MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS (H-3-A)
A FAMILY OF ALGORITHMS FOR THE
ESTIMATION OF THE PARAMETERS OF THE STABLE LAWS AND
THE PARAMETERS OF ATTRACTING STABLE LAWS

by

T. A. Delehanty
and
A. S. Paulson

School of Management
Rensselaer Polytechnic Institute
Troy, NY 12180-3590
(518)266-6586

The view, opinions, and/or findings contained in this report are those of the
author(s) and should not be construed as an official Department of the Army
position, policy, or decision, unless so designated by other documentation.

Note: This paper is not to be quoted without the permission of the author(s).
For a list of Working Papers available from the School of Management,
please contact Sheila Chao, School of Management.
DESCRIPTION AND PURPOSE

Potential application of the stable laws has long been hindered by the unavailability of generally available, well-documented algorithms. This paper removes this deficiency by presenting an algorithm for estimation of stable law parameters, with the goal of facilitating the application of stable laws in modeling and inference frameworks. The stable laws have steadily increased in importance to the statistical community since the paper of Mandelbrot (1963). Their role as the only laws possessing domains of attraction makes the stable laws an appealing probabilistic model, and they are capable of modeling a wide range of skewness, heavy tailedness, and central peakedness. Procedures for estimation of stable law parameters have been described by Mandelbrot (1963), DuMouchel (1971), Fama and Roll (1971), Paulson, Holcomb, and Leitch (1975), Koutouvelis (1980, 1981), Feuerverger and McDunnough (1981a, 1981b), and Brockwell and Brown (1981). Because of the intractability of stable densities, attention has centered in recent years on Fourier-based procedures, using the empirical characteristic function. Such procedures should have an adaptive nature (Paulson, Holcomb, and Leitch, 1975; Paulson, Delehanty, and Brothers, 1982; Paulson and Delehanty, 1982).

We present an iterative and adaptive algorithm for joint estimation of stable law parameters, using the empirical characteristic function. The algorithm is flexible in that either of two procedures may be selected, and subsets of the parameters may be allowed to vary freely, with others constrained or held constant. The statistical rationale for the procedures is described in the companion paper by Paulson and Delehanty (1982). The algorithm may also be used to provide informal estimates of the parameters.
A FAMILY OF ALGORITHMS FOR THE ESTIMATION OF THE PARAMETERS OF THE STABLE LAWS AND THE PARAMETERS OF ATTRACTING STABLE LAWS

by

T. A. Delehanty

and

A. S. Paulson*

Rensselaer Polytechnic Institute

Key words: stable laws, parameter estimation, adaptive estimation, empirical characteristic function, domains of attraction, sensitivity analysis, nonlinear optimization, constrained estimation

LANGUAGE: FORTRAN 66

of the stable law to which a sample distribution is attracted.

THEORY AND NOTATION

Nondegenerate stable random variables $X$ may be defined by the characteristic function

$$
\phi(u) = \mathbb{E} \{ \exp(iuX) \} = \exp \{ iu\mu - |\sigma u|^\alpha (1 + i\beta \text{sgn}(u) \chi(u, \alpha)) \},
$$

where $i^2 = -1$, $0 < \alpha \leq 2$, $|\beta| \leq 1$, $\sigma > 0$, and

$$
\chi(u, \alpha) = \begin{cases} 
\tan \frac{\pi \alpha}{2}, & \alpha \neq 1 \\
\frac{2}{\pi} \log |u|, & \alpha = 1.
\end{cases}
$$

Here $\alpha$, the characteristic exponent, is a measure of heavy tailedness and central peakedness, $\beta$ is a skewness measure, $\sigma$ is a scale parameter, and $\mu$ is a location parameter unless $(\alpha = 1, \beta = 0)$, when the function of location parameter is assumed by $\mu + \frac{2}{\pi} \log u$. The only stable laws whose densities are expressible in closed form are the Gaussian ($\alpha = 2, \beta = 0$), the Cauchy ($\alpha = 1, \beta = 0$), and the reciprocal of a $\chi^2$ variate on one degree of freedom ($\alpha = 1, \beta = -1$).

Let $X_1, \ldots, X_n$ be a stable random sample. The empirical characteristic function is

$$
\phi_n(u) = n^{-1} \sum_{j=1}^{n} \exp(iuX_j).
$$

Let $\psi(u) = \text{Re} \phi(u) + \text{Im} \phi(u)$, $\psi_n(u) = \text{Re} \phi_n(u) + \text{Im} \phi_n(u)$. Estimators interior to the parameter space can be viewed as zeros of the systems

Formulation A

$$
\frac{q}{j} \sum_{j=1}^{q} \frac{\partial \psi(u_j)}{\partial \theta} (\psi(u_j) - \psi_n(u_j))W_j = 0,
$$
Formulation B

\[
\sum_{j=1}^{q} \sum_{k=1}^{q} \frac{\partial \psi(u_j)}{\partial \theta} K_{jk} \left( \psi(u_k) - \psi_n(u_k) \right) w_j w_k = 0, \tag{5}
\]

for \( \theta = \alpha, \beta, \sigma, \mu \). The grid \( \{u_j | j = 1, \ldots, q\} \) is symmetric about zero but does not include the origin, and \( K_{jk} \) denotes the \( j,k \) element of the inverse matrix \( (K_{jk})^{-1} \), where

\[
K_{jk} = n \text{cov}(\psi_n(u_j), \psi_n(u_k)) = \text{Re} \, \phi(u_j - u_k) + \text{Im} \, \phi(u_j + u_k) - \psi(u_j) \psi(u_k). \tag{6}
\]

The weights \( \{w_j | j = 1, \ldots, q\} \) also depend on the parameters \( \alpha, \beta, \sigma, \mu \), and are described in the Numerical Method section. Both Formulations A and B represent modified, weighted \( \chi^2 \) minimum procedures, corresponding to the respective objective functions

A: \( S_n = \sum_{j=1}^{q} \left( \psi(u_j) - \psi_n(u_j) \right)^2 w_j, \tag{7} \)

B: \( Q_n = \sum_{j=1}^{q} \sum_{k=1}^{q} \left( \psi(u_j) - \psi_n(u_j) \right) w_j K_{jk} w_k (\psi(u_k) - \psi_n(u_k)). \tag{8} \)

The following points are critical for practical application:

1) The shapes of \( \psi \) and \( \psi_n \) are highly dependent on location and scale parameters, and so should be standardized;

2) The estimators are improved by making the gridpoints and weights depend on \( \alpha \) and \( \beta \).
3) Since the procedures are adaptive \( \{u_j\}, \{w_j\} \) and \( \{K_{jk}\} \) depend on unknown parameters, algorithms must be iterative;

4) Since \( \alpha, \beta \) and \( \sigma \) are always constrained, each iteration involves solution of a nonlinear optimization problem with variable bound constraints.

Our procedures may therefore be summarized as follows, where a tilde indicates estimators, their values, or adaptively standardized quantities.

Begin with initial guesses for the parameters. At each iteration, compute and save \( \{u_j\}, \{w_j\} \), possibly \( \{K_{jk}\} \), and standardized empirical characteristic function values \( \{\hat{\psi}_n(u_j)\} \), based on the latest \( (\hat{\alpha}, \hat{\beta}) \). The objective \( S_n \) or \( Q_n \) is then minimized (an "optimization subproblem"), and cumulative location and scale estimates \( (\hat{\ell}, \hat{s}) \) are updated. Iteration stops when values of \( \sigma \) and \( \mu \) minimizing \( S_n \) or \( Q_n \) are acceptably close to unity and zero, respectively.

Estimators whose values are not on a bound are asymptotically multivariate Gaussian distributed. The asymptotic covariance matrices \( \Sigma \) of the estimators are derived in Paulson and Delehanty (1982). The basic formula is

\[
\Sigma_i = H_i^{-1} V_i H_i^{-1}, \quad i = A, B. \tag{9}
\]

There are two particularly appealing ways to approximate \( \Sigma \). In "approximation (i)", expectations are approximated from the data: \( H \) is computed by differencing the objective at the final optimum, and

\[
V_A = 4\varphi_k^T K D, \tag{10}
\]

\[
V_B = 4\varphi_k^{-1} f_k(w) K^{-1} D. \tag{11}
\]
Here

\[ D_{ij} = \frac{\partial^2 \psi(u_i)}{\partial \theta \partial \theta'} w_j, \tag{12} \]

\( \theta \) ranging over the parameters free of bounds,

\[ \tilde{K}_{jk} = n^{-1} \sum_{m=1}^{n} (\psi(u_j) - \cos_j \tilde{x}_m - \sin_j \tilde{x}_m)(\psi(u_k) - \cos_k \tilde{x}_m - \sin_k \tilde{x}_m), \tag{13} \]

\[ f(\omega) = R_{jk} w_j w_k. \tag{14} \]

By location and scale invariance, \((\tilde{\omega}, \tilde{\mu})\) are set to \((1, 0)\) during these computations, and \(\sum \) scaled. In "approximation (ii)", expectations are calculated analytically, so \(\tilde{K}\) replaces \(\tilde{k}\) in \((10), (11), \) and \((14), \)

and factors of 2 are omitted. The expected Hessian has elements

\[ H_{A\theta \theta'} = \sum_{j=1}^{q} \frac{\partial^2 \psi(u_j)}{\partial \theta \partial \theta'} w_j, \tag{15} \]

where \(\theta\) and \(\theta'\) range over free parameters.

To analyze domains of attraction, we use what we refer to as the \(k\text{-sum procedure}. If \(k\) is a positive integer, the power

\[ \phi_n^k(u) = n^{-k} \sum_{j=1}^{n} \cdots \sum_{j_k=1}^{n} \exp(\imath u(x_{j_1} + \cdots + x_{j_k})) \tag{16} \]

is the characteristic function corresponding to the \(k\text{-th convolution} \)

power of the empirical distribution function \(F_n(x)\), and can be interpreted as empirical characteristic function of all possible \(k\)-sums

\(\{x_{j_1} + \cdots + x_{j_k}\}\), sampling with replacement from \(F_n(x)\). We add real and imaginary parts and standardize, giving \(\psi_n^k(u)\), say, and estimate \((\tilde{\alpha}_k, \tilde{B}_k, \tilde{\omega}_k)\) for different values of \(k\). If the sample distribution is
attracted to a stable law with parameters \((\alpha, \beta, \sigma, \mu)\), the sequence of normalized estimators \((\tilde{\alpha}_k, \tilde{\beta}_k, \tilde{\sigma}_k/k^{1/2}, \tilde{\mu}_k/k)\), for reasonable values of \(k\), should approach \((\alpha, \beta, \sigma, \mu)\). In particular, a rapid rise in \(\tilde{\sigma}_k\) may indicate that a stable model is not appropriate, a possible alternative being a mixture of finite variance components with differing scale parameters.

The \(k\)-sum procedure can thus be used in a sensitivity analysis, to examine how well the data support the stability assumption. Other possible tools for sensitivity analysis are varying the mechanism (to be described below) underlying the weights \(\{w_j\}\), and comparing approximations (i) and (ii) of the estimated asymptotic covariance matrix, provided \(n\) is large enough for approximation (i) to be accurate.

**NUMERICAL METHOD**

The main computational task required is solution of bound-constrained nonlinear optimization problems. Although Formulations A and B lead to nonlinear least squares problems, current algorithms for nonlinear least squares do not allow constraints (Hiebert, 1981). Numerical Algorithms Group (NAG) subroutine E04KBF (NAG, 1981) is used for optimization. E04KBF is a quasi-Newton procedure, requiring an objective function and analytical first partial derivatives. It is substantially faster than the gradient projection routine used by Paulson, Holcomb and Leitch (1975), although the latter is very reliable. The other complicated numerical procedure required is inversion of a positive definite symmetric matrix \((K, H \text{ or } \tilde{H})\), for which NAG subroutine F01ABF is used. Various NAG utility procedures are also used, see Auxiliary Algorithms. The use of NAG
procedures inhibits transportability in that the algorithm, as presented, is only usable at installations having the NAG Library. However, listings of rapid, high-quality algorithms for constrained optimization have not appeared in the literature (see Chambers, 1977, pp. 159-160; the situation described there has not improved). Given that E04KBF is used, reliance on additional NAG Library procedures is expedient.

We require a minimum of \( q=20 \) gridpoints \( \{u_j\} \), and prefer \( q=20 \) or 40, since they are reasonable values in practice, and have been tested extensively. Only the positive gridpoints are explicitly required, due to symmetry of the grid and the Hermitian property of characteristic functions. They are computed as follows: An endpoint \( U \) is chosen as

\[
3, \hat{a} \geq 1.8; 3.3, 1.8 > \hat{a} > 1.7; 3.6, 1.7 > \hat{a} > 1; 5, \hat{a}=1; 4, 1 > \hat{a} > .9; 5, .9 > \hat{a} > .8; 7, .8 > \hat{a} > .6; 10, \hat{a} < .6.
\]

An inner number \( I \) of points is selected close to the origin: \( I=2 \) if \( q<30 \) and \( 3 \) if \( q \geq 30 \), 1 being subtracted if \( q \leq 5 \). The inner \( I \) points are spaced as follows: if \( \alpha>1 \), the \( \alpha \)-optimal values of Feuerverger and McDunnough (1981b) for the nearest (larger) \( \alpha \) are used; if \( \alpha \leq 1 \), the first \( I \) points giving \( q/2 \) equal increments of \( \log (u + \alpha^2) \) between 0 and \( U \) are multiplied by \( \frac{k}{\alpha} \) \( \alpha^2 \) (\( \alpha = \max(\hat{a}, .3) \)). The remaining points are logarithmically spaced out to \( U \): if \( \hat{a}>1 \), the function \( \log (1 + u/2) \) is used, and if \( \hat{a} \leq 1 \), \( \log (u + \alpha^2) \) is used. This rather complicated ad hoc scheme was developed through graphical inspection of \( \hat{y}(u) \) and \( \hat{y}_n(u) \), comparisons of asymptotic efficiencies, and parameter estimation for real and simulated data. No claims of optimality are made, but the scheme provides high efficiencies if efficiency is preferred, or good matches between \( \hat{y} \) and \( \hat{y}_n \) if curve fitting is preferred. The point of stratified and logarithmic spacing is
to emphasize \( u \) values near the origin. Details when \( a \neq 1 \) reflect the fact that \( \psi(u) \) has a sharp cusp near the origin, but decays slowly. The stepwise nature of the scheme is not deemed a serious drawback.

