United States
Department of
Agriculture

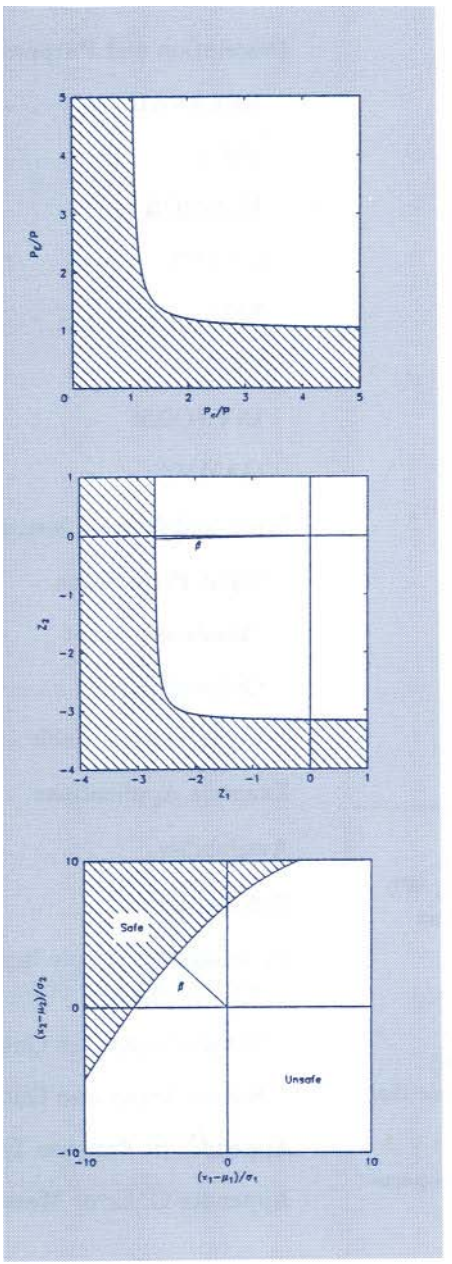
**Forest Service**

Forest
Products
Laboratory

General
Technical
Report
FPL-GTR-72

# Fortran Programs for Reliability Analysis

**John J. Zahn**



## Abstract

This report contains a set of FORTRAN subroutines written to calculate the Hasofer–Lind reliability index. Nonlinear failure criteria and correlated basic variables are permitted. Users may incorporate these routines into their own calling program (an example program. RELANAL, is included) and must provide a failure criterion subroutine (two example subroutines. LINFAIL and YLINEN. are included).



Keywords: Reliability. FORTRAN. safety. design

July 1992

Zahn. John J. 1992. Fortran programs for reliability analysis. Gen. Tech. Rep. FPL–GTR–72 Madison. WI: U.S. Department of Agriculture. Forest Service. Forest Products Laboratory. 25 p.

A limited number of free copies of this publication are available to the public from the Forest Products Laboratory. One Gifford Pinchot Drive. Madison. WI 53705–2398 Laboratory publications are sent to more than 1,000 libraries in the United States and elsewhere.

The Forest Products Laboratory is maintained in cooperation with the University of Wisconsin.


## Contents

*Page*

# Fortran Programs for Reliability Analysis

**John J. Zahn,** Research General Engineer
Forest Products Laboratory, Madison, Wisconsin

## Introduction

The high level of interest in structural reliability among wood engineers prompted development of these programs. The author served on a task committee of the American Society of Civil Engineers whose task was to develop a Load and Resistance Factor Design pre-standard (American Society of Civil Engineers, 1988) and wrote these programs as a part of that effort.

The programs are intended for teaching and research applications. The user should be familiar with the basics of structural reliability analysis. An excellent introductory text is that of Thoft–Christensen and Baker (1982), which is listed in the references. The theory is briefly reviewed here under the heading Discussion of Main Algorithm. Most users should be those who write their own Fortran programs for particular reliability analysis applications. These subroutines are designed to calculate the so-called Hasofer–Lind reliability index (Hasofer and Lind, 1974; Thoft–Christensen and Baker, 1982) by a trial and error technique that is well suited to civil engineering design applications. The programs are quite general; the failure criterion may be nonlinear, and the variables it contains may be correlated. The user must provide the failure criterion subroutine and prescribe the correlation matrix if the variables are not independent.

The code is compact and well suited for use on personal computers. The language conforms to the ANSI FORTRAN 77 standard.

The heart of these software programs is the subroutine EL2SRCH and its subprograms DIST, INVNORM, and GAMMA. When linked with a user-supplied subroutine FAIL, these programs perform the trial and error search technique described in Thoft–Christensen and Baker (1982). See Discussion of Main Algorithm and Description and Purpose of Programs for complete details.

An example program RELANAL is provided showing how these subroutines can be incorporated into a larger reliability analysis program. Two examples of the user-supplied subroutine FAIL, namely LINFAIL and YLINEN, are also shown.

This manual contains a structure chart for program RELANAL, program descriptions, input/output summaries, and example applications. Availability of the programs is discussed under the heading Availability. Complete program listings are provided in Appendix B to further clarify the meaning of the algorithms and to enable users to modify the programs to suit their needs. Error messages are shown in Appendix C.

## Discussion of Main Algorithm

In the reliability analysis performed in these programs, the Hasofer–Lind reliability index β is calculated by a process of minimization. That index is approximately related to the probability of failure by the equation

$$P_f = \Phi(-\beta), \qquad \beta = -\Phi^{-1}(P_f) \tag{1}$$

in which Φ is the standard normal cumulative distribution function (CDF). Equation (1) is exact when the failure criterion is linear and all random variables have normal distributions. β is defined as follows:

Given a vector of basic variables $\boldsymbol{X}$,

a failure surface $\partial\omega$ on which the failure criterion $g(\boldsymbol{X}) = 0$, and

a safe region $g(\boldsymbol{X}) > 0$,

introduce reduced variables $\boldsymbol{Z}$

$$\boldsymbol{Z} \equiv \boldsymbol{S}_x^{-1}(\boldsymbol{X} - \boldsymbol{\mu}_x) \tag{2}$$

in which $\boldsymbol{S}_x$ is a diagonal matrix of standard deviations of $\boldsymbol{X}$, and $\mu_x$ is the mean of $\boldsymbol{X}$. Then the Hasofer–Lind reliability index b is defined as (Fig. 1)

$$\beta \equiv \min_{\boldsymbol{Z} \in \partial\omega} \sqrt{\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}} \tag{3}$$

The point on the failure surface at which $\boldsymbol{Z}$ has minimum magnitude is called the design point and will be denoted $\boldsymbol{Z}^*$. The vector $\boldsymbol{Z}$ can be written as the product of its magnitude β and a unit vector $\boldsymbol{\alpha}$:

$$\boldsymbol{Z} \equiv \boldsymbol{\alpha}\beta \tag{4}$$

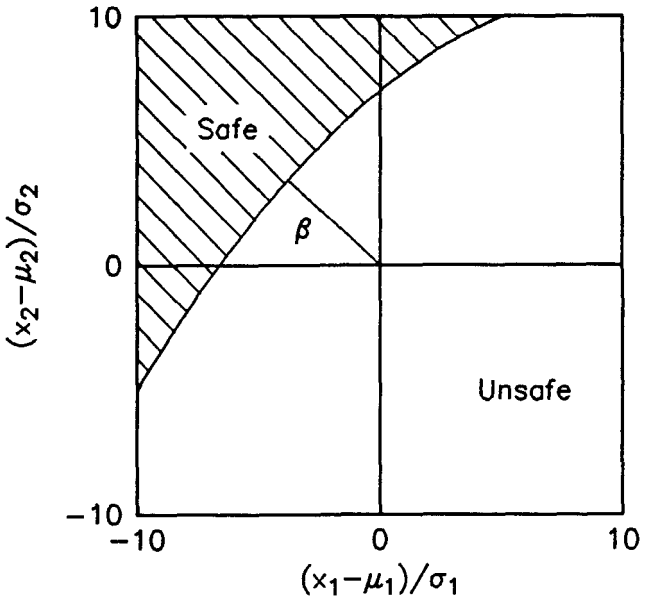



*Figure 1–Definition of Hasofer–Lind reliability index β. Safe region is shaded.*

When $\boldsymbol{Z}$ is a minimum the unit vector $\boldsymbol{\alpha}$ will be normal to the surface $\partial\omega$. Therefore we use a vector c which is normal to that surface to construct $\boldsymbol{\alpha}$. Let $h(\boldsymbol{Z})$ be $g(\boldsymbol{X})$ in the $\boldsymbol{Z}$ space. Then a suitable choice for $\boldsymbol{c}$ is

$$\boldsymbol{c} \equiv \frac{\partial h}{\partial \boldsymbol{Z}} \tag{5}$$

and

$$\boldsymbol{\alpha} = -\frac{\boldsymbol{c}}{|\boldsymbol{c}|} \tag{6}$$

Using the chain rule for partial differentiation, (4) becomes

$$\boldsymbol{c} = \boldsymbol{S_x b} \tag{7}$$

in which

$$\boldsymbol{b} \equiv \frac{\partial g}{\partial \boldsymbol{X}} \tag{8}$$

The following search loop[1] will converge to b if the initial guess for the design point is sufficiently close:

Initialize $b_i := \pm 1, \quad \beta := \beta_0$
Step 1–Construct $\boldsymbol{c}$ normal to failure surface at the estimated design point:
$\boldsymbol{c} := \boldsymbol{S_x b}$
Reduce $\boldsymbol{c}$ to a unit vector a:
$|\boldsymbol{c}| := \sqrt{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{c}}$
$\boldsymbol{\alpha} := -\boldsymbol{c}/|\boldsymbol{c}|$
Get estimated design point in $\boldsymbol{X}$-space:
$\boldsymbol{X} := \boldsymbol{\mu}_x + \boldsymbol{S}_x\boldsymbol{\alpha}\beta$
Evaluate $\boldsymbol{b} \equiv \partial g/\partial \boldsymbol{X}$ (subroutine)
Solve $g(\boldsymbol{X}) = 0$ for $\beta$ (subroutine)

Iterate from step 1 until convergence is obtained.

