Issue No. 20 *1987(B)*

The GENSTAT Newsletter

NAG NUMERICAL ALGORITHMS GROUP

Editors

P W Lane
Rothamsted Experimental Station
Harpenden
Hertfordshire
United Kingdom AL5 2JQ

K I Trinder
NAG Central Office
Mayfield House
256 Banbury Road
Oxford
United Kingdom OX2 7DE

Printed and produced by the Numerical Algorithms Group

ISSN 0269-0764

GENSTAT NEWSLETTER
Issue No. 20

NP1600

1987 October

Contents

Published Twice Yearly by
Rothamsted Experimental Station Statistics Department
and the Numerical Algorithms Group Ltd

Editorial

This issue of the Newsletter consists mostly of articles based on presentations at the Genstat Conference in Pavia, held in September 1987. We would like to take this opportunity to thank all those who gave presentations at the Conference, and to thank the organizers, particularly Carlo Berzuini of the University of Pavia and Roger Payne of Rothamsted.

Genstat 5 was demonstrated at the Conference, both on VAX VMS and IBM PC systems. Since then, the VAX VMS version has been released by NAG. The high-resolution graphics of this version are produced in a form suitable for the GHOST80 graphical system: other versions will soon be available for the GKS and GINO-F systems. The Sun Unix version of Genstat will be released soon after the time this Newsletter appears; the graphics facility will be available via the Sun CGI or GKS systems. The IBM PC version of Genstat has been delayed by various problems, which have now been resolved. It will be available before long, although initially without any high-resolution graphics. Many other conversions are underway: contact NAG to get details of these.

The Genstat 5 Manual was published in October by Oxford University Press. Initially, it is in hardback only and costs £45. The Reference Summary was published as a separate booklet in November by NAG, and is priced at £2.50, with a minimum order of four copies. Both of these publications are available now from NAG. The introductory book has been rewritten, and was published early in 1988 by Oxford University Press as 'Genstat 5: an Introduction'.

Following the release of Genstat 5, the developers of Genstat would like your help in two particular areas. Firstly, there is a need for procedures for the Library which has replaced the Genstat 4 Macro Library. If you wrote a macro for the old Library, or if you write a general program that is likely to be useful for other users, please develop it into a procedure and submit it to the new Library: details of procedure writing and submission are given in an article in this Issue. Secondly, reaction is needed about the new style of diagnostics produced by Genstat. If you encounter a diagnostic that is obscure, or that gives wrong or misleading information about its cause, please send a copy of the program and the offending message to NAG.

One of the articles in this Issue concerns the new facilities in Genstat for allowing extensions at the Fortran level. This theme has been chosen as the subject of a one-day Conference to be held at Rothamsted on 28 April. Some presentations will describe how to use the new facilities; others will describe their application, for example in accessing subprograms from the NAG Fortran Library, and in developing new areas of analysis eventually to be incorporated in Genstat. For further information, please contact:

Roger Payne
Statistics Department
Rothamsted
Harpenden
United Kindom AL5 2JQ

Changing from Genstat 4 to Genstat 5

H Talbot
Edinburgh University Computing Service
West Mains Road
Edinburgh
United Kingdom

The Computer Service

Edinburgh University has approximately 4000 accredited users of its mainframe services with a further 500 users of micros. This population consists largely of University lecturing or research staff as well as undergraduate and postgraduate students. In addition we have a few commercial users and a rather larger number of Government supported scientific research workers. Of our user community, approximately half use some sort of statistical package with perhaps 200 using Genstat. Some of these obtain help from a special Genstat clinic which is run by the Scottish Agricultural Statistical Service (SASS). However all University users must use the Computing Service for Genstat advice. At the moment this is given by two people who have largely learnt the hard way, and neither of whom have any formal statistical training.

For some time we have adopted a policy of not giving courses but writing code for users to get them started. Indeed, with Genstat being such a difficult package to learn, we have encouraged as many as possible to use other packages where these are appropriate. This is mostly because our users make infrequent use of statistical packages and so there is little point in them learning a difficult package when there are easier ones. Those mainframe users who make considerable use of Genstat are largely agriculturalists who get their guidance from SASS.

We have many students who are with the University for fairly short periods of time and do not have time to learn Genstat but still require its facilities. The best we can do for these is to involve their supervisors and work with both student and staff to provide the required code.

We expect Genstat 5 in the autumn. From then on the incidence of Genstat enquiries will rise considerably. At the moment we have, on average, three to four enquiries a week about Genstat. The new version of Genstat will at least double that. But the main bulk of our work will be in persuading people to try the new goods.

Changing Mainframe Computers

There are many similarities between our recent change over of mainframe computers and the change in the two versions of Genstat. The two machines were running similar operating systems, but there were differences. In some cases the differences were quite fundamental. In the use of packages, users who had previously enjoyed an environment in which they did not need to know much about the computer, suddenly found that a knowlegdge of how to define files became vital. We have had a considerable amount of trouble both during and after the change. Much is the result of users refusing to take advice offered to them. Indeed steering the course between the users' happiness and the advisors' sanity is very difficult.

In order to meet the need for conversion advice when Genstat changes, we must be prepared. Our recent experiences when we changed from one type of mainframe to another suggests we should take the following points into account:

(1) Centrally run conversion courses are poorly attended. Of our thousands of mainframe users less than 200 actually bothered to attend a course. The reason for this is probably a lack of interest.

(2) No amount of advertising will persuade people to make the transfer during a period of overlap. About 20% will leave it to the last minute or later. The one great advantage in a period of overlap is that the computer centre have a very good opprotunity to try out the new system without any pressure. If it does not work then it is easy to tell the user to revert to the old machine.

(3) Running a parallel service only helps the committed. The difficult customers are still late runners. Our experiences with SPSS and SPSS-X, when we had both running for over four years, still resulted in a large degree of panic when SPSS was removed with some users unable to understand that SPSS-X might be a good path to take.

(4) The less computer literate people take the whole procedure in a much more organised fashion and seek help early on. The users with large problems are the ones who wander in at the last minute with something which will not work.

(5) There is always a proportion of people who regard conversion as a computing centre service activity which they do not need to know too much about. The User Support staff will always do it for them.

Our plans for Conversion

Many Genstat users are scientists with very little time to learn Genstat, never mind convert to a new version of the language. We regard a smooth transition as being vital if our users are going to continue with the language. With considerable experience in computing, we should easily be able to provide the tools for the job. One could argue that it might be better done centrally by the providers of Genstat but failing that we intend to provide a package for our users consisting of:

(1) A concisely written document showing the differences in the two languages.

(2) On-line help information.

(3) A computer program to convert from one to the other.

Conversion Documentation

The document and on-line help information will be closely related. Indeed they may even be the same. We have a system where any machine readable text is easily converted to on-line help information. In this document there will be sections of the form:

(1) Data structures.
(2) Reading data.
(3) Printing data.
(4) Plotting.
(5) Analysis of variance.
(6) Linear regression analysis.
(7) Using external files.

Within each section, words will be kept to a minimum whilst wide use will be made of examples. Taking the first of these sections as an example our document will be of the form:

Genstat 4

```
'variate' A,B $ 20
'variate' C $ 2 = 1,2
'factor' E,F $ 2 = 1,2
```

Genstat 5

```
variate [nvalues=20] A,B
variate [nvalues=2; values=1,2] C
factor [levels=2; values=1,2] E,F
```

Writing a document of this nature without a new manual is very difficult. I have acquired a great amount of paper on the new instructions but I am left with many holes and question marks. However since we are likely to get the new version before the manual, we will continue to try to find solutions to the problems.

Conversion Program

We have done a great amount of work in writing a program to take the most commonly used directives and convert them into the new language. There is still much work to be done on this but so far the directives available are:

```
BLOCK                    OUTPUT
CLOSE                    PAGE
COVARIATE                PRINT
ENDMACRO                 READ
EOD                      REFERENCE
FACTOR                   REPEAT
FOR                      STOP
GENERATE                 TREATMENT
INPUT                    UNIT
MACRO                    VARIATE
NAME
```

The program has been written in a modular form in Fortran so that further directives can be added easily. Before it is released to users it will contain the linear regression and analysis of variance directives. Throughout the program I have used extra instructions rather than complicate the logic. In the case of `PRINT` and `READ`, they have many options and parameters in common but to put them together would have added an extra layer of logical complication. There is no intention on my part to include all the directives. My aim has been firstly, to include all those of a simple nature and secondly, to include all those more complicated ones which will allow conversion of analysis of variance and linear regression programs. Many of our users are likely to carry out such techniques as time series and multivariate analysis with other packages. However, I would be interested to hear from anyone who considers such techniques as vital parts of a conversion program, with a view to possible collaboration.

The Ideal Solution

We would like to provide both a teaching video and a computer aided learning program so that our users can learn when it suits them. Unfortunately we do not have facilities to make either, but it seems that since the language has been made so much more logical the subject now lends itself to a video with perhaps a computer aided learning package to back it up. A video on the subject of conversion would not be a viable idea but there is scope for a new video of the language, based on an introductory text.

The Analysis of a Mixed Model

M E van den Bol
ITI-TNO
PO Box 214
2600 GA Delft
The Netherlands

1. What is a Mixed Model?

A mixed model is a linear model with several error terms called random effects. The model may be described as:

$$\underline{y} = X\alpha + \underline{Z}_1\underline{b}_1 + \underline{Z}_2\underline{b}_2 + \underline{e}$$

with the following definitions:

$\underline{y}$ is a vector of N observations,
α is a vector of p fixed effect parameters,
$\underline{b}_1$ and $\underline{b}_2$ are vectors of unobservable random effects
$\underline{e}$ is a vector of residual error terms.
$\underline{e}$ and $\underline{b}_i$ are independently distributed Normal variables,
σ_1^2, σ_2^2 and σ_e^2 are called the components of variance.

(Random variables are underlined)

2. Some History

The idea that experimental error can arise from several different sources, and the importance of identifying these sources has been known for a long time.

Astromoner Airy [1] postulated that his observations on a particular night were subject to errors from two different sources:

- an error peculiar to that night, caused by "atmospheric and personal circumstances",
- a random error of measurement.

He was clearly aware that if the error peculiar to a night proved to be larger than the error of measurement, then the total number of nights over which observations were taken could be more important than the total number of observations.

Uses in the biological science were pioneered by Fisher [6]. In 1925 he described a method of estimating the size of error from different sources: that of equating sums-of-squares to their expected values in terms of error components. This method was limited however to balanced data.

A technique for analysing unbalanced data, classified by no more than one factor, was proposed by Cochran [2]. He again used the principle of equating sums-of-squares to their expectations.

Henderson [9] showed how these ideas could be extended to all types of unbalanced data. He described several methods which gave "unbiased" estimates of the components of variance, the most popular being Henderson's method III.

More recently, techniques to estimate variance components by maximum likelihood (ML) have been developed. These were first suggested by Crump [3], [4] and set out in a general form by Hartley and Rao [7].

Advantages of the maximum likelihood approach are:

- it is conceptually simple and always well defined,
- it requires no assumptions concerning the structure or balance of the data, and
- estimates of functions of the components of variance are easily obtained, along with the approximate standard errors.

A disadvantage is the fact that ML estimators differ from the analysis of variance estimators in the case of balanced data and that the computational burden may be quite heavy.

The analysis of variance estimators have been shown to be

- the best quadratic unbiased estimators in balanced data and
- the best unbiased estimators if the data are balanced and Normally distributed.

The ML estimators are generally biased downwards, sometimes dramatically so; [8], [11].

The problem of bias can be overcome by the use of residual maximum likelihood (REML), so called because residuals are used in the estimation procedure. This technique has also been called restricted maximum likelihood. Since contrasts between unknown fixed or treatment effects cannot provide any information on the error structure, the REML technique sets out to maximize the joint likelihood of all error contrasts which have zero expectation. There are broad analogies with analysis-of-variance techniques where both treatment sums-of-squares and degrees of freedom are subtracted in order to estimate the error distributions. Indeed, for balanced data, REML and ANOVA estimators are identical.

The general techniques for REML analysis of unbalanced data were developed by Patterson and Thompson, [10], [11].

3. Maximum Likelihood

Model: $\underline{y} = X\alpha + Z\underline{b} + \underline{e}$

with $E(\underline{b}) = E(\underline{e}) = 0$
$Var(\underline{b}) = E(\underline{bb}') = D$
$Var(\underline{e}) = E(\underline{ee}') = R$
$Var(\underline{y}) = E(\underline{yy}') = V = ZDZ' + R$

When we are interested in the fixed effects and we know the matrix V, then the generalized least-square solution for α is:

$$\hat{\alpha} = (X'V^{-1}X)^{-1}X'V^{-1}y$$

A problem is that the matrix V has dimension $N \times N$, where N is the total number of observations and may be very large, and is generally not diagonal.

Henderson indicated in 1959 that a set of equations not involving V^{-1} can be established for deriving $\hat{\alpha}$. Suppose that the effects represented by $\underline{b}$ are in fact fixed and not random, then the Normal equations would be:

$$\begin{bmatrix} X'R^{-1}X & X'R^{-1}Z \\ Z'R^{-1}X & Z'R^{-1}Z \end{bmatrix} \begin{bmatrix} \hat{\alpha} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} X'R^{-1}y \\ Z'R^{-1}y \end{bmatrix}$$

Amend this by adding D^{-1} to the lower right-hand sub-matrix $Z'R^{-1}Z$:

$$\begin{bmatrix} X'R^{-1}X & X'R^{-1}Z \\ Z'R^{-1}X & Z'R^{-1}Z + D^{-1} \end{bmatrix} \begin{bmatrix} \hat{\alpha}^* \\ \hat{b}^* \end{bmatrix} = \begin{bmatrix} X'R^{-1}y \\ Z'R^{-1}y \end{bmatrix}$$

These equations are called the Mixed Model Equations (MME's).

The solution $\hat{\alpha}^*$ proves to be identical to the generalized least-squares solution $\hat{\alpha}$ when the covariance matrix V is known, because solution of the MME's gives:

$$(X'WX)\hat{\alpha}^* = X'Wy$$

with $W = R^{-1} - R^{-1}Z(Z'R^{-1}Z+D^{-1})^{-1}Z'R^{-1}$
and WV may be shown to be equal to I, hence $W = V^{-1}$.

