Applied and
Computational
Mathematics
Division

NISTIR 92–4834

Center for Computing and Applied Mathematics

User's Reference Guide for ODRPACK Version 2.01 Software for Weighted Orthogonal Distance Regression

Paul T. Boggs, Richard H. Byrd,
Janet E. Rogers and Robert B. Schnabel

June 1992

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, MD 20899

User's Reference Guide for ODRPACK Version 2.01 Software for Weighted Orthogonal Distance Regression [1]

Paul T. Boggs [2] Richard H. Byrd [3] Janet E. Rogers [4]
Robert B. Schnabel [5]

[1] Contribution of the National Institute of Standards and Technology (formerly the National Bureau of Standards), and not subject to copyright in the United States.

[2] Applied and Computational Mathematics Division, National Institute of Standards and Technology, Gaithersburg, MD 20899

[3] Department of Computer Science, University of Colorado, Boulder, CO 80309 (Research supported in part under National Science Foundation Grants DCR-8403483 and CCR-870143, and Army Research Office Contracts DAAG 29-84-K-0140 and DAAL 03-88-0086)

[4] Applied and Computational Mathematics Division, National Institute of Standards and Technology, Boulder, CO 80303-3328

[5] Department of Computer Science, University of Colorado, Boulder, CO 80309 and Applied and Computational Mathematics Division, National Institute of Standards and Technology, Boulder, CO 80303-3328 (Research supported in part under National Science Foundation Grants DCR-8403483 and CCR-870143, and Army Research Office Contracts DAAG 29-84-K-0140 and DAAL 03-88-0086)

Abstract

ODRPACK is a software package for *weighted orthogonal distance regression*, i.e., for finding the parameters that minimize the sum of the squared weighted orthogonal distances from a set of observations to the curve or surface determined by the parameters. It can also be used to solve the nonlinear ordinary least squares problem. The procedure has application to curve and surface fitting, and to measurement error models in statistics. ODRPACK can handle both explicit and implicit models, and will easily accommodate complex and other types of multiresponse data. The algorithm implemented is an efficient and stable trust region Levenberg-Marquardt procedure that exploits the structure of the problem so that the computational cost per iteration is equal to that for the same type of algorithm applied to the nonlinear ordinary least squares problem. The package allows a general weighting scheme, provides for finite difference derivatives, and contains extensive error checking and report generating facilities.

Keywords: orthogonal distance regression; measurement error models; errors in variables; nonlinear least squares.

Categories: G2E, I1B1

Revision History

2.01 (06-19-92) ODRPACK 2.01 corrects two minor errors. The first affected the derivative checking procedure, and the second affected the values specified by `NDIGIT`. The solutions found by ODRPACK 2.00 were not affected in any way by the first of these errors, and were not seriously affected by the second.

Version 2.01 also encorporates several other minor modifications that further improve the code. The most significant of these modifications changes how the scale values are used within the solution procedure. This change should increase the robustness of the procedure for poorly scaled problems, and will result in slightly different computed results for all problems. Another of these changes is to the report generated by the derivative checking procedure. This report now includes the user supplied derivative value, as well as the relative difference between the user supplied value and the finite difference value it was checked against. This should help users assess the validity of the checking procedure results.

The "ODRPACK 2.01 User's Reference Guide" is essentially the same as the version 2.00 Guide (NISTIR 89-4103, Revised). The major differences are that

- the minimum length of array `WORK` has been increased by $(p+m)q$ locations, and
- the sample programs and output shown in Chapter 2, and the discussion of the derivative checking results in Chapter 5 have been modified to reflect the above mentioned changes.

2.00 (03-04-92) ODRPACK 2.00 adds several new features to those available in version 1.8 and earlier versions.

- With version 2.00, ODRPACK can now easily accommodate complex and other types of multiresponse data, i.e., data where each observation is multidimensional.

- It can handle implicit as well as explicit models.
- The weighting scheme has been enhanced to allow for instances when the components of a multidimensional observation are correlated.
- A facility for computing central finite difference derivatives has been added.
- The amount of work space required for the computations has been reduced for most problems.

Because of these new features, the argument lists for the ODRPACK 2.00 user callable subroutines, including the user supplied subroutine `FCN` *that now comprises the functionality of subroutines* `FUN` *and* `JAC`*, are not the same as for earlier versions.* The interpretations of some arguments have changed, and new arguments have been added. In addition, the arguments have been reordered, with arguments having similar purpose now grouped together. Those who have been using an earlier release of ODRPACK should be especially careful to examine the new calling sequences, and the definitions for subroutine arguments `FCN`, `JOB`, `WE` and `WD`.

1.80 (12-18-90) ODRPACK 1.80, a test release with only limited distribution, differs from prior releases in several respects.

- The number of function evaluations required to find the solution has been reduced, and the "restart" facility has been modified to better accommodate cases where the user supplied subroutines `FUN` and/or `JAC` are very time consuming.
- The printed reports have been redesigned to identify parameters that induce rank deficiency, and to include a 95% confidence interval for the estimated parameters.
- Several enhancements have been added to the covariance matrix computations, including the option of constructing the covariance matrix without incurring any additional derivative evaluations.
- The finite difference approximation to the Jacobian matrix has been improved.

