Computing Aspects of a Nonlinear Method of Sire Evaluation for Categorical Data

I. MISZTAL, D. GIANOLA, and J. L. FOUILLEY
Department of Animal Sciences
University of Illinois
Urbana 61801

ABSTRACT

The threshold model equations are shown to correspond to a pseudo-linear model. A modification of Jacobi iteration "on data" is presented that does not require setting up the system of equations explicitly. Two iterative threshold model programs were developed: these involve absorption-inversion and iteration "on data". The programs were applied to a set of simulated data. Normal termination did not occur when the data contained fixed effects with observations only in an extreme category, and methods for solving such problems are described. The absorption-inversion program behaved erratically when the model was not of full rank; numerical accuracy problems precluded detection of dependencies between thresholds and other effects. The iteration "on data" version was fastest when only a single round of modified Jacobi was performed for each Newton-Raphson round. A round of iteration for the absorption-inversion program consumed 837, 11.8, and 1.2 s of central processing unit time on a Personal Computer AT, an IBM 3081 mainframe, and a CRAY X-MP/48 supercomputer, respectively. Corresponding times for the iteration "on data" program were 55.2, 2.6, and .57 s. The cost of a threshold model analysis would be three to five times larger than that of a linear model.

INTRODUCTION

Threshold models have been suggested and applied to animal breeding problems with categorical data (3, 4, 5, 6, 8, 9, 10, 11). These models assume the existence of an underlying, unobservable normal variable that is categorized through fixed thresholds. The basic theory of threshold models with reference to animal breeding has been given by Gianola and Foulley (3, 4), Harville and Mee (6), and Gilmour et al. (5). Provided that the assumptions hold, evaluations based on threshold model analysis have given a larger rate of response to selection than those based on linear models (8, 10). The extra response is expected to increase as incidence becomes more extreme, as heritability and number of fixed factors increase, and as the number of response categories decreases.

The potential usefulness of a threshold model analysis is impaired by its computing difficulty. Programs are difficult to write and to test because the necessary equations involve a large number of normal probability integrals, as opposed to linear functions of the data in mixed linear model equations. In threshold models, solutions need to be obtained iteratively, and in each round a linear system of equations must be solved. When this system is large, it must also be solved iteratively, which may involve setting up the coefficient matrix on disk many times, a very expensive procedure.

Another problem in computing threshold models is slow or lack of convergence when all responses for a given class of a fixed factor occur in an extreme category. The solutions to such classes tend to plus or minus infinity as iteration proceeds. Consequently, convergence rate is poor. Also, the values of denominators in certain formulae approach zero, thus causing division by zero errors. To overcome this problem, Harville and Mee (6) suggested either treating fixed factors as random or eliminating the observations causing the problem. The first solution is restrictive and may violate the as-
The objective of this paper was to describe a strategy for solving large threshold model equations and to illustrate its properties using alternative computer systems. An iterative scheme implemented in FORTRAN-77 was employed to solve large threshold models using personal computers, mainframes, or supercomputers.

A LINEAR REPRESENTATION OF THE THRESHOLD MODEL

Consider the linear model:

\[ y = X\beta + Zu + e \]  \[ 1 \]

with \( E(y) = X\beta \), and:

\[
\begin{bmatrix}
\mathbf{u} \\
\mathbf{e}
\end{bmatrix} \sim N \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix} \right)
\]

where \( y \) is a vector of underlying, unobservable variables, \( \beta \) and \( u \) are vectors of fixed and random effects, respectively, \( X \) and \( Z \) are known incidence matrices, and \( e \) is a vector of residuals. Assume \( G \), the variance-covariance matrix of random effects, to be known. Suppose that categorical responses are generated through a set of ordered thresholds \( t = \{ t_1 < t_2 < \ldots < t_n \} \) that partition the real line into mutually exclusive and exhaustive intervals. Estimates of \( t \) and \( \beta \), and predictions of \( u \) can be obtained solving the equations given by Gianola and Foulley (4):

\[
\begin{bmatrix}
T[i-1] \\
X'W[i-1]X \\
Z'W[i-1]X
\end{bmatrix}
\begin{bmatrix}
\Delta t[i] \\
\Delta \beta[i] \\
\Delta u[i]
\end{bmatrix} =
\begin{bmatrix}
p[i-1] \\
X'v[i-1] \\
Z'v[i-1] - G^{-1}u[i-1]
\end{bmatrix}
\]

[2]

where \( T \), \( L \), \( W \), \( p \), and \( v \) involve normal distribution functions, and \( W \) is a diagonal matrix. The above equations follow from Newton-Raphson or Fisher's scoring iterative algorithms, and \( i \) is iterate number. These methods require solving a linear system of equations successively until solutions converge, i.e., when the \( \Delta \) are sufficiently small. System [2] resembles the mixed linear model equations. However, matrices and vectors due to threshold effects are created differently, the right hand sides are a function of \( G \) and \( u \) and the equations are solved for differences between consecutive iterations.

