 6) User summary.
   7) Function call at current low.
   8) Prediction or 9) Profile plots.
  10) Covariance matrix.
  11) Search Globally.
  12) Quit.

 ENTER CODE:
#11                                                Try a global search
                                                   to get an idea of the
      GLOBAL SEARCH                                parameter values
      -------------

 SET SEARCH BOUNDS:                                Search is done
 -----------------                                 within this range

 CHOOSE:

 1) Review current bounds.                       ! Displays max and min
                                                 ! for all params
 2) Set bounds.                                  ! Allows user to
                                                 ! specify max & min
 3) Take default bounds of POINT +- STEP.        ! Gives default bound
                                                 ! values
 4) Edit POINT and STEP with editor.       ! Allows user to reset
                                           ! defaults

 ENTER CODE: ( 1 - 4 or <CR> if finished.)!
#2                                         ! Set min and max

 Enter low value for range of parameter 1 ! Enter values
#40                                        !
 Enter high value for range of parameter 1!
#60                                        !
 Enter low value for range of parameter 2 !
#.1                                        !
 Enter high value for range of parameter 2!
#1                                         !
 Enter low value for range of parameter 3 !
#0                                         !
 Enter high value for range of parameter 3!
#.5                                        !

  CHOOSE:

  1) Review current bounds.                !
  2) Set bounds.                           !
  3) Take default bounds of POINT +- STEP. !
  4) Edit POINT and STEP with editor.      !
                                           !
  ENTER CODE: ( 1 - 4 or <CR> if finished.)!
#<CR>                                      ! Bounds are set
 Enter Q, the number of points to search: !
 Default of Q = 10 * N.                    !
#64                                        ! 4x4x4=64 gives nice
 Enter file name to write to.              ! grid values
 Enter <CR> if no output file is desired.  !
#<CR>                                      ! No output file
 Select Random or Grid search:             ! desired
 Enter R or G                              !
#G                                         ! Request a grid search
                                           !
                                           ! Output looks like:

  VAL: 451.86045 POINT:   44.000000    0.28000000    0.10000000
  VAL: 906.05667 POINT:   48.000000    0.46000000    0.20000000
  VAL: 2311.9458 POINT:   52.000000    0.64000000    0.30000000
  VAL: 4212.6078 POINT:   56.000000    0.82000000    0.40000000
  VAL: 228.90367 POINT:   48.000000    0.28000000    0.10000000
  VAL: 1496.4592 POINT:   52.000000    0.46000000    0.20000000
  VAL: 3565.6761 POINT:   56.000000    0.64000000    0.30000000
  VAL: 1423.8246 POINT:   44.000000    0.82000000    0.40000000
  VAL: 361.64637 POINT:   52.000000    0.28000000    0.10000000
  VAL: 2504.2889 POINT:   56.000000    0.46000000    0.20000000
  VAL: 1135.2687 POINT:   44.000000    0.64000000    0.30000000
  VAL: 1896.5065 POINT:   48.000000    0.82000000    0.40000000
  VAL: 850.08854 POINT:   56.000000    0.28000000    0.10000000
  VAL: 733.08140 POINT:   44.000000    0.46000000    0.20000000
```

```
VAL: 1501.8100 POINT:   48.000000     0.64000000     0.30000000
VAL: 2826.1009 POINT:   52.000000     0.82000000     0.40000000
VAL: 443.75653 POINT:   44.000000     0.28000000     0.20000000
VAL: 825.54508 POINT:   48.000000     0.46000000     0.30000000
VAL: 2172.9144 POINT:   52.000000     0.64000000     0.40000000
VAL: 4804.7660 POINT:   56.000000     0.82000000     0.10000000
VAL: 201.33166 POINT:   48.000000     0.28000000     0.20000000
VAL: 1388.4657 POINT:   52.000000     0.46000000     0.30000000
VAL: 3393.2262 POINT:   56.000000     0.64000000     0.40000000
VAL: 1732.6273 POINT:   44.000000     0.82000000     0.10000000
VAL: 311.48438 POINT:   52.000000     0.28000000     0.20000000
VAL: 2365.6177 POINT:   56.000000     0.46000000     0.30000000
VAL: 1053.3355 POINT:   44.000000     0.64000000     0.40000000
VAL: 2290.2783 POINT:   48.000000     0.82000000     0.10000000
VAL: 774.21468 POINT:   56.000000     0.28000000     0.20000000
VAL: 676.85592 POINT:   44.000000     0.46000000     0.30000000
VAL: 1392.9508 POINT:   48.000000     0.64000000     0.40000000
VAL: 3314.3245 POINT:   52.000000     0.82000000     0.10000000
VAL: 440.00958 POINT:   44.000000     0.28000000     0.30000000
VAL: 749.95912 POINT:   48.000000     0.46000000     0.40000000
VAL: 2598.8948 POINT:   52.000000     0.64000000     0.10000000
VAL: 4608.6269 POINT:   56.000000     0.82000000     0.20000000
VAL: 178.43574 POINT:   48.000000     0.28000000     0.30000000
VAL: 1285.6173 POINT:   52.000000     0.46000000     0.40000000
VAL: 3918.8406 POINT:   56.000000     0.64000000     0.10000000
VAL: 1628.9343 POINT:   44.000000     0.82000000     0.20000000
VAL: 266.30477 POINT:   52.000000     0.28000000     0.30000000
VAL: 2232.2817 POINT:   56.000000     0.46000000     0.40000000
VAL: 1308.7050 POINT:   44.000000     0.64000000     0.10000000
VAL: 2158.8255 POINT:   48.000000     0.82000000     0.20000000
VAL: 703.61666 POINT:   56.000000     0.28000000     0.30000000
VAL: 625.30724 POINT:   44.000000     0.46000000     0.40000000
VAL: 1728.8496 POINT:   48.000000     0.64000000     0.10000000
VAL: 3152.0564 POINT:   52.000000     0.82000000     0.20000000
VAL: 441.00752 POINT:   44.000000     0.28000000     0.40000000
VAL: 990.54082 POINT:   48.000000     0.46000000     0.10000000
VAL: 2454.4983 POINT:   52.000000     0.64000000     0.20000000
VAL: 4410.4264 POINT:   56.000000     0.82000000     0.30000000
VAL: 160.66312 POINT:   48.000000     0.28000000     0.40000000
VAL: 1608.5078 POINT:   52.000000     0.46000000     0.10000000
VAL: 3741.5148 POINT:   56.000000     0.64000000     0.20000000
VAL: 1525.4549 POINT:   44.000000     0.82000000     0.30000000
VAL: 226.61803 POINT:   52.000000     0.28000000     0.40000000
VAL: 2647.0595 POINT:   56.000000     0.46000000     0.10000000
VAL: 1220.8147 POINT:   44.000000     0.64000000     0.20000000
VAL: 2026.9390 POINT:   48.000000     0.82000000     0.30000000
VAL: 638.87225 POINT:   56.000000     0.28000000     0.40000000
VAL: 793.15865 POINT:   44.000000     0.46000000     0.10000000
VAL: 1614.2649 POINT:   48.000000     0.64000000     0.20000000
VAL: 2988.5962 POINT:   52.000000     0.82000000     0.30000000


VALUE ON ENTRY OF:   25207.363              ! SOS minimum using
FOUND AT POINT:                             ! SIMPLEX default
 1.00000      1.00000    1.00000            ! values
 WHILE SEARCH GIVES VALUE OF:  160.663        Note improvement-
 AT POINT:                                    this gives a likely
  48.0000    0.28000   0.40000                starting point


 ACCEPT NEW POINT FROM SEARCH? (<Y>, N)       Point found in global
#Y                                            search is new CURRENT
                                              point in SIMPLEX
 Repeat with same bounds? (Y/<N>):
#N                                            Return to SIMPLEX main

                   SIMPLEX MENU
                   ------------
                                              SIMPLEX main menu
                   as above....

 ENTER CODE:
#2                                            Choose to edit the
                                              algorithm data

  3 VARIABLE(S)
                                            . Editor lists the
 IP) INITIAL POINT:                         ! algorithm data
  1) 48.00000  2) 0.280000  3) 0.400000     !
 IS) INITIAL STEP:                          !
  1) 0.100000  2) 0.100000  3) 0.100000     !
 SL) REQUIRED SIMPLEX LIMIT =               !
     0.10000000E-05 REL.                    !
 MF) MAX. OF  400 FUNCTION CALLS            !
                                            !
 TF) TERMINAL DISPLAY FREQUENCY:     1      !
                                            !
 FF) FILE WRITE FREQUENCY:           0      !
                                            !
 SR) STEP REDUCTION FRACTION:  0.10000000   !
                                            !
 ENTER CODE TO EDIT PARTICULAR FIELD,       !
 "AL" TO EDIT ALL FIELDS, "H" FOR HELP,     !
 <CR> TO SEE CURRENT VALUES,                !
 OR "Q" TO EXIT EDITOR                      !
#IS                                         ! First set reasonable
                                            . initial steps, initial
 <CR> TO ACCEPT CURRENT OR DEFAULT VALUE.   ! point is okay
                                            !
 CURRENT VALUE OF STEP #  1 :  0.10000000   !
 NEW VALUE?                                 !
#10                                         ! Gives 50 +/- 10 for
                                            ! YINF
 CURRENT VALUE OF STEP #  2 :  0.10000000   !
 NEW VALUE?                                 !
#<CR>                                       ! Accept default of 0.25
                                              +/- 0.1 for K
 CURRENT VALUE OF STEP #  3 :  0.10000000
 NEW VALUE?
#.3                                         ! Gives TZERO = 0.25
                                            ! +/- 0.3
 ENTER CODE TO EDIT PARTICULAR FIELD,
 "AL" TO EDIT ALL FIELDS, "H" FOR HELP,
 <CR> TO SEE CURRENT VALUES,
 OR "Q" TO EXIT EDITOR
#SL                                         ! Now reset SIMPLEX
                                            ! limit
 <CR> TO ACCEPT CURRENT OR DEFAULT VALUE.

 CURRENT SIMPLE LIMIT TYPE IS RELATIVE .
 NEW TYPE? (REL or ABS):
#<CR>                                       ! Accept default of
                                            ! relative
 CURRENT SIMPLEX LIMIT:  0.10000000E-05     !
 NEW VALUE?
#.001                                       ! Set SIMPLEX limit at
                                            ! 1E-3
 ENTER CODE TO EDIT PARTICULAR FIELD,       !
 "AL" TO EDIT ALL FIELDS, "H" FOR HELP,     !
 <CR> TO SEE CURRENT VALUES,                !
 OR "Q" TO EXIT EDITOR                      !
#<CR>                                         Review current values

  3 VARIABLE(S)

 IP) INITIAL POINT:
  1) 48.00000  2) 0.280000  3) 0.400000
 IS) INITIAL STEP:
  1) 10.00000  2) 0.100000  3) 0.300000
 SL) REQUIRED SIMPLEX LIMIT =               !
     0.10000000E-02 ABS.                    !
                                            !
 MF) MAX. OF  400 FUNCTION CALLS            !