Here we take our first guess to be some starting value $\beta_0$ and $b_i = +1$ for resistances and $b_i = -1$ for loads. To ensure convergence, a good starting value $\beta_0$ must be found by trial or past experience. Starting the loop with the multiplication by $\boldsymbol{Sx}$ is motivated by the desire to start with a slightly better first guess for the $\boldsymbol{a_i}$ than if the $\boldsymbol{c_i}$ had simply been initialized to ±1.

## Linear Approximation of *g*

Solving $g = 0$ for β each time through the loop may be difficult when $g(\boldsymbol{X})$ is nonlinear. As long as we are relying upon convergent iteration we may as well approximate $g(\boldsymbol{X})$ with a local linear approximation each time through the loop. Then both approximation processes can converge simultaneously. This replaces the solution of $g(\boldsymbol{X}) = 0$ with the simple replacement statement

$$\beta := \beta + \frac{g}{|\boldsymbol{c}|} \tag{9}$$

This result is easily derived as follows:

Let $\boldsymbol{Z}$ be the current trial value for which $g$ is small but not yet zero and $\boldsymbol{Z}'$ be the next trial value. Then by Taylor's theorem

$$\begin{aligned} g(\boldsymbol{Z}') &= g(\boldsymbol{Z}) + \sum_i \frac{\partial g}{\partial Z_i}(Z_i' - Z_i) \\ &= g + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{\alpha}(\beta' - \beta) \end{aligned} \tag{10}$$

We want $g(\boldsymbol{Z}') = 0$ and by construction of a (6) we have that $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{\alpha} = -|\boldsymbol{c}|$. Therefore (9) follows.

Solving for b is the "analysis" problem. The corresponding "design" problem is one in which b is specified and the mean of one of the basic variables is to be found. Again, replacing g with a linear Taylor series approximation results in a simple replacement statement:

$$\mu_j := \mu_j - \frac{g}{b_j} \tag{11}$$

This too is easily established. Applying Taylor's theorem in the $\boldsymbol{X}$ space,

$$g(\boldsymbol{X}') = g(\boldsymbol{X}) + \sum_i \frac{\partial g}{\partial X_i}(X_i' - X_i) \tag{12}$$

This time only the *j*-th *X* varies so the summation reduces to *bj(μ'j - μj)* and (11) follows.

[1] The := sign used here is a "replacement" operator. For example, $\boldsymbol{x} := \boldsymbol{x} + 1$ means that $\boldsymbol{x}$ is replaced by $\boldsymbol{x} + 1$.

### Example of a Nonlinear Failure Criterion

Suppose we wish to analyze the reliability of a column design using Ylinen's column formula as the nonlinear failure criterion. The column formula is (Ylinen, 1956)

$$P = U - \sqrt{U^2 - \frac{P_c P_e}{c}} \tag{13}$$

in which

$$U \equiv \frac{P_c + P_e}{2c} \tag{14}$$

and $P$ is the failing load, $P_c$ is the crushing resistance, $P_e$ is the Euler buckling resistance, and $0 < c < 1$ is an adjustable parameter. The two resistances $P_c$ and $P_e$ are obtained from distributions for compressive strength $F_c$ and modulus of elasticity $E$ as follows:

$$P_c = AF_c \tag{15}$$

$$P_e = BE \tag{16}$$

in which $A$ is the cross section area, $B = \pi^2 A/(L/r)^2$, and $L/r$ is the slenderness ratio (ratio of effective length to radius of gyration of cross section).

For simplicity, we shall suppose that the load is determinate so that the problem is statistically two dimensional.

The evaluation of $g$ and its derivatives $\boldsymbol{b}$ would be encoded as follows:

$$U := (AF_c + BE)/(2c)$$
$$V := AF_cBE/c$$
$$R := \sqrt{U^2 - V}$$
$$g := U - R - P$$
$$\partial g/\partial F_c := (1 - U/R + BE/R)A/(2c)$$
$$\partial g/\partial E := (1 - U/R + AF_c/R)B/(2c)$$

For this two-dimensional problem, the safe region would appear as in Figure 2. Figure 3 shows the safe region in the $\boldsymbol{Z}$–plane.

### Non-normal Variables

The simple relationship of the Hasofer–Lind reliability index β to the probability of failure $p_f$ shown in Equation (1) is exact only if all the basic variables are normal and the failure criterion is linear. This relation is approximately true even when the failure criterion is nonlinear so long as it is approximately linear near the design point.

To extend β to non-normal variables, it is sufficient to replace the distributions of non-normal variables with local normal approximations. Rackwitz and Fiessler (1977) proposed the simple expedient of choosing the normal approximation such that its CDF has the same ordinate and slope as the non-normal CDF at the design point (Fig. 4). We do this as follows:

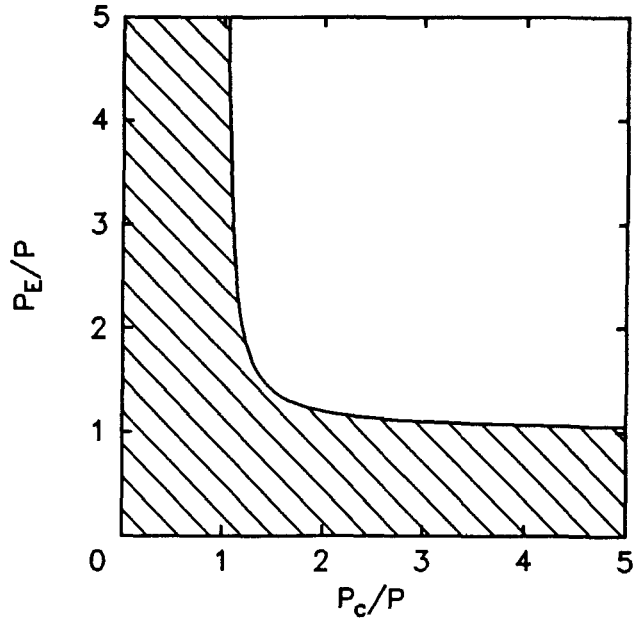



***Figure 2–Ylinen failure criterion in X plane with $c$ = 0.80. Safe region is shaded.***

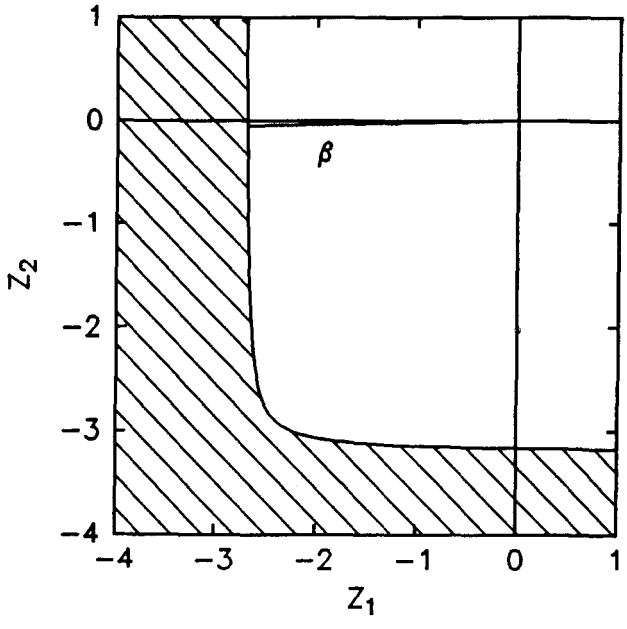



**Figure 3–Ylinen failure criterion in Z plane. Safe region is shaded.**

Suppose that $X_j$ is non-normal with CDF $F_j(X_j)$ and probability density function (PDF) $f_j(X_j)$. Recall that the PDF is merely the derivative (slope) of the CDF. Let the corresponding normal approximations be $\Phi$ and $\phi$ with mean $\mu_{rfj}$ and standard deviation $\sigma_{rfj}$. Equating ordinates requires

$$F_j(X_j^*) = \Phi\left(\frac{X_j^* - \mu_{rfj}}{\sigma_{rfj}}\right) \tag{17}$$

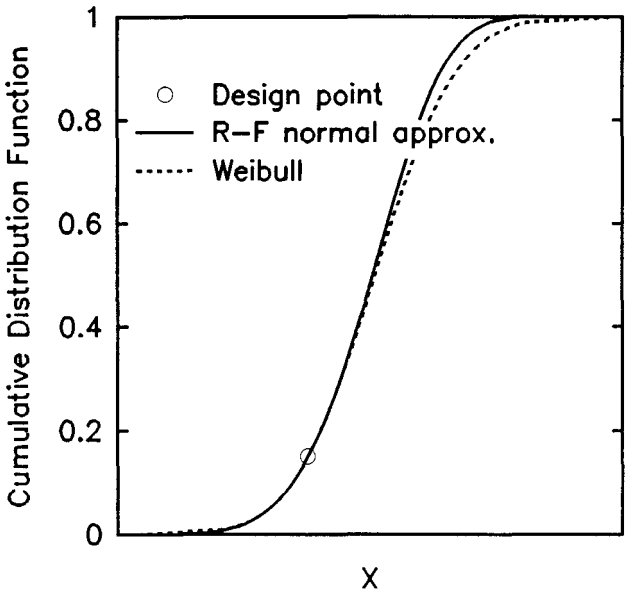



***Figure 4–Definition of Rackwitz–Fiessler normal approximation. Ordinate and slope are matched at design point.***

and equating slopes requires

$$f_j(X_j^*) = \phi\left(\frac{X_j^* - \mu_{\mathrm{rf}j}}{\sigma_{\mathrm{rf}j}}\right)\frac{1}{\sigma_{\mathrm{rf}j}} \qquad (18)$$

This Rackwitz–Fiessler algorithm can be encoded as

$$z := \Phi^{-1}[F_j(X_j)]$$
$$\sigma_{\mathrm{rf}j} := \phi(z)/f_j(X_j)$$
$$\mu_{\mathrm{rf}j} := X_j - \sigma_{\mathrm{rf}j} z$$

in which the superscript denoting design point has been omitted because at any particular iteration of the loop $\boldsymbol{X}$ is only approaching $\boldsymbol{X}^*$. That is, the design point is only reached when the loop has converged. This coding can be inserted at the end of the loop given previously. Throughout the rest of the loop, μrfj and qrfj should be used in place of the actual mean and standard deviation of $X_j$. The algorithm should be applied to every non-normal $X_j$.