The MME's arise also from the joint density of $\underline{y}$ and $\underline{b}$. On assuming Normality for $\underline{e}$ and $\underline{b}$: $\underline{e} \sim N(0,R)$ and $\underline{b} \sim N(0,D)$, this joint density is

$$f(y,b) = g(y|b).h(b) = C\ exp[-\tfrac{1}{2}(y-X\alpha-Zb)'R^{-1}(y-X\alpha-Zb)]\ exp[-\tfrac{1}{2}b'D^{-1}b]$$

where C is a constant. Maximizing with respect to α and b leads at once to the MME's.

The solution for $\hat{\alpha}^*$ is of interest because α is a vector of fixed effects in the model. The solution for b^* is, in many situations, of interest also. It is an estimator of the conditional mean of $\underline{b}$ given y:

First $Cov(\underline{b},\underline{y}) = E(\underline{b}(\underline{y}-X\alpha)') = E(\underline{b}(Z\underline{b}+\underline{e})') = (E\underline{b}\underline{b}')Z' = DZ'$

Then, on assuming Normality:

$$E(\underline{b}|y) = E(\underline{b}) + Cov(\underline{b},\underline{y})\ [Var(\underline{y})]^{-1}[\underline{y}-E(\underline{y})] = DZ'V^{-1}(y-X\alpha)$$

The MME's lead to the same solution for b^* with α replaced by α^*, which is the maximum likelihood estimator of α. Hence b^* is the maximum likelihood estimator of $E(\underline{b}|y)$, the mean of $\underline{b}$, for a given set of observations y. It is, as mentioned by Henderson, the "estimated genetic merit" used by animal breeders. They are used in order to decide which animals are best in some sense.

The maximum likelihood equations for estimating variance components from unbalanced data cannot be solved explicitly. However, Hartley and Rao [7] have developed a general set of equations from which specific estimates are obtained by iteration, involving extensive computations.

Large sample variances can be obtained from the information matrix:

$$Var(\hat{\alpha}) = (X'V^{-1}X)^{-1}$$

$$Cov(\hat{\alpha},\hat{\sigma}^2) = 0$$

$$Var(\hat{\sigma}^2) = 2\left[tr\left(V^{-1}\ \frac{\partial V}{\partial \sigma_i^2}\ V^{-1}\ \frac{\partial V}{\partial \sigma_j^2}\right)\right]^{-1} \qquad \text{for } i,\ j = 1,\ldots,q$$

4. Residual Maximum Likelihood

A disadvantage of the maximum likelihood method is the fact the ML estimators are biased downwards whereas the REML estimators are unbiased.

The difference between the two methods is analogous to the well known difference between two methods of estimating the variance σ^2 of a Normal distribution, given a random sample of n values. Both methods use the same sum-of-squares of deviations. But whereas one method equates this sum-of-squares to $(n-1)\sigma^2$, the other equates this sum to $n\sigma^2$. The estimate from $(n-1)\sigma^2$ is an unbiased estimate of σ^2, whereas the estimate from $n\sigma^2$ maximizes the likelihood of the sample and has smaller mean-square error.

Patterson and Thompson suggest replacing the vector y by a vector of error contrasts Qy, where Q is any full set of independent rows or columns of

$$I - X(X'X)^{-1}X'$$

the projection matrix on the space orthogonal to the columns of X.

Contrasts of the fixed effects are thus excluded from the likelihood function. Since contrasts between unknown fixed effects cannot provide any information on the error structure, the REML technique sets out to maximize the joint likelihood of all error contrasts which have zero expectation.

5. The Program

Most of the general statistical software packages do not have subroutines for analysing mixed models with unbalanced data. Looking at the formulas the reason is obvious. Both ML and REML solutions can be expressed in the Mixed Model Equations:

$$\begin{bmatrix} X'R^{-1}X & X'R^{-1}Z \\ Z'R^{-1}X & Z'R^{-1}Z + D^{-1} \end{bmatrix} \begin{bmatrix} \hat{\alpha} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} X'R^{-1}y \\ Z'R^{-1}y \end{bmatrix}$$

for the corresponding

Model: $\underline{y} = X\alpha + \underline{Zb} + \underline{e}$

with $E(\underline{b}) = E(\underline{e}) = 0$
$Var(\underline{b}) = E(\underline{bb}') = D$ with dimension $q \times q$
$Var(\underline{e}) = E(\underline{ee}') = R$ with dimension $N \times N$
$Var(\underline{y}) = E(\underline{yy}') = V = ZDZ' + R$

The program has been written for a special error structure, the split-plot design, with whole plots and sub-plots. Then the matrices D and R are of a simple diagonal structure:

$$R = \sigma_e^2 I_N \text{ and } D = \sigma_b^2 I_q.$$

After substituting these values in the MME's we get the following equations:

$$\begin{bmatrix} X'X & X'Z \\ Z'X & Z'Z + \lambda Iq \end{bmatrix} \begin{bmatrix} \hat{\alpha} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} X'y \\ Z'y \end{bmatrix} \quad \text{with } \lambda = \frac{\sigma_e^2}{\sigma_b^2}$$

First absorb the α equations in the b equations: $\hat{\alpha} = (X'X)^{-1}X'(y - Z\hat{b})$ and substitute this expression for $\hat{\alpha}$ in the b equations:

$$\{Z'(I_N - X(X'X)^{-1}X')Z + \lambda I_q\}\hat{b} = Z'(I_N - X(X'X)^{-1}X')y$$

Setting $M = (I_N - X(X'X)^{-1}X')$ gives

$$(Z'MZ + \lambda I_q)\hat{b} = Z'My$$

and therefore

$$\hat{B} = (Z'MZ + \lambda I_q)^{-1}Z'MY$$

These calculations involve inversions of the matrices

- $X'X$, a symmetric matrix of dimension p, p being the number of fixed effect variates
- $Z'MZ + \lambda I_q$, a symmetric matrix of dimension $q \times q$ being the number of whole plots.

These matrices may be very large.

We chose always to invert the matrix $X'X$, involving the fixed effects, and to rewrite $Z'MZ + \lambda I_q$ as follows:

$$\{(Z'Z + \lambda I_q) - Z'X(X'X)^{-1}X'Z\}^{-1}$$

because

$$M = I_q - X(X'X)^{-1}X'$$

Let $A = Z'Z + \lambda I_q$; the inverse of $Z'MZ + \lambda I_q$ may then be expressed as:

$$a^{-1} + A^{-1}Z'X(X'X - X'ZA^{-1}Z'X)^{-1}X'ZA^{-1}$$

where the diagonal matrix A is easy to invert. Furthermore we need to invert a matrix with the dimension $p \times p$ of the fixed effects.

The program reads in the dependent variate and the independent variates belonging to the fixed effect parameters. These fixed effect variates may be covariates and/or factors. These factors however may not be coded like the factors in Genstat; they must be coded as dummy variates with zeros and ones or minus-ones, one associated with each parameter of a factor.

The data must be sorted according to whole plot level. Thus it is not necessary to form dummy variates for the factor defining the whole plots. The program takes care of that. It also assumes an intercept in the model. The variates defining the fixed effects are collected in the full rank matrix X and the variates defining the whole plots are called Z. The matrices $X'X$, $X'Y$, $X'Z$, $Y'Y$, $Y'Z$ and $Z'Z$ will be formed and saved in a userfile.

Furthermore a few programs have been written to perform the mixed model analyses. They have the same input and output, but the computations are somewhat different:

- REML, inverting matrices of dimension p,
- REML, inverting matrices of dimension p and q, both using an EM-algorithm [5] and
- I-MINQUE (Rao) using the Fisher scoring method.

The output of the programs consists of:

- the variance components and their ratio at each iteration step (with a maximum of 20 steps),
- covariance matrix for the fixed effect estimates,
- covariance matrix for the estimates of the variance components,
- chi-square tests for combinations of fixed effects,
- estimates of the fixed effects and their standard errors,
- approximate tests using the t distribution: $t = a/sa$.

Satterthwaites's [12] approximation provides:

- approximate degrees of freedom,
- a two-sided P-value,
- predictions of the random effects (BLUP) and their standard errors,
- standardized whole plot errors plotted against the whole plot number and against the Normal scores.

Some problems have been encountered in programming the formulas.

One problem is presumably an error in the Genstat language. Execution of the following statements where `n1` is the number of sub-plots in a whole plot:

```
'SCALAR' yty
'FOR' n1 = hn1
   'MATRIX' y $ n1,1
      read in data into y
   'CALCULATE' yty = yty + TPDT(y; y)
'REPEAT'
```

breaks off with an integer overflow, a Fortran error, after a few whole plots with about 80 observations per whole plot. We could not find any reason for this error, but we got rid of it by declaring

```
'SCALAR' hyty
```

and by changing the `'CALCULATE'` statement into two separate `'CALCULATE'` statements:

```
'CALCULATE' hyty = TPDT(y; y)
'CALCULATE' yty = yty + hyty
```

This kind of error only appeared with large datasets (about 10,000 observations).

Furthermore the matrix XX had to be scaled before inversion, otherwise the inversion would result in an error. The reason may be the accuracy of the Vax computer. We did not have this problem when using the Cyber computer.

6. Some Concluding Remarks

The reason for developing this program is the fact that a few colleagues and I have been studying the mixed model theory. At the time we did not have any programs available. The programming helped us in understanding the theory. We planned to develop another program for analysing mixed models with three components of variance. However we did not succeed in carrying out that plan into effect.

Recently we purchased a program developed by the Scottish Agricultural Statistics Service in Edinburgh in co-operation with CSIRO Division of Mathematics and Statistics. This program can handle a wide range of models, with no restriction on the number of fixed and random effects or covariates. Up to now we have not been able to study this program and to compare the possibilities, so we have not yet decided whether we will go on programming.

7. References

[1] Airy, G.B.
On the Algebraical and Numerical Theory of Errors of Observations and the Combination of Observations,
Cambridge and London, Macmillan and Co., 1861.

[2] Cochran, W.G.
The use of the Analysis of Variance in Enumeration by Sampling,
Journal of the American Statistical Society, **34**, pp. 492-510, 1939.

[3] Crump, S.L.
The Estimation of Components of Variance in Multiple Classifications,
Ph. D. Thesis, Iowa State University, Ames, Iowa, p. 78, 1947.

[4] Crump, S.L.
The Present Status of Variance Component Analysis,
Biometrics, **7**, pp. 1-16, 1951.

[5] Dempster, A.P., Selwyn, M.R., Patel, C.M. and Roth, A.J.
Statistical and Computational Aspects of Mixed Model Analysis,
Applied Statistics, **33**, pp. 203-214, 1984.

[6] Fisher, R.A.
The Correlation between Relatives on the Supposition of Mendelian Inheritance,
Transactions of the Royal Society, Edinburgh, **52**, pp. 399-433, 1918.

[7] Hartley, H.O. and Rao, J.N.K.
Maximum Likelihood Estimation for the Mixed Analyses of Variance Model,
Biometrika, **54**, pp. 93-108, 1967.

[8] Harville, D.A.
Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems,
Journal of the American Statistical Association, **72**, pp. 320-340, 1977.

[9] Henderson, C.R.
Estimation of Variance and Covariance Components,
Biometrics, **9**, pp. 226-252, 1953.

[10] Patterson, H.D. and Thompson, R.
Recovery of Inter-block Information when Block Sizes are Unequal,
Biometrika, **58**, pp. 545-554, 1971.

[11] Patterson, H.D. and Thompson, R.
Maximum Likelihood Estimation of Components of Variance,
Proceedings of the 8th International Biometric Conference, pp. 197-207, 1975.

[12] Satterthwaite, F.E.
Biometrics Bulletin **2**, p. 110, 1946.

The Use of Genstat as a Data Organiser for Long-Season Data

J Fenlon
Glasshouse Crops Research Station
Institute for Horticultural Research
Littlehampton
Sussex
United Kingdom

1. Introduction

Experiments on protected crops (typically, but not exclusively, tomatoes and cucumbers grown under glass) have a long growing season, the harvesting of fruit possibly extending from late February through to October. During this time fruit will be picked two or three times a week. Such an experiment may involve 60 or more plots, and the yield from each plot at each harvest will be weighed and graded in up to ten fruit categories. Thus a single experiment can generate upwards of 50,000 data values.

For the statistician this generally poses two problems: firstly, how to organise the data, and secondly, how to render it in a readily interpretable form to the experimenter. This paper will primarily address the first question but will also illustrate some of the ways of tackling the second.

Only within the last couple of years has automatic data capture of yield and quality values from multiple-harvest crops become a reality. Prior to that all data were manually recorded and delays in transferring recorded data to the computer prevented all but the most rudimentary form of continuous crop monitoring. In previous seasons the analysis of fully graded data was only performed after all the data had been collected: several macros written in Genstat 4.04 allowed a flexibility in tabulation, plotting and analysis of variance. For this season a new suite of programs has been written in Genstat 5 which progressively updates file storage and presents the user with interim summaries of his experiment in tabular form. The programs use backing store to hold (a) design details, table headings, etc., (b) individual harvest data, and (c) cumulative weekly summaries for each experimental plot. The program is fully interactive and generates all filenames and sub-filenames internally so that it can be operated by a naive Genstat user.

To demonstrate the use of the programs a data set from a 1986 experiment has been used. The data set will also serve to demonstrate how other features of Genstat such as graphics, analysis of designed experiments, etc. can be linked with the use of backing store to provide more extensive analyses of the data.

2. Data Collection

Individual plots (consisting of perhaps 30 individual plants in the case of a tomato crop) are harvested and transferred to a packing shed where they are graded into three quality Classes: I, II and III. Class I fruit are further divided into six size grades (A-F) depending on fruit diameter; modern equipment is photometric, though most older grading machines rely on a system of two divergent bars slightly inclined to the horizontal, down which the fruit roll until they fall through into a collection tray. Trays are weighed on an electronic balance linked to a BBC micro-computer. A driver program written in Basic captures the data from the balance once it has been activated by the appropriate plot number. The program is menu-driven and has various facilities for data-checking and plot accounting. The data are recorded on floppy-disc and at the completion of a harvest run are transferred to a VAX 11/750. The format of a typical data file is shown in Table 1. One interesting feature is that the plots can be offered to the balance in random order as the Genstat program will do all the necessary sorting.

```
24   220786   (Harvest no. and Date)

6      0     0  471 2610  227   58   487 1196   62
61     0     0 1882 6228 1479  216   219  932  256
 .     .     .    .    .    .    .     .    .    .
 .     .     .    .    .    .    .     .    .    .
 .     .     .    .    .    .    .     .    .    .
46     0     0 2945 2400  243    0  1882 1395   92
:
       A     B    C    D    E    F
Plot                Class I              II   III    Waste
```

Table 1.
Format of typical data file

3. Data Organisation Program

A schematic diagram of the operation of the program is given in Figure 1. The program runs interactively with primary control to a terminal. Nevertheless all the operator has to do is to name the appropriate data files for the current week's update and pass control to the main (secondary channel) program, as all other files and subfiles are generated within that program. Such a procedure enables simple tabulated output to be returned direct to the screen, although a fuller, more complete summary is also output to a disc file which can be printed after the interactive run is completed.