The weights \( \{ w_j \} \) are computed as follows:

Under Formulation A,

\[
W_j = \frac{\left| \phi(u_j) \right|^{2\lambda}}{K_t(u_j, u_j)} = \frac{\exp(-2\lambda |u_j|^\alpha)}{K_t(u_j, u_j)},
\]

and under Formulation B,

\[
W_j = \left| \phi(u_j) \right|^{\lambda} = \exp(-\lambda |u_j|^\alpha),
\]

where \( \lambda \) and \( \tau \) are supplied by the user, \( 0 \leq \tau \leq 1 \),

\[
K_t(u, u) = 1 + \tau(\text{Im} \tilde{\phi}(2u) - \tilde{\phi}^2(u)),
\]

and \( \lambda \) is recommended nonnegative. Rationale for these weights, and some corresponding asymptotic efficiencies, are in Paulson and Delehanty (1982). We recommend \( \tau = 1 \) under Formulation A. Under Formulation B, it is convenient to let \( \tau > 0 \) represent a fraction of the average diagonal element by which to inflate \( K \), giving a matrix we shall call \( A \). We have only found this inflation necessary if \( a \) is very close to two, when \( \tau = 0.01 \) suffices.

To use the quantity \( \lambda \) as a tool for sensitivity analysis, we interpret it as a damping factor, lessening the effects of noise in \( \tilde{\psi}_n(u) \) for larger \( |u| \). If the data are truly stable and the sample size is fairly large (say 150 or more), estimates should change little as \( \lambda \) varies, say, from 0 to 1. Large discrepancies in the estimates for different values of \( \lambda \) indicate problems with the data or the stable assumption or both. It may not be easy to isolate the difficulty but further study is
definitely required.

For the k-sum procedure, k>1, the situation regarding gridpoints and weights changes. Tests so far indicate that when \( \tilde{a} \geq 1 \), Formulation A, with gridpoints equispaced from 0 to U, gives better results than "efficient configurations" used for k=1. Reasons for this are unclear. A possible explanation is that when \( \tilde{a} \geq 1 \) and k>1, \( \tilde{\psi}_n^k(u) \) is so smooth that estimation is practically equivalent to deterministic curve fitting, and implicit or explicit emphasis on gridpoints near the origin neglects important curvature for large \(|u|\). Accordingly, when k>1 and \( \tilde{a} \geq 1 \), we equispace gridpoints and set all weights to 1. When \( \tilde{a} \leq 1 \), \( \tilde{\psi}_n^k(u) \) has a sharp cusp near the origin and remains a jagged curve as k increases, due to the presence of very large observations. In this situation, we set all weights to 1 and use basically the same gridpoint scheme as when k=1, omitting only multiplication of the inner points by \( \alpha^{2/4} \). In either case, Formulation A is recommended.

An important question is how large k may be taken. Equation (16) suggests that we cannot expect to take k arbitrarily large. There seems to be a tendency for \( \tilde{a} \) to increase and \( \tilde{b} \) to drift if k is too large, though this may be partially due to suboptimal gridpoints or weighting. It appears that when n is large, say 500 or more, k may safely be taken up to 20. Care is required for smaller n, and when \( \alpha \) is small or very near two.

Implicit standardization is carried out as follows. Let k be a positive integer, and \((\tilde{\lambda}, \tilde{s})\) cumulative location and scale estimates. Then
\[ \psi_n^{j,k}(u_j) = \rho_{jk}(\cos \gamma_{jk} + \sin \gamma_{jk}), \quad (20) \]

where
\[ \rho_{jk} = |\phi_n(u_j/\tilde{s})|^k \quad (21) \]
and
\[ \gamma_{jk} = k \arg \phi_n(u_j/\tilde{s}) - \tilde{z}_u_j/\tilde{s}. \quad (22) \]

No problems of principal values arise, and complex arithmetic is not used. The FORTRAN mathematical library function ATAN2 computes arguments.

The estimator \( \hat{\alpha} \) may be bounded in (closed) subintervals of \([\delta,1-\epsilon],[1,1], \) or \([1+\epsilon,2] \) unless \( \hat{B} \) is fixed at 0, when \([\delta,2] \) is possible (\( \delta \) and \( \epsilon \) are small positive numbers), while \( \hat{B} \) may be bounded in subintervals of \([-1,1] \). Estimators \( \hat{\sigma} \) and \( \tilde{\mu} \) may be constrained arbitrarily in \([\delta,\infty) \) and \((-\infty,\infty) \), respectively, unless \( \hat{a} \) is fixed at 1 and \( \hat{B} \) is not fixed at 0, when \( \tilde{\mu} \) cannot be constrained. Bounds on \( \sigma \) and \( \mu \) are internally set for use in subproblems. These bounds must be wide enough to allow the "true values" to be found, but narrow enough to deter straying into undesirable regions, particularly \( \sigma = \infty, \) \( |\mu| = \infty \). The ad hoc bounds of \([-5,5] \) for \( \mu \) and \([0.2,5] \) for \( \sigma \) work well in practice. If \( \hat{\sigma} \) or \( \tilde{\mu} \) are initially constrained, their internal bounds are adaptively modified, see the Algorithm for description.

Initial guesses for the parameters are required. We do not find their specification particularly important, provided \( \hat{\alpha} \) is on the correct side of 1 in the nonsymmetric case. We have used the median and semi-interquartile range as guesses for \( \tilde{\mu} \) and \( \hat{\sigma} \), and averages of upper and lower bounds for \( \hat{\alpha} \) and \( \tilde{\sigma} \). If \( \hat{\alpha} \) is anticipated less than 1.2, say, it is
worthwhile to put more effort into initial guesses, since fewer iterations will be required (the semi-interquartile range will overestimate \( \sigma \), and if \( \alpha \) is near but different from 1, the median is nearer \( \mu - \frac{2}{\pi} \beta \sigma \log \sigma \) than \( \mu \)).

Convergence is judged by a tolerance on subproblem solutions, 
\[
\max(|\alpha^{(m)} - |\alpha^{(m-1)}|, |\beta^{(m)} - |\beta^{(m-1)}|) \text{ if } \check{\sigma} \text{ and } \check{\mu} \text{ are fixed},
\]
with a maximum allowable number of iterations. Attainable tolerances depend on \( n \), but more strongly on the underlying parameters. If \( \check{\alpha} \) is near two, \( \check{\psi}_n \) is very smooth and stringent tolerances such as \( 10^{-6} \) may be attained. If \( \check{\alpha} \approx 1.2 \), \( \check{\psi}_n \) has many small oscillations due to large observations, and, especially for smaller samples, it may be preferable to terminate after a fixed number of iterations. Good estimates are generally obtained within five iterations, fewer if initial guesses are good; if stringent tolerances are required, or for difficult problems (skewed distributions with \( 0.9 \leq \check{\alpha} \leq 1.1 \)) more may be required. Convergence is typically slower under Formulation B, since the weighting mechanism is more complicated.

Approximation of asymptotic covariance matrices requires little description. We note that for approximation (i) and the \( q \) values we use, it is faster to define a vector
\[
\delta_j^T = (\check{\psi}(u_1) - \cos u_1 \check{x}_j - \sin u_1 \check{x}_j, \cdots, \check{\psi}(u_q) - \cos u_q \check{x}_j - \sin u_q \check{x}_j),
\]
and cumulate
\[
\check{\psi} = \frac{1}{n} \sum_{j=1}^{n} (\check{p}^T \delta_j) (\check{p}^T \delta_j)^T
\]
under Formulation A, or
\[ \hat{V} = 4n^{-1} \sum_{j=1}^{n} (g^T A^{-1} \delta_j)(g^T A^{-1} \delta_j)^T \]  

(25)

under Formulation B, than to cumulate \( \hat{R} \). The matrix \( D \) is computed by

the function/gradient subroutine. E04KBF returns an approximate Hessian, which could conceivably be used for \( \hat{H} \) in approximation (i). Rather

often, however, E04KBF will terminate with its failure indicator set to

3 and the Hessian set to the identity matrix, even though the optimum may be reliable. It is therefore simpler to compute \( \hat{H} \) by differencing.

The following procedures is used: Set an initial Hessian to 0, and the

steplength to \( 10^{-3} \). Successively divide the steplength by \( \sqrt{10} \) and approx-

imate the Hessian by differencing; three-point differencing for the diagonal, and four-point for off-diagonal elements. Compare elements of successive approximations by maximum relative or absolute differences, according as the element of the latest approximation exceeds 1 in absolute value or not. A tolerance of \( 10^{-6} \) is used for this convergence criterion. If convergence has not occurred with a steplength of \( 10^{-5} \), the result with steplength \( 10^{-4} \) is used.

Approximation (i) of the asymptotic covariance matrix is rather expensive to compute. It should not be computed for smaller sample sizes, as it implicitly involves estimation of \( b_q(b_q+1) \) covariances.

Following is an informal description, in Algorithm form, of the basic routine STABLE. Approximate asymptotic covariance matrices may also be computed, but this presents no logical difficulties, so is omitted.
Algorithm

Produces estimates \((\tilde{\sigma}, \tilde{\beta}, \tilde{\gamma}, \tilde{u})\) for k-sums \((k \geq 1)\), based on a sample \((x_1, \ldots, x_n)\).

Input parameters: \(k, n, \{x_j\}, q, \lambda, \tau,\) Formulation \((A\) or \(B)\), convergence tolerance \(\varepsilon\), maximum number \(M\) of iterations, and flags whether \(\tilde{\sigma}\) and \(\tilde{u}\) are constrained.

Input/output parameters: \((\tilde{\sigma}, \tilde{\beta}, \tilde{\gamma}, \tilde{u})\) are initial guesses on entry and estimates on exit, \((\sigma_L, \sigma_u, \mu_L, \mu_u)\) and \((\sigma_L, \sigma_u, \mu_L, \mu_u)\) are lower bounds. The \((\sigma, \mu)\) bounds are changed, but restored on exit. In the special case where \(\tilde{\alpha}\) is fixed at 1, \(\tilde{u}\), on entry, is the initial guess for location \(u + \frac{2}{\pi} \log \sigma\).

Auxiliary quantities \(\tilde{\zeta}\) and \(\tilde{s}\) are cumulative location and scale estimates. Entry values of \((\sigma_L, \sigma_u, \mu_L, \mu_u)\) are stored in \((b_1, b_2, b_3, b_4)\). On entry and exit, \((\tilde{\sigma}, \tilde{u}, \sigma_L, \mu_L, \sigma_u, \mu_u)\) are normalized.

S1 [Initialize.]

Set \(\tilde{\xi} = k \tilde{u}, \tilde{s} = k^{1/\tilde{\alpha}} \tilde{\sigma}, m = 0.\)

Save \((b_1, b_2, b_3, b_4) = (\sigma_L, \sigma_u, \mu_L, \mu_u)\).

if \(\tilde{u}\) is unconstrained then set \(\mu_L = \tilde{u}, \mu_u = 5;\)
else if \(\tilde{u}\) is fixed then set \(\mu_L = \mu_u = 0;\)
else set \(\mu_L = k \mu_L, \mu_u = k \mu_u.\)

if \(\tilde{\sigma}\) is unconstrained then set \(\sigma_L = 0.2, \sigma_u = 5;\)
else if \(\tilde{\sigma}\) is fixed then set \(\sigma_L = \sigma_u = 1;\)
else set \(\sigma_L = k^{1/\tilde{\alpha}} \sigma_L, \sigma_u = k^{1/\tilde{\alpha}} \sigma_u.\)
S2  [Looping point for iteration; save adaptive quantities for sub-problem.]

Increment \( m + 1 \).

Save \( \alpha^{(m-1)} + \tilde{\alpha}, \beta^{(m-1)} + \tilde{\beta} \).

\[ \text{if} \ \tilde{\sigma} \ \text{is constrained but not fixed then set} \]
\[ \sigma_L = \max(0.2, b_1 / \delta), \]
\[ \sigma_u = \min(5, b_2 / \delta). \]

\[ \text{if} \ \tilde{\mu} \ \text{is constrained but not fixed then set} \]
\[ \mu_L = \max(-5, (b_3 - \tilde{\ell}) / \delta), \]
\[ \mu_u = \min(5, (b_4 - \tilde{\ell}) / \delta). \]

Set \( \tilde{\delta} + 1 \), \( \tilde{\mu} + 0 \).