If $F_j$ is the lognormal with mean $\mu_j$, coefficient of variation $CV_j$, and lower limit $x_{j0}$, then it is easily shown that (17) and (18) can be reduced in closed form. The result can be encoded as

$$\omega := \sqrt{1 + [CV_j\mu_j/(\mu_j - x_{j0})]^2}$$
$$\sigma_{\mathrm{rf}j} := (X_j - x_{j0})\sqrt{2\ln\omega}$$
$$\mu_{\mathrm{rf}j} := X_j + (X_j - x_{j0})\ln[(\mu_j - x_{j0})/((X_j - x_{j0})\omega)]$$

## Correlated Variables

The essential method here is to use a similarity transformation to a space in which the variables are uncorrelated and then transform back, However, no actual transformation need be performed-only some matrix multiplications with the correlation matrix $\boldsymbol{R}$. The derivation of the equations may be found in Zahn (1989).

The resulting search loop contains the new vectors $\boldsymbol{d}$ and $\boldsymbol{\gamma}$ which are analogous to $\boldsymbol{c}$ and $\boldsymbol{\alpha}$ respectively. If $\boldsymbol{X}$ is uncorrelated, then $\boldsymbol{R}$ is the unit matrix and $\boldsymbol{d}$ and $\boldsymbol{\gamma}$ reduce identically to $\boldsymbol{c}$ and $\boldsymbol{\alpha}$ defined previously. Because the existence of the similarity transformation is always assured, the algorithm always works, at least for normal $\boldsymbol{X}$.

The only problem arises when some of the basic variables are non-normal. The standard definition of correlation in terms of covariance applies only to normal variables. No definition exists for a correlation matrix if the variables are non-normal, although some alternative measures of association appear in the statistical literature for a few special cases.

If one employs the Rackwitz–Fiessler algorithm as shown above, then *substitute normal variables* appear in $\boldsymbol{X}$, and $\boldsymbol{R}$ must be understood as defining the correlation of this substitute $\boldsymbol{X}$. The substitute $\boldsymbol{X}$ will have a substantially different covariance matrix than the original $\boldsymbol{X}$, and the appropriate $\boldsymbol{R}$ matrix to be used in this algorithm is unknown.

Nevertheless, if one is only interested in studying the qualitative effects of correlation, or the effect of perfect correlation, then a user-supplied correlation matrix is sufficient, even when the variables are non-normal. Therefore, for generality, EL2SRCH employs a search loop with both a correlation matrix and the Rackwitz–Fiessler algorithm. Therefore the main algorithm has the following final form.

## Main Algorithm

Initialize $b_i := \pm 1$, $\beta := \beta_0$

Step 1–Construct $\boldsymbol{d}$ (analogous to $\boldsymbol{c}$):

$$\boldsymbol{d} := \boldsymbol{S_x b}$$

Reduce $\boldsymbol{d}$ to $\boldsymbol{\gamma}$ (analogous to $\boldsymbol{\alpha}$):

$$|\boldsymbol{c}| := \sqrt{\boldsymbol{d}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{d}}$$
$$\boldsymbol{\gamma} := -\boldsymbol{R}\boldsymbol{d}/|\boldsymbol{c}|$$

Get estimated design point in $\boldsymbol{X}$-space:

$$\boldsymbol{X} := \boldsymbol{\mu_x} + \boldsymbol{S_x}\boldsymbol{\gamma}\beta$$

Evaluate $g(\boldsymbol{X})$ and $\boldsymbol{b} \equiv \partial g/\partial \boldsymbol{X}$ **(subroutine)**

Either $\beta := \beta + g/|\boldsymbol{c}|$ **(analysis)**

or $\mu_j := \mu_j - g/b_j$ **(design)**

For all non-normal $X_j$ get normal approximations:

$$z := \Phi^{-1}[F_j(X_j)]$$
$$\sigma_{\mathrm{rf}j} := \phi(z)/f_j(X_j)$$
$$\mu_{\mathrm{rf}j} := X_j - \sigma_{\mathrm{rf}j} z$$

Iterate from step 1 until convergence is obtained.

## Notation

File names, including program names, are shown in capital letters:

INPT

File contents are shown in typewriter font:

```
'DE',3,0.25
```

Generic file names are shown in angle brackets:

<filename>.DAT

This means you are to supply your own file name:

JOE.DAT

or whatever is appropriate. Generic file contents are shown similarly but in typewriter font:

```
<beta>
```

This means you are to supply a numerical value such as 2.75, or whatever is appropriate. Fortran variable types such as real, integer, character*4, etc. are shown in parentheses, such as **<k>** (integer) where needed.

## Structure Chart

The structure chart for program RELANAL is shown in Figure 5.

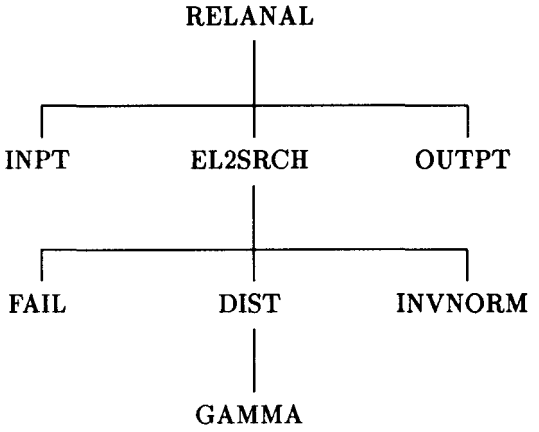



***Figure 5–Structure chart for program RELANAL.***

## Description and Purpose of Programs

The programs are described in the order of their appearance on the structure chart for the example application RELANAL.

### RELANAL

Program RELANAL has two uses: reliability analysis and reliability-based design. It can be run in two modes:

(1) Analysis mode. In the limited context of this manual, "reliability analysis" means the following: Given a failure criterion (or safety checking equation) and distributions for each of its random variables, RELANAL calculates the Hasofer–Lind reliability index $\beta$.

(2) Design mode. "Design" in this context means that one uses the failure criterion as a design equation and solves for the mean of one of the random variables. Given the reliability index and all distributions except one, for which the type and COV are specified, RELANAL calculates its mean. If the nominal design value is defined as some characteristic value of the distribution other than the mean, the user must calculate the nominal from its definition.

How to specify analysis or design mode is discussed under the heading Input Preparation Instructions for RELANAL.

### INPT

The purpose of program INPT is to provide input to RELANAL. Input is read from <filename>.DAT, which is prepared in advance. An output log file <filename>.LOG is opened for use by all programs, and a copy of all input is written to the log file.

### EL2SRCH

Program EL2SRCH provides a search loop that minimizes the Hasofer–Lind reliability index $\beta$. It is the heart of this set of programs. See Discussion of Main Algorithm.

### OUTPT

Program OUTPT prints the final values of each random variable at the design point, along with its associated component of $\boldsymbol{\gamma}$. The magnitude of its $\boldsymbol{\gamma}$ component is a rough measure of the importance of the variable in determining the reliability index $\beta$.

Finally, it prints the converged value of b (analysis mode) or $\mu_j$, (design mode).

### FAIL

Program FAIL calculates the failure function $g(\mathbf{X})$ and its partial derivatives $\boldsymbol{b}$. If you wish to associate names with the basic random variables $\boldsymbol{X}$, a character*4 array XNAME is provided for that purpose. Another real*8

array, PARAMS, is provided for passing parameters between EL2SRCH and FAIL.

This program must be user written. However, a linear version of FAIL is supplied in file LINFAIL.FOR. The linear version uses XNAME to distinguish between loads and resistances. If XNAME(j)='**LOAD**', then X(j) is a load, and if XNAME(j)='**RESI**', then X(j) is a resistance. It uses PARAMS(1) to pass **<N>** where **<N>** is the number of variables in $g(\boldsymbol{X})$, that is, the dimension of $\boldsymbol{X}$.

### DIST

Program DIST evaluates the CDF and the PDF of a specified distribution. The distribution must be one of the following: Weibull, Gumbel, or Frechet. The search loop handles normal and lognormal distributions in closed form.

### INVNORM

Function INVNORM evaluates the inverse normal CDF. It uses a Hastings polynomial approximation with absolute error less than $4.5 \times 10^{-4}$ (Abramowitz and Stegun, eds., 1972).

### GAMMA

Function GAMMA evaluates the gamma function $\Gamma(x)$ where $\Gamma(n) = (n - 1)!$. It uses a Hastings polynomial approximation with absolute error less than $3 \times 10^{-7}$ (Abramowitz and Stegun, eds., 1972).

## Input and Ouput Instructions for RELANAL

### Input Preparation

These instructions are also shown on the listing of program INPT. All input is obtained from an input file prepared in advance. INPT prompts for the input filename. The input file is prepared as follows. Note that when inputing character variables you must include the single quotes as shown. Either commas or blanks may be used as separators.

### Mandatory Input

You must enter basic variables, one per line, as follows:

**<class>, <type>, <mean>, <std dev>, <min>, <name>**

in which

**<class>** (character*2) is **'RE'** or **'LO'**, where **'RE'** denotes a resistance variable and **'LO'** denotes a load variable.

**<type>** (character*2) is **'NO'** or **'LO'** or **'WE'** or **'GU'** or **'FR'**. This specifies the distribution type. **'NO'** denotes normal, and the other specifications denote lognormal, Weibull, Gumbel, and Frechet respectively.

**<mean>** (real) is the mean value,

**<std dev>** (real) is the standard deviation,

**<min>** (real) is the lower limit (this must be included but will be ignored for those distributions that have −∞ as their lower limit), and

**<name>** (character*4) is **'RESI'** or **'LOAD'**. These are variable names that LINFAIL uses to distinguish between resistances and loads.