 TF) TERMINAL DISPLAY FREQUENCY:     1

 FF) FILE WRITE FREQUENCY:           0
                                            !
 SR) STEP REDUCTION FRACTION:  0.10000000   !
                                            !
 ENTER CODE TO EDIT PARTICULAR FIELD,       !
 "AL" TO EDIT ALL FIELDS, "H" FOR HELP,     !
 <CR> TO SEE CURRENT VALUES,                !
 OR "Q" TO EXIT EDITOR                      !
#Q                                          ! Values are okay --
                                            ! return to main menu
                      SIMPLEX MENU
                      ------------
                                              SIMPLEX main menu

                      as above

 ENTER CODE:                                !
#5                                          ! Attempt minimization
 INITIAL POINT:                             ! Initial point as found
   48.000000     0.28000000     0.40000000! in global search
                                            !
```

```
INITIAL FUNCTION VAL:   160.66312          Function value at
                                           initial point
INITIAL HIGH:           974.86160          Worst value in
                                           constructed SIMPLEX
INITIAL LOW:            143.03086          Best value in
                                           constructed SIMPLEX

                                           First iteration:

   6 CALLS,    1 ITERS,  0  RESTARTS.
 0 VALUES IMPROVED.  ACTION ACCEPTED: CONTRACTION
CURRENT MINIMUM FOUND AT:
  48.000000         0.28000000         0.70000001
CURRENT MINIMUM:   143.03086    CURRENT MAXIMUM:   441.70366

                                          ! Second iteration:

   7 CALLS,    2 ITERS,  0  RESTARTS.
 2 VALUES IMPROVED.  ACTION ACCEPTED: REFLECTION
CURRENT MINIMUM FOUND AT:
  48.000000         0.28000000         0.70000001
CURRENT MINIMUM:   143.03086    CURRENT MAXIMUM:   375.89989

                                           Third iteration

   9 CALLS,    3 ITERS,  0  RESTARTS.
 0 VALUES IMPROVED.  ACTION ACCEPTED: CONTRACTION
CURRENT MINIMUM FOUND AT:
  51.055556         0.27351852         0.51388889
CURRENT MINIMUM:   126.60105    CURRENT MAXIMUM:   182.53134

                      .                   !
                      .                   ! SIMPLEX doing well,
                      .                   ! lets return later,
                      .                   ! near the end...
                      .                   !
                      .                   ! 41st iteration:

  79 CALLS,   41 ITERS,  0  RESTARTS.
 2 VALUES IMPROVED.  ACTION ACCEPTED: REFLECTION
CURRENT MINIMUM FOUND AT:
  57.315559         0.16418968         0.15333421
CURRENT MINIMUM:   3.9800285    CURRENT MAXIMUM:   3.9812958

                                          ! Max and min are very
                                          ! close, so we should
                                          ! be near the minimum

  81 CALLS,   42 ITERS,  0  RESTARTS.
 0 VALUES IMPROVED.  ACTION ACCEPTED: CONTRACTION
CURRENT MINIMUM FOUND AT:
  57.291145         0.16441514         0.15506405
CURRENT MINIMUM:   3.9799164    CURRENT MAXIMUM:   3.9807709
```

```
  87 CALLS,   42 ITERS,  0 RESTARTS.           !
                                                !
 MINIMUM OF    3.9799164     FOUND AT:          ! Found a minimum!
 57.291145     0.16441514     0.15506405        !
                                                !
                                                ! Return to SIMPLEX
                     SIMPLEX MENU               ! main with good point
                     ------------               !
                                                !
                     as above....               !
                                                !
 ENTER CODE:                                    !
#4                                              ! We want to save good
                                                ! point on file, so we
 CURRENT ALG DATA FILE NAME IS: DATA.ALG        ! save algorithm data
 ENTER NEW FILE NAME,                           !
 <CR> TO SAVE DATA IN CURRENT FILE,             !
 OR ! TO RETURN TO THE MENU.                    !
#MUSSELS.ALG                                    ! Save in MUSSELS.ALG
                                                !
                     SIMPLEX MENU               !
                     ------------               ! SIMPLEX main menu
                                                !
                     as above                   !
                                                !
 ENTER CODE:                                    !
#8                                              ! Now plot to see how
                                                ! good fit is
                    PLOTTER MENU                ! Plotter main menu
                    ------------                !
                                                !
  1) RESET HIGH OR LOW RESOLUTION               !
        CURRENTLY=LOW                           !
  2) RESET DISPLAY ON OR OFF                    !
        CURRENTLY=ON                            !
  3) RESET LOW RESOLUTION SIZE                  !
      CURRENT HEIGHT= 22 , WIDTH=  80           !
  4) CREATE PLOT                                !
  5) SAVE CURRENT PICTURE ON FILE               !
  6) DISPLAY CURRENT PICTURE                    !
  7) LOAD AND DISPLAY A SAVED PLOT              !
  8) <EXIT>                                     !
                                                !
  ENTER CODE:                                   !
#3                                              ! Set width of plot to
  ENTER HEIGHT AND WIDTH OF GRAPH,              ! fit on page
  SEPARATE WITH SPACE                           !
  <22,80>                                       !
#20 60                                          !
                                                !
                                                !
                    PLOTTER MENU                !
                    ------------                ! Plot menu
 1) RESET HIGH OR LOW RESOLUTION
       CURRENTLY=LOW
 2) RESET DISPLAY ON OR OFF
       CURRENTLY=ON
 3) RESET LOW RESOLUTION SIZE
     CURRENT HEIGHT= 20 , WIDTH=  60
 4) CREATE PLOT
 5) SAVE CURRENT PICTURE ON FILE
 6) DISPLAY CURRENT PICTURE
 7) LOAD AND DISPLAY A SAVED PLOT
 8) <EXIT>

 ENTER CODE:
#4

        PLOT TYPES                               Menu for various
        ----------                               plots available

 CHOOSE ONE OF THE FOLLOWING OPTIONS:

        1) Y VS. X
        2) Y VS. Y
        3) X VS. X
        4) RESIDUALS VS. Y
        5) RESIDUALS VS. X
        6) <EXIT>

  ENTER CODE:
#1                                               Plot Y observed &
                                                 predicted on X
  SET RANGES FOR X DATA:                         Select range of X
                                                 Default is min and
  COMPUTED MIN=   1.0000000                      max of X values
  COMPUTED MAX=   16.000000
  DO YOU WISH TO CHANGE THESE VALUES?
  Y,<N>?
#Y                                               Reset range for nice
  PLEASE ENTER MINIMUM                           axis
#0
  PLEASE ENTER MAXIMUM
#20                                            ! Range is now from 0
                                               ! to 20
  SET RANGES FOR Y DATA:                       ! Defaults of min of
  COMPUTED MIN=   7.3600001                    ! predicted and
  COMPUTED MAX=   54.160000                    ! observed and max of
  DO YOU WISH TO CHANGE THESE VALUES?          ! pred and obs
  Y,<N>?
#Y                                               Reset for nicer axis
  PLEASE ENTER MINIMUM
#0
  PLEASE ENTER MAXIMUM
#100                                             Range is now from 0
  PLEASE ENTER THE PLOT TITLE                    to 100
```

```
#OBS. AND PRED. LENGTH AT AGE FOR FRESHWATER MUSSELS

  ENTER THE LABEL FOR THE HORIZONTAL AXIS    Enter title and
# AGE IN YEARS                               labels
  ENTER THE LABEL FOR THE VERTICAL AXIS
# LENGTH IN MM

  CHOOSE Y PREDICTED, Y OBSERVED,
  OR EXIT TO MAIN MENU
  P,O,<E>?
#O                                          ! Plot observed lengths
  CHOOSE PLOT CHARACTER <*>                 ! first
#<CR>                                       ! Data points marked
  CHOOSE PLOT TYPE (<1>-6)                  ! with a "*"
#1                                          ! Scatter plot
  ENTER STOP CHARACTER (2-9,A-Z,a-<z>)
#<CR>                                       ! Default of z for
                                            ! stop character
                                            ! Resultinn plot:

         OBS. AND PRED. LENGTH AT AGE FOR FRESHWATER MUSSELS
           |
  L        |
  E        |
  N      80+
  G        +
  T        |
  H      60+                                      *
           +                          * *  * *  *
  I      40+                   * *  *
  N        |              * *
           +         * *
  M      20+      *
  M        |    *
           | *
            -+--+-+--+-+--+-+--+-+--+-+--+-+--+-+--+-+--+-+--+-+
                       5            10         15

                              AGE IN YEARS
                                            !
  PRESS <CR> TO CONTINUE                    ! Pause to inspect
#<CR>                                       ! graph
  CHOOSE Y PREDICTED, Y OBSERVED,           !
  OR EXIT TO MAIN MENU                      !
  P,O,<E>?
#P                                           This time plot
 PLOT PREDICTED POINTS AT DATA POINTS OR     predicted points
 DISTRIBUTED THROUGH RANGE? (P,<D>)
#<CR>
  HOW MANY POINTS ON THE PREDICTED CURVE?    Default gives many
 < 50>                                       points
#<CR>                                        Accept default
```

```
  CHOOSE PLOT CHARACTER <*>
#P                                              Plotting character is
  CHOOSE PLOT TYPE (<1>-6)                      a "P"
#<CR>                                           Scatter plot
  ENTER STOP CHARACTER (2-9,A-Z,a-<z>)
#<CR>
                                               ! New plot looks like:

         OBS. AND PRED. LENGTH AT AGE FOR FRESHWATER MUSSELS
             |
 L           +
 E           |
 N        80+
 G           +
 T           |
 H        60+                                  *  PPPPPPPP
             +                           P*P*PP*P*PP*PPPP
 I        40+                      PP*P*PP*
 N           |                PP*P*
             +            P*P*
 M        20+          P*P
 M           |       P*
             +   *P
              P+--+-+--+-+--+-+--+-+--+-+--+-+--+-+--+-+--+-+
                          5          10          15

                              AGE IN YEARS

                                                Fit appears to be good
  PRESS <CR> TO CONTINUE                       !
#<CR>                                          !
  DO YOU WANT A NEW PREDICTION?                 Allows us to get new
  Y,<N>?                                        predicted curve
#N                                              We don't want one
  CHOOSE Y PREDICTED, Y OBSERVED,
  OR EXIT TO MAIN MENU
  P,O,<E>?                                     !
#<CR>                                          ! Exit to plotter menu
                                               !
                PLOTTER MENU                   !
                ------------                    Main menu for plotter

                as above....