Modified Equations

Equations [2] can be modified so that they can be solved using techniques suitable for a standard linear model analysis. Define:
\[ \Delta t[i] = t[i] - t[i-1] \]
\[ \Delta \beta[i] = \beta[i] - \beta[i-1] \]
\[ \Delta u[i] = u[i] - u[i-1] \]

and write [2] as:

\[ C \begin{bmatrix} t[i] \\ \beta[i] \\ u[i] \end{bmatrix} = r - C \begin{bmatrix} t[i-1] \\ \beta[i-1] \\ u[i-1] \end{bmatrix} \]

[3]

\[ \left[ \begin{array}{c} t[i] \\ \beta[i] \\ u[i] \end{array} \right] = r \]

[4]

where \( C \) and \( r \) are the coefficient matrix and the right-hand side of Equations [2], respectively. Moving the \([i-1]\) solutions to the right-hand side of Equations [4], one obtains, after some algebra:

\[ \left[ \begin{array}{c} t[i] \\ \beta[i] \\ u[i] \end{array} \right] = C^{-1} \left[ \begin{array}{c} t[i-1] \\ \beta[i-1] \\ u[i-1] \end{array} \right] \]

[5]

Denote:

\[ \tilde{L}[i] = L[i-1] W[i-1]^{-1} \]
\[ \varphi[i] = (W[i-1]^{-1})' \{ v[i-1] + L[i-1] t[i-1] + W[i-1](X \beta[i-1] + Z u[i-1]) \} \]
\[ q[i] = L[i-1] (X \beta[i-1] + Z u[i-1]) - \varphi[i] + p[i-1] + T[i-1] t[i-1] \]
\[ H[i] = T[i-1] - \tilde{L}[i-1] W[i-1] \tilde{L}[i] \]

[6]

Then, system [2] can be expressed as follows:

\[ \left[ \begin{array}{c} \tilde{L}[i-1] W[i-1] \tilde{L}[i-1] + H[i-1] \\ X' W[i-1] \tilde{L}[i-1] \\ Z' W[i-1] \tilde{L}[i-1] \end{array} \right] \begin{bmatrix} t[i] \\ \beta[i] \\ u[i] \end{bmatrix} = \left[ \begin{array}{c} \tilde{L}[i-1] W[i-1] \varphi[i] \\ X' W[i-1] \varphi[i] \end{array} \right] \]

[7]
The preceding equations can be viewed as corresponding to a pseudo-linear model for the vector of pseudo-observations:

\[ y_{i-1} = \tilde{L}_{i-1}t_{i-1} + X\beta_{i-1} + Zu_{i-1} + e_{i-1}, \]

with \( e_{i-1} \) being a residual such that \( \text{Var}(e_{i-1}) = (W_{i-1})^{-1} \). Note, however, that [7] involves the matrix \( H \) and the vector of \( q \) of order \( n \times n \) and \( n \times 1 \), respectively (\( n \) is the number of thresholds). The form of [7] suggests that given an additional subroutine for calculating pseudo-observations, \( H \) and \( q \), existing programs for linear models can be modified to accommodate a threshold model. Further, features of a linear model program, such as variance component estimation, can be exploited to advantage in the threshold model analysis to the extent that they are applicable. The pseudo-linear model has additional computational cost due to the need to calculate pseudo-observations, \( H \) and \( q \).

In general, Equations [2] and [7] are not full rank, except with completely random models. If the first threshold (\( t_1 \)) is set to zero, as by Harville and Mee (6), then the part of the pseudo-observations, \( H \) and \( q \) pertaining to that particular threshold do not need to be computed. In particular, there is only one threshold when the number of categories is two, so the parts of equations [7] corresponding to the threshold effect can be eliminated (3, 4) yielding:

\[
\begin{bmatrix}
X'W_{i-1}X & X'W_{i-1}Z \\
Z'W_{i-1}X & Z'W_{i-1}Z + G^{-1}
\end{bmatrix}
\begin{bmatrix}
\beta_{i-1} \\
u_{i-1}
\end{bmatrix}
= 
\begin{bmatrix}
X'W_{i-1}y_{i-1} \\
Z'W_{i-1} \tilde{u}_{i-1}
\end{bmatrix}
\]

Numerical Issues

Computing solutions to threshold models poses several problems not encountered in a linear model analysis. In linear models having less than full rank, a generalized inverse or imposing constraints on solutions lead to the same estimates of estimable functions. In threshold models, some matrices such as \( T \) or \( L \) are calculated with lower accuracy than other matrices, and a generalized inverse may fail to detect dependencies between thresholds and other effects. Consequently, solutions could be inaccurate. However, inverses will be calculated correctly if either the model is reparameterized to full rank explicitly or constraints are imposed to the solutions.