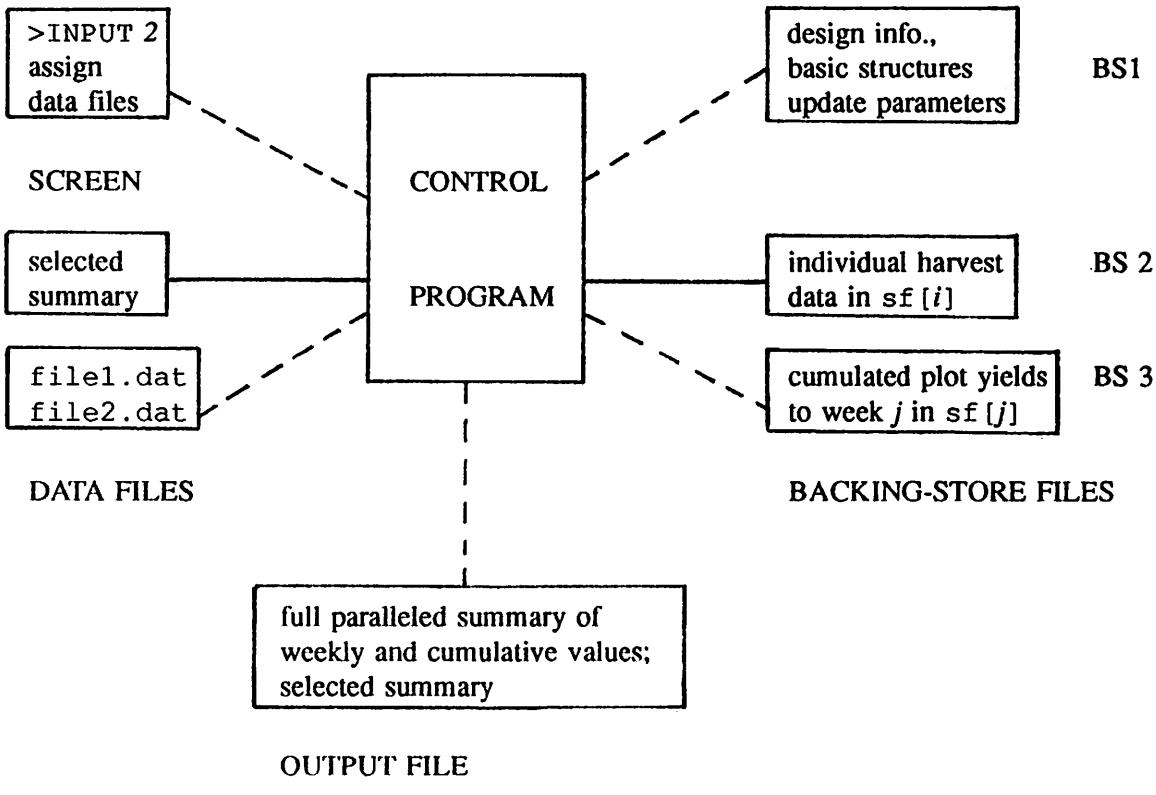

Figure 1.
Principle of update program

Once the experimental design of the trial has been determined an initial program can be set up which reads in all the basic design information for the experiment: e.g. number of plots (units), factors, table structures, conversion factors, etc.; also included are text vectors which are used for table headings, and as basic building blocks for constructing headings, files, labels, etc. All these structures are then transferred to a backing-store file which is automatically connected to the update program.

To run the update program the experimenter has to type

```
G5 IN2=UPDATE.PRG
```

and pass control to the secondary input channel once the Genstat prompt appears on the screen. The program opens the first backing-store file and calls down all the design information and basic structures. It then returns to primary input requesting the names of the data files to be used in the update: these are simply input as text strings from the terminal keyboard. The files are opened in turn and a check is made to ensure that files are attached in the correct chronological sequence. The data are read, some calculations are made and the data are sent to a second backing-store file where they are stored in a sub-file indexed by harvest number. When all the data have been read and stored, and the current harvest number read back into the primary backing-store file, these two files are closed. During the reading process the current week's data have been cumulated and these data are output to a file called `WEEK`*n*`.LIS` where *n* corresponds to the current week number. The data are tabulated by treatment but obviously the form of tabulation is the prerogative of the experimenter and statistician. A third backing-store file containing cumulative weekly data is now opened and the cumulation prior to the current update is retrieved. To this is added the current week's values and this new data set is sent back to backing-store as a new sub-file `sf[`*n*`]`, *n* again corresponding to the week number. This file is now closed, and the remainder of the program comprises calculation, organisation and tabulation of this latest data set which is output in simple tabular form to the screen with a more extensive output to the line file `WEEK`*n*`.LIS`. A skeleton listing of the program is given at the end of this article.

At any time during the experiment the cumulated data in the third backing store file is available for more thorough statistical analysis. These data can be retrieved very simply and attached to analysis-of-variance or graphics procedures, or indeed any other Genstat code that is relevant to a particular experiment. Some discussion of this is given below.

4. Example

The data presented in Table 1 is taken from a glasshouse experiment conducted at the Institute of Horticultural Research, Littlehampton in 1986. A 4×2 factorial of four summer CO_2 enrichment levels each at two ventilation temperatures was done on a tomato crop. Such environmental experiments require separate compartments of a glasshouse at the very least, and this experiment was done in eight identical stand-alone glasshouses each of 168 m^2 ground area. Such resources are very limited and it is frequently not possible to replicate except over years. In this experiment a linear response to CO_2 level was anticipated at both temperatures so that it was intended that the four degrees of freedom for deviation from linear response would be used as residual error. A fuller description of the experiment and its statistical rationale is given in Slack, Fenlon and Hand [3]. Within each house there were two blocks, each block containing a double row of 28 plants of each of four cultivars. Plots were harvested twice weekly from late April to the end of August, there being 36 harvests in all from 64 plots.

Table 2 shows an example of the typical screen output when the update program is run:

		Total marketable yield (kg/plant)				
	cvar	calypso	counter	criterium	marathon	mean
temp	CO2					
21%C	A	7.549	8.818	9.593	7.941	8.475
	S1	8.521	8.999	9.771	8.260	8.888
	S2	8.878	9.409	10.402	9.312	9.500
	S3	9.660	10.107	10.744	9.435	9.987
	Mean	8.652	9.333	10.128	8.737	9.212
26%C	A	7.500	7.882	8.413	7.333	7.782
	S1	8.352	8.765	9.199	8.908	8.006
	S2	8.726	9.265	9.549	8.715	9.064
	S3	9.471	9.638	9.808	9.112	9.507
	Mean	8.512	8.888	9.242	8.517	8.790
Mean	A	7.524	8.350	9.003	7.637	8.128
	S1	8.436	8.882	9.485	8.584	8.847
	S2	8.802	9.337	9.976	9.014	9.282
	S3	9.566	9.873	10.276	9.273	9.747
	Mean	8.582	9.111	9.685	8.627	9.001

Table 2.

Total marketable yield to the end of week 13 (29 July 1986)

The tabulated values are simple averages of the values of those plots assigned to a particular treatment. The predicted means (from the model described above) can only be derived under the ANOVA directive with the correct model specified. Nevertheless, the averages are a good indication of the progress of the experiment and free the statistician from the monitoring task which should be the responsibility of the experimenter anyway. The whole plot (i.e. glasshouse) analysis of variance for the data above is shown in Table 3 in which also are presented the means from the linear regression model.

```
***** Analysis of variance *****
Variate: wk13

Source of variation          d.f.       s.s.       m.s.     v.r.  F pr.

house stratum
CO2                             1   22.38622   22.38622   82.16  <.001
  Lin                           1   22.38622   22.38622   82.16  <.001
  Assigned to error             2    0.33511    0.16755    0.61
temp                            1    2.85864    2.85864   10.49  0.032
CO2.temp                        1    0.01650    0.01650    0.06  0.818
  Lin.temp                      1    0.01650    0.01650    0.06  0.818
  Assigned to error             2    0.75483    0.37741    1.39
Residual                        4    1.08994    0.27248

Total                           8   26.35130


***** Tables of means *****

Variate: wk13

Grand mean  9.001

      CO2         A        S1        S2        S3
            8.208     8.737     9.266     9.795

     temp      21%C      26%C
            9.212     8.790

      CO2      temp      21%C      26%C
        A              8.441     7.975
       S1              8.955     8.518
       S2              9.470     9.061
       S3              9.984     9.605


*** Standard errors of differences of means ***

Table                    CO2          temp          CO2
                                                    temp

                                  (smoothed)
rep.                      16            32             8
s.e.d.                0.1846        0.1305        0.2610
```

Table 3.
Analysis-of-Variance Table and Table of Means

5. Some statistical problems

At the outset it was suggested that one of the statistician's problems is to summarise the mass of data from such experiments into a readily interpretable form. Inevitably the experimenter will be interested in certain obvious variates such as the total marketable yield over the season, Class I yield and total marketable value, but he will also want to know how these variates and some derivatives change over the season. Analysis of variance is a useful tool in the horticultural statistician's armoury but it must be appreciated that these data are repeated measurements on the same groups of plants and are multivariate in the sense of compositional data. Consequently univariate analysis of variance must be interpreted with care. The type of approach advocated by Rowell and Walters [2] has been tried on these data, but fluctuations in light and temperature generate considerable noise in the data which are not best catered for in polynomial smoothing.

An unpublished manuscript by my colleague Rodney Edmondson [1] looks at the regression of cumulative treatment means against cumulative overall means and this appears to be a promising approach.

The concept of a missing value takes on a different meaning in this type of data: during the course of an experiment individual plants may die from a variety of causes, they may cease to produce fruit whilst still remaining vegetative, or they may fail to produce fruit for a period; cases of catastrophic failure of a complete plot are rare, though there have been incidents of collapse of a support wire resulting in considerable stem breakage. The loss of a few plants towards the end of an experiment is to be expected but it needs to be verified that plant loss is not treatment dependent. Although failure of a complete plot is rare, when it does occur it creates considerable problems because of the highly structured nature of many designs.

The compositional nature of much of the yield data from multiple-harvest crops has not been fully explored although increasing demands for fruit quality mean that its significance will increase. The fact that Class I fruit are classified according to size means that under suitable transformation data can be analysed as grouped Normal (or possibly some other distribution) data as described by Thompson and Baker [4]. Certainly in the experiment described in the example the size of fruit was found to change not only with treatment (i.e. CO_2 level and temperature) but also with time. A more complex compositional problem relates to certain experiments where it is not possible to grade all fruit at all harvests. Generally only one replicate will be graded at any harvest and the statistical problems relate to determining a balancing design for which replicate to weigh at which harvest, and whether it is possible to use this partial information to estimate the composition of ungraded fruit.

6. Outline of update program

```
JOB 'Update program'
```

Declare various scalars

```
OPEN NAME='BS1.DAT','BS2.DAT','BS3.DAT'; CHANNEL=1,2,3; FILE=backing
RETRIEVE [CHANNEL=1; SUBFILE=sf[1]] plot, factors
 &    [SUBFILE=sf[2]] declared tables
 &    [SUBFILE=sf[3]] text vectors
 &    [SUBFILE=sf[4]] cumharv  "update parameter"
UNITS [64] plot
PRINT 'Please enter the names of the 2 files for this week'
TEXT [NVALUES=1] file[1,2]
READ [CHANNEL=1] file[]
INPUT 2
OPEN NAME=file[]; CHANNEL=3,4; FILE=input; WIDTH=132
READ [CHANNEL=3; END=*] harv,date
```

Calculate test for correct sequence

```
READ [CHANNEL=3] plot,v[1...6,8,9,11,15]
CLOSE 3
```

Calculate variates v[7,10,12]

```
STORE [CHANNEL=2; SUBFILE=sf[harv]] harv,date,plot,v[2...12]
CALCULATE w[2...12] = v[2...12]
```

Repeat for channel 4

```
STORE [CHANNEL=1; SUBFILE=sf[4]; METHOD=overwrite] cumharv
CLOSE 1,2; backing
```

Calculate further variates required in summaries

```
"Derive name of output file and create it"
CALCULATE week = harv/2
CONCATENATE [outfile] f[1],k[week],f[2]
OPEN NAME=outfile; CHANNEL=2; FILE=output
```

Tabulate and print summaries to channel 2

```
CALCULATE pweek = week-1
RETRIEVE [CHANNEL=3; SUBFILE=sf[pweek]] pweek,pdate,plot,z[1...18]; \
   week,date,plot,x[1...18]
```

Calculate new cumulative totals

```
STORE [CHANNEL=3; SUBFILE=sf[week]] week,date,plot,x[1...18]
CLOSE 3; backing
```

Further tabulate and print to channels 1 and 2

```
CLOSE 2; output
STOP
```

7. Conclusions

In Genstat 4.04, macros were used to analyse this type of experiment, though such analyses only took place after all the data had been collected. The use of procedures was considered for Genstat 5 but the individual nature of each experiment suggested otherwise. Nevertheless the basic update program outlined below will be fairly consistent across all trials. This is the first year that such an interactive approach to data analysis has been tried and it has been very successful; the limited coding for Genstat 5 makes the problem look almost trivial, but such a program would have been almost impossible in Genstat 4.04. Immediate access to updated data has proved beneficial to the experimenter, and indeed has highlighted several shortcomings in the grading and recording of data which had not emerged previously.

8. References

[1] Edmondson, R.N.
Analysis of repeated observations data by regression on the response means.
Unpublished manuscript, 1986.

[2] Rowell, J.G. and Walters, D.E.
Analysing data with repeated observations on each experimental unit.
Journal of Agricultural Science, Cambridge 87, pp. 423-432, 1976.

[3] Slack, G., Fenlons, J.S. and Hand, D.W.
The effects of summer CO_2 enrichment and ventilation temperatures on yield, quality and value of glasshouse tomatoes.
To be published in *Journal of Horticultural Science*, **63**, 1986.

[4] Thompson, R. and Baker, R.J.
Composite link functions in generalised linear models.
Applied Statistics, **30**, pp. 125-131, 1981.