  ENTER CODE:
  #4                                           ! Create plot: this time
                                               ! try residuals
          PLOT TYPES                           !
          ----------                           !
 CHOOSE ONE OF THE FOLLOWING OPTIONS:

        1) Y VS. X
        2) Y VS. Y
        3) X VS. X                          !
        4) RESIDUALS VS. Y
        5) RESIDUALS VS. X
        6) <EXIT>
 ENTER CODE:
#4                                           Call up residuals vs
                                             Y

 CHOOSE Y OBSERVED OR <Y PREDICTED>.
 O, <P>?
#<CR>                                        Plot against Y
                                           . predicted
 SET RANGES FOR Y DATA:                    !
 COMPUTED MIN=   7.4308090                 !
 COMPUTED MAX=   53.057774                 !
 DO YOU WISH TO CHANGE THESE VALUES?       !
 Y,<N>?                                    !
#Y                                         ! Reset range
 PLEASE ENTER MINIMUM                      !
#0                                         !
 PLEASE ENTER MAXIMUM                      !
#60                                        !
 DO YOU WISH TO NORMALIZE THE RESIDUALS?   !
  Y, <N>?                                  !
#Y                                           Look at normalized
                                             residuals
 SET RANGES FOR RESIDUALS :
 COMPUTED MIN=  -1.7272640
 COMPUTED MAX=   2.2032764
 DO YOU WISH TO CHANGE THESE VALUES?
 Y,<N>?
#Y
 PLEASE ENTER MINIMUM
#-2
 PLEASE ENTER MAXIMUM
#2
 PLEASE ENTER THE PLOT TITLE               !

# RESIDUALS FOR LENGTH AT AGE FITTED TO VONB

 ENTER THE LABEL FOR THE HORIZONTAL AXIS   !
# PRED LEN. (MM)                           !
 ENTER THE LABEL FOR THE VERTICAL AXIS     !
# RESIDUALS                                !
 CHOOSE PLOT CHARACTER <*>                 !
#<CR>                                      ! Plot using default "*"
 CHOOSE PLOT TYPE (SCATTER-O OR <COUNT-1>) !
 O,1?                                      !
#<CR>                                      !
 ENTER STOP CHARACTER (2-9,A-Z,a-<z>)      !
#<CR>                                      !
```

```
             RESIDUALS FOR LENGTH AT AGE FITTED TO VONB
       2.0+
          |
R         +                    *
E      1.0+
S         |
I         +
D         |
U       .0+
A         |     *
L         +
S     -1.0+
          |
          +           *
          |                                      *
      -2.0+---+---+---+---+---+----+---+---+---+---+---+---+
                 10      20      30      40      50

                        PRED LEN. (MM)


 PRESS <CR> TO CONTINUE                       Pause
#<CR>

                PLOTTER MENU
                ------------

                as above....

 ENTER CODE:
#<CR>                                         Exit plotter

!
! NOTE: The procedure for creating high resolution plots is
! similar to the above method, except that the user specifies
! different codes for plot types and characters, and must of
! course have access to a high resolution plotting terminal.
! See Section 5 for details.
.
                                             !
                 SIMPLEX MENU
                 ------------                 SIMPLEX main menu

                 as above....

 ENTER CODE:
#10                                           Look at parameter
                                              covariances
 COVARIANCE CALCULATOR.
```

```
Three "grids" are used to calculate the  ! Instructions and
information, the base grid being formed  ! information
using the current point and step.  Three !
scale factors must be supplied, and the  ! 3 grids are used
initial steps will be multiplied by these ! because the method
scale factors to adjust the size of the  ! used to calc the
grid. Please edit the point and step first! covariances is
(if desired).                            ! affected by scale. If
                                         ! two grids agree we
  3 VARIABLE(S)                          ! assume that answer is
                                         ! correct
 IP) INITIAL POINT:
 1) 57.29114  2) 0.164415  3) 0.155064     Edit point and step
 IS) INITIAL STEP:                         used to build base
 1) 10.00000  2) 0.100000  3) 0.300000     grid

 ENTER CODE TO EDIT PARTICULAR FIELD,
 "AL" TO EDIT ALL FIELDS, "H" FOR HELP,
 <CR> TO SEE CURRENT VALUES, OR "Q" TO
 EXIT EDITOR
#Q                                         We don't want to
                                           change these values
 POINT:
 1) 57.29114  2) 0.164415  3) 0.155064

 INITIAL STEP:
 1) 10.00000  2) 0.100000  3) 0.300000

 GRID FACTORS:                             Default grid factors
 1) 0.100000  2) 0.99E-02  3) 0.10E-02

 INPUT NEW VALUE OF FACTOR #  1 (> 0)
 PRESS RETURN TO ACCEPT DEFAULT
#<CR>                                      Accept grid size
 INPUT NEW VALUE OF FACTOR #  2 (> 0)      defaults
 PRESS RETURN TO ACCEPT DEFAULT
#<CR>
 INPUT NEW VALUE OF FACTOR #  3 (> 0)
 PRESS RETURN TO ACCEPT DEFAULT
#<CR>

 New values for factors:
  1) 0.100000  2) 0.99E-02  3) 0.10E-02

 Factors ok?
 <Y>/N
#<CR>                                      Factors are okay

 Please choose correction constant        !
 (inverse of Hessian will be scaled by    !
 this number to obtain Covariance matrix).!
```

```
CHOOSE:
1)  Constant = 1  ;  if objective
    function is log likelihood
2)  User chosen constant
3)  Constant = 0.5 * SIGMA ** (-2);
    Use if minimizing on sum of squares.
4)  Constant = user chosen constant *
    0.5 * SIGMA ** (-2).
ENTER 1,2,<3>,4:
#<CR>                                        This is sum of
COVARIANCE CONSTANT =  1.63320013            squares problem, take
OBTAINED FROM MINIMUM OF 3.97991640          default
AND ESTIMATED STAN. DEV. OF 0.91844225

Calculating F on grid # 1                    Crunching
AVERAGE F:   18.137886
STD DEV F:   14.820796
Now calculating Hessian #  1
Inverting Hessian #  1
DETERMINANT:  0.1045E+09

Calculating F on grid # 2
AVERAGE F:   4.1211892
STD DEV F:  0.14794104                   !
Now calculating Hessian #  2
Inverting Hessian #  2
DETERMINANT:  0.9460E+08

Calculating F on grid # 3                !
AVERAGE F:   3.9813291                   !
STD DEV F:  0.16608362E-02               !
Now calculating Hessian #  3             !
Inverting Hessian #  3                   !
DETERMINANT:  0.9450E+08                 !
                                         !
                                         !
Calculations completed.  Enter choice:   !
D - display results on screen,           ! Displayed below. The
F - print results to file,               ! results from grids 2
R - run user final routine, or           ! and 3 are fairly
Q - quit and return to main menu.        ! close, so we accept
Choice:                                  ! these as the correct
#D                                       ! values. We could re-do
                                         ! with a grid factor of
                                         ! .01 to check values


OUTPUT FROM COVARIANCE MATRIX FROM SIMPLEX MINIMIZER

POINT
 1)  57.291145      2) 0.16441514       3) 0.15506405
STEP FOR DIFFERENCING
 1)  10.000000      2) 0.10000000       3) 0.30000001
```

```
COVARIANCE CONSTANT  1.000000000     * 0.5 * SIGMA**-2

GRID #           1                 2
SCALE FACTOR:   .100              .100E-01          .100E-02

                                 FUNCTION VALUES
AVERAGE F:        18.13788632       4.121189160       3.981329099
STD DEV F:        14.82079593      0.1479410426      0.1660836190E-02
COF VAR F:         0.81711814       0.35897659E-01    0.41715622E-03

PAR#                           STANDARD DEVIATIONS
 1               0.6236633402      0.6521245397      0.6524315524
 2               0.5519759413E-02 0.5801261418E-02 0.5804290173E-02
 3               0.6757293868E-01 0.6949696734E-01 0.6951788838E-01

PAR#                         COEFFICIENTS OF VARIATION
 1                 0.10885859E-01    0.11382641E-01    0.11388000E-01
 2                 0.33572087E-01    0.35284228E-01    0.35302650E-01
 3                 0.43577437        0.44818233        0.44831725

PAR#, PAR#                        CORRELATIONS
 1, 2              -0.92494678       -0.93158925       -0.93165596
 1, 3              -0.56895712       -0.59740172       -0.59769399
 2, 3               0.76523097        0.77986622        0.78001727

PAR#, PAR#                        COVARIANCES
 1, 1            0.3889559619      0.4252664153      0.4256669306
 1, 2            -.3184103008E-02 -.3524337152E-02 -.3528089866E-02
 1, 3            -.2397742590E-01 -.2707465094E-01 -.2710880770E-01
 2, 2            0.3046774398E-04 0.3365463404E-04 0.3368978441E-04
 2, 3            0.2854207191E-03 0.3144187211E-03 0.3147385262E-03
 3, 3            0.4566102042E-02 0.4829828469E-02 0.4832736805E-02

                                  DETERMINANTS
COV MATRIX:      0.9569502553E-08 0.1057089487E-07 0.1058193995E-07
COR MATRIX:        0.17684917        0.15292345        0.15268768

Calculations completed.  Enter choice:
D - display results on screen,
F - print results to file,
R - run user final routine, or
Q - quit and return to main menu.
Choice:
#Q                                             Finished, return to
                                               SIMPLEX
                  SIMPLEX MENU
                  ------------                 SIMPLEX main menu

                  as above.
 ENTER CODE:
#9                                         ! Create profile plots

 PROFILER                                  !
 --------                                  !
  Please choose profile parameter:
#1                                           Profile along YINF
 Input point   57.291145                     Minimum we found with
 gives function value:  3.9799164            minimizer
 Enter low value of search range:
#50                                        ! Vary YINF from 50 to
 Enter high value of search range.         ! 70
#70

 Input minimum is at:    57.291145           Check bounds
 Low end of range is:    50.000000
 High end of range is:   70.000000
                                           !
 Range of profile parameter okay? (<Y>,N)  !
#<CR>                                      ! Okay
 How many points from low to minimum?      !
#10                                        ! Number of points on
 How many points from minimum to high?     ! high and low profiles
#10                                        !

 Initial point is:   57.291145             ! Minimum point from
                                           ! SIMPLEX
 Minimizing on point:  56.562031           ! 2nd point of profile
 Please edit alg data: do not edit         ! edit alg data before
 profile parameter, STEP 1 will be set     ! minimizing on it
 to zero.

 Press <CR> to continue.
#<CR>
                                           !
  3 VARIABLE(S)                            ! SIMPLEX editor
                                           !
 IP) INITIAL POINT:                        !
 1) 56.56203  2) 0.164415  3) 0.155064
 IS) INITIAL STEP:
 1) 0.00E+00  2) 0.100000  3) 0.300000       Note step for profile
 SL) REQUIRED SIMPLEX LIMIT =                parameter is set to 0
     0.10000000E-02 ABS.                   !
 MF) MAX. OF  400 FUNCTION CALLS           !
 TF) TERMINAL DISPLAY FREQUENCY:     10    !
 FF) FILE WRITE FREQUENCY:            0    !
 SR) STEP REDUCTION FRACTION:  0.10000000  !
                                           !
 ENTER CODE TO EDIT PARTICULAR FIELD,
 "AL" TO EDIT ALL FIELDS, "H" FOR HELP,
 <CR> TO SEE CURRENT VALUES,
 OR "Q" TO EXIT EDITOR                     !
#SL                                          Raise SIMPLEX limit
                                             to speed up search
 <CR> TO ACCEPT CURRENT OR DEFAULT VALUE.  !
 CURRENT SIMPLE LIMIT TYPE IS ABSOLUTE .
 NEW TYPE? (REL or ABS):
#<CR>
 CURRENT SIMPLEX LIMIT:  0.10000000E-02
 NEW VALUE?
#.0001
 ENTER EDIT CODE:
#TF
 CURRENT TERMINAL OUTPUT FREQUENCY:   10
 NEW VALUE?
#0                                            Suppress output

 ENTER EDIT CODE:                            !
#Q                                           ! Begin calculating the
                                             ! profile, starting at
 *** Calculating point #  2 **************! 2nd point
                                             !
   68 CALLS,   31 ITERS,  0 RESTARTS.       !
 MINIMUM OF    4.3987860    FOUND AT:       !
 56.562031    0.17065154    0.20033245      !