A problem peculiar to threshold models (irrespective of the number of categories of response) occurs when all observations in a given fixed effect fall in an extreme category. For example, suppose that in [1] the only effect in the model is \( \mu \), a general mean, and that there are two categories. In this case the likelihood function (4, 6) is proportional to \( P^{n_1} (1 - P)^{n_2} \), where \( P \) is the probability of response in the first category with \( P = \Phi(\mu) \), \( \Phi(\cdot) \) is the normal distribution function), and \( n_1 \) (\( n_2 \)) is the number of observations in the first (second) category. If all responses are in the first category, the likelihood is proportional to \( [\Phi(\mu)]^{n_1^n} \), and the value of \( \mu \) maximizing this is infinity, as \( \Phi(\infty) = 1 \). Hence, the sequence of values of \( \mu \) in the course of iteration would not be a convergent one. This does not occur with random factors because of the effect of the prior distribution. The extreme category problem (ECP) extends straightforwardly to multiply classified data. In these cases, "automatic"
reparameterization can be dangerous. For example, a system of equations incorporating the restriction that the sum of all classes of a factor of zero will not converge, because an anomalous solution for a class of such factor would cause the solutions for other classes to be anomalous as well. Similarly, setting the first class of a fixed factor to 0 would cause problems if this class has an ECP because the solution for such class tends to either plus or minus infinity.

Formulae for \( T, W, L, v, \) and \( p \) in Equations [2] involve the probability \( P_j = \Phi(t_k - \mu_j) - \Phi(t_{k-1} - \mu_j) \) in denominators, where \( \mu_j = x_j B + z_j u \) is a location parameter for observation \( j \). Whenever \( \mu_j \) is negative and large, the \( \Phi \) are close to 1, so \( P \) is calculated with low numerical accuracy. For example, when \( \mu_j = -5.5, t_{k-1} = -5, t_k = 0, \) and \( P_j \) is calculated with seven-digit precision:

\[
P_j = \Phi(5.5) - \Phi(5) \approx 1.0000000 - 0.9999997 = 3 \times 10^{-7}
\]

which means that only one significant decimal digit is retained. This will cause a reduction of accuracy in formulae where \( P_j \) appears. In extreme cases, where \( \mu_j \) is sufficiently negative, \( P_j \) will be computed as zero resulting in a subsequent "division by zero" error. Such an error can also occur in a non-ECP observation when \( \mu_j \) is strongly negative or when the thresholds are very close in value.

Difficulties caused by the ECP can be remedied in at least two ways. First, one may not allow the values of \( P_j \) to drop below a certain value, which we refer to as \( P_{\text{min}} \), thus eliminating the division by zero error. Second, solutions, tending to plus or minus infinity can be identified before causing problems and set to a sufficiently large number (in absolute sense), for which the normal integral is close to 1 or 0 (say, -10 or +10). We refer to this technique as intercepting. This does not create problems in the system of equations because observations having such values for location parameters do not contribute to the coefficient matrix, as shown below for the two category case.

Suppose that all \( n \) observations in class \( j \) of a fixed factor occur only in the first of two possible categories of response. The pseudo-weight \( w_{jj} \) for this level will be (3):

\[
w_{jj} = \frac{\Phi(t_0 - \mu_j) - \Phi(t_1 - \mu_j)}{\Phi(t_1 - \mu_j) - \Phi(t_0 - \mu_j)} = \frac{\phi^2(t_1 - \mu_j)}{\phi^2(\infty) - \phi^2(t_1 - \mu_j)}
\]

because \( t_0 = \infty \), and where \( \phi \) is the standard normal density. When \( \mu_j \) tends to \( -\infty \):

\[
W_{jj} = \lim_{\mu_j \to -\infty} \frac{\phi^2(t_1 - \mu_j)}{\phi^2(\infty) - \phi^2(t_1 - \mu_j)} = \frac{\phi^2(\infty)}{\phi^2(\infty)} = 0
\]

which also holds if all responses are in the other category, because of symmetry. Whenever the absolute value of \( \mu_j \) is large, the weight \( w_{jj} \) approaches 0, so observations in this level make a negligible contribution to the coefficient matrix. However, such observations contain information on the side of the distribution (with respect to the threshold) the particular class should be. Knowledge that the pseudo-weights \( w_{jj} \) tend to 0 in the progress of iteration can be used to intercept such classes automatically. Whenever \( w_{jj} \) reaches an arbitrarily low value (\( D_{\text{min}} \) say), such an effect is tagged as creating an ECP.