Compute and save positive gridpoints \( \{ u_j | j = q/2 + 1, \ldots, q \} \),
weights \( \{ w_j | j = 1, \ldots, q \} \), and standardized empirical characteristic
function values \( \{ \psi_n^k(u_j) | j = 1, \ldots, q \} \).

\[ \text{if Formulation B} \ \text{then compute and invert} \ A. \]

S3  [Subproblem.]

Solve the optimization problem

\[ \min_j \sum_{j=1}^{q} (\psi(u_j) - \tilde{\psi}_n^k(u_j))^2 w_j \]  \ (Formulation A)

or

\[ \min_{i=1}^{q} \sum_{j=1}^{q} w_i (\psi(u_i) - \tilde{\psi}_n^k(u_j)) A^{ij} (\psi(u_j) - \tilde{\psi}_n^k(u_j)) w_j \]  \ (Formulation B),

yielding new \( (\tilde{\alpha}, \tilde{\beta}, \tilde{\sigma}, \tilde{\mu}) \).

Su  [Update and test convergence.]

Set \( \tilde{\ell} = \tilde{\ell} + \tilde{s} \ \tilde{\sigma} \).

\[ \text{if} \ \tilde{\sigma} = 1 \ \text{then set} \ \tilde{\ell} = \tilde{\ell} + \frac{2}{\pi} \tilde{\sigma} \ \tilde{\sigma} \ \log \tilde{\sigma}. \]
Set $\bar{s} + \bar{\alpha} \bar{\phi}$.

**If** $\bar{\alpha}$ and $\bar{\mu}$ are fixed then error $+ \max (|\bar{\alpha} - \bar{\alpha}^{(m-1)}|, |\bar{\beta} - \bar{\beta}^{(m-1)}|)$;

**else** error $+ \max (|\bar{\alpha} - \bar{\alpha}|, |\bar{\mu}|)$.

**If** error $\geq \epsilon$ and $\mathbf{m} < M$ then go to S2.

**S5** [Final estimates.]

Set $(\sigma_L, \sigma_u, \mu_L, \mu_u) = (b_1, b_2, b_3, b_4)$.

**If** $\bar{\alpha} = 1$ then set $\bar{\zeta} + \bar{\zeta} - \frac{2}{\pi} \bar{\beta} \bar{s} \log \bar{s}$.

Set $\bar{\mu} + \frac{\bar{\zeta}}{k}$, $\bar{s} + \frac{\bar{\zeta}}{k^{1/\bar{\alpha}}}$.
SUBROUTINE STABLE (X,N,MODE,KSUM,XLAM,TAU,NPTS,TOL,MAXIT,XL,XH,NPAR, ISCLBD,LOCBND,ICOV,VCV1,VCV2,WORK,LWORK,IWORK,LIWORK,IFault)

Formal parameters

X  Real array (N)           input: sample
N  Integer                 input: sample size
MODE Integer              input: formulation; if zero, then Formulation B is used, else Formulation A
KSUM Integer              input: convolution power k
XLAM Real                 input: \( \lambda \)
TAU Real                  input: \( \tau \)
NPTS Integer              input: q
TOL Real                  input: convergence tolerance
MAXIT Integer             input: maximum allowable number of iterations
XL Real array (NPAR)      input: lower bounds for parameters \( (a, \beta, \sigma, \mu) \); the third and fourth elements change during execution output: input values are restored
XH Real array (NPAR)      input: upper bounds for parameters; the third and fourth elements change during execution output: input values are restored
NPAR Integer              input: number of parameters (4)
ISCLBD Integer           input: flag if \( \tilde{\sigma} \) is constrained: if negative, \( \tilde{\sigma} \) is fixed at \( XB(3) \); if zero, \( \tilde{\sigma} \) is free to vary and initial values of XL(3) and XH(3) are irrelevant; if positive, \( \tilde{\sigma} \) is constrained in \([XL(3),XH(3)]\)
LOCBND Integer input: flag if $u$ is constrained: if negative, $u$ is fixed at $X_B(4)$; if zero, $u$ is free to vary and initial values of $XL(4)$ and $XH(4)$ are irrelevant; if positive, $u$ is constrained in $[XL(4),XH(4)]$

ICOV Integer input: flag for computation of covariance matrices: if negative, neither approximation (i) nor (ii) is computed; if zero, both approximations are computed; if positive, only approximation (ii) is computed

VCV1 Real array(NPAR,NPAR) output: covariance matrix approximation (i) if requested; the strict lower triangle contains correlations, the upper triangle contains covariances (times $n$); if a parameter is on a bound, the corresponding elements are zero

VCV2 Real array(NPAR,NPAR) output: covariance matrix approximation (ii) if requested; the strict lower triangle contains correlations, the upper triangle covariances (times $n$); if a parameter is on a bound, the corresponding elements are zero

WORK Real array (LWORK) workspace: output: some elements may be of interest on output (see Restrictions)

LWORK Integer input:

IWORK Integer array(LIWORK) workspace: output: some elements may be of interest on output (see Restrictions)

LIWORK Integer input:

IVNIT Integer input: if positive, unit number for output (see Additional Comments); if zero or negative, no output is produced

IFault Integer output: failure indicator
Failure indicators

IFault = 0 indicates success. Nonzero values of IFAULT are due to two types of errors. The first type is input errors, detected in STABLE;

IFault will be

1. if MAXIT≤0;
2. if N<50 (see Restrictions);
3. if KSUM ≤ 0;
4. if τ<0 or τ>1 and MODE≠0;
5. if NPTS<20 or mod(NPTS,2)≠0;
6. if TOL≤0;
7. if NPAR≠4;
8. if insufficient workspace was allotted (see Restrictions);
9. if improper bounds were supplied. The following conditions cause this failure:
   XL(i)>XB(i) or XL(i)>XH(i) or XB(i)>XH(i), i=1,2
   XL(1)≤0 or XH(1)>2
   XL(1)<-1 or XH(1)>1
   (XL(2)≠0 or XH(2)≠0) and (XL(1)<1 and XH(1)≥1
   or XL(1)≤1 and XH(1)>1
   XB(3)≤0
   ISCLBĐ0 and (XL(3)>XB(3) or XL(3)>XH(3) or XB(3)>XH(3))
   ISCLBD<0 and XL(3)≠XH(3)
   LOCBD<0 and (XL(4)>XB(4) or XL(4)>XH(4) or XB(4)>XH(4))
   LOCBD<0 and XL(4)≠XH(4)
   LOCBD<0 and XL(1)=XH(1)=1 and (XL(2)≠0 or XH(2)≠0).

On input errors, STABLE terminates immediately, without performing any computations. The second type of error occurs after some computation.

IFault will be

10. if A was found numerically non positive definite;
11. if A was found ill-conditioned;
12. if too many function evaluations were required during solution of a subproblem;
13. if iteration converged, but the most recent E04KBF fault indicator was 3 and internal checks were not met. These checks are
   (i) ||c||_2 < 10 * X02AAF(DUMMY), and
   (ii) K < 1/||G||, as recommended by E04KBF documentation, where ||G|| is the norm of the projected gradient and K the estimated condition number of the projected Hessian matrix;
if there were repeated problems with overflow in the Cholesky factors of the projected Hessian;

if iteration converged, but the most recent E04KBF fault indicator was 5 and internal checks were not met;

if convergence did not occur in MAXIT iterations;

if convergence did not occur in MAXIT iterations, the most recent E04KBF fault indicator was 3, and internal checks were not met;

if convergence did not occur in MAXIT iterations, the most recent E04KBF fault indicator was 5, and internal checks were not met.

Conditions IFAULT=10 and 11 are detected in SETECF (they are caused by $\tau$ being too small under Formulation B), the remainder in STABLE.

IWORK(2) and IWORK(3) (see Restrictions) are failure indicators for asymptotic covariance matrix versions (i) and (ii) respectively. Zero indicates success, 1 that $H$ was non positive definite, and 2 that $H$ was ill-conditioned, the failures detected in SETVCV. If IFAULT=1-12 or 14, covariance matrices are not computed, and their fault indicators are set to the corresponding value of IFAULT.

Auxiliary algorithms

The user has only to call STABLE. Auxiliary procedures fall into two groups: those supplied here, and NAG Library procedures. The following subroutines are supplied:

SUBROUTINE GRIDWT(PAR,NPAR,XLAM,TAU,PTS,NPTS2,WT,NPTS,MODE,KSUM): computes gridpoints and weights;

SUBROUTINE CHARFN(U,PAR,NPAR,RE,XIM): computes real and imaginary parts of standard stable characteristic function $\Phi(u)$;

SUBROUTINE FUNCT(IFLAG,N,XC,FC,GC,IW,LIW,W,LW): objective function/gradient evaluation;
SUBROUTINE SETECF(X,N,PAR,NPAR,MODE,TAU,SIGMA,XMU,KSUM,IA,NPTS2,NPTS,PTS,ECF,A,AINV,WORK,IFault): computes standardized empirical characteristic function values \( \hat{\Psi}(u) \), computes and inverts A under Formulation B;

SUBROUTINE MONIT(N,XC,FC,GC,ISTATE,GPJNRM,COND,POSDEF,NITER,NF,IW,LIW,W,LW): monitors the progress of E04KBF;

SUBROUTINE VARIAB(ICOV,X,N,PAR,NPAR,MODE,SIGMA,XMU,ISUB,NVAR,PTS,NPTS2,WT,ECF,NPTS,DERIV,WORK,HOLD,A,IA,AINV,VCV1,VCV2,H,NVAR1,V,IV,W,LW): computes approximate asymptotic covariance matrices;

SUBROUTINE VMATRIX(X,N,MODE,XMU,SIGMA,PTS,NPTS2,WT,ECF,WORK,NPTS,DERIV,V,HOLD,NVAR): computes \( \hat{\Psi} \) for version (i) of asymptotic covariance matrix;

SUBROUTINE DAPROD(FAC1,IFAC1,NPTS,FAC2,WORK,NVAR): auxiliary matrix multiplication for VARIAB;

SUBROUTINE HVPROD(FAC1,IFAC1,NVAR,FAC2,NPTS,VH,IVH): auxiliary matrix multiplication for VARIAB;

SUBROUTINE SETVCV(ISUB,NVAR,H,NVAR1,V,WORK,VCV,NPAR,SIGMA,IFAULT): auxiliary routine for VARIAB;

SUBROUTINE HESDIF(PAR,NPAR,ISUB,H,SAVEI,SAVE2,NVAR,IW,LIW,W,LW): computes an approximate Hessian by differencing for version (i) of asymptotic covariance matrix.

The following NAG Library procedures are used:

REAL FUNCTION X02AAF(DUMMY): returns the smallest positive \( \varepsilon \) such that \( 1.0 + \varepsilon > 1.0 \);

SUBROUTINE E04KBF(N,FUNCT,MONIT,IPRINT,LOCSCCH,INTYPE,MINLIN,MAXCAL,ETA,XTOL,STEPMX,FEST,IBOUND,BL,BU,X,HESL,LH,HESD,ISTATE,F,G,IW,LIW,W,LW,IFAIL): solves optimization problems. Control parameters are set as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPRINT</td>
<td>0</td>
</tr>
<tr>
<td>LOCSCCH</td>
<td>.TRUE.</td>
</tr>
<tr>
<td>INTYPE</td>
<td>3 for subproblems after the first if parameters which are not fixed are not on bounds, else 0</td>
</tr>
<tr>
<td>MINLIN</td>
<td>NAG Library routine E04LBS</td>
</tr>
<tr>
<td>MAXCAL</td>
<td>400</td>
</tr>
<tr>
<td>ETA</td>
<td>0.9</td>
</tr>
<tr>
<td>XTOL</td>
<td>( 10.0 \sqrt{X02AAF(DUMMY)} ) explicitly, so it is available on exit</td>
</tr>
<tr>
<td>STEPMX</td>
<td>0.25</td>
</tr>
<tr>
<td>FEST</td>
<td>0.0</td>
</tr>
<tr>
<td>IBOUND</td>
<td>0</td>
</tr>
</tbody>
</table>

SUBROUTINE F01ABF(A,IA,N,B,IB,Z,IFAIL): inverts the positive definite symmetric matrix A;
SUBROUTINE FOCAF(A,M,N,IFAIL): sets matrix A to zero;

SUBROUTINE FO1CMF(A,LA,B,LB,M,N): copies elements of matrix A into matrix B;

SUBROUTINE FO1CKF(A,B,C,N,IP,M,Z,IZ,IOPT,IFAIL): matrix multiplication A=BC, where B or C may be overwritten.

RESTRICTIONS

We require the sample size N at least 50, since for smaller samples \( \hat{\psi}_n(u) \) is not generally sufficiently smooth to allow accurate estimation. Since \( \hat{\alpha} \) and \( \hat{\beta} \) are bounded in the narrow ranges \((0,2]\) and \([-1,1]\) and have standard errors decreasing as \( N^{-3} \), it is preferable to have \( N \geq 100 \). For \( N \) less than 150, say, relatively large values of \( \lambda \) may be preferred, to damp out noise in \( \hat{\psi}_n(u) \). We further require \( NPTS \geq 20 \).

Extended work vectors WORK and IWORK are required, in order to communicate information to FUNCT and MONIT without using COMMON blocks. To aid readers who may wish to adapt the algorithm to installations not having the NAG Library, we describe the use of these work vectors.