Prediction on the Scale of the Linear Predictor for Generalized Linear Models

M S Ridout
Institute of Horticultural Research
East Malling
Maidstone
Kent
United Kingdom ME19 6BJ

1. Introduction

When the explanatory variables in a regression model are quantitative (that is, variates), the regression coefficients usually provide a satisfactory summary of the fitted model. But when the model involves qualitative explanatory variables (factors) it is usual to present tables of means or adjusted means to summarize their effects.

The PREDICT directive enables such tables to be formed. PREDICT operates by first forming a full table of predictions, classified by every factor in the model, and then reducing this table by averaging over levels of factors which are not required in the final table of predictions. The averaging may be simple or weighted and there is considerable flexibility in the type of weighting that can be specified. If the model includes variates, predictions are formed at their mean value by default although it is possible to specify an alternative value, or indeed a set of values which then appear as an additional classifying factor in the final table of predictions.

For generalized linear models (GLMs) the situation is complicated slightly by the existence of two scales – the scale on which the response variable is measured and the scale on which the linear model structure is assumed to exist (the linear-predictor scale).

PREDICT produces predictions on the response variable scale. The full table of predictions is first formed on the linear-predictor scale and these predictions are then transformed to the scale of the response variable before the averaging over unwanted factors takes place.

In factorial experiments the prediction for a particular treatment combination may be regarded as an estimate of the average response that would have been obtained if every experimental unit had received that treatment combination [2]. In this sense the predictions are analogous to the treatment combination means from ANOVA. But designed experiments are comparative and a more important property required of the treatment means is that any comparison of these means provides an estimate of the corresponding comparison of the 'true' means. To produce quantities with the same property for a GLM, predictions must be made on the linear predictor scale since this is the scale on which the model is assumed additive.

There is a clear analogy with the analysis of transformed data when the usual practice is to present treatment means and standard errors on the transformed scale, for the purposes of inference. In addition back-transformed means are usually presented to assist in interpretation. With GLMs one could either back-transform the predictions made on the linear-predictor scale, or simply use PREDICT. Analogous possibilities exist in the analysis of transformed data and their implications are discussed in [3].

It is clear from this discussion that it will often be useful to be able to form predictions on the linear-predictor scale when analysing data with a GLM. The purpose of this article is to point out how this may be achieved and to provide three procedures to assist in the process.

2. Method

It is well known that the maximum-likelihood estimates of the parameters (excluding the dispersion parameter) of a GLM may be obtained by performing a sequence of weighted regressions involving an adjusted response variate which, together with the weights, changes with each successive fit. Indeed, this algorithm, the method of iterative weighted least squares, is that used by Genstat.

This is the key to persuading PREDICT to work on the scale of the linear predictor. Essentially all that is necessary is to calculate the adjusted response variate and the weights from the fitted GLM and to refit the model with DISTRIBUTION=Normal and LINK=identity, thereby performing a further iteration of the fitting process. Since Genstat has already decided that the fitting process has converged there will be little change in the parameter estimates. Similarly the covariance matrix of parameter estimates will be essentially unchanged provided the DISPERSION parameter is the same as was used in the GLM fit. The newly fitted model may be termed the 'equivalent least-squares fit'.

If PREDICT is used now, the full set of predictions on the linear-predictor scale will again be produced, but since LINK=identity no transformation occurs and the averaging over unwanted factors takes place on this scale.

There is, however, a complication with weighting. For an unweighted regression, the default method of averaging over unwanted factors is to form weighted averages, the weights for different levels of a factor being determined by the overall replication of the factor levels in the whole set of data. This corresponds to the ADJUSTMENT=marginal option setting. If factor levels are equally replicated in the data it is the same as results from the alternative ADJUSTMENT=equal setting. But the equivalent least-squares fit is a weighted regression, and the ADJUSTMENT=marginal setting then bases the weights for averaging on the sum of the regression weights for different factor levels.

Averaging using the regression weights is not a sensible thing to do so the ADJUSTMENT=marginal setting cannot be used. This leaves two possibilities – setting ADJUSTMENT=equal or supplying an explicit set of weights using the WEIGHTS option. Fortunately ADJUSTMENT=equal is often what is required, particularly in the analysis of designed experiments. But it is important to remember to set this option, even when factor levels are equally replicated.

3. Procedures

Three procedures are provided – EQLSFIT performs an equivalent least-squares fit and BTRAN 'back-transforms' a set of predictions on the linear-predictor scale to obtain a set of predictions on the response variable scale. Finally SED is a short procedure which forms a matrix of standard errors of differences for a set of estimates from their covariance matrix.

3.1. EQLSFIT

The procedure EQLSFIT performs an equivalent least-squares fit for the most recently fitted model. No output is produced. EQLSFIT has options DISTRIBUTION, LINK, EXPONENT, DISPERSION, WEIGHTS, OFFSET and parameters Y and NBINOMIAL. These are defined in exactly the same way as the corresponding options and parameters of the MODEL directive and should be set precisely as in the MODEL statement that defined the GLM. EQLSFIT reports an error if DISTRIBUTION=Normal and LINK=identity since there is then no distinction between the 'equivalent least-squares fit' and the fit already obtained. Apart from this, EQLSFIT does not check that sensible parameters and options have been supplied.

The most important thing to note about EQLSFIT is that it redefines the current regression model. Thus if further GLM fitting is to take place after using EQLSFIT, a fresh MODEL statement is required to respecify the GLM in place of the DISTRIBUTION=Normal, LINK=identity settings used in EQLSFIT.

EQLSFIT uses an auxiliary procedure CONVERT which converts the DISTRIBUTION and LINK settings to numeric codes NDIST and NLINK. The coding used is shown in the listing of CONVERT. The efficiency of EQLSFIT may be improved slightly by supplying NDIST and NLINK directly as options (with default values 4 and 3 respectively) in place of DISTRIBUTION and LINK, and deleting the line:

```
CONVERT DISTRIBUTION; LINK; NDIST; NLINK
```

One minor point is that the DISTRIBUTION and LINK options are checked less carefully than Genstat itself checks them in the MODEL statement. For example the six possible settings of DISTRIBUTION each begin with a different letter so CONVERT checks only the first letter. p and pois are both interpreted as poisson but so too are pareto and pearsontype2. But this is of no real importance because any setting which is acceptable for the MODEL statement is acceptable for EQLSFIT and will be interpreted in the same way.

3.2. BTRAN

The procedure BTRAN transforms predictions produced on the linear-predictor scale back to the response variable scale and displays the results. The specification of BTRAN is as follows:

Options

OVERWRITE = *string*		Whether to overwrite the predictions supplied to the procedure with the back-transformed values; standard errors and the covariance matrix are also overwritten if these were supplied (yes, no); (default n)
DISTRIBUTION = *string*	)	
LINK = *string*	)	Set as for EQLSFIT
EXPONENT = *scalar*	)	

Parameters

PREDICTIONS = *table*	Predictions on linear-predictor scale saved using PREDICTIONS option of PREDICT
SE = *table*	Standard errors of predictions on linear-predictor scale saved using SE option of PREDICT

The back-transformation is determined solely by the link function – the DISTRIBUTION option need not be set unless LINK=canonical. Based on the link function, the predictions are back-transformed and printed. Approximate standard errors are also printed if the SE parameter is set. These standard errors are the usual first-order approximations (see, for example, Kendall and Stuart, [1], section 10.6). By default the procedure exits with all parameters unchanged. However, if the OVERWRITE option is set to yes, PREDICTIONS contains the back-transformed predictions on exit and if the SE parameter was supplied it is overwritten with the approximate standard errors of the back-transformed predictions.

3.3. SED

The procedure SED may be used to obtain a matrix of standard errors of differences of predictions from their covariance matrix (saved with the VCOVARIANCE option of PREDICT). The s.e.d.s are calculated as a symmetric matrix whose $(i,j)^{th}$ element is the s.e.d. between the i^{th} and j^{th} predictions. The diagonal elements are set to missing values. This matrix is automatically printed by the procedure and may optionally be saved for future use. The procedure has no options and just two parameters

VCOVARIANCE = *symmetric matrix*	Covariance matrix of predictions
SED = *symmetric matrix*	To save s.e.d.s

4. An Example

Snedecor and Cochran ([4], p.289, Example 15.11.1) give wireworm counts from a field experiment involving five treatments (K,M,N,O,P) arranged in a single 5×5 Latin square. The treatments involved soil fumigation in the previous year but no details are given. The counts are quite small and Snedecor and Cochran consider an analysis of variance after the transformation

$$\sqrt{\text{Count}+1}$$

Here we consider instead an analysis based on a Poisson model using the canonical (log) link function.

The first step is to fit the GLM

```
UNITS [25]
FACTOR [LEVELS=5; VALUES=5(1...5)] ROWS
  &    [VALUES=(1...5)5] COLUMNS
  &    [LABELS=!T(K,M,N,O,P)] TREATMNT
READ   [PRINT=data] TREATMNT,WIREWORMS; FREPRESENTATION=labels

P  3    O  2    N  5    K  1    M  4
M  6    K  0    O  6    N  4    P  4
O  4    M  9    K  1    P  6    N  5
N 17    P  8    M  8    O  9    K  0
K  4    N  4    P  2    M  4    O  8
:

MODEL [DISTRIBUTION=poisson] WIREWORMS
TERMS ROWS+COLUMNS+TREATMNT
FIT [PRINT=*] ROWS
ADD [PRINT=*] COLUMNS
ADD [PRINT=model,accumulated,estimates] TREATMNT
```

giving the following analysis-of-deviance table:

```
*** Accumulated analysis of deviance ***

Change                                      mean   mean deviance
             d.f.      deviance      deviance         ratio
+ ROWS          4        15.695         3.924          2.41
+ COLUMNS       4         4.158         1.040          0.64
+ TREATMNT      4        25.193         6.298          3.87
Residual       12        19.508         1.626
Total          24        64.555         2.690
```

The magnitude of the residual deviance suggests the possibility of overdispersion. To get a better idea of this we calculate the Pearson statistic:

```
RKEEP FITTEDVALUES=FV; DF=RESDF
CALC PEARSON = SUM( (WIREWORMS-FV)**2 / FV )
  &  PROB = 1-CHISQ(PEARSON; RESDF)
PRINT PEARSON,RESDF,PROB; DECIMALS=2,0,3

   PEARSON        RESDF         PROB
     18.01           12        0.115
```

In this example, without the benefit of experience of similar data, it is difficult to decide whether an adjustment for overdispersion is called for. The Pearson statistic is not significantly large, even at the 10% level. On the other hand if the usual adjustment for overdispersion were made, standard errors would be increased by 22.5% ($\sqrt{18.01/12} = 1.225$). Fortunately the decision is not crucial for the present discussion and for simplicity we continue to assume that there is no overdispersion.

Parameter estimates are as follows:

```
*** Estimates of regression coefficients ***

                              estimate        s.e.           t
Constant                        -0.046       0.507       -0.09
ROWS 2                           0.267       0.347        0.77
ROWS 3                           0.487       0.331        1.47
ROWS 4                           1.012       0.308        3.28
ROWS 5                           0.442       0.340        1.30
COLUMNS 2                       -0.355       0.283       -1.25
COLUMNS 3                       -0.398       0.286       -1.39
COLUMNS 4                       -0.368       0.276       -1.33
COLUMNS 5                       -0.305       0.289       -1.05
TREATMNT M                       1.671       0.450        3.71
TREATMNT N                       1.712       0.444        3.86
TREATMNT O                       1.580       0.452        3.50
TREATMNT P                       1.380       0.462        2.98
* MESSAGE: s.e.s are based on dispersion parameter with value 1
```

The PREDICT directive may be used to obtain predictions for TREATMNT:

```
   PREDICT [PRINT=description,predictions,se] TREATMNT

*** Predictions from regression model ***

The predictions have been standardized by averaging
fitted values over the levels of some factors:

        Factor   Weighting policy   Status of weights
       COLUMNS   Marginal weights   Constant over levels of other factors
          ROWS   Marginal weights   Constant over levels of other factors

     Table contains predictions followed by standard errors

 Response variate: WIREWORM

    TREATMNT
           K        1.196       0.490
           M        6.360       1.167
           N        6.629       1.147
           O        5.809       1.098
           P        4.754       1.007

* Standard errors are approximate, since model is not linear
```

These predictions have been obtained as follows.

(1) Calculating predicted values of the linear predictor for all combinations of levels of TREATMNT, ROWS and COLUMNS (a total of 125 predicted values). These values are obtained directly from the parameter estimates. For example, the predicted value for treatment M in row 2, column 3 is

Constant + ROWS 2 + COLUMNS 3 + TREATMNT M = 1.494

(2) Transforming these values back to the scale of the response variable, using the inverse of the link function. For example, the predicted wireworm count for treatment M in row 2, column 3 is

$exp(1.494) = 4.45$

(3) Averaging the 25 predicted counts for each level of TREATMNT.

An equivalent least-squares fit may be obtained using:

```
EQLSFIT [DISTRIBUTION=poisson] WIREWORMS
```

Use of the RDISPLAY directive will confirm that all parameter estimates and standard errors are, to three decimal places, identical to those previously obtained, except for one standard error which differs by one digit in the third decimal place.

Use of the PREDICT directive with ADJUSTMENT=equal now gives predictions on the scale of the linear predictor. The predictions and their SEs and covariance matrix are saved for subsequent use.

```
PREDICT [PRINT=description,predictions,se; ADJUSTMENT=equal;\
         PREDICTIONS=P; SE=S; VCOVARIANCE=V] TREATMENT

*** Predictions from regression model ***

The predictions have been standardized by averaging
fitted values over the levels of some factors:

        Factor   Weighting policy   Status of weights
       COLUMNS      Equal weights   Constant over levels of other factors
          ROWS      Equal weights   Constant over levels of other factors

     Table contains predictions followed by standard errors

Response variate: Z
                      P           S
    TREATMNT
           K      0.110       0.412
           M      1.781       0.184
           N      1.822       0.181
           O      1.690       0.191
           P      1.490       0.212
```

Note that these predictions are directly related to the parameter estimates. For example, the difference between the predictions for treatments M and K is 1.781-0.110 = 1.671. As expected, this is precisely the estimate of the TREATMNT M parameter in the fitted model.

The covariance matrix of these predictions is:

```
PRINT V

             V

1      0.16977
2      0.00030     0.03370
3      0.00253     0.00061     0.03285
4      0.00091     0.00029     0.00176     0.03665
5      0.00023     0.00038     0.00075     0.00001     0.04482

               1           2           3           4           5
```

Note that the off-diagonal elements of the covariance matrix are small compared with the diagonal elements. Thus the predictions are only slightly correlated in this example.