The only other mandatory input is

**'ST'** (character*2).

This terminates subroutine INPT and starts subroutine EL2SRCH. If a user written subroutine FAIL requires input, you could read such input from the .DAT file also. In that case the FAIL input would follow **'ST'.**

### Optional Input

Optional input is also entered one statement per line as follows:

**'AN'** or

**'DE',<k>,<COV>,**

where **'AN'** (character*2) prescribes the analysis mode (solving for b), and **'DE'** (character*2) prescribes the design mode (solving for the mean of one variable). In design mode, **<k>** (integer) is the index of the desired variable and **<COV>** (real) is its coefficient of variation. The index of a variable depends on the order in which it is read from the input file (see Mandatory Input). The first variable has index 1; the second has index 2, etc.

**'EP',<epsilon>,**

where **<epsilon>** (real) is the relative error in b or the mean of variable $k$, $\mu_k$, at output (default value of epsilon is 0.0001). If the search loop halts after only two executions, you may need to supply a smaller value of epsilon in order to get things started. This can happen if the basic variables vary enormously in magnitude. In general, scaling the basic variables so that they differ by no more than a factor of one million is good practice.

**'BE',<beta>**

where **<beta>** (real) is the reliability index (default is 3.0). In analysis mode this is the initial guess, and in design mode it is the target value.

**'IN'** or

**'CO'**

where **'IN'** (character*2) denotes independence and **'CO** (character*2) denotes correlation. If correlation is specified, then the upper triangular correlation matrix (elements above the diagonal) must be input immediately following, one element per line:

**'RM',<i>,<j>,<r>**

where **'RM'** (character*2) identifies the input line as a correlation matrix element, **<i>** and **<j>** (integer) are the row and column indices of the matrix element, and **<r>** (real) is its value.

### Input/Output Table

Table 1 shows the arguments of each program and whether they are input, output, or both. The program arguments are also described in comments at the head of each listing.

## Example Applications

Program RELANAL references INPT, OUTPT, and EL2SRCH. See Input and Output for detailed input preparation instructions.

Program EL2SRCH finds β if you specify **'AN'** (analysis mode) or $\mu_j$, if you specify **'DE'** (design mode). It calls subprogram FAIL to calculate $g$ and $\boldsymbol{b} = \partial g/\partial \boldsymbol{X}$. Two examples of FAIL are provided; a linear g is in file LINFAIL, and a nonlinear g corresponding to Ylinen's column formula is in file YLINEN. Either one may be linked into the system, or the user may write a version of FAIL that meets specific needs. EL2SRCH also calls subprogram DIST during execution of the Rackwitz–Fiessler algorithm. The following non-normal distributions are supported: Weibull, Gumbel, and Frechet. If another distribution is desired, it must be user supplied (the CDF and the PDF must be returned by DIST). DIST in turn calls GAMMA when evaluating the Weibull and Frechet distributions.

The result (β or $\mu_j$) is output by OUTPT which also gives the final values of the vector of basic variables $\boldsymbol{X}$ and the vector $\boldsymbol{\gamma}$. The components of $\boldsymbol{\gamma}$ indicate the relative importance of the corresponding basic variable. (See Appendix A.)

## Availability

These programs are available on diskette from Forest Products Laboratory. Requesters should send a blank formatted diskette along with their request. Future updates are not contemplated.

## References


**Abramowitz, M.; Step, I.A., eds.** 1972. Handbook of mathematical functions with formulas, graphs, and mathematical tables. Applied Mathematics Series 55. Washington, DC: U.S. Department of Commerce, National Bureau of Standards.

**Hasofer, A.M.; Lind, N.C.** 1974. An exact and invariant first order reliability format. Journal of the Engineering Mechanics Division, American Society of Civil Engineers. EM1:111–121.

**ASCE.** 1988. Load and resistance factor design for engineered wood construction. A pre-standard report. American Society of Civil Engineers, New York.

**Rackwitz, R.; Fiessler, B.** 1977. An algorithm for calculation of structural reliability under combined loading. Berichte zur Sicherheitstheorie der Bauwerke, Lab. f. Konstr. Ingb. Munich, Germany.

**Thoft-Christensen P.; Baker, MJ.** 1982. Structural reliability theory and its applications. Springer–Verlag, New York.

**Ylinen, A.** 1956. A method of determining the buckling stress and the required cross-sectional area for centrally loaded straight columns in the elastic and inelastic range. International Association for Bridges and Structural Engineering, Zurich, Switzerland, 16: 529–550.

**Zahn, J.J.** 1989. Empirical failure criteria with correlated resistance variables. Journal of Structural Engineering. 116(11): 3122–3137.

## Table 1–Input/Output Summary

| Program | Argument | Type | Description | IN | OUT |
|---|---|---|---|---|---|
| DIST | X | Real*8 | One of the basic variables | * | |
| | I | Integer | Index of variable X | * | |
| | J | Integer | Indicates distribution type | * | |
| | Xo | Real*8 Array | Lower limits of distributions | * | |
| | MEAN | Real*8 Array | Means of distributions | * | |
| | STDV | Real*8 Array | Standard deviations of distributions | * | |
| | CKPR | Logical | Controls checkprinting | * | |
| | CDF | Real*8 | Cumulative distribution function of X | | * |
| | PDF | Real*8 | Probability density function of X | | * |
| EL2SRCH | DISTYP | Integer Array | Distribution type indicators | * | |
| | PROBTYP | Character*2 | ANalysis or DEsign | * | |
| | BETA | Real*8 | Reliability index | * | * |
| | K | Integer | Design variable index | * | |
| | COV | Real*8 | Design variable COV | * | |
| | EPS | Real*8 | Relative error allowed | * | |
| | N | Integer | Number of basic variables | * | |
| | CORR | Character*1 | (Y/N)Are basic variables correlated? | * | |
| | X | Real*8 Array | Vector of basic variables | * | * |
| | GAMA | Real*8 Array | Unit vector in X space | | * |
| | PARAMS | Real*8 Array | User's values passed to FAIL | * | |
| | XNAME | Character*4 Array | User's names for basic variables | * | |
| | B | Real*8 Array | Partial derivatives of G | | * |
| | R | Real*8 Array | Correlation matrix | * | |
| | xo | Real*8 Array | Lower limits of basic variables | * | |
| | MEAN | Real*8 Array | Means of basic variables | * | * |
| | STDV | Real*8 Array | Std dev's of basic variables | * | |
| | CKPR | Logical | Controls checkprinting | * | |
| FAIL | X | Real*8 | Vector of basic variables | * | |
| | B | Real*8 | Partial derivatives of basic variables | | * |
| | G | Real*8 | Value of failure function | | * |
| | PARAMS | Real*8 | User's values passed to FAIL | * | * |
| | XNAME | Character *4 | User's names for basic variables | * | |
| INPT | DISTYP | Integer Array | Distribution type indicators | | * |
| | PROBTYP | Character*2 | ANalysis or DEsign | | * |
| | BETA | Real*8 | Reliability index | | * |
| | K | Integer | Design variable index | | * |
| | COV | Real*8 | Design variable COV | | * |
| | EPS | Real*8 | Relative error allowed | | * |
| | N | Integer | Number of basic variables | | * |
| | CORR | Character*1 | (Y/N) Are variables correlated? | | * |
| | XNAME | Character*4 Array | User's names for basic variables | | * |
| | B | Real*8 | Partial derivatives of G | | * |
| | R | Real*8 | Correlation matrix | | * |
| | xo | Real*8 | Lower limits of basic variables | | * |
| | MEAN | Real*8 | Means of basic variables | | * |
| | STDV | Real*8 | Std dev's of basic variables | | * |
| | CKPR | Logical | Controls check printing | | * |
| OUTPT | N | Integer | Number of basic variables | * | |
| | PROBTYP | Character*2 | Analysis or Design | * | |
| | BETA | Real*8 | Reliability index | * | |
| | K | Integer | Design variable index | * | |
| | X | Real*8 | Vector of basic variables | * | |
| | GAMA | Real*8 | Unit vector in X space | * | |
| | XNAME | Character*4 Array | User's names for basic variables | * | |
| | MEAN | Real*8 | Means of basic variables | * | |

# Appendix A. Sample Input and Output for RELANAL

## Sample Input and Output with LINFAIL

When the INPT module prompts you with

```
CHECKPRINT?(Y/N)
```

simply enter **N.** Entering **Y** would turn on checkprinting, a useful device when searching for errors. When the INPT module prompts you with

```
dat filename=?
```

simply enter the filename SAMPLE and the input data will be taken from the file SAMPLE.DAT shown in the next section. Output will appear on the screen, and the file SAMPLE.LOG will be created with a complete record of both input and output. In the next section several examples are shown.

After the linear failure criterion LINFAIL has been linked into the executable program, the following input file SAMPLE.DAT will produce the subsequent output file SAMPLE.LOG:

File SAMPLE.DAT:

```
'RE' 'WE' 4885.525 1645.183 77. 'RESI'
'LO' 'NO' 1102.5 110.25 0 'LOAD'
'AH'
'BE'2.
```

File SAMPLE.LOG:

```
 RE WE   4885.5250000   1645.1830000     77.0000000 RESI
 LO NO   1102.5000000    110.2500000      0.0000000 LOAD
 AN
 BE   2.00000
 ST
 Beta=   2.29431
 Beta=   2.52068
 Beta=   2.54779
 Beta=   2.56395
 Beta=   2.56406
 Design point:
 RESI  X( 1)=0.1135026191E+04  GAMA( 1)=-.9934030325E+00
 LOAD  X( 2)=0.1134915884E+04  GAMA( 2)=0.1146752593E+00
 Beta= 2.5641
```

This example illustrates an application to non-normal basic variables. See the INPT listing for an explanation of the input file SAMPLE.DAT. Note that the output log echoes the input and shows the successive trial values of $\beta$ as well as the final design point. The relative magnitudes of the components of $\gamma$ show the relative significance of each basic variable in determining the value of the reliability index $\beta$.