The required length of WORK is

\[
10 + 11NPAR + NPAR(NPAR-1)/2 + (3+NPAR)NPTS + NPTS + NPTS/2 \text{ if } MODE=0, \text{ with an additional } NPTS^2(2NPTS+1)
\]

required if \( MODE=0 \). Some sample lengths are

<table>
<thead>
<tr>
<th>MODE</th>
<th>NPTS=20</th>
<th>NPTS=40</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1030</td>
<td>3600</td>
</tr>
<tr>
<td>nonzero</td>
<td>210</td>
<td>360</td>
</tr>
</tbody>
</table>

The subvector \( W \) is passed to E04KBF, FUNCT, and MONIT.
<table>
<thead>
<tr>
<th>WORK starting point</th>
<th>W starting point</th>
<th>Elements</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
<td>Convergence criterion</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>1</td>
<td>Objective function value on exit from E04KBF</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>NPAR</td>
<td>Projected gradient on exit from E04KBF</td>
</tr>
<tr>
<td>(other addresses internally computed)</td>
<td>-</td>
<td>1</td>
<td>Old $\alpha$ value for convergence testing</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>1</td>
<td>Old $\beta$ value for convergence testing</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>1</td>
<td>$\bar{\alpha}$ lower bound on entry</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>1</td>
<td>$\bar{\beta}$ upper bound on entry</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>1</td>
<td>$\bar{\mu}$ lower bound on entry</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>1</td>
<td>$\bar{\mu}$ upper bound on entry</td>
</tr>
<tr>
<td></td>
<td>NPAR*(NPAR-1)/2</td>
<td>HESL factor for E04KBF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPAR</td>
<td>HESD factor for E04KBF; workspace for VARIAB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9*NPAR</td>
<td>E04KBF workspace; broken up into H and V matrices and used as workspace in VARIAB</td>
<td></td>
</tr>
<tr>
<td>(other addresses internally computed)</td>
<td>1</td>
<td></td>
<td>On exit from E04KBF, estimated condition number of projected Hessian</td>
</tr>
<tr>
<td></td>
<td>NPTS/2</td>
<td>Positive gridpoints PTS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPTS</td>
<td>Weights WT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPTS</td>
<td>Empirical characteristic function values ECF, workspace in VARIAB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPAR*NPTS</td>
<td>Partial derivatives of $\Psi(u)$ at gridpoints, workspace in VARIAB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPTS</td>
<td>Workspace for FUNCT and VARIAB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(NPTS+1)*NPTS</td>
<td>If MODE=0, used for A matrix (the extra row is required by NAG Library routine F01ABF); workspace in VARIAB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPTS*NPTS</td>
<td>If MODE=0, used for $A^{-1}$; workspace in VARIAB</td>
<td></td>
</tr>
</tbody>
</table>
The required length of IWORK is 7 + NPAR. The subvector IW is passed to E04KBF, FUNCT, and MONIT.

<table>
<thead>
<tr>
<th>IWORK starting point</th>
<th>IW starting point</th>
<th>Elements</th>
<th>Use for</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
<td>Iteration count</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>NPAR</td>
<td>ISTATE vector for E04KBF, workspace for VARIAB.</td>
</tr>
</tbody>
</table>

If covariance matrices are requested, on exit IWORK(2) contains a fault indicator for approximation (i), IWORK(3) contains a fault indicator for approximation (ii), and IWORK(4) contains the number of iterations required to compute the approximate Hessian for approximation (i).

Workspace for E04KBF, HESDIF

1Stores MODE

4Stores output unit number IUNIT

5Stores NPTS

6Stores 1 less than the address of PTS(1) in W

PRECISION

Double precision will be required on computers with 32 bit wordlength.

The precision used by the local NAG Library implementation should be adequate. To change the precision:

- change all REAL declarations to DOUBLE PRECISION;
- replace constants by double precision versions, constants \( \pi, \frac{2}{\pi}, \sqrt{10} \) typed in to machine accuracy;
- declare NAG Library function X02AAF as DOUBLE PRECISION;
- change the precision of FORTRAN library functions, i.e., ABS to DABS, ATAN2 to DATAN2, SIGN to DSIGN, etc. FLOAT(I) can be replaced by DBLE(FLOAT(I)).
If extremely large observations are present in the sample, there may be a loss of significant figures when computing sines and cosines in SETECF and VMATRX. This should not occur when real data is used, but can be a problem with simulated data for small $a$.

**TIME**

Execution times depend on the quality of initial guesses and properties of the real data used, and vary somewhat throughout the parameter space. As a rough guide, we give some statistics for simulated data, using a moderately difficult situation with $a>1$. Tables 1a and 1b provide approximate running times for Formulation A, $q=40$, and Formulation B, $q=20$, $n=100, 200, 500, 1000, 2500$. Timing starts upon entry to STABLE. Samples from $S(1.3, -0.5, 3, 15)$ were generated using the method of Chambers, Mallows, and Stuck (1976). Initial guesses for $a, \beta, \sigma, \mu$ in all cases were $1.05 = \frac{1}{2}(1.01+2), 0, \frac{1}{2}(x_{0.75} - x_{0.25})$, and $x_{0.5}$, the sample median, respectively. Because of skewness, the median is not a good estimator of $\mu$ in this case. Five iterations were used. Time required to compute asymptotic covariance matrices includes approximations (i) and (ii), except where noted. Timings are for a double precision version of the algorithm, compiled by the IBM FORTRAN H Extended compiler, and run on an IBM 370/3033.

The following qualitative points are clear from this rather restricted set of timings. There is a substantial overhead, which may crudely be assumed fixed, associated with nonlinear optimization, although E04KBF solves the optimization subproblems rapidly. For large samples, run time is dominated by evaluation of the empirical characteristic function, and thus is asymptotically linear in $n$ for a fixed number of iterations.
Table 1a
Timings for Formulation A, q=40, on Simulated Samples
from s(1.3, -.5, 3, 15); λ=1 for n<200 and .5 for n>200, τ=1

<table>
<thead>
<tr>
<th>n</th>
<th>Iterations</th>
<th>Estimation time (sec)</th>
<th>Convergence criterion</th>
<th>Covariance matrix time</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>5</td>
<td>0.7</td>
<td>5.4(-4)</td>
<td>0.1*</td>
</tr>
<tr>
<td>200</td>
<td>5</td>
<td>1.0</td>
<td>2.3(-2)</td>
<td>0.1*</td>
</tr>
<tr>
<td>500</td>
<td>5</td>
<td>1.8</td>
<td>1.8(-4)</td>
<td>0.8</td>
</tr>
<tr>
<td>1000</td>
<td>5</td>
<td>3.2</td>
<td>3.3(-5)</td>
<td>1.2</td>
</tr>
<tr>
<td>2500</td>
<td>5</td>
<td>7.5</td>
<td>4.8(-6)</td>
<td>2.5</td>
</tr>
</tbody>
</table>

*Sample size too small to compute approximation (i), only approximation (ii) computed.

Table 1b
Timings for Formulation B, q=20, on Simulated Samples
from s(1.3, -.5, 3, 15); λ=τ=0

<table>
<thead>
<tr>
<th>n</th>
<th>Iterations</th>
<th>Estimation time (sec)</th>
<th>Convergence criterion</th>
<th>Covariance matrix time</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>5</td>
<td>0.9</td>
<td>1.2(-2)</td>
<td>0.3</td>
</tr>
<tr>
<td>200</td>
<td>5</td>
<td>1.2</td>
<td>3.2(-2)</td>
<td>0.4</td>
</tr>
<tr>
<td>500</td>
<td>5</td>
<td>1.6</td>
<td>1.8(-4)</td>
<td>0.5</td>
</tr>
<tr>
<td>1000</td>
<td>5</td>
<td>2.3</td>
<td>5.7(-5)</td>
<td>0.7</td>
</tr>
<tr>
<td>2500</td>
<td>5</td>
<td>4.4</td>
<td>1.5(-4)</td>
<td>1.4</td>
</tr>
</tbody>
</table>
Approximation (ii) of the asymptotic covariance matrix is quite easy to compute, while approximation (i) is highly time-consuming.

For fixed $k>1$ with the $k$-sum procedure, one iteration generally suffices, provided estimates from the nearest value of $k$ are used, and the estimates don't change much. For mixtures of very different distributions, or if the exponent $\alpha$ is near unity, more are required.

**ADDITIONAL COMMENTS**

Although output need not be produced, we recommend calling STABLE with IUNIT>0, so the user will have a record of how estimation progressed.

The following information will then be printed out:

- by MONIT: number of E04KBF iterations and function evaluations, objective function value, norm of projected gradient, subproblem solution, projected gradient, and estimated condition number of projected Hessian;

- by STABLE: E04KBF fault indicator, and value of convergence criterion;

- by HESDIF(if called): number of iterations needed to compute approximate Hessian, and steplength used.

Use of STABLE in "batch mode" has drawbacks. For instance, most faults arising in E04KBF are not diagnosed until iteration ceases. In practice, such faults may likely be due to the initial $\alpha$ being on the wrong side of 1. Further, when $\alpha$ is small, convergence tolerances are difficult to interpret, and the user may prefer direct control of iteration. We therefore prefer to use STABLE interactively, a copy of the output described above being directed to the terminal, and the user deciding after each iteration whether he wishes to continue. Required modifications are simple.
Faster and/or more compact codings of the Algorithm are possible, for instance, if \( B \) is known to be zero, if only Formulation A or Formulation B is desired, or if asymptotic covariance matrices are not desired. Generality is achieved at a price in efficiency.
REFERENCES


C**************************************************************************************STAB 001
C CENTRAL SUBROUTINE OF ESTIMATION PROCESS.
C CALLED BY - MAIN
C CALLS - GRIDWT, SETCF, VARIAB
C CALLS TO EONKBF VIA EXTERNAL SMTH - FUNCT, MONIT, EONLBS
C N.A.G. PROCEDURES CALLED -
C X02AAF (RETURNS MACHINE PRECISION),
C EONKBF (QUASI-NEWTON OPTIMIZATION WITH BOUNDED VARIABLES),
C F01CAF (SETS A MATRIX TO ZERO).
C**************************************************************************************STAB 010
C**************************************************************************************STAB 011
SUBROUTINE STABLE(X, N, MODE, KSUM, XLAM, TAU, NPTS, TOL, MAXIT,
*X, XB, XN, NPAR, ISLBD, LOCBND, ICOD, VCVI, VCV2, WORK, LWKWF,
*IMWK, LWKRF, UNIT, IFault)
C**************************************************************************************STAB 014
C EXTERNAL ROUTINES
C EXTERNAL EONLBS, FUNCT, MONIT
C**************************************************************************************STAB 016
C ARGUMENTS
C INTEGER N, MODE, KSUM, NPTS, MAXIT, NPAR, ISLBD, LOCBND, ICOD,
C *IMWK, LWKRF, IMWK(LWRF), UNIT, IFault
C REAL X(N), XLAM, TAU, TOL, XL(NPAR), XB(NPAR), XH(NPAR),
C *VCV1(NPAR, NPAR), VCV2(NPAR, NPAR), WORK(LWRF)
C**************************************************************************************STAB 021
FUNCTION CALLED
REAL X02AAF
C**************************************************************************************STAB 023
C LOCAL SCALARS
C SPECIFICALLY FOR EONKBF PARAMETERS -
C******************************************************************************STAB 025
LOGICAL LOCSCN
INTEGER LW, LIV, MAXCAL, IBOUND, IPRINT, LHESL, INTYPE
REAL TA, XTL, TOL, STEPMX, FEST
C SCRATCH VARIABLES, SUBSCRIPT INDICATORS, AND CONSTANTS -
C INTEGER INO, INDI, IOFRLW, NPTS2, IA, NVAR, NVAR, NPTS, NMT,
C *MECF, MDERIV, MVWORK, MA, MAINV, MHESL, MHESD, MIV, MV, MV
C REAL SIGMA, XNU, XK, ZERO, PI2, THOVP1, ONE, TWO, SQRT10, FIVE,
C *TEN
C DATA LOCSCN, MAXCAL, IBOUND, IPRINT, ETA, STEPMX, FEST
C /* TRUE, NOD, 0, 0, 0, 0, 0.2, 0.0
C * TRUE, NOD, 0, 0, 0, 0, 0.2, 0.0
DATA ZERO, PI2, THOVP1, ONE, TWO, SQRT10, FIVE, TEN
*/0.0, 0.2, 0.6366197724, 1.0, 2.0, 3.162277660, 5.0, 10.0/
C**************************************************************************************STAB 037
C ON INPUT ERRORS (IFault = 1 - 9), EXIT IMMEDIATELY.
C**************************************************************************************STAB 039
C WORK(1) = 0
C WORK(1) = ZERO
C IFault = 0
C IF (MAXIT .LE. 0) RETURN
C IFault = 1
C IF (MAXIT .LE. 0) RETURN
C IFault = 2
C IF (N .LT. 50) RETURN
C IFault = 3
C IF (KSUM .LE. 0) RETURN
C IFault = 4
C IF (TAU .LT. ZERO .OR. MODE .NE. 0 .AND. TAU .GT. ONE) RETURN
C IFault = 5
C IF (NPTS .LT. 20 .OR. MOD(NPTS,2) .NE. 0) RETURN
C IFault = 6
C IF (TOL .LE. ZERO) RETURN
C IFault = 7
C IF (NPAR .NE. 0) RETURN
C IFault = 8
C NPTS2 = NPTS / 2
C NPTS = (NPAR*NPAR - NPAR) / 2
C LW = 10 * NPAR + LHESL + (3 + NPAR) * NPTS + NPTS2
C IF (MODE .EQ. 0) LW = LW + NPTS * (2*NPTS + 1)
C**************************************************************************************STAB 060
IF (LWORK .LT. LW .OR. LIWORK .LT. NPAR + 7) RETURN

CHECKING OF PARAMETER BOUNDS (IFAILT = 9 IF WRONG)

IF (X(1) .GT. XB(1) .OR. XL(1) .GT. XH(1) .OR. XB(1) .GT. XH(1)) RETURN

*RETURN

IF (XL(1) .LE. ZERO .AND. XB(1) .GT. XH(1)) RETURN

IF (XL(2) .LE. ZERO .AND. XL(2) .GT. XH(2)) RETURN

*RETURN

IF (XL(2) .LE. ONE .AND. XB(2) .GT. XH(2)) RETURN

IF (XL(1) .LT. ONE .AND. XL(1) .LE. ONE .AND. XB(1) .GT. XH(1) .LT. ONE RETURN

*RETURN

IF (XB(3) .LE. ZERO) RETURN

IF (ISCLBD .LT. 0 .AND. XL(3) .GT. XB(3) .OR. XL(3) .GT. XH(3)) RETURN

*RETURN

IF (LOCBD .LT. 0 .AND. XL(4) .GT. XB(4) .OR. XL(4) .GT. XH(4)) RETURN

*RETURN

IF (LOCBD .LT. 0 .AND. XL(4) .LE. XH(4)) RETURN

IF (LOCBD .LT. 0 .AND. XL(4) .EQ. XH(4)) RETURN

*RETURN

INITIAL ADJUSTMENT OF LOCATION/SCALE PARAMETERS/BOUNDS.