The procedure SED may be used to obtain standard errors of differences of these predictions.

```
SED V

*** Matrix of S.E.D.s derived from covariance matrix V ***

1           *
2      0.4504           *
3      0.4445      0.2556           *
4      0.4523      0.2641      0.2569           *
5      0.4627      0.2789      0.2760      0.2854           *

               1           2           3           4           5
```

Thus an s.e.d. of about 0.45 is appropriate for comparing treatment K with any of the other treatments. The appropriate s.e.d. for comparing any other two treatments is about 0.27. These standard errors would have to be multiplied by 1.225 if an adjustment for overdispersion were thought necessary (of course this can be done automatically by setting DISPERSION=1.5 in EQLSFIT). Again we can note that the s.e.d. for comparing treatment M with treatment K is precisely the standard error of the TREATMNT M parameter in the fitted model.

If the squareroot link function is used instead of the log link the situation is rather different. The predictions on the linear-predictor scale are then uncorrelated and have equal variances. There is therfore a single s.e.d. for comparing any two treatments, as in analysis of variance. This arises because the square-root function is the (first order) variance-stabilising transformation for the Poisson distribution.

Thus the squareroot link offers some simplicity in terms of presentation of standard errors. The log link, on the other hand, has advantages in terms of interpretation since additive effects on the log scale imply simply multiplicative effects on the original scale of measurement.

Finally, back-transformed predictions and approximate standard errors may be obtained using the procedure BTRAN.

```
BTRAN [DISTRIBUTION=Poisson] P; S

*** BACK-TRANSFORMED PREDICTIONS ***

Link function:  Log

Table gives back-transformed predictions followed by approximate SEs

 TREATMNT
        K       1.116       0.4599
        M       5.934       1.0894
        N       6.185       1.1210
        O       5.420       1.0376
        P       4.436       0.9391
```

These predictions may be compared with the predicted counts obtained earlier from direct use of the PREDICT command.

5. References

[1] Kendall, M.G. and Stuart, A.
The Advanced Theory of Statistics.
4th Edition. Charles Griffin and Co. Ltd., London and High Wycombe, 1977.

[2] Lane, P.W. and Nelder, J.A.
Analysis of covariance and standardization as instances of prediction.
Biometrics, **38**, pp. 613-622, 1982.

[3] Morris, G.E.L.
The presentation of treatment responses from block experiments after analysis of variance of transformed data.
Ann. Appl. Biol., **107**, pp. 571-580, 1985.

[4] Snedecor, G.W. and Cochran, W.G.
Statistical Methods.
7th Edition, Iowa State University Press, 1980.

6. Appendix

Procedure EQLSFIT

```
PROCEDURE 'EQLSFIT'
PARAMETER 'Y','NBIN'
OPTION 'DISTRIBUTION','LINK','EXPONENT','DISPERSION','WEIGHTS','OFFSET'; \
   MODE=2(T),4(P); DEFAULT='N','C',-2,-99,*,*
   "
    Convert DISTRIBUTION and LINK to numerical values NDIST and NLINK.
    Refuse to deal with DISTRIBUTION=Normal, LINK=identity
   "
   CONVERT DISTRIBUTION; LINK; NDIST; NLINK
   IF NDIST.EQ.4 .AND. NLINK.EQ.3
      PRINT '*** FAULT in procedure EQLSFIT ***'
      PRINT [SQUASH=yes] 'EQLSFIT cannot be used with DIST=Normal and', \
         'LINK=Identity'; FIELDWIDTH=1
      SKIP [FILE=output] 2
      EXIT [CONTROL=proc]
   ENDIF
   "
    Extract required quantities from the most recently fitted regression
    model and set PHI to be the dispersion parameter. PHI is determined
    from the DISPERSION option as for the MODEL directive, i.e.
     If DISPERSION=*, PHI=residual mean deviance from previous fit
     If DISPERSION=d, PHI=d
     If DISPERSION is unset, PHI=1 for DIST=M,B or P. For other distributions
                             PHI is set as for DISPERSION=*
   "
   RKEEP Y=Y; FITTED=FV; DEVIANCE=DEV; DF=DF; TERMS=MODEL; \
      ITERATIVEWEIGHTS=W; LINEARPRED=LP
   IF UNSET(DISPERSION)
      CALC PHI = DEV/DF
   ELSE
      CALC PHI = DISPERSION * (DISPERSION.NE.-99) + \
                 (NDIST.LE.3) * (DISPERSION.EQ.-99)
        &  PHI = PHI * (PHI.NE.0) + DEV/DF * (PHI.EQ.0)
   ENDIF
   "
    Set GDASH to be the derivative of the link function evaluated at
    the fitted values
   "
   CASE NLINK
      CALC GDASH = 1/FV    "Log"
```

```
   OR
      CALC GDASH = NBIN/(FV*(NBIN-FV))  "Logit"
   OR
      CALC GDASH = 1  "Identity"
   OR
      CALC GDASH = -1/(FV*FV)  "Reciprocal"
   OR
      CALC GDASH = EXPONENT*FV**(EXPONENT-1)  "Power"
   OR
      CALC GDASH = 2.506628*EXP(0.5*NED(FV/NBIN)**2)/NBIN  "Probit"
   OR
      CALC GDASH = 0.5/SQRT(FV)  "Squareroot"
   OR
      CALC GDASH = 1/((NBIN-FV)*LOG(NBIN/(NBIN-FV)))  "Comploglog"
   ENDCASE
   "
    Form adjusted response variable Z and fit the equivalent normal model
   "
   CALC Z = LP+(Y-FV)*GDASH
   IF .NOT. UNSET(WEIGHTS)
      CALC W = W*WEIGHTS
   ENDIF
   IF UNSET(OFFSET)
      MODEL [DISTRIBUTION=Normal; DISPERSION=PHI; WEIGHT=W] Z
   ELSE
      MODEL [DISTRIBUTION=Normal; DISPERSION=PHI; WEIGHT=W; OFFSET=OFFSET] Z
   ENDIF
   FIT [PRINT=*] #MODEL
ENDPROC
```

Procedure CONVERT

```
PROCEDURE 'CONVERT'
PARAMETER 'DISTRIBUTION','LINK','NDIST','NLINK'
   "
    Converts strings DISTRIBUTION and LINK, set as for the MODEL directive,
    to numeric codes NDIST and NLINK defined as follows

      DISTRIBUTION   NDIST                    LINK      NLINK
      --------------------                  -----------------
       Multinomial      1                      Log         1
           Poisson      2                    Logit         2
          Binomial      3                 Identity         3
            Normal      4               Reciprocal         4
             Gamma      5                    Power         5
     Inversenormal      6                   Probit         6
                                         Squareroot         7
                                Complementaryloglog         8
   "
CONCATENATE [DIST1] DISTRIBUTION; WIDTH=1
& [LINK1] LINK; WIDTH=1
CALC NDIST = SUM((!T(M,m,P,p,B,b,N,n,G,g,I,i) .IN. DIST1) * !(2(1...6)))
& NLINK = SUM((!T(L,l,I,i,R,r,P,p,S,s,C,c) .IN. LINK1) * !(2(1,3,4,5,7,8)))
& NLINK = NLINK + (NLINK .IN. !(1,5) .AND. NDIST .EQ. 3)
ENDPROC
```

Procedure BTRAN

```
PROCEDURE 'BTRAN'
PARAMETER 'PREDICTIONS','SE'
OPTION 'OVERWRITE','DISTRIBUTION','LINK','EXPONENT'; MODE=3(T),P; \
   DEFAULT='N','N','C',-2
   "
    Convert DISTRIBUTION and LINK to numerical values NDIST and NLINK.
   "
   CONVERT DISTRIBUTION; LINK; NDIST; NLINK
   "
    Using NLINK, set P to be back transformed predictions and DERIV to be
    the derivative of the inverse link function evaluated at PREDICTIONS
   "
   CASE NLINK
      CALC P = EXP(PREDICTIONS)
      & DERIV = P
   OR
      CALC P = EXP(PREDICTIONS)/(1+EXP(PREDICTIONS))
      & DERIV = P*(1-P)
   OR
      CALC P = PREDICTIONS
      & DERIV = 1
   OR
      CALC P = 1/PREDICTIONS
      & DERIV = -P*P
   OR
      CALC P = PREDICTIONS**(1/EXPONENT)
      & DERIV = EXPONENT*P/PREDICTIONS
   OR
      CALC P = NORMAL(PREDICTIONS)
      & DERIV = EXP(-0.5*PREDICTIONS**2)/2.506628
   OR
      CALC P = PREDICTIONS*PREDICTIONS
      & DERIV = 2*PREDICTIONS
   OR
      CALC P = 1-EXP(-EXP(PREDICTIONS))
      & DERIV = (1-P)*EXP(PREDICTIONS)
   ENDCASE
   "
    Calculate new SEs from old SEs if these were supplied
   "
   IF .NOT. UNSET(SE)
      CALC S = ABS(DERIV)*SE
   ENDIF
   "
    Print back transformed predictions and their SEs if available
   "
   TEXT T[1...8]; VALUES='Log','Logit','Identity','Reciprocal','Power', \
      'Probit','SquareRoot','Complementary Log Log'
   PRINT '*** BACK-TRANSFORMED PREDICTIONS ***'
   PRINT [IPRINT=*; SQUASH=yes] 'Link function:',T[NLINK]; FIELDWIDTH=1
```

```
   IF UNSET(SE)
      PRINT [IPRINT=*] P
   ELSE
      SKIP [FILE=output] 1
    PRINT [SQUASH=y] 'Table gives back-transformed predictions followed ', \
        'by approximate SEs'; FIELDWIDTH=1
      PRINT [IPRINT=*] P,S
   ENDIF
   "
    If OVERWRITE=yes overwrite PREDICTIONS and (possibly) SE
   "
   CONCATENATE OVERWRITE; WIDTH=1
   EXIT [CONTROL=proc] OVERWRITE.NES.'Y' .AND. OVERWRITE.NES.'y'
   CALC PREDICTIONS = P
   IF .NOT. UNSET(SE)
      CALC SE = S
   ENDIF
ENDPROC
```

Procedure SED

```
PROCEDURE 'SED'
PARAMETER 'VCOVARIANCE','SEDMATRIX'
   "
    Forms and prints a matrix of s.e.d.s from a covariance matrix
   "
   DIAGONAL D
   CALC D = VCOVARIANCE
   & SED = VCOVARIANCE.EQ.VCOVARIANCE
   & SED = SQRT(PRODUCT(D; SED)+PRODUCT(SED; D)-2*VCOVARIANCE)
   & SED = SED*(SED/SED)
   PRINT '*** Matrix of S.E.D.s derived from covariance matrix', \
      !P(VCOVARIANCE),'***'; FIELDWIDTH=1
   & [IPRINT=*] SED
   "
    Save s.e.d.s if required
   "
   IF .NOT.UNSET(SEDMATRIX)
      CALC SEDMATRIX = SED
   ENDIF
ENDPROC
```

Genstat 5 Procedure Library: Instructions for Authors

R W Payne
P G N Digby
Statistics Department
Rothamsted Experimental Station
Harpenden
Hertforshire
United Kingdom AL5 2JQ

The Genstat 5 Procedure Library is controlled by an Editorial Committee whose current members are listed at the end of this note.

Procedures submitted to the Library are checked to ensure that they are useful; also and more importantly, that they are reliable. To save time and effort in refereeing we would ask you to ensure that you supply all the information listed in Section 1 below, and that you try to follow the guidelines of style given in Sections 3 and 4. Remember that the easier a procedure is to assess, the more likely the referee is to be sympathetic and the faster you are likely to receive a report!

The procedures in the Library are divided into modules according to the type of facility offered. The Secretary of the Editorial Committee can provide an up-to-date list of modules and their contents, in order to save duplication of effort. We also keep a list of facilities for which we would like to see procedures written, if you need any suggestions.

1. Submissions

Procedures should be sent to the Secretary of the Editorial Committee and should consist of the following.

(a) The code of the procedure following as far as possible the style described in Section 4.

(b) Information for the Library on-line help (produced by procedure LIBHELP). This includes sections of description which are stored in Genstat 5 text structures for printing by LIBHELP when requested by users. There should also be a simple example, to illustrate how to use the procedure; output from this should not be too voluminous and should be interspersed with captions to explain what is going on. The input statements of the example are stored in a text structure which can be accessed by a procedure called LIBEXAMPLE and then run as a macro (that is by use of the special symbols ##). Further details are given in Section 8.

(c) Test programs for the referees to use to check that the procedure produces correct and accurate results. These should be well commented and should check every aspect of the procedure. Please also supply evidence that the output is correct: for example by showing that the results agree with examples taken from books or papers, or by including simple examples that can be checked by hand.

(d) Any additional information to assist the editor and referees, for example references to books or papers to explain the method, flowcharts (if appropriate), comparisons with existing procedures, and so on.

Items (a)–(c) should be in a computer readable form. The preferred method is to use a 5¼" IBM PC standard format floppy disc. Magnetic tape or floppy discs for other types of computer may also be acceptable, and communication is possible over some computer networks, but please discuss this with the Secretary first. Whichever method is used, you should also send a written version through the post as a check.

2. Arguments

Information is transferred to and from a procedure by means of the arguments specified by its options and parameters. Please try to avoid making the user specify information that can easily be determined within the procedure itself. For example, there is no need to require the length of a

vector to be specified as well as the vector itself; this can easily be determined by using the NVALUES function in a CALCULATE statement. The GETTATRIBUTE directive can also be useful in this context (see Section 6.3.1 of the Genstat 5 Reference Manual).

Where some of the arguments are vectors, you should state what will happen if they are restricted.

3. Syntax

The rules of syntax for the use of a procedure are exactly the same as those of the standard Genstat 5 directives. Moreover, procedures in the Library are accessed automatically so that a user need not know whether a particular statement uses a procedure or a directive. For efficiency, procedures that are very popular may be supplied as directives in future releases of Genstat 5 (and the converse may also be true). Consequently it is important that the names of Library procedures and their options and parameters follow the same conventions of type, ordering and vocabulary as have been set for the definition of the directives.

(a) When deciding on a name for each option or parameter, check whether there is a suitable name already in use (in directives or other procedures in the Library) before you invent another one; also check that its existing purpose is similar to that required in your procedure. For example, if you need an option to control the maximum number of iterations for an algorithm, you should use the name MAXCYCLE (as in directives RCYCLE, ANOVA and ESTIMATE), and not some new name. On the other hand, you should not use MAXCYCLE for a different purpose – for example to define an attribute of a cyclic design. Alternatively, you may be able to use one of the standard prefixes; see (g) below.