## Sample Input and Output with YLINEN

If the nonlinear failure criterion YLINEN has been linked into the executable program, the following input file YLINEN1.DAT will produce the subsequent output file YLINEN1.LOG:

File YLINEN1.DAT:

```
'RE' 'WE' 10000. 3000. 000. 'FC'
'RE' 'NO' 1904200. 476050. 0. 'E'
'LO' 'NO' 7929.8 792.98 0. 'P'
'AN'
'BE' 6.0
'EP' .00001
'ST'
5.25
40.
.80
```

File YLINEN1.LOG:

```
 RE WE   10000.000000000   3000.000000000      0.000000000 FC
 RE NO 1904200.000000000  476050.000000000      0.000000000 E
 LO NO    7929.800000000    792.980000000      0.000000000 P
 AN
 BE   6.00000
 EP   0.00001
 ST
 Area=  5.25000
 L/r= 40.00000
 Ylinen c=   0.80000
 Beta=  5.91305
 Beta=  3.64985
 Beta=  3.49692
 Beta=  3.46289
 Beta=  3.46202
 Beta=  3.46201
 Designpoint:
 FC   X( 1)=0.9961587145E+04 GAMA( 1)=-.70284589633-02
 E    X( 2)=0.2586822504E+06 GAMA( 2)=-.9984373093E+00
 P    X( 3)=0.8081998680E+04 GAMA( 3)=0.5543951864E-01
 Beta=3.4620
```

This example illustrates the remarkable stability of the search algorithm even when the failure criterion is highly nonlinear, and the initial guess is far from correct. Such stability is by no means assured in all cases.

The next example is similar to the previous one except that the resistance variables are assumed to be perfectly correlated.

Here the input file is YLINEN2.DAT and the output file is YLINEN2.LOG:

File YLINEN2.DAT:

```
'RE' 'WE' 10000. 3000. 000. 'FC'
'RE' 'NO' 1904200. 476050. 0. 'E'
'LO' 'NO' 7929.8 792.98 0. 'P'
'AN'
'BE' 6.0
'CO'
'RM' 1,2,1.
```

```
'RH' 1,3,0.
'RM' 2,3,0.
'EP' .00001
'ST'
5.25
40.
.80
```

Here we have specified the following correlation matrix:

$$R = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

by specifying the three upper triangular elements.

File YLINEN2.LOG:

```
 RE WE   10000.000000000    3000.000000000       0.000000000 FC
 RE NO 1904200.000000000  476050.000000000       0.000000000 E
 LO NO    7929.800000000     792.980000000       0.000000000 P
 AN
 BE   6.00000
 CO
 RM  1  2   1.00000
 RM  1  3   0.00000
 RM  2  3   0.00000
 EP   0.00001
 ST
 X( 1) reset to  100.00000
 Area=   5.25000
 L/r=  40.00000
 Ylinen c=   0.80000
 Beta=   5.90325
 Beta=   3.61051
 Beta=   3.23928
 Beta=   0.53472
 Beta=   2.70554
 Beta=   2.88399
 Beta=   2.99032
 Beta=   2.99765
 Beta=   3.00000
 Beta=   3.00000
 Design point:
 FC    X( 1)=0.1898752664E+04  GAMA( 1)=-.9943602857E+00
 E     X( 2)=0.4841041616E+06  GAMA( 2)=-.9943602857E+00
 P     X( 3)=0.8182098059E+04  GAMA( 3)=0.1060548078E+00
 Beta= 3.0000
```

This example illustrates what the search algorithm does when a trial value of a basic variable falls below its prescribed lower limit $X_0$. It resets that variable slightly above $X_0$. Such situations could destabilize the search loop and prevent it from terminating. If the program resets every time through the loop, it will prompt for a new value of FACTOR after 10 successive resets (see listing of EL2SRCH for the meaning of FACTOR). Adjusting FACTOR up or down may get the loop unstuck.

# Appendix B. Program Listings

## RELANAL

```
C  RELANAL
      PROGRAM  RELANAL
C RELIABILITY  ANALYSIS  PROGRAM  USING  EL2SRCH  (EXTENDED  LEVEL  2)
C
      REAL*8 B(10),R(10,10),X0(10), MEAN(10), STDV(10)
C
C B    = vector of partial derivatives  of  failure  functicm G;  calculated
C        by subroutine  FAIL
C  R     =Correlation  matrix
C XO   = vector of lower limit values of basic variables
C MEAN = vector of means of basic variables
C STDV = vector of std dev's of basic variables
C
      LOGICAL  CKPR
C
C CKPR = Logical  (Checkprinting  on)  used during  check  out
C
      INTERGER DISTYP(10)
C
C  DESTYP  IS  O-NOrmal,  1-LOgnormal,   2-Weibull,   3-GUmbel,   4-FRechet
C
      REAL*8   BETA,PS,COV,PARAMS(10)
C
C  BETA   = Reliability  index
C  EPS    = relative  error permitted  in BETA or MEAN(K)  at  exit
C  COV    = Coefficient  of variation  of K-th distribution  in DEsign mode
C PARAMS = array of values passed to and from subroutine  FAIL;  included
C
      REAL*8  X(10) ,GAMA(10)
C
C X = vector  of basic  variables
C GAMA = auxiliary  vector  analogous  to alpha
C
      CHARACTER*2  PROBTYP
C
C PROBTYP = ANalysis  or DEsign
C
      CHARACTER*1  CORR,ANS
C
C  CORR = Yes  or No  (Are basic  variables  correlated?)
C
      CHARACTER*4  XNAME(10)
C
C XNAME = CHARACTER*4  array of basic  variable  names  passed  to  subroutine
C   FAIL;  included for users  convenience
C
C  START
C
       PRINT  *,'CHECKPRINT?(Y/N)'
       READ  100,ANS
 100   FORMAT(A1)
       CKPR=.FALSE.
       IF(ANS.EQ.'Y'.OR.ANS.EQ.'y') THEN
          CKPR=.TRUE.
       END  IF
C
C  INPUT
C
       CALL    INPT(DISTYP,PROBTYP,BETA,K,COV,EPS,N,CORR,XNAME,B,R,XO,
      1MEAN,STDV,CKPR)
```

```
      PARAMS(1)=N
C
C  SEARCH  LOOP
C
      CALL    EL2SRCH(DISTYP,PROBTYP,BETA,K,COV,EPS,N,CORR,X,GAMA,PARAMS,
     1 XNAME,B,R,XO,MEAN,SIDV,CKPR)
C
C  PRINT  RESULTS
C
      CALL,  OUTPT(N,PROBTYP,BETA,K,X,GAMA,XNAME,MEAN)
      END
```