As noted earlier, another possibility would be to exclude from the analysis the observations creating the ECP (6). To the extent that extreme responses are associated with classes of fixed factors, deleting such observations would lead to a biased sample. Hence, we prefer the course of action described.

**Solving the System of Equations**

Equations [7] constitute a linear system of equations stemming from one step of scoring or Newton-Raphson (NR). For small problems, the system can be stored in matrix form in the memory of a computer and then solved by some finite method, e.g., inversion. In a large linear model problem, the coefficient matrix can be created on disk once and the equations solved iteratively. However, creating the coefficients on disk several times, as required by a threshold model analysis, can be expensive. An alternative method would be to iterate "on data" (13, 16). In this approach data are read one or more times per round of iteration but the coefficient matrix is never created. However, more computations are done per round.
We will concentrate on Jacobi iteration "on data", as described by Misztal (12). This procedure is simple but it may diverge, especially in layouts with many factors. The problem can be circumvented if instead of updating solutions for all factors (a feature of Jacobi iteration), only solutions for one factor are updated per pass of the data. Such iteration could be called single factor Jacobi (SFJ). For layouts with fixed and random factors (with a diagonal variance-covariance structure), this would be equivalent to Gauss-Seidel iteration (13), which converges for most situations found in animal breeding. For more complex models (e.g., including a relationship matrix), SFJ can have a slower rate of convergence than Gauss-Seidel. This can be alleviated using second-order Jacobi, which has been found to perform as well as successive overrelaxation when the numerator relationship matrix is included (12). In general, SFJ retains the simplicity of Jacobi and the convergence rate of Gauss-Seidel, and data preparation is facilitated because files do not need to be sorted. The need to read data as many times as there are factors in the model is a disadvantage of SFJ. Although data can be read fewer times, simplicity is sacrificed. For example, data could be read one time less if sorted (13). Also, one could process more than one factor per pass of the data, hoping that convergence rate would be unaffected.

The proposed strategy for solving threshold model equations involves a combination of NR (or scoring) and SFJ. Each step of NR leads to equations such as Equation [7], and SFJ is then used to solve these. When SFJ converges, one proceeds to another step of NR and 20 rounds of SFJ are needed to solve the system of linear equations arising in each step of NR, a total of 100 linear rounds would be performed.

**COMPUTER INDEPENDENT OPTIMIZATION**

Desirable features of a program designed to carry out a threshold model analysis would be simplicity, computer-independence (portability), and speed, to some extent (15). It is useful to identify speed optimization strategies that do not affect adversely the first two features. In general, speed is usually affected by small, computationally intensive sections of a program, and a large gain in speed can be realized by optimizing such bottlenecks. In what follows, FORTRAN-77 is assumed to be the language of choice because it is available on most computers and is widely used in animal breeding.

Suppose Equation [7] is to be solved by inversion. The computing cost of inverting a matrix is roughly proportional to the cube of its rank. The cost of other computations in the analysis varies approximately linearly with the size of the data set. If the coefficient matrix is large the cost of inversion will greatly exceed other costs, and only optimization of the inversion routine can yield a sizable increase in efficiency. However, if Equation [7] is solved by iteration, the major bottleneck would be reading the data or the coefficient matrix (14, 16). In a threshold model analysis, this is even more important, because data are read more times even when the coefficient matrix is stored in memory. A bottleneck specific to a threshold model analysis can be the calculation of distribution functions.

**Read and Write Optimization**

The only statements in standard FORTRAN-77 that read and write to or from disk are READ and WRITE, the fastest option being unformatted. The average data transfer rates of unformatted READ and WRITE statements, as a function of the size of the data transferred in a single statement, are shown in Table 1 for different computers. Reading the data 2 words at a time (typical of an iteration with the coefficient matrix on disk) or 5 words at a time can be over 20 times slower than reading the data 1000 words at a time. Thus, a program can be speeded up by: 1) reading many coefficients per statement, or 2)

<table>
<thead>
<tr>
<th>Size of transfer (words)</th>
<th>PC AT</th>
<th>IBM 3081</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.7</td>
<td>18.2</td>
<td>49.0</td>
</tr>
<tr>
<td>5</td>
<td>9.3</td>
<td>44.1</td>
<td>119.0</td>
</tr>
<tr>
<td>10</td>
<td>13.9</td>
<td>86.2</td>
<td>233.6</td>
</tr>
<tr>
<td>1000</td>
<td>35.7</td>
<td>187.5</td>
<td>2884.6</td>
</tr>
</tbody>
</table>

Journal of Dairy Science Vol. 72, No. 6, 1989
by replacing READ and WRITE statements by a subroutine called IOF ("input/output fast"). Such a subroutine works as an interface between the main program and the external storage. The main program sends all data that would otherwise be transferred by the write statement to the subroutine; this, in turn, collects that data in a large buffer. When this is full, the subroutine performs a large data transfer without intervention from the main program. When the main program requires some data, the subroutine reads a large data buffer and dispenses these data in such quantities as the main program demands.