 *** Calculating point #  3 **************!
                                             ! Note here: YINF
   44 CALLS,   18 ITERS,  0 RESTARTS.       ! increases in even
 MINIMUM OF    5.7301709    FOUND AT:       ! steps, Minimum
 55.832916    0.17727307    0.24586886      ! increases and K and
                                             ! TZERO also respond
                                             !
                                             !
                                             !


                                             !
 *** Calculating point # 10 **************!
                                             !
   48 CALLS,   20 ITERS,  0 RESTARTS.       ! We asked for only 10
 MINIMUM OF    58.430474    FOUND AT:       ! points...
 50.729115    0.23545740    0.51588916      !

 *** Calculating point # 11 **************!
                                             !
   45 CALLS,   19 ITERS,  0 RESTARTS.       ! but profiler always
 MINIMUM OF    75.553139    FOUND AT:       ! calculates odd number
 50.000000    0.24556838    0.54614626      ! of points. Note values
                                             ! of YINF and minimum
 *** Calculating point #  2 **************!
                                             !
   68 CALLS,   34 ITERS,  0 RESTARTS.       ! 2nd half of profile
 MINIMUM OF    4.9965620    FOUND AT:       !
 58.562031    0.15438531    0.71809889E-01!
```

```
*** Calculating point #  3 ***************!
                                          !



*** Calculating point # 11 ***************!
                                          !
  55 CALLS,   26 ITERS,  0 RESTARTS.      !
MINIMUM OF   50.378141     FOUND AT:      ! Final point
70.000000    0.973781E-01  -0.67002972    !

CREATE PLOT FROM RESULTS?   <Y>,N)
#Y                                          Plot profiles

                   PLOT MENU
                   ---------

  1) RESET RESOLUTION   CURRENTLY=LOW
  2) RESET LOW RESOLUTION SIZE
     CURRENT HEIGHT= 22 , WIDTH=  80
  3) <CREATE PLOT>
  4) DISPLAY PLOT
  5) SAVE PLOT
   ) SAVE PROFILE VALUES                    Option 6 writes param
   ) EXIT                                   values and func
                                            values to file
  ENTER CODE:
 #1                                         Select high res plot

                   PLOT MENU
                   ---------

                   as above....

  ENTER CODE:
#3                                          Create plot

 PROFILE PARAMETER IS NUMBER  1.
 ENTER PARAMETER NO. FOR HORIZONTAL AXIS.
 < 1>
#<CR>                                       YINF on horizontal
                                            axis
 ENTER PARAMETER OR RESPONSE NO. FOR
 VERTICAL AXIS, RESPONSE IS < 4>
#<CR>                                       Func values on
                                            vertical axis
 SET RANGES FOR PARAMETER DATA:
 COMPUTED MIN=   50.000000
 COMPUTED MAX=   70.000000
 DO YOU WISH TO CHANGE THESE VALUES?
 Y,<N>?
#<CR>                                       YINF axis range is
                                            okay
 SET RANGES FOR FUNCTION VALUES:
 COMPUTED MIN=   3.9799163
 COMPUTED MAX=   75.553139
 DO YOU WISH TO CHANGE THESE VALUES?
 Y,<N>?
#Y                                          Reset function axis
  PLEASE ENTER MINIMUM                      range
#0
  PLEASE ENTER MAXIMUM
#100
  PLEASE ENTER THE PLOT TITLE
#PROFILE ON YINF
  PLEASE ENTER HORIZONTAL AXIS LABEL
#YINF
  PLEASE ENTER VERTICAL AXIS LABEL
#SUM OF SQUARES
  CHOOSE PLOT TYPE (<0>-6)
#1                                         ! Line graph with
  CHOOSE LINE TYPE (0-9 <0>-solid line)    !
#<CR>                                      ! solid line...
  CHOOSE A <SMOOTH-0> OR JAGGED-1 LINE
  0,1?
#<CR>                                      ! smoothed
                                           ! See results in Fig 1.3
                  PLOT MENU                ! of introduction
                  ---------

                  as above

  ENTER CODE:
#3                                         ! Now look at relation
                                           ! between YINF and K
 PROFILE PARAMETER IS NUMBER  1.
 ENTER PARAMETER NO. FOR HORIZONTAL
 AXIS. < 1>
#<CR>                                       YINF on horizontal

 ENTER PARAMETER OR RESPONSE NO. FOR
 VERTICAL AXIS, RESPONSE IS < 4>
#2                                         ! Parameter 2 (K)
                                           ! plotted on vertical
 SET RANGES FOR PARAMETER DATA:            ! axis
 COMPUTED MIN=   50.000000
 COMPUTED MAX=   70.000000
 DO YOU WISH TO CHANGE THESE VALUES?
 Y,<N>?
#<CR>                                      ! YINF ranges okay
                                           !
 SET RANGES FOR FUNCTION VALUES:
 COMPUTED MIN= 0.97365782E-01
 COMPUTED MAX= 0.24556838
 DO YOU WISH TO CHANGE THESE VALUES?
 Y,<N>?
#Y                                          Reset for nicer
  PLEASE ENTER MINIMUM                      axis
```

```
#0
  PLEASE ENTER MAXIMUM
#.4
  PLEASE ENTER THE PLOT TITLE
# CORRELATION OF YINF AND K
  PLEASE ENTER HORIZONTAL AXIS LABEL       !
# YINF
  PLEASE ENTER VERTICAL AXIS LABEL
#K                                         !
  CHOOSE PLOT TYPE (<0>-6)
#1                                          Line graph with
  CHOOSE LINE TYPE (0-9 <0>-solid line)
#0                                          solid line...
  CHOOSE A <SMOOTH-0> OR JAGGED-1 LINE
  0,1?                                      smoothed
#<CR>                                       See similar plot in
                                            introduction, Fig. 1.6
                 PLOT MENU
                 ---------

                 as above

  ENTER CODE:
#7                                          Exit plotter
                  SIMPLEX MENU
                  ------------

                  as above

 ENTER CODE:
#9                                          Profiler again, this
                                            time do a slice
 PROFILER                                  !
 --------                                  ! Input is same as for
                                           ! profile above, until
                                           ! edit phase




 ENTER CODE TO EDIT PARTICULAR FIELD,
 "AL" TO EDIT ALL FIELDS, "H" FOR HELP,
 <CR> TO SEE CURRENT VALUES,               !
 OR "Q" TO EXIT EDITOR                     !
#IS                                         Set non-profile
 <CR> TO ACCEPT CURRENT OR DEFAULT VALUE.   parameter steps to 0

 CURRENT VALUE OF STEP #  1 :  0.00000000
 NEW VALUE?
#<CR>                                       Keep profile parameter
 CURRENT VALUE OF STEP #  2 :  0.10000000   as is
 NEW VALUE?
```

```
#0                                              ! Now K is constant
 CURRENT VALUE OF STEP #  3 :  0.30000001 !
 NEW VALUE?                                     !
#0                                              ! And TZERO is constant
                                                !
 ENTER EDIT CODE:                               !
#TF                                             !
 <CR> TO ACCEPT CURRENT OR DEFAULT VALUE.       !
                                                !
 CURRENT TERMINAL OUTPUT FREQUENCY:   10        !
 NEW VALUE?                                     !
#0                                              ! Suppress output
 ENTER EDIT CODE:                               !
#Q                                              !
                                                !
 *** Calculating point #  2 ***************!
                                                
   15 CALLS,     1 ITERS,  0 RESTARTS.
 MINIMUM OF    8.2943907     FOUND AT:          !
 56.562031     0.16441514    0.15506405         !
                                                !
 *** Calculating point #  3 ***************!
                                                !
                       .                        !
                       .                        !
                       .                        !


 *** Calculating point # 11 ***************!

   15 CALLS,     1 ITERS,  0 RESTARTS.            K and TZERO have not
 MINIMUM OF    1301.0090     FOUND AT:            changed, minimum is
 70.000000     0.16441514    0.15506405           very high

 CREATE PLOT FROM RESULTS?  (<Y>,N)             !
#<CR>                                           ! Create plot of slice
                                                !
                 PLOT MENU                      !
                 ---------                      !
                                                !
                 as above....                   !

  ENTER CODE:                                   !
#1                                                Select high resolution
                 PLOT MENU
                 ---------

                 as above....