## INPT

```
C  INPT
      SUBROUTINE      INPT(DISYP,PROBTP,BETA,K,COV,EPS,N,CORR,XNAME,
     1B,R,XO,MEAN,STDV,CKPR)
C
C INPUT   SUBROUTINE
C
C INPUT:
C All  input is read from <filename>.DAT.   See instructions  below.
C
C OUTPUT:
C DISTYP   =  0-(NOrmal,   1-WEibull,   2-LOgnormal,  3-GUmbel,   4-FRechet
C PROBTYP  =  'AN'alysis  or  'DE'sign
C BETA     =  Reliability  index-initial  guess
C K        = Index of element  of MEAN vector which is to be adjusted  in DEsign
C COV      = Coefficient,  of variation of K-th distribution  in DEsign mode
C EPS      = relative error permitted in BETA or MEAN(K)  at  exit
C N        =  number  of  basic-variables
C CORR     =  'Y'es an 'N'o (Are basic variables correlated?)
C XNAME    = CHARACTER*4 array of basic variable names passed to subrtn FAIL
C B        =  vector  of  partial  derivatives  of  failure function  G-initial  values
C R        =  Correlatian  matrix
C XO       = vector of lower limit values of basic variables
C MEAN     = vector of means of basic variables
C STDV     = vector of std dev's of basic variables
C CKPR     = Logical  (Checkprinting on)  used during check out
C
       REAL*8   BETA,EPS,COV
       INTERGER DISTYP(10)
       cHARACTER*2   PROBTYP,WD,TP
       CHARACTER*1   CORR
       CHARACTER*4   XNAME(10)
       CHARACTER*12    FILENAME
       REAL*8     B(10),R(10,10)XO(10),MEAN(10),STDV(10)
       LOGICAL   CKPR
C
C  WD = temp storage for char*2 input
C  TP = char*2 distribution type
C  FILENAME is the input  and  output  <filename>
C
C  ************************************************************************
C  *               INPUT PREPERATION INSTRUCTIONS                         *
C  ************************************************************************
C  You  MUST enter basic variables,  on per card,  as follows:
C    <class>,<type>,<mean>,<std    dev>,<min>,<name>
C  where <class> is 'RE (for REsistance) or 'LO'  (for LOad),
C     <type>     is     'NO'rmal     or     'WE'ibull     or     'GU'mbel     or
C                    'FR'echet (you must include the single quotes)
C       <mean> is the mean value (REAL)
C       <std dev> is the standard deviation (REAL)
C       <min> is the lower limit (REAL), and
```

```
C       <name> is 'RESI'stance or 'LOAD' (used by LINFAIL)
C Note that either commas or blanks may be used as separators.
C You may optionally specify:
C   'AN'alysis  (default) or
C    'DE'sign,<K>,<COV>
C where <K> is the index of the design variable (INTEGER) and
C       <COV>, is its coefficient of variation (REAL)
C   'EP'silon,<epsilon>  (default is .0001)
C where <epsilon> is the relative error in MEAN(K) or BETA at
C output (REAL).  If the search loop "converges" after only two
C executions, you my need to supply a smaller value of epsilon in
C order to get things started.  This can happen if the basic variables X
C vary enormously in magnitude.  In general, it is good practice to
C scale the basic variables so that they are of roughly similar
C magnitude (differ no more than a factor of one million).
C   'BE'ta,<beta>  (default is 3.)
C where <beta> is the reliability index (REAL).  In ANalysis mode this
C is the initial guess, and in DEsign mode it is the target value.
C   'IN'dependent  (default) or
C    'CO'rrelated
C If COrrelated, then the upper triangular correlation matrix (above
C diagonal) must be input as follows, one per card:
C   'RM' ,<i>,<j>,<R(i,j)>
C where 'RM' identifies it as a correlation matrix value, and
C <i>,<j> are row an colums indices of the matrix element, and
C <R(i,j)> is the value of the matrix element
C Finally, you MUST supply the last card as follows:
C   'ST'
C in order to STart the program.
C ************************************************************
C *         END OF INPUT PREPARATION INSTRUCTIONS                    *
C ************************************************************
C
C INITIALIZE AND SET DEFAULTS
C
      N=O
      PROBTYP='A'
      CORR='N'
      BETA=3.
      EPS=.0001
C
c OPEN INPUT FILE 10 AND OUTPUT LOG FILE 11
C
100   PRINT 3000
3000  FORMAT(' dat filename=? ')
      READ  3001,FILENAME
3001  FORMAT(A9)
      LAST=MIN(LEN(FILENAME),INDEX(FILENAME,'    ')-1)
      IF(LAST.GT.8)THEN
         PRINT *,'dat filename must not exceed 8 characters'
         GO TO 100
      END IF
      OPEN(10,FILE=FILENAME(:LAST)//'.DAT')
C
C NEXT OPEN OUTPUT LOG FILE
C
      OPEN(11,FILE=FILENAME(:LAST)//'.LOG')
C
C INPUT
C
      N=1
C READ CONTROL  WORD
1     READ(10,*,ERR=101,END=109)WD
      IF(CKPR)PRINT *,WD,'  '
```

```
2       IF(WD.EQ.'ST')THEN
          WRITE(11,*)WD,'   '
          GO TO 6
        ELSE IF (WD.EQ.'LO')THEN
          B(N)--1
          GO TO 3
        ELSE IF  (WD.EQ.'RE')THEN
          B(N)=1.
          GO TO 3
        ELSE IF (WD.EQ.'AN')THEN
          WRITE(11,*)WD, ' '
          PROBTYP='AN'
          GO TO 1
        ESLE IF (WD.EQ.'DE')THEN
          PROBTYP='DE'
          BACKSPACE  10
          READ(10,*,ERR=102,END=109)WE,K,COV
          IF(CKPR)PRINT  *,WD,'  ',K,'  ',COV
          WRITE(11,203)WD,K,COV
203         FORMAT(A3,I3,F10.5)
          GO TO 1
        ELSE   IF(WD.EQ.'CO')THEN
          CORR='Y'
          WRITE(11,*)WD,'   '
          GO TO 5
        ELSE   IF(WD.EQ.'IN')THEN
          CORR='N'
          WRITE(11,*)WD,'   '
          GO TO 1
        ELSE   IF(WD.EQ.'EP')THEN
          BACKSPACE   10
          READ(10,*,ERR=103,END=109)WD,EPS
          IF(CKPR)PRINT  *,WD,'  ',EPS
          WRITE(11,204)WD,EPS
204       FORMAT(A3,F10.5)
          GO TO 1
        ELS    IF(WD.EQ.'BE')THEN
          BACKSPACE  10
          READ(10,*,ERR=104,END=109)WD,BETA
          IF(CKPR)PRINT *,WD,' ', BETA
          WRITE(11,205)WD,BETA
205       FORMAT(A3,F10.5)
          GO TO 1
        ELSE
          STOP  'in  INPT:  Unrecognizable  input'
        END  IF
3       BACKSPACE  10
        READ(10,*,ERR=105,END=109)WD,TP
        IF(CKPT)PRINT  *,WD,'  ',TP
        IF(TP.EQ.'NO')THEN
          DISTYP(N)=0
          GO TO 4
        ELSE   IF(TP.EQ.'LO')THEN
          DISTYP(N)=1
          GO TO 4
        ELSE   IF(TP.EQ.'WE')THEN
          DISTYP(N)=2
          GO TO 4
        ELSE   IF(TP.EQ.'GU')THEN
          DISTYP(N)=3
          GO TO 4
        ELSE   IF(TP.EQ.'FR')THEN
          DISTYP(N)=4
          GO  TO  4
```

```
      ELSE
         STOP 'in INPT:  unrecognizable  distribution  type'
      END IF
4     BACKSPACE  10
      READ(10  ,*,ERR=l06,END=109)WD,TP,MEAN(N),STDV(N),XO(N),XNAME(N)
      IF(CKPR)PRINT *,WD,' ',TP,'  ',MEAN(N),'   ',STDV(N),'   ',XO(N),
     1' ',XNAME(N)
      WRITE(1l,200)WD,TP,MEAN(N),STDV(N)XO(N),XNAME(N)
200   FORMAT(2A3,3F18.9,A5)
      N=N+1
      GO TO 1
5     READ(10,*,ERR=107,END=109)WD
      IF(CKPR)PRINT *,WD,' '
      IF(WD.EQ. 'RM')THEN
         BACKSPACE  10
         READ(10,*,ERR=108,END109)WD,I,J,R(I,J)
         IF(CKPR)PRINT *,WD,'  'I,'J  ',J,'  ',R(I,J)
         WRITE(11,201)WD,I,J,R(I,J)
201      FORMAT(A3,2I3,F10.5)
      ELSE
         GO TO 2
      END IF
      GO TO 5
6     N=N-1
      IF(CORR.EQ.'N')RETURN
      DO 7 I=1,N-1
      R(I,I)=1.
      DO 7 J=I+I,N
7     R(J,I)=R(I,J)
      R(N,N)=l.
      RETURN
C
101   STOP 'in INPT:  Can''t read control word'
102   STOP 'in INPT:  Can''t read K,COV  following ''DE'''
103   STOP 'in INPT:  Can''t read EPS following ''EP'''
104   STOP 'in INPT:  Can''t read BETA following ''BE'''
105   STOP 'in INPT:  Can''t read distribution type following ''RE'' or
     1''LO'''
106   STOP 'in INPT:  Can''t read data following distribution type'
107   STOP 'in INPT:  Can''t read RMatrix following ''CO'''
108   STOP 'in INPT:  Can''t read data following ''RM'''
109   STOP 'in INPT:  Premature  End-Of-File'
      END
```