Reading many coefficients per statement is possibly faster, but it complicates programming. The second approach delegates all the complexity of data transfers to a subroutine. The speed of a 5-word transfer using formatted, unformatted, or IOF options for three types of computers is given in Table 2. The IOF was fastest in all cases, surpassing the unformatted READ/WRITE option by a factor of at least 2.

### Probability Functions

We evaluated three methods for computing normal probability functions. The first uses functions from the IMSL computer library, available in a range of computers (17). The second employs formulae from Abramowitz and Stegun (1), using the EXP function and a Taylor expansion. Third, we propose a table approach, where probabilities are stored into a table, and a linear interpolation between tabulated points is performed. The table can be generated using algorithms such as those described by Ducrocq and Colleau (2). The speed of the three methods is presented in Table 3. The table option was the fastest because it uses relatively few arithmetic operations.

### NUMERICAL BENCHMARKING

#### Simulation of Data

Data were generated using the model:

$$y_{ijkl} = h_i + g_j + s_{jk} + e_{ijkl}$$

where \(y_{ijkl}\) is the liability (underlying variable) of cow \(i\), daughter of sire \(k\) in group \(j\), and producing in herd-year-season \(i\). The values of herd-year-seasons \((h_i)\), groups \((g_j)\), sires \((s_{jk})\), and residuals \((e_{ijkl})\) were sampled from independent normal distributions with null means and variances .30, .22, .03, and .45, respectively. This gives an intra-herd-group heritability of \(h^2 = .25\) and a variance ratio \(\sigma^2_h/\sigma^2_s = 15\). A total of 5000 records were generated, representing 800 \(h\) classes, 18 \(g\) classes, and 200 sires. The layout was unbalanced, and the number of records per class of the appropriate factor ranged between 1 to 17 (herd-year-seasons), 233 to 372 (groups), and 11 to 54 (sires). Liability was trichotomized to yield incidence of 80, 15, and 5% in the corresponding categories. The data set contained 321 \(h\) effects with observations in the first category only, representing a total of 1843 records. These observations were deleted to form a reduced data set free of the ECP. There was no class of the \(h\) factor with observations in the third category only. In the analysis, \(h\) and \(g\) effects were treated as fixed.

#### Programs

Two programs were developed for conducting the analysis: one used iteration "on data", 

<table>
<thead>
<tr>
<th>Computer</th>
<th>Option</th>
<th>PC AT</th>
<th>IBM 3081</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IMSL (EXP + ERF)</td>
<td>47</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXP + Taylor expansion</td>
<td>.6</td>
<td>28</td>
<td>119</td>
</tr>
<tr>
<td></td>
<td>&quot;Table&quot; approach</td>
<td>1.4</td>
<td>50</td>
<td>228</td>
</tr>
</tbody>
</table>

1Option not available at the time of the study.

---

The table option was the fastest because it uses relatively few arithmetic operations.
and the other used absorption and inversion with full storage in memory of the coefficient matrix. Both programs employed the pseudo-linear model concept, and used IOF read and write routines and the table approach for computing probability functions. The programs can be applied to either a threshold or a linear model and are available upon request. Unless noted otherwise, the minimum allowable value for $P_{jk}$ was set to $10^{-10}$, the first threshold ($t_1$) was set to 0, and classes with ECP were not intercepted ($D_{min} = 0$).

**Comparisons**

Several computing options were compared, as discussed later. Except where noted, the convergence criterion was:

$$C_i = \frac{||(\mathbf{u}_i - \mathbf{u})^T(\mathbf{u}_i - \mathbf{u})||^5}{(\mathbf{u}^T \mathbf{u})^5}$$

where $i$ is the NR round number for the matrix version, or the total number of SFJ rounds for iteration “on data”. Only solutions for random effects were monitored because these do not change with different constraints on fixed effects imposed automatically by different computing procedures. The “converged” sire solutions $\mathbf{u}$ were obtained by continuing iteration until no further changes in solutions occurred within six significant digits. A $C_t$ equal to $10^{-p}$ corresponds approximately to $p$ significant decimal digits of accuracy (14). The criterion $C_t$ is reliable because it contrasts current solutions with “true” ones. However, its application is limited in practice because “true” solutions are often not known. A comparison used a criterion based on differences between all solutions in consecutive rounds:

$$C_i = \frac{||(\mathbf{b}_i - \mathbf{b}_{i-1})^T(\mathbf{b}_i - \mathbf{b}_{i-1})||^5}{(\mathbf{b}_i^T \mathbf{b}_i)^5}$$

Above, $\mathbf{b}$ is a solution vector. Criterion $C_d$ is less accurate than $C_t$, does not require converged solutions.