SAVE BOUNDS FOR EXIT.

WORK(NPAR + 5) = XL(3)
WORK(NPAR + 6) = XH(3)
WORK(NPAR + 7) = XL(4)
WORK(NPAR + 8) = XH(4)

XK = FLOAT(KSUM)

LOCATION

XMU = XK * XB(4)
IF (LOCBD .LT. 0) GO TO 10
IF (LOCBD .GT. 0) GO TO 20

XL(4) = -FIVE
XH(4) = FIVE
GO TO 30

GO TO 30

10 XL(4) = ZERO
XH(4) = ZERO
GO TO 30

20 XL(4) = XK * XL(4)
XH(4) = XK * XH(4)

SCALE

30 XTOL = XK ** (ONE/XB(1))
SIGMA = XTOL * XB(3)
IF (ISCLBD .LT. 0) GO TO 40
IF (ISCLBD .GT. 0) GO TO 50

XL(3) = PI2
XH(3) = FIVE
GO TO 60

GO TO 60

40 XL(3) = ONE
XH(3) = ONE
GO TO 60

50 XL(3) = XTOL * XL(3)
XH(3) = XTOL * XH(3)

EXPLICITLY SET XTOL TO ED04KBF DEFAULT VALUE, USING X02AAF,
SO THAT IT IS AVAILABLE ON EXIT ON ED04KBF.

60 XTOL = TEN * SQR(ED04KBF(XTOL))
C MEMORY MANAGEMENT
= 2 + NPAR
IWORK(MW + 2) = MODE
IWORK(MW + 3) = UNIT
IWORK(MW + 4) = NPTS
IWORK(MW + 5) = 9 * NPAR + 2
LIW = 6
LKV = LW - 8 - 2 * NPAR - LHESL
MIESL = 9 + NPAR
MIESD = MIESL + LHESL
MV = 1 + NPAR
NPTS = MV + 9 * NPAR + 2
MWT = MPTS + NPTS
MCF = MWT + NPTS
MODERIV = MCF + NPTS
C AVOID UNCLEAR REFERENCES TO A AND A INVERSE IF MODE .NE. 0
MA = MODERIV
MAINV = MODERIV
IA = NPTS + 1
IF (MODE .NE. 0) GO TO 70
MA = MA + (NPAR + 1) * NPTS
MAINV = MA + IA * NPTS
C LOOOING POINT FOR ITERATION
70 IWORK(1) = IWORK(1) + 1
IOFLM = 0
C IF LOCATION/SCALE PARAMETERS ARE CONstrained, UPDATE
THEIR BOUNDS.
IF (ISCLBD .LE. 0) GO TO 80
XL(3) = AMIN1(ONE, AMAX1(FI2, WORK(NPAR + 5)/SIGMA))
STOB 150
XL(3) = AMAX1(ONE, AMIN1(FIVE, WORK(NPAR + 6)/SIGMA))
STOB 151
80 IF (LOCBNL .LE. 0) GO TO 90
XL(4) = AMIN1(ZERO, AMAX1(-FIVE, WORK(NPAR + 7) + XMU)/SIGMA)
STOB 152
XL(4) = AMAX1(ZERO, AMIN1(FIVE, WORK(NPAR + 8) + XMU)/SIGMA)
STOB 153
C CHANGE PARAMETERS TO REFLECT FUTURE STANDARDIZATION.
90 XBI(3) = ONE
XBI(4) = ZERO
WORK(NPAR + 3) = XBI(1)
WORK(NPAR + 4) = XBI(2)
STOB 154
C SET INTYPE = 3 IF POSSIBLE, SO OLD HESSIAN CAN BE USED.
INTYPE = 0
IF (IWORK(1) .EQ. 1) GO TO 110
STOB 155
INTYPE = 3
STOB 156
DO 100 I = 1, NPAR
STOB 157
IF (IWORK(1 + 1) .GT. 0) GO TO 100
STOB 158
INTYPE = 0
STOB 159
GO TO 110
STOB 160
100 CONTINUE
C SET GRIDPOINTS, WEIGHTS.
110 CALL GRIDMT(XB, NPAR, XLM, TAU, WORK(NPTS), NPTS2, WORK(NWT),
* NPTS, MODE, KSUM)
STOB 161
C SET E, CH, F. VALUES, ALSO A AND A INVERSE, IF MODE = 0, USING
FIRST COL. OF DERIV AS WORKSPACE.
CALL SETECF(X, N, XB, NPAR, MODE, TAU, SIGMA, XMU, KSUM, IA,
* NPTS, NPTS, WORK(NPTS), WORK(MWT), WORK(MCF), WORK(MA), WORK(MAINV),
* WORK(MDERIV), IFault)
IF (IFault .GT. 0) IFault = 9 + IFault
IF (IFault .EQ. 10 OR. 1Fault .EQ. 11) GO TO 180
STOB 162
C IN THE RARE EVENT THAT THERE ARE NO FREE VARIABLES, VARIAB
C IS NOT CALLED AND COVARIANCE MATRICES ARE SET TO ZERO.
IF (NVAR .GT. 0) GO TO 170
IF (ICOV .EQ. 0) CALL F01CAF(VCV1, NPAR, NPAR, IWORK(2))
CALL F01CAF(VCV2, NPAR, NPAR, IWORK(3))
GO TO 190
STAB 246
STAB 246
STAB 247
170 NVAR1 = NVAR + 1
STAB 248
STAB 249
STAB 250
STAB 251
STAB 252
STAB 253
STAB 254
STAB 255
STAB 256
STAB 257
STAB 258
STAB 259
STAB 260
STAB 261
STAB 262
STAB 263
STAB 264
STAB 265
STAB 266
STAB 267
STAB 268
STAB 269
STAB 270
STAB 271
STAB 272
STAB 273
STAB 274
STAB 275
STAB 276
STAB 277
STAB 278
STAB 279
STAB 280
STAB 281
STAB 282
STAB 283
STAB 284
STAB 285
STAB 286
STAB 287
STAB 288
STAB 289
STAB 290
STAB 291
STAB 292
STAB 293
STAB 294
STAB 295
STAB 296
STAB 297
STAB 298
STAB 299
STAB 300
STAB 301
STAB 302
STAB 303
STAB 304
STAB 305
STAB 306
STAB 307
STAB 308
STAB 309
STAB 310
STAB 311
STAB 312
STAB 313
STAB 314
STAB 315
STAB 316
STAB 317
STAB 318
STAB 319
STAB 320
STAB 321
STAB 322
STAB 323
STAB 324
STAB 325
STAB 326
STAB 327
STAB 328
STAB 329
STAB 330
STAB 331
STAB 332
STAB 333
STAB 334
STAB 335
STAB 336
STAB 337
STAB 338
STAB 339
STAB 340
STAB 341
STAB 342
STAB 343
STAB 344
STAB 345
STAB 346
STAB 347
STAB 348
STAB 349
STAB 350
STAB 351
STAB 352
STAB 353
STAB 354
STAB 355
STAB 356
STAB 357
STAB 358
STAB 359
STAB 360
STAB 361
STAB 362
STAB 363
STAB 364
STAB 365
STAB 366
STAB 367
STAB 368
STAB 369
STAB 370
STAB 371
STAB 372
STAB 373
STAB 374
STAB 375
STAB 376
STAB 377
STAB 378
STAB 379
STAB 380
STAB 381
STAB 382
STAB 383
STAB 384
STAB 385
STAB 386
STAB 387
STAB 388
STAB 389
STAB 390
STAB 391
STAB 392
STAB 393
STAB 394
STAB 395
STAB 396
STAB 397
STAB 398
STAB 399
STAB 400
STAB 401
STAB 402
STAB 403
STAB 404
STAB 405
STAB 406
STAB 407
STAB 408
STAB 409
STAB 410
STAB 411
STAB 412
STAB 413
STAB 414
STAB 415
STAB 416
STAB 417
*ONEPT7, ONEPT8, TWO, THREE, THRPT3, THRPT6, FOUR, FIVE, SEVEN, TENCGRID 018
DATA ZERO, PT025, PT035, PT045, PT104, PT105, PT106, PT107, GRID 019
*PT107, PT3, PT5, PT6, PT8, PT9, ONE, ONEPT2, ONEPT4, ONEPT6, GRID 020
*ONEPT7, ONEPT8, TWO, THREE, THRPT3, THRPT6, FOUR, FIVE, SEVEN, GRID 021
*THREEDRG, FIVE, SEVEN, TENCGRID 022
*0.075, 0.3, 0.5, 0.6, 0.8, 0.9, 1.0, 1.2, 1.4, 1.6, 1.7, 1.8, 2.0, GRID 023
*3.0, 3.3, 3.6, 4.0, 5.0, 7.0, 10.0/ GRID 024
GRID 025
GRID 026
GRID 027
GRID 028
GRID 029
GRID 030
GRID 031
GRID 032
GRID 033
GRID 034
GRID 035
GRID 036
GRID 037
GRID 038
GRID 039
GRID 040
GRID 041
GRID 042
GRID 043
GRID 044
GRID 045
GRID 046
GRID 047
GRID 048
GRID 049
GRID 050
GRID 051
GRID 052
GRID 053
GRID 054
GRID 055
GRID 056
GRID 057
GRID 058
GRID 059
GRID 060
GRID 061
GRID 062
GRID 063
GRID 064
GRID 065
GRID 066
GRID 067
GRID 068
GRID 069
GRID 070
GRID 071
GRID 072
GRID 073
GRID 074
GRID 075
GRID 076
GRID 077
C CASE WHEN ALPHA LE 1 - SUBTRACT 1 FROM NO. OF INNER PTS
C IF ALPHA LE 1/2, START BY EQUISPACING ALL POINTS FOR
GRID 078
C GRID 079
GRID 080
GRID 081
GRID 082
GRID 083
GRID 084
GRID 085
GRID 086
GRID 087
GRID 088
GRID 089
GRID 090
GRID 091
GRID 092
GRID 093
GRID 094
GRID 095
GRID 096
GRID 097
GRID 098
GRID 099
GRID 100
GRID 101
GRID 102
GRID 103
GRID 104
GRID 105
GRID 106
GRID 107
GRID 108
GRID 109
GRID 110
GRID 111
GRID 112
GRID 113
GRID 114
GRID 115
GRID 116
GRID 117
GRID 118
GRID 119
GRID 120
GRID 121
GRID 122
GRID 123
GRID 124
GRID 125
GRID 126
GRID 127
GRID 128
GRID 129
GRID 130
GRID 131
GRID 132
GRID 133
GRID 134
GRID 135
GRID 136
C*******************************************************************************
C COMPUTES REAL AND IMAGINARY PARTS OF STANDARD STABLE
C CHARACTERISTIC FUNCTION (SIGMA = 1, MU = 0).
C  CALLING GRIDWT, SETECF, VARIAB                       CHAR 004
C**********************************************************************
C SUBROUTINE CHART(U, PAR, NPAR, RE, XIM)                          CHAR 005
C ARGUMENTS                                                      CHAR 006
INTEGER NPAR                                                    CHAR 007
REAL U, PAR(NPAR), RE, XIM                                     CHAR 008
C LOCAL SCALARS                                                CHAR 009
REAL ALPHA, XMOD, ZER0, ONE, PI2Y2                             CHAR 010
DATA ZERO, ONE, PI2Y2 /0.0, 1.0, 1.570796327/                 CHAR 011
C RE = ABS(U)                                                   CHAR 012
ALPHA = PAR(1)                                                 CHAR 013
IF (ALPHA .NE. ONE) GO TO 10                                    CHAR 014
XIM = ZERO                                                     CHAR 015
IF (U .NE. ZERO) XIM = ALOG(RE) / PI2Y2                        CHAR 016
GO TO 20                                                       CHAR 017
10 XIM = TAN(PI2Y2*ALPHA)                                      CHAR 018
20 XMOD = RE ** ALPHA                                          CHAR 019
XIM = -PAR(2) * SIGM(XMOD,U) * XIM                            CHAR 020
XMOD = EXP(-XMOD)                                              CHAR 021
RE = XMOD * COS(XIM)                                           CHAR 022
XIM = XMOD * SIN(XIM)                                          CHAR 023
RETURN                                                         CHAR 024
C**********************************************************************
C FUNCTION/DERIVATIVE EVALUATION.                                FUNC 001
C SUBROUTINE FUNC(IFLAG, M, XC, FC, GC, IW, LIW, W, LW)          FUNC 002
C ARGUMENTS                                                      FUNC 003
INTEGER IFLAG, M, LIW, IW(LIW), LW                             FUNC 004
REAL XC(M), FC, GC(M), W(LW)                                   FUNC 005
C LOCAL SCALARS                                                FUNC 006
LOGICAL ALF1, LFLAG                                            FUNC 007
INTEGER ILOW, NPTS, NPTS2, IND, INDI, ISUB, ISUB1, IPTS, IW,   FUNC 008
*IECF, IDERIV, IPSI                                           FUNC 009
REAL ALPHA, SIGMA, XMU, OMEGA, XMOD, PTSI, SINE, COSINE,        FUNC 010
*PISEC2, SUEXP, XLOGSU, FAC, Z, Z1, ZERO, ONE, PI2Y2           FUNC 011
DATA ZERO, ONE, PI2Y2 /0.0, 1.0, 1.570796327/                 FUNC 012
C ALPHD = XC(1)                                                 FUNC 013
BSIGN = XC(2)                                                   FUNC 014
SIGMA = XC(3)                                                  FUNC 015
XMU = XC(4)                                                    FUNC 016
ALF1 = ALPHA .EQ. ONE                                         FUNC 017
LFLAG = IFLAG .EQ. 0                                           FUNC 018
C VARIABLES TO AID ADDRESSING IN W VECTOR - IPTS = ONE LESS THAN FUNC 019
C POSITION OF FIRST POSITIVE GRIDPOINT IN W VECTOR, ETC.        FUNC 020
NPTS = IW(5)                                                   FUNC 021
NPTS2 = NPTS / 2                                               FUNC 022
IND = NPTS + 1                                                 FUNC 023
ILOW = NPTS2 + 1                                               FUNC 024
IPTS = IW(6)                                                   FUNC 025
IW = IPTS + NPTS                                              FUNC 026
IECF = IW + NPTS                                              FUNC 027
IDERIV = IECF + NPTS                                          FUNC 028
IPSI = IDERIV + M * NPTS                                       FUNC 029
NPTS2 = IPTS - NPTS                                           FUNC 030