  ENTER CODE:
#<CR>                                           ! Create plot
```

```
 PROFILE PARAMETER IS NUMBER  1.
 ENTER PARAMETER NO. FOR HORIZONTAL
 AXIS. < 1>
#1                                         ! YINF on horizontal
                                           ! axis
 ENTER PARAMETER OR RESPONSE NO. FOR
 VERTICAL AXIS, RESPONSE IS < 4>
#<CR>                                        Function values
                                             (response) on
 SET RANGES FOR PARAMETER DATA:              vertical axis
 COMPUTED MIN=   50.000000
 COMPUTED MAX=   70.000000
 DO YOU WISH TO CHANGE THESE VALUES?
 Y,<N>?
#<CR>                                      !
                                           !
 SET RANGES FOR FUNCTION VALUES:           !
 COMPUTED MIN=   3.9799163                 !
 COMPUTED MAX=   1301.0090
 DO YOU WISH TO CHANGE THESE VALUES?
 Y,<N>?
#Y
  PLEASE ENTER MINIMUM
#0                                           The higher slice
  PLEASE ENTER MAXIMUM                     ! values are not too
#100                                       ! interesting; stop at
  PLEASE ENTER THE PLOT TITLE              ! SOS=100
# SLICE ON YINF. K and TZERO fixed at opt.!
  ENTER THE LABEL FOR THE HORIZONTAL AXIS !
# YINF                                     !
  ENTER THE LABEL FOR THE VERTICAL AXIS    !
# SUM OF SQUARES                           !
  CHOOSE PLOT TYPE (<0>-6)                 !
#1                                         ! Line graph with
  CHOOSE LINE TYPE (0-9 <0>-solid line)    !
#4                                         ! dashed line,...
  CHOOSE A <SMOOTH-0> OR JAGGED-1 LINE     !
  0,1?                                     ! smoothed.
#<CR>                                      ! See results in
                                           ! introduction, Fig. 1.3
                    PLOT MENU              !
                    ---------              !
                                           !
                    as above....           !
                                           !
  ENTER CODE:                              !
#7                                         ! Return to SIMPLEX main
                                           !
                     SIMPLEX MENU
                     ------------          !
                                           !
                     as above....
 ENTER CODE:                                     !
#12                                              ! Exit
FORTRAN STOP                                     ! Program completed
$                                                ! Back to DCL
```

## ACKNOWLEDGEMENTS

We would like to give special credit to Skip McKinnell, Head of the Computer Centre at the Pacific Biological Station, for his active work on the SIMPLEX package from its earliest beginnings. He provided much of the incentive, technical support, and advice needed to produce the final product. Most of the plotting software was written by Donna Sweeney, while the interrupt capabilities were provided by Marc Hamer. We appreciate the programming advice received both from Marc and from Marilyn Marshall thoughout the project. We extend special thanks to Craig Clarke, John Shelbourn, Jake Schweigert, John Jensen, and Laura Richards, who helped test SIMPLEX and made numerous suggestions for improvement. Finally, we are grateful to Jim Radziul and Kim Hyatt, who graciously provided some useful sample data. Jim also gave us the benefit of his critical comments on both SIMPLEX and this report.

## REFERENCES

Evans, J.W. and R.J. Craig. 1978. Function maximization using a modified Nelder-Mead simplex search procedure. Dept. of Statistics, University of Kentucky, Technical Report 130: 22p.

Kalbfleisch, J. G. Probability and statistical inference (lecture notes for Mathematics 233). Dept. of Statistics, University of Waterloo, Waterloo, Ontario. 250p (approx).

Kendall, M. and A. Stuart. 1979. The advanced theory of statistics. Vol. 2. Inference and relationship. 4th Edition. MacMillan Publishing Co. (New York) 748 p.

Nash, J.C. 1979. Compact numerical methods for computers: linear algebra and function minimization. Halsted Press, John Wiley and Sons (New York): 227p.

Nelder, J.A., and R. Mead. 1965. A simplex method for function minimization. Computer J. 7: 308-313.

O'Neill. R. 1971. Algorithm AS 47, function minimization using a simplex procedure. Appl. Stat. 20: 338-345.

Ricker, W. E. 1975. Computation and interpretation of biological statistics of fish populations. Fish. Res. Board Can. Bull. 191: 382 p.

Schnute, J., and D. Fournier. 1980. A new approach to length-frequency analysis: growth stucture. Can. J. Fish. Aquat. Sci. 37:1337-1351.

Schnute, J. 1982. A manual for easy nonlinear parameter estimation in fishery research with interactive microcomputer programs. Can. Tech. Rep. Fish. Aquat. Sci. 1140: xvi + 115p.

Schnute, J. 1983. A new approach to estimating populations by the removal method. Can. J. Fish. Aquat. Sci. 40: 2153-2169.

Schnute, J., and S. McKinnell. 1984. A biologically meaningful approach to response surface analysis. Can. J. Fish. Aquat. Sci. 41:936-953.

Walmsley, D.A. 1981. The simplex method for minimization of a general function. Transport and Road Research Laboratory, Dept. of Environment and Dept. of Transport. Crowthorne, Berkshire, England. TRRL Supplementary Report 686: 9p.

## APPENDIX A. GLOSSARY OF TERMS

ALGORITHM DATA ... controls the operation of the minimization procedure.

AUXILARY PARAMETERS ... used in user function (TEMPLATE in particular) to perform special tasks.

AXIAL SEARCH ... performed when the simplex search algorithm finds a minimum, to check for premature convergence.

CENTROID ... average value of a number of points. In SIMPLEX, refers to the average parameter values of the N lowest points of the simplex.

COMMON.FOR ... file containing subroutines common to both MINSUM and MINFUN; includes editor, minimizer, covariance calculator, wide search, etc.

COMMO2.FOR ... similar to COMMOM.FOR, except minimizer uses Evans and Craig (1978) modified simplex search algorithm.

CONSTRAINTS ... used in UFUN and TEMPLATE to restrict parameters to desired values.

CONTRACTION ... Simplex high point moves in towards centroid of lower points.

CONTRACTION COEFFICIENT ... dictates how far high point moves toward the centroid; here set at 0.5.

CONVERGENCE ... occurs when function value at all points of simplex are within some limit defined by the user.

COVARIANCE MATRIX ... output frum covariance routine; the large sample approximation to the covariance matrix of the parameter estimates.

EDITOR ... used to "edit" the algorithm data.

ESTIMATES ... of the parameters which minimize the object function; what the simplex search algorithm finds.

EXTENSION ... Simplex high point moves across centroid to point twice as far from centroid as reflected point.

EXTENSION COEFFICIENT ... determines size of extension; here set at 2.0.

GLOBAL SEARCH ... Given a set of parameter bounds, GLOBAL SEARCH performs a random or grid search for a minimum.

FIXED PARAMETERS ... variation of minimization, where some parameters are held constant as the rest are minimized.

HYPERSURFACE ... a multidimensional "surface".

INTERRUPT ... on some systems, the capability for the user to interrupt the minimization procedure to make changes. On the Vax, interrupt is achieved with C.

LIKELIHOOD ... the function that describes the probability of the observed data, given values of the parameters. Maximum likehood parameter estimates are those which maximize this function (or minimize its negative logarithm).

MINFUN ... central program which minimizes user's UFUN.

MINSUM ... central program which minimizes the sum of TERMs in the user's TEMPLATE.

MULTIPLE MINIMA ... condition of a function having several local minimum points. This can be indicatve of a poor model, noisy data, or too little data.

NON-LINEAR ESTIMATION ... parameter estimation which cannot be dealt with by ordinary linear regression.

OBJECT FUNCTION ... the function for which we are trying to obtain a minimum.

PARAMETERS ... constants in a model, typically unknown (except for auxiliary parameters, which are typically known or set by the user); dependent variables of the object function. Parameters are estimated by minimizing the object function.

PENALTY FUNCTIONS ... a special part of the user routine which assigns a high value to the object function if a parameter takes on an undesirable value.

POINT ... in this manual, usually a point in N-dimensional space defined by N parameter values. "High" and "low" points correspond to high and low values of the object function, respectively.

PROFILE PLOT ... a plot of function values as one parameter is varied and other parameters are simultaneously chosen optimally for that parameter.

REDUCTION ... The simplex search algorithm can find no better points between the centroid and the high point, so all points are moved toward the minimum point.

REFLECTION ... The simplex search algorithm finds a new point opposite from the highest point across the centroid of the remaining low points.

REFLECTION COEFFICIENT ... controls how far simplex search algorithm steps away from the high point when reflecting; here set at 1.0.

RESTARTS ... If the axial search finds a better minimum than did the simplex search, the simplex procedure is restarted from the new point. More than a few restarts indicates tuning problems with the algorithm data.

SIMPLEX ... several contexts:

1) The minimization method, as outlined by Nelder and Mead (1971).

2) The actual geometric structure which is manipulated along the function surface to the minimum.
3) This entire package, including software and documentation.

SIMPLEX LIMIT ... the value defining the maximum allowable difference between high and low function values for convergence.

STEP ... part of the algorithm data. STEP defines the initial size of the simplex.

STEP SIZE REDUCTON FRACTION ... Multiplying STEP by the SSRF gives axial distances for the axial search.

TEMPLATE ... the file into which users can insert the code needed for their application.

UFUN ... the user routine written by the user giving the object function.

## APPENDIX B. SIMPLEX ITERATION

This appendix gives an algorithmic description of one iteration of the simplex search.

INPUT/OUTPUT VARIABLES:

| | |
|---|---|
| SIMPLEX (N,N+1): | Contains the entire simplex, that is, N+1 points defined by N parameters. Updated during iteration. |
| VALUES (N+1): | Contains function values at each simplex point. Updated during iteration. |

INTERNAL VARIABLES:

| | |
|---|---|
| CENTROID (N): | Location of the centroid of the N lowest simplex points. |
| L, H: | Indices of the high and low point, respectively, in SIMPLEX and VALUES. |
| CPRIME (N): | Location of the reflected point. |
| CPVAL: | Function value of the reflected point. |
| CPP (N): | Location of the extension point. |
| CPPVAL: | Function value of the extension point. |
| M: | Number of points in SIMPLEX improved by CPRIME. |

PREDEFINED CONSTANTS:

| | |
|---|---|
| ALPHA = 1.0 | reflection coefficient |
| BETA = 0.5 | contraction/reduction coefficient |
| GAMMA = 2.0 | extension coefficient |

SEARCH ALGORITHM:

1) Find H, L, and CENTROID:
   a) Find High and Low of points in VALUES, set H and L.

   b) Compute CENTROID of N lowest points (i.e. SIMPLEX(I,H) is not included):

```
DO I = 1, N
   CENTROID(I) = 0.0
   DO J = 1, N+1
      CENTROID(I) = CENTROID(I) + SIMPLEX(I,J)
   END DO
   CENTROID(I) = (CENTROID(I) - SIMPLEX(I,H)) / N
END DO
```

```
2) Reflection to CPRIME, and get function value CPVAL:
      DO I = 1 TO N
         CPRIME(I) = (1+ALPHA)*CENTROID(I)-ALPHA*SIMPLEX(I,H)
      END DO
      CALL UFUN (CPVAL, CPRIME, N, NF)

3) Find M, the number of points improved by the reflection.
      M = 0
      DO I=1 TO N+1
         IF (CPVAL.LT.VALUE(I)) M = M + 1
      END DO

4) If M = N+1 then new point is excellent.
    a) Find extension.
         DO I = 1 TO N
            CPP(I) = GAMMA*CPRIME(I)+(1-GAMMA)*CENTRIOD(I)
         END DO
         CALL UFUN (CPPVAL, CPP, N, NF)

    b) If extension improves reflected point, accept new point
         SIMPLEX (1...N,H) = CPP,  VALUE(H) = CPPVAL

    c) Else accept reflection.
         SIMPLEX (1...N,H) = CPRIME, VALUE(H) = CPVAL

5) Else if M > 1 then new point is good.  Accept reflection.
         SIMPLEX (1...N,H) = CPRIME, VALUE(H) = CPVAL

6) Else attempt contraction.
    a) If M = 1, point is just OK. Perform reflection before
         contracting.
         SIMPLEX (1...N,H) = CPRIME, VALUE(H) = CPVAL

    b) Find contraction point:
         DO I = 1 TO N
            CPRIME(I) = (1-BETA) *  SIMPLEX(I,H) +
                             BETA  *  CENTROID(I)
         END DO
         CALL UFUN (CPVAL, CPRIME, N, NF)

    c) If contraction point CPVAL is lower than VALUE(H), accept.

         SIMPLEX (1...N,H) = CPRIME, VALUE(H) = CPVAL

    d) Else reduce entire simplex, find all new values.
         DO J = 1 TO N+1
            IF J<>L THEN
               DO I = 1 TO N
                  SIMPLEX(I,J)= BETA       * SIMPLEX(I,J)
                               -(BETA - 1) * SIMPLEX(I,L)
               END DO
               CALL UFUN (VALUE(I),SIMPLEX(1,I),N,NF)
            END IF
         END DO

   End.
```

## APPENDIX C. CHOICES FOR MINSUM PLOTTING

This appendix describes, in algorithmic form, the various choices available for plotting observations, predictions, explanatory variables, and residuals. These options are discussed generally in Section 5.2; they are available to MINSUM only. There are 5 main possibilities: (1) Y vs. X, (2) Y vs. Y, (3) X vs. X, (4) residuals vs. Y, and (5) residuals vs. X.