(b) The first four characters of each option name must be different from the first four characters of the name of any other option of the procedure; similarly, the first four characters of each parameter name must be distinct from those of the other parameters. Thus for example you should not have options PRINT and PRINCIPAL in the same procedure. Also the first eight letters of the name of the procedure must not be the same as the first eight letters of any directive or any other Library procedure. (See Section 1.6.1 of the Genstat 5 Reference Manual.)

(c) If you do use an existing name for an option or parameter, ensure that it has the same mode for its setting (identifier, string, expression or formula) as in its use elsewhere. For example, if you have an option called PRINT, its setting should be one or more strings, as for example in directives ANOVA, CVA, FIT and TABULATE. (The mode is specified by the MODE parameter of the OPTION and PARAMETER directives; see Section 6.3.1 of the Genstat 5 Reference Manual.)

(d) Most option and parameter settings should be of mode identifier. Strings should occur only as the settings of VALUES options for the definition of texts, or for settings where the aim is to select one, or more, values from a predetermined set. Number lists should occur only with VALUES options for the definition of numerical structures. (See Section 1.6.4 of the Genstat 5 Reference Manual.)

(e) Try to keep to the same ordering of options and of parameters as has been used in the Genstat 5 directives. For example, you will find that PRINT always comes before CHANNEL where both occur as options of the same directive – see directives ADISPLAY, DUMP, READ, RDISPLAY, STORE, and so on.

(f) Options and parameters that have possible settings yes or no must always have no as their default.

(g) Standard prefixes have been defined for use in the names of directives and their options and parameters. Try to use these where appropriate, and avoid defining further prefixes (or causing further ambiguity in the meanings of the existing prefixes) unless this is unavoidable.

A for analysis of variance (for example in directives AKEEP and ADISPLAY, and in the ASAVE option of SET);

C for covariates (for example in option CPRINT of ANOVA and ADISPLAY, and options CREGRESSION and CSSP of AKEEP), or for clustering (for example in option CTHRESHOLD of directive HCLUSTER);

CL for column labels (for example in option CLPRINT of directive PRINT);

F for form (for example in directives FSSPM and FTSM), or for factor (for example in option FREPRESENTATION of READ), or for the F distribution (for example in option FPROBABILITY of ADISPLAY and ANOVA);

G for group (for example in parameter GTHRESHOLD of HCLUSTER, and parameter GSIMILARITY of HDISPLAY);

H for hierarchical (for example in directives HCLUSTER and HDISPLAY);

I for identifier (for example in option IPRINT of PRINT and TABULATE), or for imaginary (for example in parameters ISERIES and ITRANSFORM of FOURIER);

IN for input (for example in option INPRINT of JOB and SET, and parameter INMATRIX of FLRV and SVD);

LIB for Library (for example in procedure LIBHELP);

M for margin (for example in the MNAME parameter of PRINT);

MAX for maximum (for example in option MAXCYCLE of ANOVA and ESTIMATE, and option MAXLAG of CORRELATE and TSUMMARIZE);

MV for missing value (for example in option MVREPLACE of ESTIMATE);

N for number (for example in option NVALUES of FACTOR, TEXT, VARIATE and UNIT, in option NROOTS of CVA, FLRV, PCO, PCP and RELATE, and in option NTIMES of EXIT, FOR and RETURN);

NEW for new (for example in option NEWSTRUCTURE of COMBINE, parameter NEWSTRUCTURES of EQUATE, and parameters NEWVALUES and NEWINTERVALS of INTERPOLATE);

NO for no (for example in option NOMESSAGE of ADD, DROP, FIT, FITCURVE, FITNONLINEAR, RDISPLAY, STEP, SWITCH and TRY);

OLD for old (for example in option OLDSTRUCTURE of COMBINE, parameter OLDSTRUCTURES of EQUATE, and parameters OLDVALUES and OLDINTERVALS of INTERPOLATE);

OUT for output (for example in option OUTPRINT of JOB and SET, and option OUTCHANNEL of MERGE);

P for print (for example in option PUNKNOWN of PRINT, and options PFACTORIAL, PCONTRASTS and PDEVIATIONS of ADISPLAY and ANOVA);

R for regression (for example in directives RDISPLAY and RKEEP, and option RSAVE of SET) and for residual (for example in option RMETHOD of MODEL);

RL for row labels (for example in options RLPRINT and RLWIDTH of PRINT);

RSS for residual sum of squares (for example in parameter RSS of ROTATE);

T for time series (for example in directives TKEEP and TSUMMARIZE, and option TSAVE of SET), or for table (for example in functions TMEANS and TSUMS);

U for unadjusted (for example in option UPRINT of ADISPLAY and ANOVA);

V for variance (for example in option VCOVARIANCE of PREDICT, RKEEP and TKEEP), or variate (for example in functions VMEANS and VSUMS);

W for within-group (for example in parameter WMEANS of SSPM, parameter WMATRIX of FLRV, and parameter WSSPM of CVA);

X for x (for example in parameters XTITLE, XLOWER and XUPPER of AXES, DGRAPH and FRAME, and parameters XINPUT and XOUTPUT of ROTATE); and

Y for y (for example in parameters YTITLE, YLOWER and YUPPER of AXES, DGRAPH and FRAME, and parameters YINPUT and YOUTPUT of ROTATE).

4. Style of the Code

To help the editors and referees to assess procedures from the many different authors, we strongly encourage a standard style for procedures submitted to the Library. The aim of these stylistic rules is to make the procedures easier to read and understand, for example by consistent use of spaces, and by the use of different patterns of small and capital letters to distinguish between identifiers, strings and system words. Procedures that depart from the rules may be returned to the author for revision before any detailed refereeing is undertaken. Many of the rules are similar to those used for examples in the Genstat 5 Reference Manual – see Section 1.8.

(a) There should be only one statement per line. Continuation lines should be indented at least two characters to the right of the initial line of a statement. There should also be a consistent indentation of at least two characters for the initial line of each statement within a loop, a block-if structure or a multiple-selection structure (Section 6.2 of the Genstat 5 Reference Manual).

(b) Leave one space before the opening square bracket enclosing an option sequence and one space after the closing bracket, but do not put spaces by square brackets when they are used to enclose suffixes, nor around the $ character. Do not put spaces before the equals sign when it occurs as an option or parameter name, although you may put spaces around equals signs in expressions. Do not leave any spaces before commas nor before semi-colons, but do leave a space after a semi-colon and after a colon. Sometimes clarity can be improved by having several instead of a single space, for example to allow items in parallel lists to line up in successive continuation lines, but do this consistently within your procedure or not at all.

(c) System words (names of directives, functions, options and parameters) should be given in capitals. They should also be given in full, unless the full form is overlong and an abbreviation is self-explanatory: for example either TREAT or TREATMENTS would be acceptable instead of TREATMENTSTRUCTURE, but not GRAP instead of GRAPH.

(d) Omission of the names of parameters and options (with their accompanying equals signs) may be permitted provided the purpose of the statement remains clear. For example the name of the first parameter (or of the first option) can always be omitted; the second parameter name can often be omitted too, provided the first parameter is also specified and the directive is sufficiently well known for readers to be expected to remember the omitted names. For example

```
GRAPH Sales; Time
```

or

```
GRAPH Sales; Time; SYMBOLS='+'
```

would be acceptable, but not

```
PEN 2; 1
```

Statements that use the full power (and complexity) of the rules in Section 1.6.1 of the Genstat 5 Reference Manual will generally not be permitted. For example

```
PRINT Sales,Time; DECIMALS=2,0; 4,2
```

is not clear, and should be written as

```
PRINT Sales,Time; DECIMALS=2,0; SKIP=4,2
```

(e) Strings that occur as values of options or parameters (for example `left` in `JUSTIFIED=left`) should be given in full and in small letters. Again, if the full form is

overlong and an abbreviation is self-explanatory, this will be permitted: for example PRINT=aov would be acceptable instead of PRINT=aovtable in ANOVA, but not JUSTIFIED=l instead of JUSTIFIED=left.

(f) Structures should have identifiers that are easily associated with their purpose. Identifiers should be mainly in lower-case letters; it can be helpful to use a capital letter for the initial character (for example Sales) as this allows identifiers to be distinguished from strings, see (e) above. Clarity can also be enhanced by using several capital letters where an identifier is formed from two (or more) words or abbreviated words (for example MinSales). An important exception is that the names of the dummies used to refer to the arguments of the procedure must be all in capitals (see Section 6.3.1 of the Genstat 5 Reference Manual). You should not use identifiers that differ from each other only because one has some of its letters in lower case instead of in capitals (for example, in the procedure SQUARERT in Appendix 1, it would be wrong to use a structure with identifier RootX as there is already a structure called ROOTX); the procedure could then give incorrect results if directive SET had been used in the calling program to make capital and lower-case identifiers identical.

(g) Comments should be included wherever appropriate, to explain what is going on. In particular, there should be an initial set of comment lines immediately after the PROCEDURE statement, to give the date of the current version of the procedure, the names of its authors, and a brief description of the purpose and method of use of the procedure. Also there should be an explanatory comment alongside the definition of each option and parameter. For example:

```
'PRINT',  " (I: string {no,yes} default no) controls whether or not
            the square root is printed " \
```

The comment should indicate whether the setting of the option or parameter is for input only (I), for output only (O), or for input and output (I/O). It should list the types of structure that may occur, also the set of allowed values (where appropriate) and the default.

An example showing how the procedure SQUARERT, discussed in Section 6.3.1 of the Genstat 5 Reference Manual, could be submitted to the Library is shown in Appendix 1 – but as a means of illustrating the rules rather than as a serious submission!

5. Utilities

Several utility procedures have been written to simplify the writing of procedures for the library. These have been placed in the 'utility' module of the Library and you can use the Library procedure LIBHELP to find out the exact details of what they can do and how to use them. The Secretary of the Editorial Committee can supply details and code of further utilities that will be added in the next release of the Library; we would also welcome suggestions or submissions of new utilities if you are aware of additional facilities that need to be provided.

Please use the utility procedures, where possible, instead of writing your own code. This will help to prevent the Library becoming unduly large. Also some of the utility procedures may be replaced by directives (with identical syntax) to improve efficiency in future releases of Genstat.

6. Side Effects

A procedure may change some aspects of the Genstat environment, for example the default number of units or the information printed for a Genstat diagnostic. This may be a specified purpose of the procedure; but if it is an unwanted side-effect, directive GET should be used to store information about the current settings on entry to the procedure, and directive SET should be used before exit to reset any that have been changed. (SET and GET should not, however, be used to obtain information that should be supplied by options or parameters of the procedure.)

7. Diagnostics

If faulty information is supplied to a procedure, failure will occur during execution of one of the statements of the procedure, and Genstat will give an error diagnostic. The error message produced by Genstat will explain what has gone wrong at that particular statement, but it may not be clear to the user of the procedure how this has arisen from the input supplied nor how to correct the mistake.

One way of avoiding this is to check that the arguments of the procedure are correctly set and, if not, to print an error message and exit (see Section 6.3.1 of the Genstat 5 Reference Manual); there are utility procedures available to help with this (see Section 5 above).

You may also want to use directive SET to suppress the printing of diagnostics from some statements, and then use GET to obtain the fault number so that you can either issue your own error message or correct the situation. (Remember that you can use directive DISPLAY to print the diagnostic if it is one that you cannot handle.) However, you must reset the printing of diagnostics before the end of the procedure (see Section 6 above).

8. Documentation

It is hoped that the Library will be issued twice per year, each time with new and extended facilities. Consequently there is no immediate intention to issue a Manual for the Library, but instead to make information about the Library available from within Genstat itself. Later on, when the Library has become established, we may publish a guide to the Library with detailed descriptions of the more popular procedures.

Information about the Library, and its procedures, can be obtained during a Genstat run by using the (library) procedure LIBHELP. This information is stored in text structures in a backing-store file, formed when each release of the Library is loaded at a site. There are nine sections available for each procedure, although some may not be relevant for simple procedures. Please try to be informative but not verbose – if possible each section should contain no more than 20 lines of at most 80 characters in width (so that it will fit on a VDU screen).

(a) A single line giving the name of the procedure and a very brief description of its purpose; this will form the entry for the procedure in the index to the Library.

(b) A more detailed description of the purpose of the procedure, with an overview of how to use it and lists of the names of its options and of its parameters.

(c) Specifications of the syntax of the options (in the same format as used for directives by HELP).

(d) Specifications of the syntax of the parameters (in the same format as used for directives by HELP).

(e) A brief description of the method, with references to published articles or papers for more detail.

(f) Information about whether the procedure takes account of restrictions on input vectors.

(g) A list of procedures called by the procedure.

(h) Details (and comparison) of similar or related procedures.

(i) The list of authors of the procedure.

There is also a procedure called LIBEXAMPLE, which allows users to access a text structure containing input statements (and data) for a simple example illustrating how to use the procedure. The user would then execute the example as a macro (that is by using the compound operator ##). The example should not be too abstract, nor produce too much output. However, you should include captions to explain what is going on.

9. Reporting of Errors

Reports of errors in a Library procedure should be sent to the Secretary of the Editorial Committee. The report should consist of Genstat output illustrating the fault (with an explanation of why it is believed that this is incorrect), and a listing of the input and data files of the program that produced the output. It is hoped that users may often be able to supply these files in computer-readable form, as mentioned in Section 1 above. The Secretary will take a record of each error and will forward the details to the author of the procedure concerned.

10. Correction of Errors

The responsibility for correcting errors in procedures lies with their authors. Within one month of receipt of an error report we would expect the author to reply with a note that can be added into the errors section of the information provided by LIBHELP; this should explain the circumstances under which the procedure behaves incorrectly, and how to avoid them. At the same time we would expect a synopsis of how it is intended to correct the error.

It is hoped that all errors that are reported during the first three months of each release of the Library will be corrected in the next release – we aim to issue the Library twice per year. If no correction is received within three months, the Editors will decide whether to withdraw the procedure, or whether to suggest to the author that someone else be asked to supply a correction – and be added to the list of authors of the procedure.

11. The Editorial Committee

As of 1st September 1987, the Committee is as follows:

R W Payne (Chairman)
P G N Digby (Secretary)
G M Arnold
J Bryan-Jones
T J Cole
A W A Murray
J H Roger
K I Trinder
A J Weekes
G Tunnicliffe Wilson

Appendix 1: Example

This example shows what should be supplied if the procedure SQUARERT, described in Section 6.3.1 of the Genstat 5 Reference Manual, were to be submitted as a procedure to the Library. It also illustrates how the code of SQUARERT would need to be modified to give the extra protection against user errors required by Library procedures.