C IF ALPHA, W.R. TO OMEGA(U,ALPHA) AND ITS DERIVATIVE W. R. TO OMEGA.

037 IF (ALFI) GO TO 10

038 OMEGA = TAN(PI*Y2*ALPHA)

039 PISEC = PI*Y2 * (ONE + OMEGA*OMEGA)

040 C LOOP OVER POSITIVE GRID PTS (SGN(U) IGNORED), SAVE PSI AND ITS GRADIENT AT ALL GRID PTS (PSI(U) = RE(PHI(U)) + IM(PHI(U)) - EMPIRICAL COUNTERPARTS.)

041 10 DO 20 I = 1(LOW, NPST)

042 IND1 = IND - 1

043 ISUB = NPST2 + 1

044 PSI = W(ISUB)

045 XLOGSU = SIGMA * PSI

046 SUFX = XLOGSU ** ALPHA

047 XLOGSU = ALOG(XLOGSU)

048 IF (ALFI) OMEGA = XLOGSU / PI*Y2

049 COSINE = XMU * PSI - SUFX * BSIGN * OMEGA

050 SINE = SIN(COSINE)

051 COSINE = COS(COSINE)

052 XMOD = EXP(-SUFX)

053 SUFX = SUFX * XMOD

054 C SAVE COMPONENTS OF PSI

055 ISUB = IPSI + 1

056 ISUB1 = IECF + 1

057 W(ISUB) = XMOD * (COSINE + SINE) - W(ISUB1)

058 ISUB = IPSI + 1

059 W(ISUB) = XMOD * (COSINE - SINE) - W(ISUB1)

060 IF (LFLAG) GO TO 20

061 C CALCULATE DERIVATIVES IF REQUIRED

062 C DERIVATIVES W. R. TO ALPHA

063 FAC = XLOGSU * OMEGA

064 IF (.NOT. ALFI) FAC = FAC + PISEC

065 C FAC = FAC * BSIGN

066 Z = FAC + XLOGSU

067 Z1 = FAC - XLOGSU

068 ISUB = (DERIV + 1)

069 W(ISUB) = SUFX * (-Z*COSINE + Z1*SINE)

070 ISUB1 = DERIV + IND1

071 W(ISUB1) = SUFX * (Z1*COSINE + Z*SINE)

072 C DERIVATIVES W. R. TO BETA

073 FAC = SUFX * OMEGA

074 ISUB = ISUB + NPST

075 W(ISUB) = FAC * (SINE - COSINE)

076 ISUB1 = ISUB1 + NPST

077 W(ISUB1) = FAC * (SINE + COSINE)

078 C DERIVATIVES W. R. TO SIGMA

079 FAC = BSIGN * OMEGA

080 Z = ONE + FAC

081 Z1 = ONE - FAC

082 FAC = -ALPHA * SUFX / SIGMA

083 ISUB = ISUB + NPST

084 W(ISUB) = FAC * (Z*COSINE + Z1*SINE)

085 ISUB1 = ISUB1 + NPST

086 W(ISUB1) = FAC * (Z*SINE - Z1*COSINE)
W(ISUB1) = FAC * (Z1*COSINE - Z*SINE)

C DERIVATIVES W. R. TO MU
FAC = PSI * XMOD
ISUB = ISUB + NPTS
W(ISUB) = FAC * (-SINE + COSINE)
ISUB1 = ISUB + NPTS
W(ISUB1) = -FAC * (SINE + COSINE)
20 CONTINUE

C NOW COMPUTE OBJECTIVE FUNCTION, OPTIONALLY GRADIENT.
FC = ZERO
IF (LFLAG) GO TO NO
ILOW = IDERIV - NPTS
DO 30 I = 1, N
30 GC(I) = ZERO
40 IF (IWH(J) .EQ. 0) GO TO 70

C SUM-OF-SQUARES ESTIMATION
DO 60 I = 1, NPTS
C FN. EVALUATION
ISUB = IPSI + I
Z = W(ISUB)
ISUB = INT + I
Z1 = Z * W(ISUB)
FC = FC + Z * Z1
IF (LFLAG) GO TO 60

C GRADIENT EVALUATION IF REQUESTED.
ISUB = ILOW + I
Z1 = Z1 + Z1
DO 50 J = 1, N
ISUB = ISUB + NPTS
GC(J) = GC(J) + W(ISUB) * Z1
50 CONTINUE
60 CONTINUE
RETURN

C MATRIX ESTIMATION - FIRST MULTIPLY PSI VALUES BY WEIGHTS.
70 DO 80 I = 1, NPTS
ISUB = IPSI + I
ISUB1 = INT + I
W(ISUB) = W(ISUB) * W(ISUB1)
80 CONTINUE

C POSITION OF A INVERSE IS REQUIRED.
IND1 = IPSI + IND1 + NPTS
DO 110 I = 1, NPTS
Z = ZERO
IND1 = IND1 + NPTS
SUM OVER J OF PSI(J) * AINV(I,J)
DO 90 J = 1, NPTS
ISUB = IPSI + J
ISUB1 = IND1 + J
Z = Z + W(ISUB) * W(ISUB1)
90 CONTINUE

C MULTIPLY BY PSI(I) AND ADD TO FN. VALUE
ISUB = IPSI + 1
FC = FC + Z * W(ISUB)
IF (LFLAG) GO TO 110

C GRADIENT IF REQUESTED - FOR JTH COMPONENT OF GRADIENT ADD

C
2*(JTH DERIVATIVE AT GRIDPOINT I)*WT(I)*(RESULT OF DO 90 LOOP) FUNC 156
ISUB = IWT + I
Z = (Z + Z) * W(ISUB)
ISUB = ILOW + I
DO 100 J = 1, N
(ISUB = ISUB + NPTS)
OC(J) = GC(J) + Z * W(ISUB)
100 CONTINUE
110 CONTINUE
C
RETURN
END
C******************************************************************************
C COMPUTES EMPRICAL CH. F. VALUES, ADJUSTING FOR LOCATION.
C SCALE, AND K-SUM INDEX. FOR MATRIX ESTIMATION, THE UPPER
C TRIANGLE OF A MATRIX IS CALCULATED AND INVERTED.
C CALLED BY = STABLE
C CALLS = CHARFN
C N.A.G. SUBROUTINE CALLED -
C FOTABF (ACCURATE INVERSION OF POSITIVE DEFINITE
C SYMMETRIC MATIX).
C******************************************************************************
SUBROUTINE BETECF(X, N, PAR, NPAR, MODE, TAU, SIGMA, XMU, KSUM, *A, NPTS, YPTS, ECF, A, AINV, WORK, IFULT)
C ARGUMENTS
INTEGER N, NPAR, MODE, KSUM, IA, NPTS, YPTS, IFULT
REAL X(M), PAR(NPAR), TAU, SIGMA, XMU, YPTS(NPTS), ECF(YPTS), *A(IA,NPTS), AINV(NPTS,YPTS), WORK(NPTS)
C LOCAL SCALARS
INTEGER IND, IND1, IND2, IND3
REAL PTIS, PTF, RC, XIM, RE1, XIM1, RPI, RMI, RP11, RMI1, ZERO, *HALF
DATA ZERO, HALF /0.0, 0.5/
C
C CALCULATION OF E.CH.F. VALUES
PTIS = FLOAT(N)
XIM1 = FLOAT(KSUM)
RP1 = XIM1 * HALF
IND = NPTS + 1
IND1 = NPTS + 1
DO 20 I = 1, IND, NPTS
IND2 = I - NPTS
PTIS = PTIS(IND2) / SIGMA
IND2 = IND1 - I
RE = ZERO
XIM = ZERO
C
DO 10 J = 1, N
RPI1 = X(J) * PTIS
RE = RE + COS(RPI1)
XIM = XIM + SIN(RPI1)
10 CONTINUE
C
RE = RE / PTIS
XIM = XIM / PTIS
RMI = (RE*RE + XIM*XIM)**RPI
RP11 = XIM1 * ATAN2(XIM,RE) - XMU * PTIS
RE = RMI * COS(RPI1)
XIM = RMI * SIN(RPI1)
ECF(I) = RE + XIM
ECF(IND2) = RE - XIM
20 CONTINUE
C SET UPPER TRIANGLE OF A IF REQUESTED. A GENERATED FROM
C POSITIVE GRIDPOINTS ONLY - ANTIDIAGONAL COMPUTED TWICE.
C IF (MODE .NE. 0) RETURN
C FIRST FILL WORK WITH (RE + IM) (PHI) TO SAVE EVALS.
C DO 30 J = IND, NPTS
C IND2 = I - NPTS2
C CALL CHARMF(PTS(I), PAR, NPAR, RE, XIM)
C IND2 = IND - I
C WORK(I) = RE + XIM
C WORK(IND2) = RE - XIM
30 CONTINUE
C COMPUTATION OF A.
C DO 40 J = IND, NPTS
C IND2 = I - NPTS2
C PTSJ = PTSJ(IND2)
C IND2 = IND - I
C RPI = WORK(J)
C DO 40 J = IND, J
C IND3 = J - NPTS2
C PTSJ = PTSJ(IND3)
C IND3 = IND - J
C RP11 = WORK(J)
C RP11 = WORK(IND3)
C CALL CHARMF(PSTI + PTSJ, PAR, NPAR, RE, XIM)
C CALL CHARMF(PTSJ - PTSJ, PAR, NPAR, RE1, XIM1)
C A(J, I) = RE + XIM - RP1 * RP11
C A(IND2, J) = RE - XIM1 - RMI * RP11
C A(IND2, IND3) = RE1 - XIM - RMI * RMI1
40 CONTINUE
C DIAGONAL OF A ADDITIVELY INFLATED BY TAU TIMES AVERAGE OF
C DIAGONAL ELEMENTS.
C IF (TAU .EQ. ZERO) GO TO 70
C PTSI = ZERO
C DO 50 I = 1, NPTS
C PTSI = PTSI + A(I, I)
C 50 PTSI = TAU * PTSI / FLOAT(NPTS)
C DO 60 I = 1, NPTS
C A(I, I) = A(I, I) + PTSI
C 60 CONTINUE
C INVERT A USING N.A.G. ROUTINE F01ABF
C 70 IFAULT = 1
C CALL F01ABF(A, IA, NPTS, AINV, NPTS, WORK, IFAULT)
C EXIT IF A FOUND NON POSITIVE DEFINITE OR ILL-CONDITIONED.
C IF (IFAIL = 0) RETURN
C ARRANGE A INVERSE SO IT IS COMPLETELY FILLED.
C DO 80 I = 1, NPTS
C DO 80 J = 1, I
C AINV(I, J) = AINV(J, I)
C 80 CONTINUE
C RETURN
C END
C******************************************************************************
C MONITORING OF EUKBF.
C******************************************************************************
C
C CALLING - EDNKF

SUBROUTINE MONIT(N, XC, FC, GC, ISTATE, GPJNRM, COND, POSDEF, NITER, NF, IV, LW, LW)

* ARGUMENTS
LOGICAL POSDEF
INTEGER N, ISTATE(N), NITER, NF, LW, IV(LW), LW
REAL XC(N), FC, GC(N), GPJNRM, COND, IV(LW)

C LOCAL SCALAR
INTEGER IUNIT

C STORE HESSIAN CONDITION NUMBER AND PROJECTED GRADIENT NORM IN WORK VECTOR
UNIT = 9 * N + 1
W(UNIT) = COND
W(UNIT + 1) = GPJNRM
IF (UNIT .LE. 0) RETURN

C WRITE DETAILS OF OPTIMIZATION PROCESS IF IUNIT .GT. 0
WRITE (UNIT, 10) NITER, NF, FC, GPJNRM
WRITE (UNIT, 20) (XC(I), I=1, N)
WRITE (UNIT, 30) (GC(I), I=1, N)
WRITE (UNIT, 40) COND

10 FORMAT (5H0, N, 5X, 8HM EVALS, 8X, 8HM VALUE, 5X, 2H0)
20 FORMAT (10H0 SOLUTIONS, 4E13.5)
30 FORMAT (10H0 PROJ GRAD, 4E13.5)
40 FORMAT (5H0 ESTIMATED CONDITION NUMBER OF PROJECTED HESSIAN, 10H0)
50 FORMAT (5H0, 5H0)

C CALLS TO ASYMPTOTIC COVARIANCE MATRICES. IFAIL1 AND IFAIL2 ARE FAILURE INDICATORS FOR MATRIX INVERSION REQUIRED
C FOR VCV1, VCV2 RESPECTIVELY.
C CALLED BY - STABLE
C CALLS - HESDEF, FUNCT, CHARFM, SETCV, VMATRIX, DAPROD, HYPROD
C N.A.G. SUBROUTINES CALLED -
C FOTCAF (SETS A MATRIX TO ZERO), FOTCMF (SET ONE MATRIX TO ANOTHER).