## EL2SRCH

```
C  EL2SRCH
      SUBROUTINE    EL2SRCH(DISTYP,PROBTYP,BETA,K,COV,EPS,N,CORR,X,GAMA,
     1  PARAMS,XNAME,B,R,XO,MEAN,STDV,CKPR)
C EXTENDED LEVEL 2 SEARCH   ALGORITHM
C
C INPUT:
C  DISTYP  =  0-NOrmal,   1-LOgnormal,   2-Weibull,   3-GUmbel,   4-FRechet
C PROBTYP =  'AN'alysis  or  'DE'sign
C BETA    = Reliability index-Initial guess  (ANalysis)  or target  (DEsign)
C K       = Index of element of MEAN vector which is to be adjusted in DEsign
C COV     = Coefficient  of variation of K-th distribution in DEsign mode
C EPS     = relative error permitted in BETA or MEAN(K)  at exit
C N       = number of basic variables
C CORR    = 'Y'es or 'N'o (Are task variables correlated?)
C R       = Correlation matrix
C XO      = vector of lower limit values of basic variables
C MEAN    = vector of means of basic variables
C STDV    = vector of std dev's of basic variables
```

```
C  CKPR    = LOgical (Checkprinting on) used during check out
C  PARAMS  = array of values passed to and from subroutine FAIL; included
C            for users convenience
C  XNAME   = CHARACTER*4 array of basic variable names passed to subroutine
C            FAIL; included for users convenience
C
C   OUPUT:
C  X    = vector of basic variables, evaluated at design point
C  GAMA = unit vector pointing to design pint in X space G
C  B    = vector of partial derivatives of failure function G; calculated
C         by subroutine FAIL
C  BETA = Reliability index at desing point (ANalysis mode)
C  MEAN = vector of means of basic variables, j-th component adjusted to
C         meet target beta (DEsign mode)
C
       INTEGER DISTYP(10)
       CHARACTER*2  PROBTYP
       REALM BETA,COV,EPS,X(10) ,GAMA(10) ,PARAMS(10)
       CHARACTER*4  XNAME(10)
       CHARACTER*1  CORR
       REAL*8   B(10),R(10,10),XO(10),MEAN(10),STDV(10)
       LOGICAL  CKPR
C
       REAL*8   G,PDFZ,CC,CDFX,PDFX,Z,OM
C
C  G    = failure function; calculated by subroutine FAIL
C  PDFZ = stardard normal pdf; used in Rackwitz-Fiessler algorithm
C  CC   = magnitude associated with trial vector D
C  CDFX,PDFX,Z used in Rackwitz-Fiessler algorithm
C  OM   = lognormal parameter used in Rackwitz-Fiessler algorithm
C
       REAL*8   T(10),RFMU(10),RFSX(10),D(10)
C
C  T          = temp storage vector
C  RFMU,RFSX  = mean and std-dev vectors of Rackwitz-Fiessler Normals
C  D          = trial vector
C
       REAL*8  INVNORM
C
C  Name of double precision Function subroutine used in R-F algorithm
C
C   INITIALIZE
C
       DO 1 I=I,N
       RFMU(I)=(I)
1      RFSX(I)=STDV(I)
       IF(CKPR)PRINT   *,'B=',B
       IF(CKPR)WRITE(11,*)'B=',B
       ICOUNT=1
       FACTOR=.01
C
C  START SEARCH LOOP
C
2       CONTINUE
C
C  GET D
C
C                                    d := S b
       Do 9 I=1,N
9       D(I)=RFSX(I)*B(I)
       IF(CKPR)PRINT  *,'D=',D
       IF(CKPR)WRITE(11,*)'D=' ,D
C                                    |c|  := sqrt(d' R d)
       IF(CORR.EQ.'Y')  THEN
```

```
        DO 3 I=1,N
        T(I)=0
        DO 3 J=1,N
3        T(I)=T(I)+R(I,J)*D(J)
      ELSE
        DO 4 I=1,N
4       T(I)=D(I)
      END IF
      CC=0
      DO 5 I=1,N
5     CC=CC+D(I)*T(I)
      CC=DSQRT(CC)
      IF(CKPR)PRINT   *,'CC=',CC
      IF(CKPR)WRITE(11,*)'CC=',CC
C
C GET ALPHA (OR GAMA)
C
C                                     gama  :=-R  d/|c|
      DO 6 I=1,N
6      GAMA(I)=-T(I)/CC
      IF(CKPR)PRINT *,'ALPHA (or GAMMA) = ',GAMA
      IF(CKPR)WRITE(11,*)'ALPHA (or GAMMA) = ',GAMA
C
C GET X
C
C                                     X : mu + S gamma beta
      DO   7I=1,N
      X(I)=RFMU(I)+RFSX(I)*GAMA(I)*BETA
C IF ANY NON-NORMAL X(I) LIES OUTSIDE ITS PERMITTED RANGE,
C IT MUST BE PUT JUST ABOVE XO(I):
      IF(DISTYP(I).NE.0.AND.X(I).LE.XO(I))THEN
C          User my need to adjust following FACTOR up or down if
C          program gets stuck in an infinite loop of resetting X.
C          FACTOR should be small but not too small.
          X(I)=XO(I)+FACTOR*(MEAN(I)-XO(I))
          PRINT 200,I,X(I)
          WRITE(11,200)I,X(I)
200       FORMAT(' X(',I2,') reset to ',F10.5)
          ICOUNT=ICOUNT+1
          IF(ICOUNT.GT.10)THEN
             PRINT *,'X has been reset 10 times with reset factor of',FA
     1CTOR
             PRINT *,'This factor should be small, but not TOO small'
             PRINT *,Enter a new value (or Ctrl-C to quit):  '
             READ *,FACTOR
             ICOUNT=1
          END IF
      END IF
7     CONTINUE
      IF(CKPR)PRINT  *,'X=',X
      IF(CKPR)WRITE(11,*)'X=',X
C
C GET G AND B
C
C                                     evaluate g(X) and b (partials)
      CALL  FAIL(X,B,G,PARAMS,XNAME)
      IF(CKPR)PRINT  *,'B=',B
      IF(CKPR)WRITE(11,*)'B=',B
      IF(CKPR)PRINT  *,'G=',G
      IF(CKPR)WRITE(11,*)'G=',G
C
C GET BETA OR MEAN(K)
C
      IF(PROBTYP.EQ.'AN')THEN
```

```
C                                     beta := beta + g/|c|
        BETA=BETA+G/CC
        BETA=DABS(BETA)
        PRINT  201,BETA
        WRITE(11,201)BETA
201     FORMAT('  Beta=',F10.5)
        IF(DABS(G/CC/BETA).LT.EPS)GO  TO  11
      ELSE
C                                     mu(j)    :(j)    -g/b(j)
        MEAN(K)=MEAN(K)-G/B(K)
        STDV(K)=COV*MEAN(K)
        PRINT  202,K,MEAN(K),K,STDV(K)
        WRITE(11,202)K,MEAN(K),K,STDV(K)
202     FORMAT('    Mean('K2,')=',F10.5,'    Stdv(',I2,')=',F10.5)
        FI(DABS(G/B(K)/MEAN(K)).LT.EPS)GO  TO  11
      END  IF
C
C  RACKWITZ-FIESSLER  ALGORITHM
C
C                                     update  S  via  Rackwitz-Fiessler
      DO  10  I=1,N
      RFSX(I)=STDV(I)
      RFMU(I)=MEAN(I)
C       IF  NORMAL,  RF  NOT  NEEDED
      IF(DISTYP(I).EQ.O.AND.CKPR)PRINT*,'Normal;  RF  not  needed'
      IF(DISTYP(I).EQ.O.AND.CKPR)WRITE(11,*)'Normal;  RF  not  needed'
      IF(DISTYP(I).EQ.O)GO  TO  10
C       IF  LOGNORMAL,  DO  RF  IN  CLOSED  FORM
      IF(DISTYP(I).EQ.1)THEN
         IF(CKPR)PRINT*,'Lognormal;  RF  in  closed  form'
         IF(CKPR)WRITE(11,*)'Lognormal;  RF  in  closed  form'
         OM=DSQRT(1.+(STDV(I)/(MEAN(I)-XO(I)))**2)
         RFSX(I)=(X(I)-XO(I))*DSQRT(2.*DLOG(OM))
         RFMU(I)=X(I)+(X(I)-XO(I))*DLOG((MEAN(I)-XO(I))/(X(I)-XO(I))/OM)
         GO  TO  10
      EN   IF
      IF(CKPR)PRINT*,'Doing  RF  algorithm'
      IF(CKPR)WIRTE(11,*)'Doing  RF  algorithm'
      II=I
      CALL   DIST(X(I),II,DISTYP(I),CDFX,PDFX,XO,MEAN,STDV,CKPR)
      IF(CDFX.EQ.1..OR.CDFX.EQ.O.)THEN
         PRINT *,'R-F  impossible  this  loop  for  variable',I
         WRITE(11,*)'R-F  impossible  this  loop  for  variable',I
         GO  TO  10
      ELSE
         Z=INVNORM(CDFX)
         PDFZ=DEXP(-Z**2/2.DO)/DSQRT(6.283185307DO)
         RFSX(I)=PDFZ/PDFX
         RFMU(I)=X(I)-RFSX(I)*Z
         FI(CKPR)PRINT   *,'Z=',Z
         IF(CKPR)WRITE(11,*)'Z=',Z
         IF(CKPR)PRINT   *,'PDFZ=',PDFZ
         IF(CKPR)WRITE(11,*)'DPFZ=',PDFZ
         IF(CKPR)PRINT  *,'RFSX=',RFSX(I)
         IF(CKPR)WRITE(11,*)'RFSX=',RFSX(I)
         IF(CKPR)PRINT  *,'RFMU=',RFMU(I)
         IF(CKPR)WRITE(11,*)'RFMU=',RFMU(I)
      END  IF
10    CONTINUE
C
C END  OF  SEARCH  LOOP
C
      GO  TO  2
11     RETURN
```

```
      END
```

## OUTPT

```
C  OUTPT
      SUBROUTINE    OUPUT(N,PROBTYP,BETA,K,X,GAMA,XNAME,MEAN))
C   OUPUT  SUBROUTINE
C
C INPUT:
C N       = number of basic variables
C PROBTYP =  'AN'alysis  or  'DE'sign
C BETA    =  Reliability  index
C K       =  Index of element  of MEAN vector which is to be adjusted in DEsign
C X       =  vector of basic variables
C GAMA    =  auxiliary vector analogous  to alpha
C XNAME   =  CHAR*4 ARRAY OF BASIC VARIABLE NAMES
C MEAN    =  vector of means of basic variables
C
C OUTPUT:
C All output  is written to file <filename>.LOG
C
C ************************************************************************
C * This routine prints the final values of the arrays X ad GAMA,  and  *
C * the final value of either BETA or MEAN(K)                            *
C ************************************************************************
C
      CHARACTER*2  PROBTYP
      REAL*8  BETA,X(10),GAMA(10)
      CHARACTER*4  XNAME(10)
      REAL*8  MEAN(10)
C
       PRINT  *,'Design  point:'
       WRITE(11,*)'Design  point:'
       DO  12  I=1,N
       PRINT   100,XNAME(9),I,X(I),I,GAMA(I)
100    FORMAT('   ',A4,'   X(',I2,')=',E16.10,'   GAMA(',I2,')=',616.10)
       WRITE(11,100)XNAME(I),I,X(I),I,GAMA(I)
12     CONTINUE
       IF(PROBTYP.EQ.'AN')THEN
          PRINT  101,BETA
          WRITE(11,101)BETA
101    FORMAT('  Beta=',F7.4)
       END  IF
       IF(PROBTYP.EQ.'DE')THEN
          PRINT  102,  MEAN(K)
          WRITE(11,102)MEAN(K)
102    FORMAT('  Design  Mean='E12.6)
       END  IF
       RETURN
       END
```

## LINFAIL

```
C  LINFAIL
      SUBROUTINE  FAIL(X,B,G,PARAMS,XNAME)
C LINEAR FAILURE SURFACE.  Here each XNAME(1)  must be LOAD or RESI
C
C INPUT:
C X      = vector  of basic variables
c PARAMS = array of values passed to and from subroutine EL2SECH;
C          here PARAMS(1)  is used to pass N  (nr basic variables)
C XNAME  = CHARACTER*4 array of basic variable names;  used here to
C          distinguish  between  loads  and resistances
C
C  OUPUT:
```

```
C B = vector of partial derivatives of failure function G
C G = linear failure function
C PARAMS
C
      REAL*8  X(10),B(10),G
      REAL*8  PARAMS(10)
      CHARACTER*4  XNAME(10)
C
      N=PARAMS(1)
      G=0
      DO1  I=1,N
      IF(XNAME(I).EQ.'LOAD')THEN
         B(I)=-1.
      ELSE IF(XNAME(I).EQ.'RESI')THEN
         B(I)=1.
      ELSE
         STOP  ' Variable names in LINFAIL must be "LOAD" or ''RESI'''
      END  IF
1      G=G+B(I)*X(I)
      END
```