The convergence of a threshold model (T) versus a linear model (L) analysis was studied by running the following programs: 1) L-iterative, 2) T-matrix, and 3) T-iterative. The T-iterative program included runs with either 1, 2, or 5 SFJ rounds per step of NR. The L-matrix program was not run because it converges in one round. The effect of imposing a constraint on the value of the first threshold was investigated running both T-programs, with and without constraints, and using the full data set. An L-iterative run was carried out as a reference point. Various means of overcoming the ECP problem were studied by running the T-iterative program with the full data set at various interceptions ($D_{min}$ values), and at various minimums for $P_{jk}$ ($P_{min}$).

**Computers**

Three types of computers were employed. The IBM 3081 is a high speed mainframe optimized for a broad range of computational loads, and representative of the computing environment typically available in animal breeding applications. The second computer was an IBM PC AT compatible with an 8-MHz clock, numerical coprocessor, and a simulated disk in memory (RAM disk); it has a relatively fast integer arithmetic but is slow in floating point arithmetic. The third computer was a CRAY X-MP/48 supercomputer with special vector capabilities that allow matrix type operations up to 10 times faster than scalar ones (7).

All computers employed optimizing FORTRAN compilers with the highest possible degree of optimization. Only default system options were used. However, computer-specific options could have a large effect on some comparisons. For example, in the IBM 3081, the FORTRAN “OPEN” statement was replaced by a specific “FILEDEF” definition with a large BLKSIZE parameter. These two options did not affect transfer rates when the size of the data transfer was small. However, when this size was large, the FILEDEF alternative increased transfer rate by a factor of 10.

**RESULTS AND DISCUSSION**

The total number of rounds required to reach several values of $C_t$ for the data set without the ECP observations is given in Table 4. The analysis employing a linear model and iteration “on data” needed between 2 and 9 rounds to...
TABLE 4. Total number of rounds required to reach several values of criterion $C_t$ by model and program.\(^1\)

<table>
<thead>
<tr>
<th>Model</th>
<th>Program</th>
<th>SFJ rounds per NR step(^2)</th>
<th>$\cdot1$</th>
<th>$\cdot01$</th>
<th>$\cdot001$</th>
<th>$\cdot0001$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>Iterative</td>
<td>. . .</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Threshold</td>
<td>Matrix</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Threshold</td>
<td>Iterative</td>
<td>5</td>
<td>12</td>
<td>21</td>
<td>26</td>
<td>31</td>
</tr>
<tr>
<td>Threshold</td>
<td>Iterative</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Threshold</td>
<td>Iterative</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>11</td>
</tr>
</tbody>
</table>

\(^1\)Data set without ECP observations, $D_{min} = 0$, $P_{min} = 10^{-10}$, $t_i = 0$.

\(^2\)SFJ = Single factor Jacobi; NR = Newton-Raphson.

Converge to the desired accuracy. The T-matrix program reached an accuracy of $10^{-1}$ in 3 rounds, and 7 rounds were needed to attain $C_t = 10^{-4}$. In the trials with the T-iterative program, the total number of round increased with the number of SFJ rounds per step of NR. Thus, the most efficient strategy (in terms of number of rounds needed to reach a certain $C_t$) was performing a single round of SFJ per step of NR. The results obtained with the L-iterative and T-iterative programs suggest that although extra rounds were required by the latter to arrive at $C_t = 10^{-1}$ (2 and 4 rounds for L-iterative and T-iterative programs, respectively), convergence rate was similar thereafter. Therefore, the two analyses seem to differ essentially in the speed with which initial accuracy is reached.

Similar results for the full data set, including observations having the ECP in classes of the herd-year-season factor, are shown in Table 5. The T-matrix program used without imposing restrictions produced erroneous solutions because the generalized inverse routine employed failed to detect dependencies between thresholds and other effects. Setting the first threshold to zero did not affect the performance of the matrix and iterative programs. This suggests that setting a threshold to 0 can be used routinely, thus saving the time of processing the corresponding equation. A comparison between the results in Tables 4 and 5 indicates that convergence was 1 to 2 rounds slower in both threshold model programs when observations with ECP were included.