```
1) Y vs. X
   a) Select X subscript, if more than one X.
   b) Set X and Y ranges.
   c) Input axis labels and title.
   d) Choose OBS, PRED, or quit (return to main, keep plot).
      LOW RES:   I)OBS: i) Choose 1) plot character;
                                  2) plot type (1-6).
                       ii) Create plot, return to d).
                II)PRED: i) Select: predictions at data points or
                                    spread over range.
                           ia) If (spread over range) then:
                               set number of points on curve;
                               set remaining X's.
                        ii) Choose 1) plot character;
                                   2) plot type (1-6).
                       iii) If new prediction wanted, call editor
                               to change parameters; go to i);
                            else go to d).
      HIGH RES:  I)OBS:  i) Choose 1) plot type (0,2-6);
                                   2) plot character.
                            Return to d).
                II)PRED: i) Select predictions at data
                            points or spread over range.
                            ia) If (spread over range) then:
                                set number of points on curve;
                                set remaining X's.
                        ii) Choose 1) plot type (0-6),
                                   2) connecting line, or
                                   3) marker.
                       iii) If new prediction wanted, call editor
                               to change parameters; go to i);
                            else goto d).

2) Y vs. Y
   a) Choose Y PRED or Y OBS for horizontal axis.
   b) Set range of Y's.  Default is lowest of OBS and PRED to
      highest of OBS and PRED.
   c) Input title and axis labels.
   d) Plot Y OBS vs. Y PRED.
      LOW RES: I) Choose plotting character.  Default *.
              II) Choose plot type (count or overprint).
      HIGH RES I) Choose marker.  Default *.
              II) Indicate if 45 degree line needed.
   e) Return to main menu.
```

3) X vs. X
   a) Select horizontal X subscript and range.
   b) Select vertical X subscript and range.
   c) Input title and labels.
   d) Plot X1 vs. X2.
      LOW RES: I) Choose plotting character. Default *.
               II) Choose plot type (count or overprint).
      HIGH RES I) Choose marker. Default *.
   e) Return to main menu.

4) Residuals vs. Y
   a) Select Y PRED or Y OBS.
   b) Set Y range.
   c) Indicate if residuals should be normalized.
   d) Set residual range.
   e) Input title and labels.
   f) Plot residual vs. Y.
      LOW RES: I) Choose plotting character. Default *.
               II) Choose plot type (count or overprint).
      HIGH RES: I) Choose marker. Default *.
   g) Return to menu.

5) Residuals vs. X
   a) Select Xi.
   b) Set X range.
   c) Indicate if residuals should be normalized.
   d) Set residual range.
   e) Input title and labels.
   f) Plot residual vs. X.
      LOW RES: I) Choose plotting character. Default *.
               II) Choose plot type (count or overprint).
      HIGH RES: I) Choose marker. Default *.
   g) Return to menu.

**The following pages contain only excerpts of the document whose title page follows this page**

1140
C2

# A Manual for Easy Nonlinear Parameter Estimation in Fishery Research with Interactive Microcomputer Programs

Jon Schnute

Department of Fisheries and Oceans
Fisheries Research Branch
Pacific Biological Station
Nanaimo, British Columbia V9R 5K6

December 1982

**Canadian Technical Report of Fisheries and Aquatic Sciences No. 1140**

ent of Canada Gouvernement du Canada
and Oceans Pêches et Océans

Canadian Technical Report of Fisheries

and Aquatic Sciences No. 1140

December 1982

# A MANUAL FOR EASY NONLINEAR PARAMETER ESTIMATION IN FISHERY RESEARCH WITH INTERACTIVE MICROCOMPUTER PROGRAMS

by

Jon Schnute

8790 756- 16

Department of Fisheries and Oceans

Fisheries Research Branch

Pacific Biological Station

Nanaimo, British Columbia V9R 5K6

Cat No. Fs 97-6/1140 ISSN 0706-6457

TABLE OF CONTENTS

Page

## ABSTRACT

Schnute, Jon. 1982. A manual for easy nonlinear parameter estimation in fishery research with interactive microcomputer programs. Can. Tech. Rep. Fish. Aquat. Sci. 1140: xvi + 115 p.

This manual describes in detail how to solve many practical problems encountered in nonlinear parameter estimation. In addition, it presents software to aid the user with three tasks: (1) finding optimal parameter estimates, (2) plotting observations and model predictions, and (3) displaying graphically the variation in likelihood (or sum of squares) when the parameters are varied from their optimal estimates. This software is coded in BASIC for the Apple II microcomputer, and it is available on a suitable 5 1/4" diskette. In many cases, the user can adapt the general software here to his or her particular problem by adding just a few lines of BASIC code.

The simplex method of searching for a function minimum lies at the heart of the discussion here. This manual describes the method completely. Although a simplex search is known to be less efficient of compute time than derivative-based methods, it has the considerable advantage of minimizing human time required for coding a particular problem. The manual places great emphasis on ease of program development, even at the expense of computer time. It also shows how to adjust the simplex method for optimal efficiency and how to apply it in other contexts besides nonlinear estimation. The discussion throughout is illustrated with numerous examples from fisheries literature, although the methods have obvious application in many fields.

The broad purpose of this manual is to make the reader as comfortable with the process of nonlinear estimation as with the much simpler standard procedure of linear regression.

Key words: parametric models, parameter estimation, nonlinear estimation, minimization, function minimization, simplex search, microcomputers, Apple II microcomputer, BASIC, likelihood, maximum likelihood, fisheries models.

## RESUME

Schnute, Jon. 1982. A manual for easy nonlinear parameter estimation in fishery research with interactive microcomputer programs. Can. Tech. Rep. Fish. Aquat. Sci. 1140: xvi + 115 p.

Le présent manuel décrit en détail la manière de résoudre plusieurs problèmes pratiques recontrés dans l'estimation non linéaire des paramètres. De plus, il présente le logiciel nécessaire pour aider l'usager dans les trois tâches suivantes: (1) la recherche des estimations optimales de paramètres, (2) le traçage des observations et des prédictions de modèles, et (3) la visualisation graphique de la variation de la possibilité (ou somme des carrés) quand les paramètres diffèrent de leurs estimations optimales. Ce logiciel est codé en BASIC pour le micro-ordinateur Apple II et il est disponible sur disquette appropriée de 5 1/4". Dans plusieurs cas, l'usager peut adapter le logiciel général à ses besoins particuliers en ajoutant quelques lignes en BASIC.

Cette méthode de transmission unidirectionnelle pour la recherche d'un minimum pour la fonction constitue la partie principale de la discussion. Toute la méthode est décrite dans le manuel. Quoique la recherche par transmission unidirectionnelle utilise moins efficacement le temps sur ordinateur que les méthodes à base dérivée, elle a l'avantage considérable de minimiser le temps qu'une personne doit passer à coder un problème. Le manuel vise surtout à faciliter l'élaboration de programmes, même aux dépens du temps sur ordinateur. Il montre aussi comment ajuster la méthode par transmission unidirectionnelle pour obtenir une efficacité maximale et comment l'appliquer dans des contextes autres que les estimations non linéaires. Toute la discussion est illustrée de nombreux exemples tirés d'ouvrages sur les pêches, mais les méthodes peuvent évidemment s'appliquer à d'autres domaines.

Le but général de ce manuel est de familiariser le lecteur avec le procédé d'estimation non linéaire autant qu'avec la procédure classique beaucoup plus simple de régression linéaire.

Mots-clés: modèles paramétriques, estimation de paramètre, estimation non linéaire, minimisation de fonction, recherche par transmission unidirectionnelle, micro-ordinateurs, micro-ordinateur Apple II, BASIC, probabilité, probabilité maximale, modèles de pêche.

## PREFACE

Fisheries literature in recent years includes an increasing number of analyses based on nonlinear parametric models. Some of these papers have been authored or co-authored by myself. Each time I've participated in such a paper, I've been left wondering just how easily readers might be able to put the ideas into practice. For linear models, one can give formulas for the parameter estimates which translate readily to a computer program. Nonlinear parametric models require the user to program an information function, such as the likelihood or a sum of squared model errors, which must then be optimized. If the reader feels comfortable with this process, then it's enough to specify the appropriate information function. If not, then he or she may be left feeling that, no matter how interesting the analysis looks, it could not actually be attempted without an enormous effort.

Current developments in microcomputing technology pose some remarkable possibilities for implementing the required optimization methods. While microcomputers are hardly the perfect environment for a large numerical project like nonlinear estimation, they can perform acceptably on many of the small- to medium-sized problems which commonly occur in fishery data analysis. The ease of program development in BASIC helps compensate for some of the inconveniences, like slow operation. Also, these small systems have highly portable hardware and software. In the Canadian Department of Fisheries and Oceans, Pacific Region, for example, there are now several Apple II microcomputers. This report, together with one diskette, makes it possible for any user with access to an Apple II to take immediate advantage of all the programs described here. I particularly hope that the programs here might prove useful to fishery management in developing countries where large computers simply aren't available.

The intent of this report is to present a tool which makes it almost as easy for the practitioner to estimate parameters in moderate-sized nonlinear models as in linear ones, where ordinary regression gives the answer in a single step. I propose to achieve this by (1) tailoring the tool so that it does as much of the practitioner's work as possible, (2) rendering its operation visible enough so that he or she can see clearly how the tool performs, and (3) providing easy graphics display to assess the outcome. Essentially, the user needs to supply only a program to calculate the information function, such as (but not necessarily) a sum of squares of model errors. Extra conditions, such as parameter constraints or Bayesian priors, can be added easily. The tool readily allows selected parameters to be fixed at specified values.

The underlying methods employed here certainly are not new. General techniques for function minimization have been studied extensively, and a variety good algorithms exist in FORTRAN to do the job effectively on a large computer. This report employs the simplex search method originally proposed by Nelder and Mead (1965). The BASIC algorithm developed here is somewhat more flexible and much more interactive than its commonly referenced FORTRAN counterpart (O'Neill, 1971). The current version is based on my practical experience over about a year and a half. This does not imply that the program is optimal or totally error-free, and I would gratefully welcome any discussion of problems or suggestions for improvement.