## DIST

```
C  DIST
      SUBROUTINE   DIST(X,I,J,CDF,PDF,XO,MEAN,STDV,CKPR)
C Routine to evaluate CDF ad PDF of Weibull, Gumbel,  ad Frechet
C   distributions
C
C INPUT:
C X    = value of basic variable
C I    = index of X (used with arrays XO,MEAN,STDV)
C J    =index which indicates distritution type
C XO   = array of lower limits of distributions
C MEAN = array of means of distributions
C STDV = array of stardard deviations of distributions
C CKPR =  Logical  (Checkprinting on)  used during check out
C
C OUTPUT:
C CDF  = cumulative distribution function value
C PDF  = probability density function value
C
      REAL*8   X,CDF,PDF,ET,AG,U,Z,T,XX
      REAL*8    A,DEL,PH,H,GAMA
      REAL*8  XO(10),MEAN(10),STDV(10)
      LOGICAL  CKPR
      IF(J.LT.2.CR.J.GT.4)THEN
         STOP  'Wrong distritution type sent to subroutine DIST'
      ELSE
          GO TO (1,2,3,4),J
      END  IF
1     STOP  'in DIST:  Weird error,  impossible to be here'
C
C  WEIBULL
C
C AG = shape parameter;  ET = scale parmeter;  XO(I)  = minimum value
C
2       A=(STDV(I)/(MEAN(I)-XO(I)))**2+1.
      IF(CKPR)PRINT  *,'Doing  Weibull'
      IF(CKPR)WRITE(11,*)'Doing   Weibull'
      XX=X-XO(I)
      IF(A.GT.2.DO)THEN
         STOP  'in DIST:  Weibull Std dev must not exceed Mean  -  Min'
      END  IF
      AG(A-1.)**(-1./1.835)-.5
```

```
C
C Bifurcation root search:
C
         DEL=1.
         PH=0
11       H=GAMMA(1.D0+2.D0/AG)/GAMMA(1.D0+1.D0/AG)**2-A
         IF(H*PH.GE.0)THEN
            DEL=DEL/2.
            GO TO 12
         ELSE
            DEL=-DEL/2.
         END IF
12       AG=AG+DEL
         IF(DABS(DEL/AG).LT..00005)  GO TO 13
         PH=H
         GO TO 11
13     ET=(MEAN(I)-XO(I))/GAMMA(1.+1./AG)
       TMP=DEXP(-(XX/ET)**AG)
       CDF=1.-TMP
       PDF=AG/ET*(XX/ET)**(AG-1)*TMP
       IF(CKPR)PRINT  *,'Weibull parameters:  Scale=',ET,'  Shape=',AG
       IF(CKPR)WRITE(11,*)'Weibull  parameters:  Scale=',ET,'  Shape=',AG
       RETURN
C
C  GUMBEL
C
3      AG=3.141592654/DSQRT(6D0)/STDV(I)
       IF(CKPR)PRINT  *,'Doing  Gumbel'
       IF(CKPR)WRITE(11,*)'Doing  Gumbel'
       U=MEAN(I)-.57722/AG
       XX=X
       T=TEXP(-AG*(XX-U))
       CDF=DEXP(-T)
       PDF=AG*T*CDF
       RETURN
C
C  FRECHET
C
4      AG=2.33/(STDV(I)/MEAN(I))**.6770001
       IF(CKPR)PRINT  *,'Doing  Frechet
       IF(CKPR)WRITE(11,*)'Doing  Frechet'
       U=MEAN(I)/GAMMA(1.-1./AG)
       XX=X
       CDF=DEXP(-(U/XX)**AG)
       PDF=CDF*AG/U*(U/XX)**(AG+1.)
       RETURN
       END
```

## INVNORM

```
C  INVNORM
       FUNCTION  INVNORM(F)
C Routine to evaluate Inverse Normal CDF
C
C Uses Hastings polynomial 26.2.23 of "Handbook of Mathematical
C Functions", 1972, M.Abramowitz and I.A.Stegun &., AMS.55, Natl.
C Bur. Stds, U.S. Dept. of Commerce
C
C Absolute error < 0.0004
C
       REAM*8  F,T,Y,INVNORM
       IF(F.LT..5)THEN
          T=DSQRT(DLOG(1/F**2))
       ELSE
```

```
          T=DSQRT(DLOG(1/(1-F)**2))
        END  IF
        Y=.001308*T+.189269)*T+1.432788)*T+1
        Y=T-((.010328*T+.802853)*T+2.515517)/Y
        IF(F.LT..5)THEN
          INVNORM=-Y
        ELSE
          INVNORM=Y
        END  IF
        RETURN
        END
```

## GAMMA

```
C  GAMMA
       FUNCTION  GAMMA(X)
C Computes  GAMMA(X),  WHERE  GAMMA(X+1)!'  GAMMA(X)=(X-1)!
C
C  Uses  Hastings  polynomial  6.1.36  of  "Handbook  of  Mathematical
C  Functions",  1972,  M.Abramowitz  and  I.A.Stegun  eds.,  AMS.55,  Natl.
C  Bur.  Stds,  U.S.  Dept.  of  commerce
C
       REAL*8  X,GAMMA,GMH,Y
C
C  GMH(Y)  is  a  Hastings  polynomial  for  GAMMA(1.+Y)
C
       GMH(Y)=1.+Y*(-0577191652D0+*(0.9878205891D0+Y*(-0897056937D0
      1   +Y*(0.918206857D0+Y*(0.765704078D0+Y*(0.482199394D0
      2   +Y*(-0.193527818D0+Y*(0.035868343D0)))))))
       IF(X.LE.0)THEN
          GAMMA=0
          RETURN
       ELSE   IF(X.GE.1.DO)THEN
          GO TO 1
       ELSE
          GAMMA=GMH(X)/X
          RETURN
       END  IF
1      IF(X.GT.2.DO)GO  TO  2
       GAMMA=GMH(X-1.)
       RETURN
2      N=IDIDNT(X-1.D0)
       FN=N
       GAMMA=GMH(X-FN-1.)
       DO  3  I+1,N
       FI=I
3      GAMMA=GAMMA*(X-FN-1.+FI)
       END
```

## YLINEN

```
C  YLINEN
       SUBROUTINE  FAIL(X,B,G,PARAMS,XNAME)
C  Ylinen's  column failure  criterion
C
C INPUT:
C X = vector of basic variables
C  PARAMS see below
C  XNAME See below
C
C OUTPUT:
C B = vector of partial derivatives  of failure function G
C  G = Ylinen failure function
C  PARAMS see below
C  XNAME see below
```

```
C
C PARAMS = array of values passed to and from subroutine EL2SRCH:
C   here PARAMS(1)  is used to pass N (nr basic variables)
C   PARAMS(2)  passes the cross sectional area
C   PARAMS(3)  passes the L/r slenderness ratio
C   PARAMS(4)  passes the parmeter  c
C   and PARAMS(5) passes 0. (first call) or 1.  (Subsequent call).
C XNAME = CHARACTER*4 array of basic variable names;  used here to
C   distinguish strength  'FC', modulus  'E',  and load  'P'.
C
      REAL*8   X(10),B(10),G,A,LR,U,V,R
      REAL*8  PARAMS(10),FC,E,P
      CHARACTER*4  XNAME(10)
      IF(PARMAS(5).EQ.O)THEN
         PRAMS(5)=1.
         READ(10,*)PARAMS(2)
         READ(10,*)PARAMS(3)
         READ(10,*)PARAMS(4)
         WRITE(11,200)PARAMS(2)
200      FORMAT('  Area=',F10.5)
         PRINT  200,PARAMS(2)
         WRITE(11,20l)PARAMS(3)
201      FORMAT('  L/r=' ,F10.5)
         PRINT  201,PARAMS(3)
         WRITE(11,202)PARAMS(4)
202      FORMAT('  Ylinen  c=',F10.5)
         PRINT  202,PARAMS(4)
      ELSE  IF(PARAMS(5).EQ.1.)THEN
         CONTINUE
      ELSE
         STOP  'Illegal  call to YLINEN'
      END  IF
      A=PARAMS(2)
      LR=PARAMS(3)
      C=PARAMS(4)
      BB=3.141592654**2*A/LR**2
      N=PARAMS(1)
      DO  1  I=1,N
      IF(XNAME(1).EQ.'FC')THEN
         FC=X(1)
         IFC=I
      ELSE  IF(XNAME(I).Eq.'E')THEN
         E=X(I)
         IE=I
      ELSE  IF(XNAME(I).EQ.'P')THEN
         P=X(I)
         B(I)=-1.
      ELSE
          STOP 'Wrong variable names  in YLINEN'
      END  IF
 1     CONTINUE
      U=(A*FC+BB*E)/2./C
      V=A*FC*BB+E/C
      R=DSQRT(U**2-V)
      G=U-R-P
      B(IFC)=(1.  -U/R+BB*E/R)*A/2./C
      B(IE)=(1. -U/R+A*FC/R)*BB/2./C
      END
```

# Appendix C. Error Messages

Error messages are shown in typewriter font followed by an explanation where necessary.

## INPT

**`dat filename must not exceed 8 characters`**

When entering the input/output filename, you do *not* include the extension **`.dat,`** and DOS does not accept filenames longer than 8 characters.

**`in INPT: Unrecognizable input`**

**`in INPT: Unrecognizable distribution type`**

**`in INPT: Cannot read control word`**

**`in INPT: Cannot read K,COV following 'DE'`**

**`in INPT: Cannot read EPS following 'EP'`**

**`in INPT: Cannot read BETA following 'BE'`**

**`in INPT: Cannot read distribution type following 'RE' or 'LO'`**

**`in INPT: Cannot read data following distribution type`**

**`in INPT: Cannot read RMatrix following 'CO'`**

**`in INPT: Cannot read data following 'RM'`**

**`in INPT: Premature End-Of-File`**

## EL2SRCH

**`X(i) reset to <number>`**

The search loop has produced a trial value for X(i) that is below its prescribed lower limit. Therefore the value of X(i) has been reset to a value slightly larger than the lower limit. If the program gets stuck in an infinite loop of resetting X(i), the user may need to adjust FACTOR up or down (see listing for meaning of FACTOR). After 10 successive resets, the program prompts for a new value of FACTOR.

**`R-F impossible this loop for variable <i>`**

Subroutine DIST has returned a value of 0.0 or 1.0 for the CDF of **non-normal variable `<i>`.** In such cases, it is impossible to perform the Rackwitz–Fiessler algorithm on that variable, and the values from the previous loop are retained. This is harmless, so long as the loop converges and the final few loops do not contain this message.

## DIST

**`Wrong distribution type sent to subroutine DIST`**

**`in DIST: Weibull Std dev must not exceed Mean-Min`**

## YLINEN

**`Illegal call to YLINEN`**

PARAMS(5) must be 0 or 1 when calling YLINEN.

**`Wrong variable names in YLINEN`**

Meaningful variable names are **`'FC'`**, **`'P'`**, and **`'E'`**.

## LINFAIL

**`Variable names in LINFAIL must be LOAD or RESI`**