C CALCULATES EMPIRICAL (VCV1) AND ASYMPTOTIC (VCV2)
C APPROXIMATIONS TO ASYMPTOTIC COVARIANCE MATRICES. IFAIL1 AND IFAIL2 ARE FAILURE INDICATORS FOR MATRIX INVERSION REQUIRED
C FOR VCV1, VCV2 RESPECTIVELY.
C CALLED BY - STABLE
C CALLS - HESDEF, FUNCT, CHARFM, SETCV, VMATRIX, DAPROD, HYPROD
C N.A.G. SUBROUTINES CALLED -
C FOTCAF (SETS A MATRIX TO ZERO), FOTCMF (SET ONE MATRIX TO ANOTHER).

SUBROUTINE VARIABICOV(X, N, X, N, PAR, NPAR, MODE, SIGMA, XMU, ISUB, VARI 010)
*VAR NPAR, PTS, NPTS2, WT, ECF, NPTS, DERIV, WORK, HOLD, IA, AINV,
*VCV1, VCV2, H, NVAR1, V, IV, LW, LW, LW, LFA1, LFA1(2)
C ARGUMENTS
INTEGER ICOV, N, NPAR, MODE, NVAR1, NSUB(NVAR), NPTS2, NPTS, IA,
*NPAR, IV(LW), LW, IV(LW), LW, IV(LW), LW, LFA1, LFA1(2)
REAL X(N), PAR(NPAR), SIGMA, XMU, PTS(NPTS2), WT(NPTS), ECF(NPTS), VARI 018
*DERIV(NPTS, NPAR), WORK(NPTS), HOLD(NVAR), A(IA, NPTS),
*PHI(NPTS, NPTS), VCV1(NPAR, NPAR), VCV2(NPAR, NPAR), NP(NVAR1, NVAR),
*VM(NVAR, NPAR), LW(LW)
C LOCAL SCALARS
LOGICAL FLAG
INTEGER IND, IND1, IND2, IND3
REAL PTSK, PTS, RTI, RT1, RT1, RM1, COVKL, COVML, COVKM, COVMM, VARI 025
*D1, D2, D3, D4, ZERO
DATA ZERO /0.0/

C ARRAY DIMENSIONS C
C...

C ANALOGUE OF SYMVARICOV, BUT VARY COVARIANCE WITH REAL DATA.
C...

C LOCAL SCALARS
LOGICAL FLAG
INTEGER IND, IND1, IND2, IND3
REAL PTSK, PTS, RTI, RT1, RT1, RM1, COVKL, COVML, COVKM, COVMM, VARI 025
*D1, D2, D3, D4, ZERO
DATA ZERO /0.0/

C ANALOGUE OF SYMVARICOV, BUT VARY COVARIANCE WITH REAL DATA.
C...

C LOCAL SCALARS
LOGICAL FLAG
INTEGER IND, IND1, IND2, IND3
REAL PTSK, PTS, RTI, RT1, RT1, RM1, COVKL, COVML, COVKM, COVMM, VARI 025
*D1, D2, D3, D4, ZERO
DATA ZERO /0.0/

C ANALOGUE OF SYMVARICOV, BUT VARY COVARIANCE WITH REAL DATA.
C...

C LOCAL SCALARS
LOGICAL FLAG
INTEGER IND, IND1, IND2, IND3
REAL PTSK, PTS, RTI, RT1, RT1, RM1, COVKL, COVML, COVKM, COVMM, VARI 025
*D1, D2, D3, D4, ZERO
DATA ZERO /0.0/
C IND = NPTS2 + 1
IND1 = NPTS + 1

C IF REQUESTED, FIRST CALCULATE HESSIAN FOR EMPIRICAL VERSION
AND STORE IN VC1. (HESDIF, FUNCT REQUIRE ECF, WORK,
PPOSSIBLY AIMW, WHICH ARE USED AS WORK AREAS BELOW.)
V AND V ARE PASSED AS WORKSPACE.
IF (ICOV .EQ. 0) CALL HESDIF(PAR, NPAR, ISUB, VC1, H, V, NVAR,
*IW, LIW, LW, LW)

C CALL FUNCT TO SET DERIV, AS GRID SEARCH PERFORMED BY EQKRF
MAY HAVE SET IT TO STRANGE VALUES. VC2 USED AS WORKSPACE.
CALL FUNCT2(NPAR, PAR, D1, VC2(1, 1), IW, LIW, LW, LW)

C SHIFT DERIV VALUES SO THEY CAN BE ADDRESSED WITHOUT THE
ISUB VECTOR. NOTE ISUB ELEMENTS ARE IN ASCENDING ORDER.
DO 20 J = 1, NVAR
IND2 = ISUB(J)
IF (IND2 .EQ. 1) GO TO 20
DO 10 J = 1, NPTS
10 DERIV(J, 1) = DERIV(J, IND2)
20 CONTINUE

C FILL ECF VECTOR WITH CH. F. VALUES, NOW THAT IT IS NO
LONGER NEEDED FOR FUNCTION EVALUATION.
DO 30 I = IND, NPTS
IND2 = I - NPTS2
CALL CHALN(POINTS(IND2), PAR, NPAR, D1, D2)
IND2 = IND1 - 1
ECT(I) = D1 + D2
ECT(IND2) = D1 - D2
30 CONTINUE
IF (MODE .EQ. 0) GO TO 100

C SUM OF SQUARES ESTIMATION SECTION.

C FIRST DO ASYMPTOTIC VERSION.
C CALCULATE UPPER TRIANGLE OF HESSIAN.
DO 50 I = 1, NVAR
DO 50 J = 1, NVAR
DI = ZERO
DO 40 K = 1, NPTS
40 DI = DI + DERIV(K, I) * DERIV(K, J) * WI(K)
50 CONTINUE

C PREMULTIPLY DERIV BY WEIGHTS TO SAVE MULTIPLICATIONS.
DO 60 I = 1, NPTS
DI = WI(I)
DO 60 J = 1, NVAR
DERIV(I, J) = DI * DERIV(I, J)
60 CONTINUE

C COMPUTE UPPER TRIANGLE OF V. BASICALLY EQUIVALENT TO
CALCULATING THE A MATRIX FOR MATRIX ESTIMATION, BUT
COMPLICATIONS ARISE IN COMPUTING BILINEAR FORMS.
CALL IDICAT(V, NVAR, NVAR, ITAIL2)
DO 90 K = IND, NPTS
IND2 = K - NPTS2
PISK = PIS(KIND)  IND2 = IND1 + K  RPI = ECF(K)  RM1 = ECF(IND2)  DO 90 L = IND, K  IND3 = L - NPT52  PISL = PIS(IND3)  IND3 = IND1 - L  RPI1 = ECF(L)  RM1 = ECF(IND3)
CALL CHARFN(PISK + PISL, PAR, NPAR, D1, D2)  CALL CHARFN(PISK - PISL, PAR, NPAR, D3, D4)  COVKL = D3 + D2 - RPI + RPI1  COVML = D1 - D4 - RM1 + RM11  COVKM = D1 + D4 - RPI + RM1  COVMM = D3 - R2 - RM1 * RM11  FLAG = K .EQ. L  
!
C LOOP TO ADD CONTRIBUTIONS TO V (BILINEAR FORMS).
DO 80 I = 1, NVAR  D1 = DERIV(K, I)  D2 = DERIV(IND2, I)  DO 80 J = 1, NVAR  D3 = DERIV(L, J)  D4 = DERIV(IND3, J)  PISL = (COVKL*D1 + COVML*D2) * D3 + (COVKM*D1 + COVMM*D2) * D4  IF (FLAG) GO TO 70  
C WHEN K .NE. L, MUST ADD SYMMETRIC CONTRIBUTION.
RPI1 = DERIV(K, J)  RM1 = DERIV(IND2, J)  D3 = DERIV(L, I)  D4 = DERIV(IND3, I)  PISL = PISL + (COVKL*RPI1 + COVML*RM1) * D3 + (COVKM*RPI1 + COVMM*RM1) * D4  70 V(I,J) = V(I,J) + PISL  80 CONTINUE  90 CONTINUE  GO TO 140  
C MATRIX ESTIMATION SECTION.
!! FILL A MATRIX COMPLETELY (UPPER TRIANGLE ONLY ON ENTRY),
!! MULTIPLY ELEMENTS OF A BY CORRESPONDING WEIGHTS, MULTIPLY
!! NEW VALUES BY WEIGHTS TO SAVE MULTIPICATIONS LATER.
100 DO 130 I = 1, NPTS  D1 = WT(I)  DO 110 J = 1, NPTS  A(I,J) = A(I,J) * D1 * WT(J)  A(J,I) = A(I,J)  110 CONTINUE  120 DERIV(I,J) = DERIV(I,J) * D1  130 CONTINUE  
C ASYMPTOTIC VERSION.
!! MULTIPLY AINV * DERIV, OVERWRITING CORNER OF AINV AND USING
!! NPT5 AS WORKSPACE.
CALL DAPROD(AINV, NPTS, NPTS, DERIV, HOLD, NVAR)
!! MULTIPLY A * (AINV CORNER), OVERWRITING CORNER OF A AND
DO 50 I = 1, N
D1 = (X(I) - XNU) / SIGMA

C WORK HOLDS VECTOR OF SINE/COSINE TERMS.
DO 10 J = IND, NPTS
IND2 = J - NPTS2
D2 = PTS(IND2) * D1
D3 = COS(D2)
D2 = SIN(D2)
IND2 = IND1 - J
WORK(J) = ECF(J) - D3 - D2
WORK(IND2) = ECF(IND2) - D3 + D2
IF (FLAG) GO TO 10

C IF MATRIX ESTIMATION, MULTIPLY ELEMENTS OF WORK BY WEIGHTS.
WORK(J) = WORK(J) * WT(J)
WORK(IND2) = WORK(IND2) * WT(IND2)
10 CONTINUE

C MULTIPLY DERIV * WORK, WRITING RESULT TO HOLD. MULTIPLICATION DONE DIRECTLY TO AVOID OVERHEAD OF SUBROUTINE CALL.
DO 30 J = 1, NVAR
D1 = ZERO
D2 = 20 K = 1, NPTS
20 D1 = D1 + DERIV(K,J) * WORK(K)
HOLD(J) = D1
30 CONTINUE

C ADD HOLD * (HOLD TRANSPOSE) TO V.
DO 40 J = 1, NVAR
D1 = HOLD(J)
40 V(J,K) = V(J,K) + D1 * HOLD(K)
40 CONTINUE

C FINAL CONSTANT FACTOR FOR V.
D1 = FOUR / FLOAT(N)
DO 60 I = 1, NVAR
DO 60 J = I, NVAR
60 V(I,J) = V(I,J) * D1
C RETURN
C END