I became involved in this project for some rather peculiar reasons. The Pacific Biological Station, where I have worked since 1976, does not (as of this writing in early 1981) have a full-scale interactive computing system. Since interactive program development and use is almost essential for efficient nonlinear estimation, I began my work on the only available resource: a Data General Nova 2 microcomputer which is operated exclusively in BASIC with about 9000 bytes in central memory available to the user for programs and data. The simplex method was the best available for tailoring to these contraints, and the programs here (wihout graphics) were originally developed for the Nova 2. Later, an Apple II microcomputer became available with an effective user space of 36000 bytes. In the context of computing at the Biological Station, this felt like a very large workspace, and the programs were enlarged to their present form, along with graphics routines to exploit the Apple's handy capabilities in that regard.

In some cases, especially when the model involves more than five parameters, the programs here may run quite slowly. Literature on the simplex algorithm commonly states that it performs poorly with more than four parameters, but I have used it successfully with up to 15. It's partly a matter of patience. Sometimes the computer has to be left alone for half-an-hour, or for several hours, or even overnight, while it searches for optimal parameter estimates. In a research context, the ease of program development is often much more important than computer run-time. If a model takes just a few minutes to program, then it can easily be tried, even if it means letting the computer work for a while. On the other hand, if a model needs hours or days of programming, then it may not be tried, even if the computer could do the work in a few minutes.

This report describes programs possible on the Apple II Plus with 48K memory and with the MICRO SIMPLEX system diskette, Version 1.1. Further details on system requirements appear in Appendix A. Appendix B lists the files available on the version 1.1 diskette, together with the sections of this report where each file is discussed. I anticipate future enhanced versions, and possibly even a complete FORTRAN counterpart. Naturally, with these possibilities in mind, I emphasize again that I would welcome all user corrections, suggestions, and proposals for future software design.

Jon Schnute
Pacific Biological Station

April, 1981

## PREFACE TO THE SECOND EDITION

Strictly speaking, this document never appeared in a first edition, except for some preprints given to a few trial users. A paper of mine (Schnute 1981) refers to this manual as "Can. Spec. Publ. Fish. Aquat. Sci. 59 (in press)". Indeed, although the manual was originally written for the Technical Report series, the editor requested that I submit it instead to the Special Publications series. However, when the editor decided to seek a review, the referee did not recommend publication without significant changes and additions. As I explain below, I agree with some of the criticism and wish I had time to produce a manual and software that dealt with all of it. Unfortunately, research on specific problems of fishery management continually takes priority. Meanwhile, almost a year and a half has gone by since the first edition was completed. I have had dozens of requests from several countries (including APPLE owners in remote places) for this paper based only on its reference in Schnute (1981), and I continue to find the programs of considerable use. I believe many practitioners may also find value in the methods here, so I have decided to make them available in the Technical Report series without further delay.

In order to clarify this manual's purpose and limitations, I'd like to address the referee's main comments here. Readers interested in making active use of this material will do well to take a few minutes to follow the discussion below. In each case I give a short paraphrase of the referee's comment (underlined), together with my reply.

1. <u>This manual describes a particular method of function minimization, but that is a very small subset of the problem of model selection and fitting</u>.

I agree. The whole point of this manual is to document a technology for getting parameter estimates easily. I assume that the user has defined the problem properly and simply wants to find the estimates. I'm trying to make this job almost as easy as in the case of linear regression, where a formula gives the answer in one step. Just as models which depend linearly on their parameters can be wrongly applied, so too can models with parameters which enter nonlinearly. The matter of selecting apprpopriate models isn't addressed here, except for some limited discussion in section 7.

2. <u>The system should have been written in a higher level language, like APPLE PASCAL (which can be compiled), rather than BASIC</u>.

In principle, that's true. The most annoying feature of BASIC is that it doesn't support true subroutines. All variables are global so that the user continually has to worry about variable names. That problem is reasonably solved here, but it can't be completely circumvented.

Unfortuntely, APPLE PASCAL suffers from serious flaws which render it awkward and even useless in this context. The fatal flaw is that it supports only single (four-byte) precision, and that just isn't enough for most practical fishery estimation problems. Also, although the operating system is conceptually elegant on the APPLE II, it's slow and awkward to use. Debugging is a serious problem compared with APPLE BASIC. Finally, APPLE PASCAL imposes on the user significant extra costs (at least for the APPLE II) in hardware and software. The material here is primarily intended for the rather large audience of APPLE II users without a particularly enhanced system.

In contrast with PASCAL, APPLE BASIC is implemented with five-byte precision, and that is usually just enough for practical problems. Since the language runs interpretively, it is remarkably easy to debug. Furthermore, there are now several good compilers available on the market, so that programs, after debugging, can be compiled and run at high speed. In fact, the speed of compiled BASIC on the APPLE II is greater than that of so-called "compiled" PASCAL, because PASCAL is actually compiled only to Pseudo-machine code ("P-code"), a slightly higher level language than machine language.

Although the software presented here works well on the APPLE II, it certainly isn't easily transportable. It has been converted to VAX BASIC without too much difficulty, but BASIC varies considerably from system to system. The ideal language for portability is FORTRAN, and plans are underway to put much of the software here into that language. When this is done, it will be made available for the convenience of non-APPLE users.

3. <u>Marquardt's gradient method could have been implemented in exactly the same way as the simplex method, with similar simplicity to the user. Furthermore, the detailed discussion of the simplex method isn't necessary for biologists</u>.

The main convenience of the simplex method is that it is a direct search method which, consequently, doesn't require the user to compute or code derivatives. Gradient-based methods want to know which way is downhill, that is, they depend on a knowledge of the derivatives. When the referee made his comment, he clearly thought of the biologist and the programmer as two different people. The trouble is, in my experience, that the biologist (happily or unhappily) often ends up writing his own programs. I know of instances where biologists have spent days calculating complex derivatives required for gradient-based methods, and this whole labor could have been spared by using the simplex method.

Another nice feature of the simplex method is that it's easy to understand how it works. Gradient-based methods are motivated by some rather complicated analysis, while the simplex method is entirely geometric. It may be that the biologist won't really care to follow the details, but I hope there will be some who are sufficiently curious that they read section 2 and come to terms with Figure 2.1.

The simplex method has the added advantage that it is reasonably robust: sooner or later it usually finds a minimum. Gradient methods sometimes fail for rather mysterious reasons and leave the user helpless in deciding what to do next. This rarely happens with the simplex method, and users who have taken the trouble to understand how it works will almost always be able to get it to converge. Section 4 includes detailed discussion of this problem; see in particular the "tuning summary" on page 30.

The biggest drawback of the simplex method is that it tends to bog down when the number of parameters becomes large. I have run this software with 15 and even 20 parameters successfully, although sometimes slowly. My philosophy is that I'd rather let the computer spend some extra time than devote my own time to coding derivatives. Still there is no question that eventually a gradient-based method becomes essential. As computing costs diminish, however, so does the need to employ a gradient method.

Readers interested in studying other minimization methods will find excellent readable description in Nash (1979), along with some interesting comparisons among methods. If I had known that Nash's book existed when I wrote this manual originally, I would have cast parts of the coding around his algorithms. His variable metric method (Algorithm 21) would be an excellent candidate for a general gradient method on a small computer. Incidentally, fisheries estimation problems frequently cannot be reduced to least squares, so that practical software must be capable of finding minima of arbitrary functions.

4. <u>No provision is made for calculating the variance-covariance matrix of the parameter estimates.</u>

That's true. Again, the main purpose of this manual is to provide software to find the estimates themselves. However, this additional feature is easy to implement, and software to do so will be included in future work. The referee suggests using Newton's divided difference method to approximate the matrix of second partial derivatives (also called the Hessian) of the negative log-likelihood at its minimum. The inverse of this matrix approximates the covariance matrix of parameters. To the reader who isn't familiar with all this terminology, that may sound like an imposing task, but it requires only a few lines of code, once the function itself has already been coded for use with the simplex algorithm.

5. <u>The matrix inversion routine in section 8.3 is not optimal for this context. Pivoting isn't needed, and the Choleski decomposition should be used.</u>

This is an excellent comment; however, it shouldn't detract from the main point of section 8.3 The idea there is that if some of the parameters enter linearly, they can be found by linear regression as part of each function calculation. This reduces the dimensionality of the problem, and potentially leads to a great improvement in speed. Recent work shows that the simplex method operating in this fashion can be much more efficient than a gradient method operating on the full set of parameters. What the referee disputes is my sledge hammer approach to linear regression. First of all, pivoting isn't necessary in the matrix inversion, as suggested in section 8.3, and secondly the Choleski decomposition is known to be the optimal method of inverting matrices of this kind. The methods described in 8.3 will work; they just aren't optimal. The book by Nash (1979) cited above gives a compact method (Algorithm 7) for Choleski decomposition.

6. <u>The system doesn't permit weighted residuals in computing the sum of squares.</u>

That's not strictly true. Since the system permits <u>any</u> function to be minimized, in particular it allows for a weighted sum of squares. However, it is true that the module TEMPLATE for the particular case of sums of squares doesn't include the possibility of variable weights. A sensitive reader will note that the data used in the example of section 1 represent <u>mean</u> lengths of fish at various ages. These means come from samples of different sizes, and there is some justification for weighting the residuals differently at each age. The correct procedure depends on the model, that is, on how one defines the population growth curve. I have chosen not to weight the residuals differently, not because that is the only way, but rather to obtain a simple demonstration problem for the software. Once again, this is a manual on the practical problems of estimation, not on the complete science of model building.