Convergence rates with varying $P_{min}$ and $D_{min}$ values for the full data set and the T-iterative program are shown in Table 6. With $P_{min} = 0$ and $D_{min} = 0$, the program aborted due to a "division by zero" error caused by a 0 value of $P_{jk}$ while processing a record with an ECP. Raising $P_{min}$ to $10^{-10}$ or $10^{-6}$ allowed the program to run normally. When $P_{min}$ was $10^{-3}$, the program achieved 4 digits of accuracy in one round less than when it was $10^{-6}$. However, the program diverged when $P_{min}$ was set at $10^{-2}$. This suggests that for this data the $P_{jk}$ for all non-ECL classes were between $10^{-6}$ and

TABLE 5. Total number of rounds required to reach a given level of criterion $C_t$.\(^1\)

<table>
<thead>
<tr>
<th>Model</th>
<th>Program</th>
<th>None</th>
<th>3</th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold</td>
<td>Matrix</td>
<td>None</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Threshold</td>
<td>Matrix</td>
<td>$h_i = g_i = 0$</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Threshold</td>
<td>Matrix</td>
<td>$t_i = 0$</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>Threshold</td>
<td>Iterative</td>
<td>None</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>Threshold</td>
<td>Iterative</td>
<td>$t_i = 0$</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>13</td>
</tr>
</tbody>
</table>

\(^1\)Full data set containing observations with extreme category problems, $D_{min} = 0$, $P_{min} = 10^{-10}$.

\(^2\)Solutions incorrect.

\(^3\)One round of single factor Jacobi per step of Newton-Raphson.
TABLE 6. Total number of rounds required to reach a given level of criterion $C_i$ for several options of the T iterative program.

<table>
<thead>
<tr>
<th>$P_{\text{min}}$</th>
<th>$D_{\text{min}}$</th>
<th>.01</th>
<th>.001</th>
<th>.0001</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>0</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>0</td>
<td>5</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>.01</td>
<td>5</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>$10^{-1}$</td>
<td>.1</td>
<td>5</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>$10^{-10}$</td>
<td>.5</td>
<td>8</td>
<td>11</td>
<td>14</td>
</tr>
</tbody>
</table>

$1$ Full data set containing observations with extreme category problems, $t_i = 0$, and 1 round of single Jacobi per step of Newton-Raphson.

$2$ Program aborted with an error message.

$3$ Program diverged.

10$^{-3}$. Thus, setting $P_{\text{min}}$ to $10^{-6}$ or $10^{-10}$ seems to be a safe way to prevent division by zero and divergence problems.

The effect of intercepting solutions having an ECP is also shown in Table 6. For $P_{\text{min}}$ of $10^{-10}$, values of $D_{\text{min}}$ (minimum allowable value of $w_{ij}$) of .01, .1, or .5 were used. If $w_{ij}$ was less than $D_{\text{min}}$, the solution for that effect was arbitrarily set to -10; this identified the effect as causing an ECP and yielded normal integrals close to zero. However, convergence was somewhat slower than with $D_{\text{min}} = 0$. Thus, intercepting did not appear advantageous in this situation, where setting $P_{\text{min}}$ to $10^{-10}$ was sufficient to solve the ECP problem.

Because the "true" solutions required to evaluate $C_i$ are seldom available in practice, the effects of intercepting solutions on $C_d$ (see [13]) are shown in Table 7. When classes having ECP were not "intercepted" ($D_{\text{min}} = 0$), the values of $C_d$ suggested that the system was converging much slower than as indicated by $C_i$ (Table 6). This was caused by changes in solutions of ECP classes, even though other solutions had converged. Intercepting such effects eventually fixed the ECP classes at -10, and $C_d$ indicated much faster convergence.

The CPU times per round of iteration used by the two threshold model program with the full data set are shown in Table 8. In all computers, a round was executed faster in the iterative version. The matrix version was about 15, 4, and 2 times slower in the PC AT, IBM 3081, and CRAY, respectively.

The decomposition of CPU time for the two threshold model programs is shown in Tables 9 and 10. In the matrix version (Table 9), the two most time-consuming parts were inversion and absorption, accounting for over 67% of the time. This illustrates that optimization of other

TABLE 7. Values of the convergence criterion $C_d$ for the T iterative program at several stages of iteration and several $D_{\text{min}}$ classes.