(a) Code of the procedure

```
PROCEDURE 'SQUARERT'
" Roger Payne, Rothamsted Experimental Station, 1/7/87.
  Calculates the square root of the value in the scalar X
  using an iterative method. The result can be printed, and
  it can be saved in scalar ROOTX. Option TRACE allows the
  progress of the iterations to be followed. "
```

```
OPTION NAME= \
  'PRINT',  " (I: string (no,yes) default no) controls whether or not the
              square root is printed " \
  'TRACE';  " (I: string (no,yes) default no) controls whether or not a
              trace of the progress of the iterations is printed " \
  MODE=t; DEFAULT='no'
PARAMETER NAME= \
 'X',     " (I: scalar) the value whose square root is to be calculated " \
 'ROOTX';" (O: scalar) the calculated square root " \
  MODE=p
" check that the arguments of the procedure are valid
  (using Library procedure CHECKARGUMENTS) "
CALCULATE E = 0
CHECKARGUMENTS [ERROR=E] PRINT,TRACE,X,ROOTX; " names of the arguments" \
  SET=3('yes'),'no';        " whether or not they must be set " \
  TYPE=2('text','scalar'); " texts indicating the allowed types for each one" \
  DECLARED=3('yes'),'no';  " whether they must already have been declared" \
  VALUES=2(!T(no,yes),*);  " the set of allowed values (if appropriate)" \
  PRESENT=3('yes'),'no'    " whether they must have values "
" the value of scalar E is set to 1 if any errors are found "
EXIT [CONTROL=procedure] E
" check for invalid setting of parameter X ( i.e. < 0 ) "
IF X < 0
  PRINT 'X < 0 :  square root cannot be calculated'
  EXIT [CONTROL=procedure]
ENDIF
" set ROOTX to a local scalar if unset "
IF UNSET(ROOTX)
  SCALAR Root
  ASSIGN Root; POINTER=ROOTX
ENDIF
" calculate convergence limit & initialize "
CALCULATE Clim = X/10000
& ROOTX = X
" loop until convergence "
FOR [NTIMES=20]
  CALCULATE Previous = ROOTX
  & ROOTX = (X/Previous + Previous)/2
  " print current estimate if TRACE set to 'yes' "
  IF 'yes'.IN.TRACE
    PRINT [IPRINT=*] ROOTX
  ENDIF
  EXIT ABS(Previous-ROOTX) < Clim
ENDFOR
" print result if PRINT set to 'yes' "
IF 'yes'.IN.PRINT
  PRINT [IPRINT=*] 'square root of ',X,' is ',ROOTX; \
    JUSTIFICATION=left
ENDIF
ENDPROCEDURE
```

(b) Information for the Library on-line help

```
Brief description
SQUARERT  calculates the square root of a scalar value
```

```
More detailed description
SQUARERT uses an iterative method to calculate the square root of a scalar
value input by parameter X. The result can be printed, by setting option
PRINT=yes, or saved by using parameter ROOTX. You can follow the progress
of the iterations by setting option TRACE=yes.
Options: PRINT,TRACE
Parameters: X,ROOTX

Specification of the syntax of the options
Options
PRINT = string  Whether or not the calculated square root is to be printed
                (no, yes); default no
TRACE = string  Whether or not the estimated value of the square root at
                each iteration is to be printed (no, yes); default no

Specification of the syntax of the parameters
Parameters
X     = scalars Values for which the square roots are to be calculated
ROOTX = scalars Scalars to store each calculated square root

Description of the method
SQUARERT uses the standard iterative method for calculating square roots, as
may be used on mechanical calculators or the cheaper electronic calculators.
The initial value is estimated by the input value of X. Then each iteration
forms a new estimate by taking the average of the current estimate and the
result of dividing X by the current estimate. The process stops when the
difference between the new and previous estimates is less than X/10000.

Information about RESTRICT
Not relevant (none of the arguments of SQUARERT is a vector).

Procedures called by SQUARERT
Procedure CHECKARGUMENTS is called to check the option and parameter settings.

Similar or related procedures
There are no similar procedures, but it is very much easier to calculate
square roots using the standard Genstat function SQRT!

Author
R.W. Payne, Statistics Department, Rothamsted Experimental Station,
            Harpenden, Herts AL5 2JQ, United Kingdom.

Example program
" Example of the use of procedure SQUARERT
  Roger Payne, Rothamsted, 1/7/87. "
PRINT !T('Use SQUARERT to calculate the square root of 48;
save the results in Root48.'); JUSTIFICATION=left
SQUARERT 48; ROOTX=Root48
PRINT Root48
PRINT !T('Use SQUARERT to calculate and print the square root of 36;
do not save the results.'); JUSTIFICATION=left
SQUARERT [PRINT=yes] 36
PRINT !T('Use SQUARERT to calculate and print the square root of 36;
print the estimate at each iteration but do not save the results.'); \
  JUSTIFICATION=left
SQUARERT [PRINT=yes; TRACE=yes] 36
```

(c) Test program

```
" Test program for the procedure SQUARERT
  Roger Payne, Rothamsted, 1/7/87. "
SQUARERT 48; ROOTX=Root48
CALCULATE Forty8 = Root48*Root48
PRINT Forty8,Root48
SQUARERT [PRINT=no; TRACE=no] 48; ROOTX=Root48
PRINT Root48
SQUARERT [TRACE=yes] 56; ROOTX=Root56
CALCULATE Fifty6 = Root56*Root56
PRINT Fifty6,Root56
SQUARERT [PRINT=yes] 56
SQUARERT [PRINT=yes; TRACE=yes] 56
SQUARERT -100
SQUARERT [PRINT='if you want'] 99
SQUARERT [TRACE=YY] 99
VARIATE [NVALUES=2] Vx
SQUARERT Vx
STOP
```

(d) Additional information

For further information, see Sections 1.7.2 and 6.3.1 of the Genstat 5 Reference Manual.

Accessing the NAG Fortran Library from Within Genstat, and Other Ways of Extending Genstat

P W Lane
Statistics Department
Rothamsted Experimental Station
Harpenden
Hertsfordshire
United Kingdom *AL5 2JQ*

R M J Iles
NAG Central Office
Mayfield House
256 Banbury Road
Oxford
United Kingdom *OX2 7DE*

J A Nelder
Imperial College of Science and Technology
Department of Mathematics
Huxley Building
180 Queen's Gate
London
United Kingdom *SW7 2BZ*

1. Introduction

Genstat is designed to be extended – not just by the developers, but also by the users. The most convenient method is by procedures, because this lets you take advantage of the high-level nature of the Genstat language to save programming time. A procedure, once written, can be used by anyone with access to it – this is easily made automatic – and its invocation looks like any of the standard Genstat directives. However, procedures can be inefficient in terms of computing time, or wasteful when low-level code already exists for a complicated task.

The new version of Genstat provides four ways of extending Genstat to new methods programmed at a low level. We list them here in increasing order of the amount of work you need to do, assuming that low-level code already exists for the task you want to do:

(a) use PASS to communicate information between Genstat and an independent program;
(b) use SUSPEND together with input and output directives, as for PASS;
(c) use OWN to invoke low-level code within a relinked version of Genstat;
(d) define new directives that invoke low-level code as for OWN.

These possibilities in combination make Genstat an ideal kernel for systems based on the low-level routines available in a library. The following sections illustrate the first two methods with experiments done in interfacing the NAG Fortran Library – an extensive and generally available library of mathematical and statistical algorithms. Section 4 illustrates the third method, in the context of fitting a nonlinear regression model. The fourth method is not discussed here: for details of this, and further examples of the other methods, see the Genstat 5 Reference Manual.

2. Differentiating a Function Using PASS

The PASS directive has a simple syntax:

PASS

Does work specified in Fortran subprograms supplied by the user, but not linked into Genstat. This directive may not be available on some computers.

Option

NAME = *text*	Filename of external executable program; default GNPASS

Parameter

pointer	Structures whose values are to be passed to the external program, and returned

PASS communicates the values of the data structures given in the parameter to an external program, whose name can be specified in the option, executes the program and then retrieves the values of the same structures, which may have been modified by the program. Much of the work done by PASS within Genstat, before and after executing the external program, is in communication between Genstat and the external program. A Fortran program called GNPASS is distributed with Genstat to serve as the main module of the external program that is to be accessed by PASS: it deals with the communications, storing values in Fortran commons, and needs to be edited only to include a call to your own Fortran routines. In fact, a PASS statement such as

```
PASS [NAME=MYPROG] !P(A,B,C)
```

could be mimicked by a short series of statements as follows, using the SUSPEND directive described in the next section:

```
OPEN 'GNPASI'; CHANNEL=2; FILETYPE=output
PRINT [CHANNEL=2; IPRINT=*; SQUASH=yes; SERIAL=yes] A,B,C
CLOSE 2; output
SUSPEND [SYSTEM='RUN MYPROG']
OPEN 'GNPASO'; CHANNEL=2; FILETYPE=input
READ [CHANNEL=2; SERIAL=yes; END=*] A,B,C
```

However, this involves you in more work, because your program MYPROG would have to deal with the details of input from the file GNPASI and output to the file GNPASO: details which are already dealt with by the distributed program GNPASS for use with PASS.

Some operating systems allow subprograms written in different languages to be linked together. For example, the VMS system on Vax computers allows Pascal and Fortran to be combined. Thus, the external program could be constructed by linking the supplied Fortran module with Pascal procedures, though information would then probably have to be passed between them by parameters of the procedures rather than by commons. However, such mixed programs will not always be portable, so we restrict our attention here to extensions using Fortran only.

The NAG Fortran Library contains a subroutine called D04AAF which provides derivatives at a given point of an arbitrary function of one variable, using an extension of the Neville algorithm. Usually, the subroutine is accessed as follows:

(a) write a Fortran driver program, to invoke D04AAF and display the results;
(b) write a Fortran function to evaluate the chosen function at a given point;
(c) compile the two Fortran modules;
(d) form an executable program from these together with the NAG Fortran Library routine;
(e) run the program.

If a new function is to be examined, or the evaluation done at a new point, each step of the process must be repeated.

All the steps above can be controlled from within Genstat by using the PASS directive. This

allows use of the standard facilities of Genstat for display and storage of results, and allows the whole process to be prepared as a procedure so that only a single command is needed to carry out all the steps. The action needed is as follows:

(a) write and compile a general driver-program for DO4AAF by editing the GNPASS subroutine, allowing the parameters and results of DO4AAF to be transmitted through the parameter of PASS;
(b) set up the Fortran code to evaluate a function as text within Genstat, and print it to an external file;
(c) compile the Fortran function by giving an operating-system command from within Genstat (using the SUSPEND directive described in the next example);
(d) form an executable program from the compiled driver-program and function and the NAG Fortran Library, by giving another operating-system command;
(e) run the new program by giving a PASS statement, supplying values of the variable at which to evaluate derivatives, and other information needed by the DO4AAF routine.

This process has been put in a procedure called DIFFERENTIATE. To use this, once the general driver-program has been made available, a single statement is needed, such as the following:

```
DIFFERENTIATE [FORTRAN='FUN=EXP(X)'; N=3] 1,3; STEP=0.05
```

This statement will estimate the first three derivatives of the function $exp(x)$ at the points x=1 and x=3 (the steplengths are needed by the NAG routine: see the NAG Fortran Library Manual, [1], for details). Here is the output, generated in interactive mode:

```
Compiling and linking Fortran ...
                          ... completed
If there are errors, type s to stop; otherwise c to continue

    6  c

FORTRAN STOP

            d1                      error
2.71828151E+00          0.16433292E-04
2.71828270E+00          0.53820924E-04
2.71826935E+00          0.97488932E-03

FORTRAN STOP

            d2                      error
7.38904142E+00          0.42477128E-04
7.38899088E+00          0.16083541E-03
7.38915634E+00          0.37408168E-02
```

The DIFFERENTIATE procedure is shown in the Appendix, together with the driver program and a procedure for providing on-line information about DIFFERENTIATE.

3. Solving Sparse-matrix Equations Using SUSPEND

The SUSPEND directive has the following syntax.

SUSPEND

Suspends execution of Genstat to carry out commands in the operating system. This directive may not be available on some computers.

Options

SYSTEM = *text*	Commands for the operating system; default: prompt for commands (interactive mode only)
CONTINUE = *string*	Whether to continue execution of Genstat without waiting for commands to complete (no, yes); default n

No parameters

SUSPEND is designed to suspend Genstat in an interactive program, and prompt for operating-system commands. Alternatively, the operating-system commands can be given as text in the SYSTEM option, in which case the directive can be used in batch programs as well.

For example, the SUSPEND directive is used in the DIFFERENTIATE procedure described in the last section to compile a Fortran module. This is achieved as follows:

```
OPEN 'FUNC.FOR'; CHANNEL=2; FILETYPE=output
PRINT [CHANNEL=2] FORTRAN; SKIP=6; FIELDWIDTH=72
CLOSE 2; FILETYPE=output
SUSPEND [SYSTEM='FORTRAN FUNC.FOR']
```

Clearly, the setting of the SYSTEM option depends on what operating system is being used.

The NAG Fortran Library contains a routine called F04AXF to solve sparse matrix equations of the form

$$Sm +* Rhs = Sol$$

where *Sm* is a sparse matrix, +* denotes matrix multiplication, and *Rhs* and *Sol* are column vectors. The original example program for the library routine has been amended to take values of *Sm* and *Rhs* from a file and put the resulting values of *Sol* in another file. Rather than modify this program into a form suitable for PASS, the SUSPEND directive has been used to solve equations from within Genstat.

A series of simple procedures in Genstat have been written to do the following operations on sparse matrices:

(a) DECSM declare a sparse matrix (stored as a pointer to five component structures);
(b) READSM read values into a sparse matrix;
(c) PRINTSM print a sparse matrix;
(d) CMTOSM convert a matrix into a sparse matrix;
(e) CSMTOM convert a sparse matrix into a matrix;
(f) ADDSM adds sparse matrices;
(g) SMSOLVE solve a sparse matrix equation.

Only the last of these requires anything other than standard Genstat statements, though ADDSM was not efficiently done, because it required both matrices to be expanded, then the resulting matrix after addition to be contracted. A similar technique could have been used for SMSOLVE, but use of the library subprogram was much less wasteful of space and time.