******************************************************************************
C MULTIPLIES THE (NPTS BY NPTS) CORNER OF THE (IFAC1 BY NPTS)
C MATRIX FACT BY THE (NPTS BY NVAR) MATRIX FAC2, OVERWRITING 
C THE UPPER LEFT (NPTS BY NVAR) ELEMENTS OF FAC1.
C UNFORTUNATELY, N.A.G. ROUTINE FOICKF DOES NOT ALLOW THIS
C KIND OF OVERWRITING.
C C ED BY - VARIAB
C*****************************************************************************
C SUBROUTINE DAPROD(IFAC1, IFAC1, NPTS, FAC2, WORK, NVAR)
C C ARGUMENTS
C INTEGER IFAC1, NPTS, NVAR
C REAL FAC1(IFAC1,NPTS), FAC2(NPTS,NVAR), WORK(NVAR)
C LOCAL Scalars
C REAL TEMP, ZERO
C DATA ZERO /0.0/
C DAPR 008
C DAPR 009
C DAPR 010
C DAPR 011
C DAPR 012
C DAPR 013
C DAPR 014
C DAPR 015
C DAPR 016
C DAPR 001
C DAPR 002
C DAPR 003
C DAPR 004
C DAPR 005
C DAPR 006
C DAPR 007
C USING HOLD AS WORKSPACE.
CALL DAPROD(A, IA, NPTS, AINV, HOLD, NVAR)  
C MULTIPLY (A CORNER TRANSPOSE) \( \times \) (AINV CORNER), GIVING  
C UPPER TRIANGLE OF V.
CALL HVPROD(A, IA, NVAR, AINV, NPTS, V, NVAR)  
C MULTIPLY (AINV CORNER TRANSPOSE) \( \times \) DERIV, GIVING UPPER  
C TRIANGLE OF HESSIAN.
CALL HVPROD(AINV, NPTS, NVAR, DERIV, NPTS, H, NVAR)  
C CODE COMMON TO MATRIX AND SUM OF SQUARES ESTIMATION.  
C COMPUTE ASYMPOTIC COVARIANCE MATRIX.
180 CALL SETCV(VISUB, NVAR, H, NVAR1, V, HOLD, VCV2, NPAR, SIGMA,  
*FAIL2)
IF (ICOV .GT. 0) RETURN  
C EMPIRICAL VERSION IF REQUESTED.
C COMPUTE EMPIRICAL V MATRIX, USING WORK AND HOLD AS WORK AREAS.  
IF (MODE .NE. 0) CALL VMATRIX(X, N, MODE, XMU, SIGMA, PTS, NPTS2,  
*WT, ECF, WORK, NPTS, DERIV, V, HOLD, NVAR)  
IF (MODE .EQ. 0) CALL VMATRIX(X, N, MODE, XMU, SIGMA, PTS, NPTS2,  
*WT, ECF, WORK, NPTS, AINV, V, HOLD, NVAR)  
C RESTORE HESSIAN FROM VCV1, COMPUTE COVARIANCE MATRIX.
CALL FOICF(VCV1, NVAR, H, NVAR1, NVAR, NVAR)  
CALL SETCV(VISUB, NVAR, H, NVAR1, V, HOLD, VCV1, NPAR, SIGMA,  
*FAIL1)
C RETURN
RETURN  
C*****************************************************************************
C Calculates empirical V matrix. Implicit is computation of  
C empirical covariance kernel, but it is faster to compute a  
C matrix product for each sample point than to cumulate the  
C whole kernel. For matrix estimation, the DERIV array  
C is actually the upper left corner of AINV.  
C Called by - VARVAB  
C N.A.G. subroutine called -  
C FOICF (SET A MATRIX TO 0).
C*****************************************************************************
SUBROUTINE VMATRIX(X, N, MODE, XMU, SIGMA, PTS, NPTS2, WT, ECF,  
*WORK, NPTS, DERIV, V, HOLD, NVAR)  
C ARGUMENTS
INTEGER N, MODE, NPTS2, NPTS, NVAR  
REAL X(N), XMU, SIGMA, PTS(NPTS2), WT(NPTS), ECF(NPTS),  
*WORK(NPTS), DERIV(NPTS, NVAR), V(NVAR, NVAR), HOLD(NVAR)
C LOCAL SCALARS  
LOGICAL FLAG  
INTEGER IND, INDI, IND2  
REAL D1, D2, D3, ZERO, FOUR  
DATA ZERO, FOUR /0.0, 4.0/  
C SET V TO 0.
FLAG = MODE .NE. 0  
IND = NPTS2 + 1  
IND1 = NPTS + 1  
CALL FOICF(V, NVAR, NVAR, IND2)  
C MAIN LOOP OVER SAMPLE.
C*****************************************************************************
DO 40 I = 1, NPTS
DO 20 J = 1, NVAR
TEMP = ZERO
DO 10 K = 1, NPTS
10 TEMP = TEMP + FAC1(I,K) * FAC2(K,J)
WORK(J) = TEMP
20 CONTINUE
DO 30 J = 1, NVAR
30 FAC1(I,J) = WORK(J)
40 CONTINUE
C RETURN

********************************************************************
C multiplies the transposed (NPTS by NVAR) corner of the
C (IFAC1 by NVAR) matrix FAC1 by the (NPTS by NVAR) matrix
C FAC2, giving the upper triangle of either V or H.
C called by - variab
C subroutine HVprod(FAC1, IFAC1, NVAR, FAC2, NPTS, VH, IVH)

C arguments
C integer IFAC1, NVAR, NPTS, IVH
C real FAC1(IFAC1,NVAR), FAC2(NPTS,NVAR), VH(IVH,NVAR)
C local scalars
C real TEMP, ZERO
C data zero /0.0/

C do 20 I = 1, NVAR
C do 20 J = 1, NVAR
TEMP = ZERO
DO 10 K = 1, NPTS
10 TEMP = TEMP + FAC1(K,I) * FAC2(K,J)
VH(I,J) = TEMP
20 CONTINUE
C RETURN

********************************************************************
C common operations in computation of covariance matrices.
C họ uses its hessian matrix h, calculates (H INVERSE) * V *
C (H INVERSE), overwriting v.
C called by - variab
C N.A.G. subroutines called -
C T010BF (accurate inversion of positive definite
C symmetric matrix),
C FO1CKF (matrix multiplication with overwriting),
C FO1CMF (set one matrix equal to another),
C FO1CAF (set a matrix to zero).
C subroutine SETVCV(ISUB, NVAR, H, NVAR1, V, WORK, VCV, NPAR, SIGMA, *FAULT)
C arguments
C integer NVAR, ISUB(NVAR), NVAR1, NPAR, *FAULT
C real h(NVAR1,NVAR), v(NVAR,NVAR), work(NVAR), vcv(NPAR,NPAR),
C sigma
C local scalars
C real temp, zero
C data zero /0.0/

C
C INVERT H, USING VCV AS WORKSPACE.
IFault = 1
CALL F01AFH(NVAR, VCV, NPAR, WORK, IFault)
IF (IFault .EQ. 0) GO TO 10
C ON FAILURE OF INVERSION, SET VCV TO 0 AND RETURN.
CALL F01ACF(VCV, NPAR, NPAR, IND)
RETURN
C FILL OUT VCV AND V (CURRENTLY ONLY HALF FULL).
10 DO 20 I = 1, NVAR
   DO 20 J = 1, I
   V(I,J) = V(J,I)
   VCV(J,I) = VCV(I,J)
20 CONTINUE
C SET H TO ITS INVERSE IN SUCH A WAY THAT MULTIPLICATION VIA
C FOICF WILL BE CORRECT. (NOTE DIMENSIONS IN FOICF CALL)
CALL FOICF(VCV, NPAR, H, NVAR, NPAR, NVAR)
C MULTIPLY H * V, OVERWRITING V.
CALL FOICFHV(H, V, NPAR, NVAR, NVAR, WORK, NVAR, 3, IFault)
C MULTIPLY V * H, OVERWRITING V.
CALL FOICFHV(V, H, NVAR, NVAR, NPAR, WORK, NPAR, 2, IFault)
C V NOW CONTAINS COVARIANCE MATRIX FOR FREE VARIABLES. ARRANGE
C ITS CONTENTS IN VCV, ADJUST SIGMA AND MU ENTRIES FOR SCALE.
CALL FOICFHV(VCV, NPAR, NPAR, IFault)
DO 30 I = 1, NVAR
   DO 30 J = 1, I
   IND = MIN(1SUB(I), 1SUB(J))
   IND1 = MAX(1SUB(I), 1SUB(J))
   VCV(IND, IND1) = V(I,J)
30 CONTINUE
C FILL LOWER TRIANGLE OF VCV WITH CORRELATIONS.
DO 50 I = 2, NPAR
   TEMP = VCV(I,I)
   IND = I - 1
   DO 50 J = 1, IND
      IF (AMIN1(TEMP, VCV(J,J)) .LE. ZERO) GO TO 50
      VCV(J,J) = VCV(J,J) / SQRT(TEMP*VCV(J,J))
50 CONTINUE
C RETURN
END
C***********************************************************************
C COMPUTES APPROXIMATE UPPER TRIANGLE OF HESSIAN BY DIFFERENCES.
C NOTE THAT WITH IFLAG = 0, FUNCT WILL NOT SET A GRADIENT.
C CALLED BY - VARIAB
C CALLS - FUNCT
C***********************************************************************
SUBROUTINE HESDIF(PAR, NPAR, ISUB, H, SAVE1, SAVE2, NVAR, IW, LW)
*W, LW)
C ARGUMENTS
**HESD 001
**HESD 002
**HESD 003
**HESD 004
**HESD 005
**HESD 006
**HESD 007
**HESD 008
**HESD 009
INTEGER NPAR, NVAR, ISUB(NVAR), LIW, IW(LIW), LW
REAL PAR(NPAR), H(NVAR, NVAR), SAVE1(NVAR, NVAR), SAVE2(NVAR, NVAR), HESD 010
*MLW)
HESD 012
C
LOCAL SCALARS
INTEGER ITER, IND, ND1, ND2, IND
REAL \*PAR, PARJ, TEM, TEM1, STEP, DENOM, ZERO, TOL, TEM2,
*STEP1, ONE, TWO, SQRT10, FOUR
DATA ZERO, TOL, TEM2, STEP1, ONE, TWO, SQRT10, FOUR /0.0, 1.0E-6, 1.0E-4,
*1.0E-4, 3.162277660E-3, 1.0, 2.0, 3.162277660, 4.0/
HESD 014
HESD 015
HESD 016
HESD 017
HESD 018
HESD 019
HESD 020
HESD 021
HESD 022
HESD 023
HESD 024
HESD 025
HESD 026
HESD 027
HESD 028
HESD 029
HESD 030
HESD 031
HESD 032
HESD 033
HESD 034
HESD 035
HESD 036
HESD 037
HESD 038
HESD 039
HESD 040
HESD 041
HESD 042
HESD 043
HESD 044
HESD 045
HESD 046
HESD 047
HESD 048
HESD 049
HESD 050
HESD 051
HESD 052
HESD 053
HESD 054
HESD 055
HESD 056
HESD 057
HESD 058
HESD 059
HESD 060
HESD 061
HESD 062
HESD 063
HESD 064
HESD 065
HESD 066
HESD 067
HESD 068
HESD 069
CALL FUNCT(0, MPAR, PAR, TEMP1, PAR, IW, LIW, W, LW)  HESD 070
TEMP = TEMP + TEMP1  HESD 071
PAR(IND3) = PARJ + STEP  HESD 072
CALL FUNCT(0, MPAR, PAR, TEMP1, PAR, IW, LIW, W, LW)
            HESD 073
H(I,J) = (TEMP - TEMP1) / DENOM  HESD 074
PAR(IND2) = PARJ  HESD 075
PAR(IND3) = PARJ  HESD 076
40 CONTINUE  HESD 077
C  F1ND DIFFERENCE, SAVE OLD HESSIAN IN SAVE1.
C  50 TEMP = ZERO  HESD 078
        DO 60 I = 1, NVAR  HESD 079
        DO 60 J = 1, NVAR  HESD 080
        PARI = H(I,J)  HESD 081
        TEMP1 = ABS(PAR1 - SAVE1(I,J))  HESD 082
        PARJ = ABS(PAR1)  HESD 083
        IF (PARJ .GE. ONE) TEMP = TEMP1 / PARJ  HESD 084
        TEMP = AMAX1(TEMP,TEMP1)  HESD 085
        SAVE1(I,J) = PAR1  HESD 086
60 CONTINUE  HESD 087
C  60 CONTINUE  HESD 088
C  THIRD ITM, SAVE OLD HESSIAN IN SAVE2 (STEPLength 1.0E-4).
C    IF (ITER .NE. 3) GO TO 80  HESD 089
        DO 70 I = 1, NVAR  HESD 090
        DO 70 J = 1, NVAR  HESD 091
        SAVE2(I,J) = H(I,J)  HESD 092
70 TEST STOPPING CRITERION FOR ITERATION.
C    80 IF (TEMP .LT. TOL) GO TO 100  HESD 093
        IF (ITER .LT. 5) GO TO 20  HESD 094
C  NO CONVERGENCE IN 5 ITNS - USE SAVE2, WITH STEP TEMPS.
        STEP = TEMPS  HESD 095
        DO 90 I = 1, NVAR  HESD 096
        DO 90 J = 1, NVAR  HESD 097
        H(I,J) = SAVE2(I,J)  HESD 098
90 EXIT, WRITE DETAILS (IND IS OUTPUT UNIT PASSED THROUGH IW),
C    AND SAVE THE NUMBER OF ITERATIONS REQUIRED.
C    100 IND = IW(4)  HESD 099
       IW(I) = ITER  HESD 100
       IF (IND .GT. 0) WRITE (IND,1000) ITER, STEP  HESD 101
C    1000 FORMAT (16HOHESSIAN DONE IW, 13, 6ITNS,, 16H STEPSIZE USED =, 
               \*E11.3)  HESD 102
C  RETURN  HESD 103
END  HESD 104
A Family of Algorithms for the Estimation of the Parameters of the Stable Laws and the Parameters of Attracting Stable Laws

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

stable laws, parameter estimation, adaptive estimation, empirical characteristic function, domains of attraction, sensitivity analysis, nonlinear optimization, constrained estimation

This paper presents several families of algorithms for estimation of the parameters of the stable laws and the parameters of attracting stable laws. The paper also presents algorithms for estimation of the parameters of stable regression and stable autoregression models.