<table>
<thead>
<tr>
<th>$D_{\text{min}}$</th>
<th>4</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.17</td>
<td>.073</td>
<td>.057</td>
<td>.046</td>
<td>.036</td>
</tr>
<tr>
<td>.01</td>
<td>.17</td>
<td>.073</td>
<td>.057</td>
<td>.046</td>
<td>.036</td>
</tr>
<tr>
<td>.1</td>
<td>.17</td>
<td>.062</td>
<td>.013</td>
<td>.025</td>
<td>.00515</td>
</tr>
<tr>
<td>.5</td>
<td>.17</td>
<td>.062</td>
<td>.017</td>
<td>.026</td>
<td>.0016</td>
</tr>
</tbody>
</table>

$1$ Full data set with extreme category problem observations, $t_i = 0$, $P_{\text{min}} = 10^{-10}$.
parts of the program would not be cost-effective. Inversion consumed 81, 46, and 22% of the total time in the PC AT, IBM 3081, and CRAY, respectively. Reading the data took not more than 3% of the time. The relatively high cost of absorption on the larger computers was due to the large number of herd-year-season classes and back solutions needed in the threshold model. However, inversion was relatively fast in the CRAY but very slow in the PC AT. This is because the inversion routine was able to exploit the fast vector features in the CRAY whereas in the PC AT, floating point operations (a main component of inversion) are relatively slow.

The iterative program (Table 10) spent 65, 52, and 55% of the time calculating threshold relating items. In general, this means that the T-iterative program would be at least two times slower than the L-iterative one, where calculation of integrals, linearization, and reading and writing of pseudo-observations is not needed. The program spent 10 to 27% of the time reading data. Altogether reading and writing took 16 to 50% of the time. If unformatted read and write rather than the optimized routine IOF had been used, this share would have been 30, 62, and 70% in the PC AT, IBM 3081, and CRAY, respectively. Hence, this optimization was important. Probability functions with the table approach did not require more than 6% of the total time in either program (this would have been no more than 12% if the other methods had been used), and optimization of this aspect had relatively little importance.

In summary, the extra computing cost of a threshold model relative to a linear model analysis was almost exclusively due to the need to iterate in the matrix version and to a "threshold overhead" in the iterative program. Assuming that a two-decimal digit accuracy is desired, the increase in total computing cost would be approximately 5 times in the matrix version and 3 times in the iterative version. Further optimization of the programs would be expected to have different effects in different computers. For example, the solutions could be obtained in the matrix program by some finite method, e.g., Gaussian elimination, whenever the inverse of the coefficient matrix is not needed. This would speed up the program several times on a PC AT, but would reduce computing time 20% in a CRAY. Exploiting symmetry when forming Equations [7] could be effective in a PC AT or an IBM 3081, but it could slow down a CRAY considerably, because vector-type processing would no longer be possible.

In conclusion, an iterative scheme employing fast reading and writing routines allows

<table>
<thead>
<tr>
<th>Computer</th>
<th>PC AT</th>
<th>IBM 3081</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>threshold-related operations</td>
<td>4</td>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>Probability functions</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Linearization</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Other</td>
<td>3</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>Reading data</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Absorption + back solution</td>
<td>12</td>
<td>37</td>
<td>45</td>
</tr>
<tr>
<td>Inversion</td>
<td>81</td>
<td>46</td>
<td>22</td>
</tr>
<tr>
<td>Other, common with linear model program</td>
<td>3</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>

1 Full data set with extreme category problem observations.

In conclusion, an iterative scheme employing fast reading and writing routines allows

**TABLE 9. Distribution of total central processing unit time spent (%) in various parts of the threshold-matrix program in three computers.**

1 Full data set with extreme category problem observations.
TABLE 10. Distribution of total central processing unit time spent (%) in various parts of the threshold-iterative program in three computers.1

<table>
<thead>
<tr>
<th>Part of program</th>
<th>Computer PC</th>
<th>AT IBM 3081</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold-related operations</td>
<td>65</td>
<td>52</td>
<td>55</td>
</tr>
<tr>
<td>Probability functions</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Linearization</td>
<td>20</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>Reading and writing pseudo-observations</td>
<td>6</td>
<td>23</td>
<td>17</td>
</tr>
<tr>
<td>Other</td>
<td>33</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Reading data</td>
<td>10</td>
<td>27</td>
<td>17</td>
</tr>
<tr>
<td>Other, common with linear model program</td>
<td>25</td>
<td>21</td>
<td>28</td>
</tr>
</tbody>
</table>

1 Full data set with extreme category problem observations.

solving large threshold model analyses in many computers. The strategy involved measures that permit conducting the analysis without needing to remove unsuitable records from the data. The costs for other data and models can be extrapolated from the results presented by using the following considerations: 1) inversion costs rise cubically with the number of equations in the system; 2) costs from reading and writing data vary almost linearly with the number of records, and less than linearly with the size of records; and 3) the number of threshold computations is proportional to the number of thresholds and to the number of records.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to Ina Hoeschele, J. Weller, and G. Wiggalas, who provided many useful comments. Support by US-Israel Binational Agricultural Research and Development Project No. US-805-84 and by the National Center for Supercomputing Applications, Champaign, IL, is gratefully acknowledged.

REFERENCES

17 User's manual for STAT/Library. 1987. IMSL Inc., Houston, TX.