The SMSOLVE procedure uses SUSPEND in much the same way as in the above example for DIFFERENTIATE. First it opens a file and prints into it the components of the sparse matrix, SM, and the variate RHS. Then it closes the file and gives a SUSPEND statement to run the external program. Finally, it reads the results output from the external program, checking for possible failure. Here is a listing of the procedure.

```
PROCEDURE 'SMSOLVE'
PARAMETER 'SM','RHS','SOL'; MODE=p
   OPEN 'extin.dat'; CHANNEL=3; FILETYPE=output
   PRINT [CHANNEL=3; SERIAL=yes; IPRINT=*] SM[],RHS
   CLOSE 3; FILETYPE=output
   SUSPEND [SYSTEM='run smsolve']
   OPEN 'extout.dat'; CHANNEL=3; FILETYPE=input
   VARIATE [NVALUES=SM[1]] SOL
   SCALAR ifail
   READ [CHANNEL=3; END=*] ifail
   IF IFAIL/=0
      PRINT [IPRINT=*] 'Fault reported: number',ifail; JUSTIFICATION=left
   ELSE
      READ [CHANNEL=3; SERIAL=yes; PRINT=*; END=*] SOL
```

```
    ENDIF
    CLOSE 3; FILETYPE=input
ENDPROCEDURE
```

This procedure uses fixed filenames, EXTOUT.DAT and EXTIN.DAT, for communication. Thus, if two users of the computer are working concurrently in the same file directory, they are likely to encounter problems. This also applies to the PASS directive, because fixed filenames are used by Genstat internally to communicate the values of data structures.

The SUSPEND directive has also been used in this way to provide a set of procedures for complex-matrix arithmetic in Genstat. A complex matrix is defined as a pointer with two matrices (for real and imaginary parts) as elements. Complex matrix addition, subtraction, multiplication, and inversion can be constructed as Genstat procedures using real matrix arithmetic. For the extraction of latent roots and vectors a NAG routine (F02AKF) was invoked using SUSPEND.

A third set of procedures has been written for evaluating gradients, hessians and Jacobians of functions, using as a kernel a Pascal procedure for algebraic differentiation of an expression.

4. Fitting Complex Nonlinear Models Using OWN

The OWN directive is not new in Genstat. However, its syntax has been modified to make it more convenient for use in Genstat 5:

OWN

Does work specified in Fortran subprograms supplied and linked by the user.

Option

SELECT = *scalar*	Set a switch, designed to allow OWN to be used for many applications; standard set-up assumes scalar in the range 0 to 9; default 0

Parameters

IN = *pointers*	Supplies input structures, which must have values, needed by the auxiliary subprograms
OUT = *pointers*	Supplies output structures whose values or attributes are to be defined by the auxiliary subprograms

To use the OWN directive to invoke your own Fortran routine, you need to modify the OWN subroutine (which is distributed with Genstat) to call your routines, and then relink Genstat, including these routines. The linking process is now simple on many computers, but is likely to be more difficult on computers such as PCs that have relatively little address space and do not have virtual memory.

As well as using the OWN directive to cause execution of some Fortran code, it is possible to make use of the code within the FITNONLINEAR directive of Genstat, which fits nonlinear models. Usually, the calculations required to specify the models are specified as Genstat expressions, and they are carried out at each step of the iterative search for the best fit, using the facilities of the CALCULATE directive. When the calculations are very extensive, it can save a lot of computer time if the calculations are supplied in Fortran, using OWN, and if the OWN, INOWN and OUTOWN options of FITNONLINEAR are set to invoke this code.

Here is a practical example of using OWN to fit a complex nonlinear diffusion model. The model is derived from standard diffusion equations and used to predict the spread of pollutants in two dimensions through soil under the influence of cultivation operations which move the soil. Here are the Genstat statements required to define and fit the model with standard expressions and the FITNONLINEAR directive.

```
"Access data:
    variates x,y storing coordinate positions of measurements of pollution
             in soil,
```

```
    variates csoil[1...6] storing observed concentrations of pollutants at
            the coordinate positions,
    variate nplough storing number of cultivations since each application of
            sewage sludge containing pollutants,
    variates csludge[1...6] storing concentrations of pollutants in each
            application of sewage sludge,
    scalars wx2, wy2 storing dimensions of treated area
(statements not shown here)"
"
Preliminary calculations for efficiency
"
POINTER [NVALUES=25] sp,xa,xb,ya,yb,nxa,nxb,nya,nyb,cc
CALCULATE sp[] = SQRT(nplough$[1...25]/2)
& (xa,ya,xb,yb)[] = (25(wx2,wy2)2+25(x,y)2*50(1,-1))/sp[]/2
"
Store calculations to be done at each iteration
"
EXPRESSION [VALUE=(nxa,nya,nxb)[] = NORMAL(((xa,ya,xb,yb)[] \
   + 25(sx,sy)2*sp[]*50(1,-1)) / SQRT(25(dx,dy)2))] e[1]
& [VALUE=cc[]=cs$[1...25]*(nxa[]+nxb[]-1)*(nya[]+nyb[]-1)] e[2]
& [VALUE=csum=VSUM(cc)] e[3]
"
Fit the model for each pollutant in turn
"
FOR yy=csoil[]; cs=csludge[]
   MODEL yy
   RCYCLE dx,dy,sx,sy
   FITNONLINEAR [CALCULATION=e] csum
ENDFOR
```

Though the calculations can be specified concisely in Genstat, using only three expression structures `e[1...3]`, they involve a lot of work because each variate consists of 158 observations, and contributions to the model have to be calculated for each of the 25 applications of sludge containing the pollutants. Therefore, the calculations were rewritten in Fortran – requiring about 50 statements, not including the calculation of the Normal probability integral achieved above with the function NORMAL. Having compiled this subprogram, and a version of the OWN subroutine modifed to call the subprogram, and relinked Genstat to include them both, the following statements were all that were needed to define and fit the model.

```
FOR yy=csoil[]; cs=csludge[]
   MODEL yy
   RCYCLE dx,dy,sx,sy
   FITNONLINEAR [OWN=1; IN=dx,dy,sx,sy,wx2,wy2,cs,nplough,x,y; OUT=csum] csum
ENDFOR
```

5. References

[1] *NAG Fortran Library Manual, Mark 12.*
Nuffield Press, Oxford, 1987.

6. Appendix

Procedure DIFFERENTIATE

```
PROCEDURE 'DIFFERENTIATE'
"
 Procedure to invoke NAG routine D04AAF to differentiate a function,
 creating a Fortran routine to evaluate the function, compiling it,
 linking an external program, and executing it.
"
   OPTION 'PRINT','FORTRAN','NDERIVATIVE','KEEP'; \
      MODE=T,P,V,T; DEFAULT='d',*,!(1),'*'
   PARAMETER 'VALUE','STEP','FAULT','DERIVATIVE','ERROR'; MODE=P
   TEXT [VALUES='REAL*8 FUNCTION FUN(X)','REAL*8 X'] start
   & [VALUES=RETURN,END] end
   " Check options"
   CONCATENATE [pcode] PRINT; WIDTH=1
   " Form program only on first call of procedure in a statement that
     specifies Fortran, and then only if Fortran is different to that
     specified last time the program was formed (stored by option KEEP)"
   IF UNSET(FORTRAN)
      CALC formp = 0
   ELSIF FORTRAN .EQS. KEEP
      CALC formp = 0
   ELSE
      CALC formp = 1
      " Reset default of KEEP option"
      TEXT KEEP; VALUES=FORTRAN
   ENDIF
   " If FORTRAN setting is not text, let compiler give fault;
     if NDER setting is too large, let NAG routine give fault
     KEEP option should not be set by user."
   CALCULATE nder = ABS(NDERIVATIVE)
   "
    Check parameters, and provide defaults
   "
   GETATT [ATTRIBUTES=type] VALUE,STEP; SAVE=pp[1,2]
   FOR p1=1,2
      IF pp[p1][1] /= 1
         PRINT 'Parameter',p1,' is not set with scalars'; FIELDWIDTH=*,2,*
         EXIT [CONTROL=proc]
      ENDIF
   ENDFOR
   "If FAULT setting is not a scalar, let IF or SCALAR give fault"
   IF UNSET(FAULT)
      SCALAR [VALUE=0] faul
      DUMMY [VALUE=faul] FAULT
   ELSIF FAULT /= 1
      SCALAR [VALUE=0] FAULT
   ENDIF
```

```
    IF UNSET(DERIVATIVE)
       DUMMY [VALUE=deriv] DERIVATIVE
    ENDIF
    IF UNSET(ERROR)
       DUMMY [VALUE=err] ERROR
    ENDIF
    " Assign values, required by PASS;
      if setting not a variate let VARIATE give fault"
    VARIATE [VALUE=#nder(0)] DERIVATIVE,ERROR
    IF formp
       GET [ENVIRONMENT=e]
       CALC inter = (e['run'] .EQS. 'interactive')
       IF inter
          PRINT [SQUASH=yes] 'Compiling and linking Fortran ...'
       ENDIF
       OPEN 'FD04AA.FOR'; CHANNEL=2; FILETYPE=output
       PRINT [CHANNEL=2; IPRINT=*; SQUASH=yes; SERIAL=yes; WIDTH=72] \
          start,FORTRAN,end; FIELDWIDTH=66; JUSTIFICATION=left; SKIP=!(*,6)
       CLOSE 2; FILETYPE=output
       " If compilation fails, let linking fail as well"
       SUSPEND [SYSTEM=!T('FORTRAN FD04AA', \
          'LINK CD04AA,GNPASS,FD04AA,NAGLIB/LIB')]
       IF inter
          PRINT [SQUASH=yes; SERIAL=yes] '                    ... completed', \
          'If there are errors, type s to stop; otherwise c to continue'
          TEXT [NVALUES=1] code
          READ [END=*; LAYOUT=fixed] code; FIELD=1
          IF code .IN. !T(s,S)
             EXIT [CONTROL=proc]
          ENDIF
       ENDIF
    ENDIF
    "
     Evaluate derivatives
    "
    PASS [NAME='CD04AA'] !P(VALUE,NDERIVATIVE,STEP,DERIVATIVE,ERROR,FAULT)
    IF pcode .IN. !T(d,D)
       " Print results"
       PRINT DERIVATIVE,ERROR; FIELDWIDTH=-20; DECIMALS=8
    ENDIF
ENDPROCEDURE
```

Subroutine CD04AA

```
SUBROUTINE 'CD04AA'
C         Subprogram to call D04AAF to differentiate a function
C         This is linked with a version of the GNPASS program that
C         has been edited to include the statement
C               CALL CD04AA
C         and with a function FD04AA
C         to produce an executable program called CD04AA
C
C         Common /DATA/ (copied from GNPASS program)
      INTEGER        MDDATA,MRDATA,MIDATA,MCDATA,MSTRUC
      PARAMETER     (MDDATA=1,MRDATA=10000,MIDATA=10000,MCDATA=1,
     -               MSTRUC=100)
      COMMON/DATA/   DDATA(MDDATA),DMV,RDATA(MRDATA),RMV,
     -               IDATA(MIDATA),IMV,CDATA(MCDATA),
     -               MODE(MSTRUC),ORIGIN(MSTRUC),NVALUE(MSTRUC)
      DOUBLE PRECISION DDATA,DMV
      REAL           RDATA,RMV
      INTEGER        IDATA,IMV,MODE,ORIGIN,NVALUE
      CHARACTER*8    CDATA
C
      REAL*8 XVAL,STEP,DERIV(14),ERROR(14),FUN
      INTEGER NDER,IFAIL
      EXTERNAL FUN
C         Copy reals (from Genstat scalars) to integers or double precision
      XVAL = RDATA(ORIGIN(1)+1)
      NDER = NINT(RDATA(ORIGIN(2)+1))
      STEP = RDATA(ORIGIN(3)+1)
C         Call NAG routine
      CALL D04AAF(XVAL,NDER,STEP,DERIV,ERROR,FUN,IFAIL)
C         Copy double precision and integers to reals
      DO 1 K1=1,NVALUE(4)
         RDATA(ORIGIN(4)+K1) = DERIV(K1)
         RDATA(ORIGIN(5)+K1) = ERROR(K1)
    1 CONTINUE
      RDATA(ORIGIN(6)+1) = IFAIL
      RETURN
      END
```

Procedure NAGHELP

```
PROCEDURE 'NAGHELP'
   PARAMETER 'ROUTINE'; MODE=T
"Give information about using NAG procedures"
   IF (UNSET(ROUTINE))
      "Give menu"
      PRINT !T('
The following routines are available through Genstat procedures:

   Routine  Procedure
   D04AAF   DIFFERENTIATE

Type NAGHELP ''NAME OF PROCEDURE'' for information, using upper case and
four or more characters; for example,
   NAGHELP ''DIFF'' '); JUST=left; SKIP=0
      EXIT [CONTROL=proc]
   ENDIF
   "Access first four characters of setting"
   CONCATENATE ROUTINE; WIDTH=4
   "Print information for each procedure"
   IF 'DIFF' .IN. ROUTINE
      SET [PAUSE=20]
      PRINT [IPRINT=*] !T('
Procedure DIFFERENTIATE

Uses NAG routine D04AAF to evaluate derivatives of a function of a
single variable, X say, at a given value of the variable.  Produces
a Fortran program called FD04AA.FOR, object module FD04AA.OBJ, and
executable program CD04AA.EXE.

Parameters
VALUE       scalars    Values of X at which to evaluate derivative
STEP        scalars    Steplengths to use for algorithm; choice can be
                       critical for success: see NAG documentation
FAULT       scalars    Settings of IFAULT parameter from NAG routine
                       see NAG documentation; 0 signals success, 1 failure
DERIVATIVE  variates   To store evaluated derivatives
ERROR       variates   To store estimates of error'); JUST=left; SKIP=0
   PRINT [IPRINT=*] !T('Options
PRINT       text       What to print (derivatives, nothing); default d
FORTRAN     text       Fortran code to calculate function value in real
                       variable FUN from value of X in real variable X;
                       initial six spaces will automatically be added;
                       default as last given in current program
NDERIV      scalar     Number of derivatives to evaluate; maximum 14,
                       negative number evaluates odd or even numbered
                       derivatives only; default 1

Example

DIFFERENTIATE [FORTRAN=''FUN=10*EXP(X)-LOG(X)''; N=5] 1,3; STEP=0.05'); \
      JUST=left; SKIP=0
   SET [PAUSE=0]
   ENDIF
ENDPROCEDURE
```