7. <u>There is no provision for residual plots, as required for valid model selection.</u>

It is certainly true that residual plots can be useful in deciding whether or not the model is really appropriate to the data. In fact, strange behavior of the residuals can suggest ways to modify the model. I repeat the point made that this manual is directed at the limited problem of estimation. A fully integrated system of software for model building would be valuable, but this document doesn't attempt such an ambitious project.

## Concluding Remarks

This manual is intended for biologists who are troubled by the problem of nonlinear parameter estimation. They may feel comfortable with linear regression, but helpless when the parameters enter the model nonlinearly. I hope to demonstrate here that there are straightforward methods of estimation which can be understood completely without sophisticated mathematics and used with ease.

Although estimation can be the most technically difficult problem of model building, it certainly isn't always the part that requires the most thought and skill. As the referee points out, this is only one step of the way. It may be used iteratively: estimate the parameters, see how well the model works, modify it, and estimate again. I have included referee's remarks in some detail here to give the reader at least a glimpse of what else might be involved. If the process of estimation itself can be made easy enough, the practitioner is freed to devote his or her creative powers to the deeper problem of proper model design.

To those who have waited for months to receive this manual, I extend my apologies for the delay.

Jon Schnute
Pacific Biological Station
Nanaimo, B.C. V9R 5K6
Canada

November, 1982

## 2. THE SIMPLEX SEARCH METHOD

From the previous section it is clear that a key purpose of the software here is to minimize a function F of several, say N, variables. (In the example above, F is the criterion function S or S*, and N is 3.) There are many known algorithms designed for this purpose, and the most efficient ones require the user to code the derivatives of F, as well as F itself. Other algorithms, called direct search methods, require information on F alone. The simplex method is a direct search method, and the Preface to this report gives some background on my reasons for choosing it. If the practitioner wants to minimize programming time, as opposed to computer run time, then a direct search method is perhaps the best. Users will have to judge from practical operation whether or not this software meets their particular needs.

The simplex method was first conceived by Spendley, Hext, and Himsworth (1962) primarily as a technique for designing experiments to locate points of optimal response for a system being actively measured. Later, Nelder and Mead (1965) pointed out that the method could also be implemented as a computer search. O'Neill (1971) formalized the procedure into a computer FORTRAN algorithm, and the BASIC software here is a variation of O'Neill's algorithm designed for greater flexibility and interactive input/output.

A simplex in N dimensions is polyhedron with N + 1 vertices and N + 1 faces. When N = 2, the simplex is a triangle, which, of course, has 3 vertices and 3 sides. When N = 3, it is a triangular pyramid, that is, a tetrahedron with 4 vertices and 4 triangular faces. The simplex search method involves inspecting the values of the function F at the N + 1 vertices of a simplex. In order to minimize F, the vertex with the highest value of F is rejected in favor of some new point. This new point, together with the remaining N vertices defines a new simplex, and the procedure continues by iteration.

Figure 2.1 illustrates the process when N = 2. Here the triangle ABC represents a two-dimensional simplex on which F is lowest at A and highest at C. Symbolically,

$$F(A) < F(B) < F(C).$$

Since F is highest at C, the idea is to move away from C towards the line AB on which F is lower, at least at the end points. This is done by the process of reflection (Figure 2.1, diagram 1) in which a new point C' is determined as the mirror image of C across the line AB. The point C' is defined so that the lines CC' and AB both have midpoint D.

Hopefully, the value of F is lower at C' than at C. Indeed, one can simply count the number of original simplex points A, B, C which have higher F-values than the new point C'. Call this number M. By definition of M, the

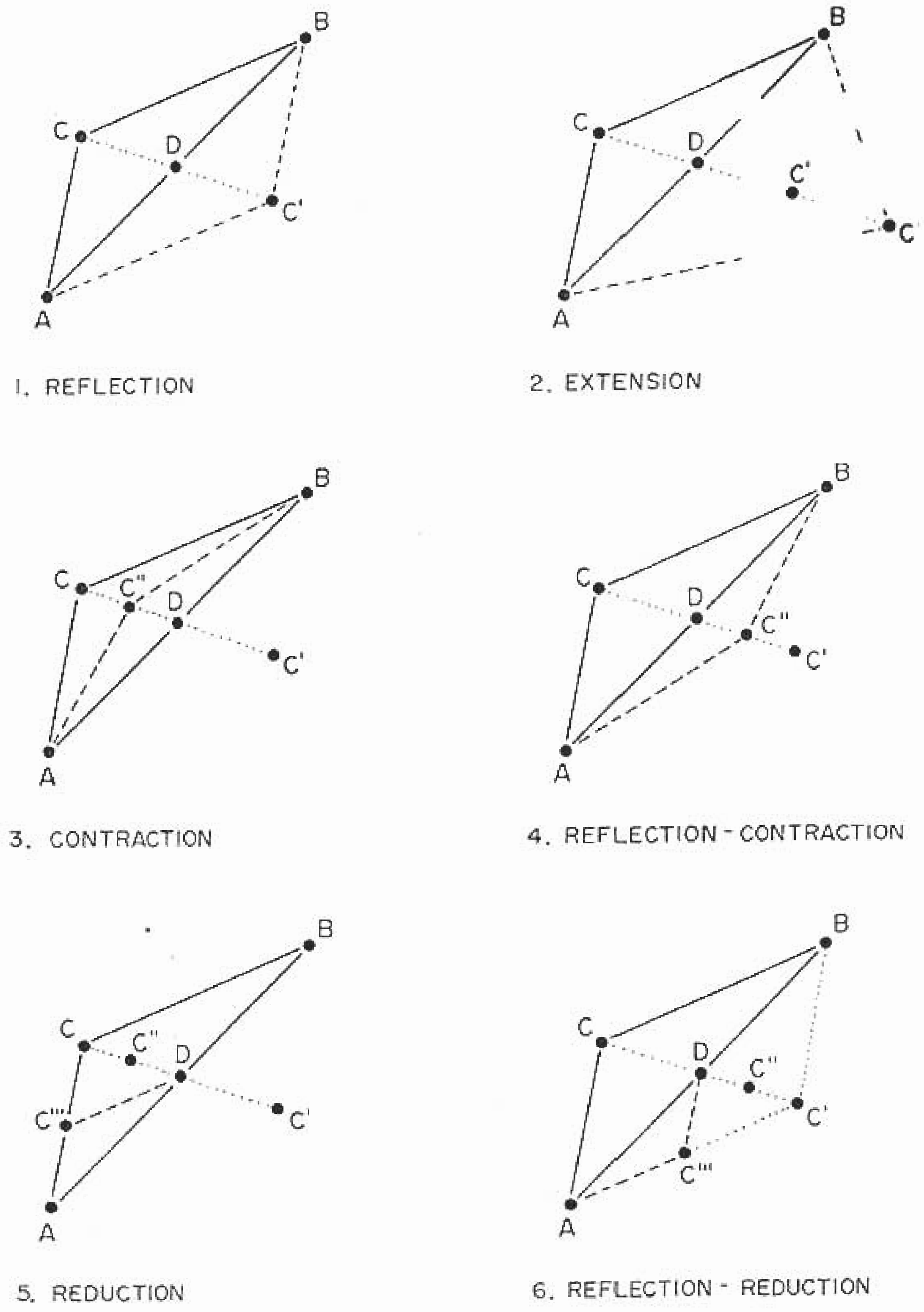


Figure 2.1. Six possible processes for one iteration of the simplex search method.

function value F(C') at the reflected point C' improves, i.e., is lower than, the value of F at M points on the original simplex ABC. There are four possibilities: M is 0, 1, 2, or 3. The next action taken depends on M.

<u>Case 1: M = 3</u>. In this case the reflected point C' improves all points on the original simplex. Thus the point C' gives the lowest value of F found so far; consequently, the step to C' has been very productive. It is therefore reasonable to extend this step by, say, the distance DC'. This gives C'' in diagram 2 of Figure 2.1. There are two possibilites: either F(C'') is lower than F(C') or not. If so, then the new point C'' is accepted to give a new simplex ABC''. This is called <u>extension</u>. Otherwise, the point C' is accepted to give a new simplex ABC', and, as mentioned above, the process is called <u>reflection</u> (diagram 1).

<u>Case 2: M = 0</u>. If case 1 is the best possible, then this is the worst possible. The reflected point C' is actually worse than any vertex on the original simplex ABC. In this case the choice is made to move from C towards the midpoint D on the line AB, but only half-way. This gives C'' in diagram 3. Hopefully F(C'') is lower than F(C), since a small step has been taken from the highest point C towards the lower points A and B. If so, then C'' is accepted to give the new simplex ABC'' (diagram 3), and the process is called <u>contraction</u>. If, by some weird quirk in the topography, C'' doesn't improve C, then drastic action is required. The decision is then made to shrink the whole simplex towards the lowest point A. Each side of the new simplex ADC''' is half the size of its original counterpart, and the process is called <u>reduction</u> (diagram 5).

<u>Case 3: M = 1</u>. In this case the reflected point C' improves the original point C, but no other. It might seem appropriate simply to accept the reflection (as in diagram 1), but think ahead. The new simplex ABC' would have the property that F is highest at C'; consequently the next iteration would just undo this one to give the old point C as the mirror image of C'. This would be wasted effort, so instead contraction to C'' (diagram 4) is attempted at once. If successful, that is, if C'' improves C', then the process is called <u>reflection - contraction</u>. Otherwise the algorithm performs <u>reflection - reduction</u> to obtain the new simplex ADC''' (diagram 6), analogous to the process of reduction in case 2.

<u>Case 4. M = 2</u>. This is the simplest case. The new point C' is good (since it improves B and C) but not very good (since it doesn't improve A). Consequently, the algorithm proceeds with <u>reflection</u> (diagram 1) in which C' is accepted to give the new simplex ABC'. Incidentally, notice that B is now the high point on the simplex, so the next step will take place in the new direction defined by reflecting B across AC'.

This whole description has been set in two dimensions, but it can be generalized at once to arbitrary N. In the original simplex ABC, let A and C again denote the low and high points; however, let B represent the remaining N-1 vertices at which F takes values between F(A) and F(C). The line AB in Fig. 2.1 now represents a face of the N-dimensional simplex, and D can be regarded as the centroid of this face. The reflected point C' is defined by reflection of C across D, exactly as before. Again let M denote the number of points in the original simplex which are improved by C'. Then M can take any of the $N + 1$ values 0, 1, . . ., N. The four cases to consider are:

Case 1 $M = N$. This case is the best possible, and it might lead to extension.

Case 2. $M = 0$. This case is the worst possible, and it leads to contraction or even reduction. Incidentally, when $N>2$, diagram 5 is slightly deceptive, because the centroid D of the face AB differs from the N-1 midpoints of the lines from A to the N-1 vertices B.

Case 3. $M = 1$. This case is almost as bad, and it leads to reflection - contraction or even reflection - reduction. When $N>2$, diagram 6 needs interpretation similar to diagram 5 in the reduction step.

Case 3. $< M < N$. This is the average case, leading to reflection.

The software presented here has as an option the ability to exhibit the value of M, the action taken on each iteration, and the resulting high and low simplex values. Typically, the algorithm moves through cycles of extensions, reflections, contractions, and back to extensions. This occurs because it first finds successful directions, exploits them, and then is forced into a reduced scale before new successful directions can be discovered. The practitioner may find himself or herself drawn into this process, as into a TV show. Watching the algorithm at work is not a bad way to learn something about the model and the data. For example, the parameters best determined become evident during the search. These are the ones which stabilize rather quickly, while the algorithm continues to make larger adjustments to others. Even covariances become somewhat evident. It's quite common to see the algorithm consistently adjust one parameter upward whenever it adjusts another parameter in some particular direction, up or down. The sensitive user may go still further and detect whole groups of parameters whose behavior seems linked during the search for lower F-values.

## 3. THE USER PROGRAM

To interface with the general software described in this manual, the user must write a program (usually rather short) which describes his or her particular problem. Typically, this involved two functions: (I) the function to be minimized, such as the sum of squares S in (1.4) or S* in (1.8), and (II) the function which defines the model, such as one of the growth models (1.1), (1.2), or (1.3). Each of the three major software packages discussed here requires information about one of these functions. MICRO SIMPLEX, which locates minima, references only the function (I) to be minimized. Similarly

## REFERENCES

Bilton, H. T., D. F. Alderdice, and J. T. Schnute. 1981. Influence of time and size of release of juvenile coho salmon on returns at maturity. Can. J. Fish. Aquat. Sci. To appear.

Nash, J. C. 1979. Compact numerical methods for computers: linear algebra and function minimization. Halsted Press, John Wiley and Sons (New York): 227 p.

Nelder, J. A., and R. Mead. 1965. A simplex method for function minimization. Computer J. 7: 308-313.

O'Neill, R. 1971. Algorithm AS 47, function minimization using a simplex procedure. Appl. Stat. 20: 338-345.

Ricker, W. E. 1975. Computation and interpretation of biological statistics of fish populations. Fish. Res. Board Can. Bull. 191: 382 p.

Schnute, Jon. 1981. A versatile growth model with statistically stable parameters. Can J. Fish. Aquat. Sci. To appear.

Schnute, Jon and David Fournier. 1980. A new approach to length-frequency analysis: growth structure. Can. J. Fish. Aquat. Sci. 37: 1337-1351.

Spendley, W., G. R. Hext, and F. W. Himsworth. 1962. Sequential application of simplex designs in optimisation and evolutionary operation. Technometrics 4: 441-461.