1.71 (07-27-89) ODRPACK 1.71 corrects an error in the code that performs the computation of finite difference derivatives with respect to the errors Δ when $m \geq 2$ and the default value of `IFIXX` is invoked. (The default value of `IFIXX` is invoked when `IFIXX(1,1)` is set to a negative value or when ODRPACK routines `DODR` or `SODR` are called.) This error could result in incorrect "fixing" of the explanatory variables, which would affect the final solution. Such "fixing" could be detected by observing the presence of estimated values for Δ that are identically zero. The error could go undetected by the user, however, if the values of Δ were not examined after the fit.

Preface

ODRPACK is a portable collection of ANSI Fortran 77 subroutines for fitting a model to data. It is designed primarily for instances when all of the variables have significant errors (see, e.g., [Fuller, 1987] and [Boggs and Rogers, 1990]), implementing a highly efficient algorithm for solving the weighted orthogonal distance regression problem [Boggs *et al.*, 1987 and 1989], i.e., for minimizing the sum of the squares of the weighted orthogonal distances between each data point and the curve described by the model equation. It can also be used to solve the ordinary least squares problem where all of the errors are attributed to the observations of the dependent variable.

ODRPACK is designed to handle many levels of user sophistication and problem difficulty.

- It is easy to use, providing two levels of user control of the computations, extensive error handling facilities, optional printed reports, and no size restrictions other than effective machine size.
- It can handle implicit as well as explicit models, and can accommodate complex and other types of multiresponse data, i.e., data where each observation is multidimensional.
- The necessary derivatives (Jacobian matrices) are approximated numerically if they are not supplied by the user, and the correctness of user supplied derivatives can be verified by the derivative checking procedure provided.
- Both weighted and unweighted analysis can be performed.
- Subsets of the unknowns can be treated as constants with their values held fixed at their input values, allowing the user to examine the results obtained by estimating subsets of the unknowns of a general model without rewriting the model subroutine.
- The covariance matrix and the standard errors of the model parameter estimators are optionally provided.
- The ODRPACK scaling algorithm automatically compensates for poorly scaled problems, i.e., problems with model parameters, and/or unknown errors in the explanatory variables that vary widely in magnitude.

- The trust region Levenberg-Marquardt algorithm implemented by ODRPACK [Boggs *et al.*, 1987 and 1989] has a computational effort per step that is of the same order as that required for ordinary least squares, even though the number of unknowns estimated in the orthogonal distance regression problem is the number of unknown model parameters plus the number of explanatory variables, while the number of unknowns estimated in the ordinary least squares problem is simply the number of unknown model parameters.
- The code is portable and is easily used with other Fortran subroutine libraries.

Computer facilities for the ODRPACK project have been provided by the National Institute of Standards and Technology (NIST), Applied and Computational Mathematics Division, and we gratefully acknowledge their support. Machine dependent constants are supplied using subroutines based on the Bell Laboratories "Framework for a Portable Library" [Fox *et al.*, 1978a]. We have also used subroutines from LINPACK [Dongarra *et al.*, 1979], and from the "Basic Linear Algebra Subprograms for Fortran Usage" [Lawson *et al.*, 1979]. The code that computes the t-values used in constructing the confidence intervals is from DATAPAC [Filliben, 1977], and the code that checks user supplied derivatives was adapted from STARPAC [Donaldson and Tryon, 1986] using algorithms described in [Schnabel, 1982].

We appreciate the comments and suggestions we have received regarding earlier versions of ODRPACK and its documentation. In particular, many of the improvements in version 2.01 are the result of the authors' collaboration with Paul D. Domich, NIST Applied and Computational Mathematics Division, and with David A. Vorp, University of Pittsburgh Department of Surgery. We wish to especially thank Paul D. Domich for many productive discussions. While it is not possible to list everyone else who has contributed to ODRPACK, we would like to thank Vincent D. Arp (formerly of the NIST Chemical Engineering Science Division), Hariharan K. Iyer (Colorado State University), Susannah B. Schiller (NIST Statistical Engineering Division), Bernard Thiesse (Ecole Nationale Supérieure d'Electrtechnique, d'Electronique d'Informatique et d'Hydraulique, Toulouse, France), and Eric J. Vanzura (NIST Electromagnetic Fields Division), for providing us with data sets that were invaluable in testing the new release.

Paul T. Boggs
Richard H. Byrd
Janet E. Rogers
Robert B. Schnabel

June 1992

Contents

1. Getting Started

ODRPACK is a portable collection of ANSI Fortran 77 subroutines for fitting a model to data. It is designed primarily for instances when all of the variables have significant errors (see, e.g., [Fuller, 1987] and [Boggs and Rogers, 1990]), implementing a highly efficient algorithm for solving the weighted orthogonal distance regression problem [Boggs *et al.*, 1987 and 1989]. It can also be used to solve the ordinary least squares problem, however, where all of the errors are attributed to the observations of only one of the variables.

We suggest that first time users of ODRPACK begin by at least skimming this chapter, and then proceed to the sample programs given in §2.C, modifying them as necessary to solve their particular problem. The sample programs are distributed in machine readable form with the ODRPACK release, and thus should be available for use as templates without requiring that the user enter the code. The subroutine arguments used in these examples are defined in detail in §2.B.ii.

Chapter 1 presents information that is especially important to first time users of ODRPACK. Users are directed to §1.A for a brief description of the orthogonal distance regression problem. This section introduces notation and provides background material for understanding the remainder of the documentation. The algorithm itself is described briefly in §1.B, and in greater detail in [Boggs *et al.*, 1987 and 1989]. Options available for specifying the problem are briefly discussed in §1.C, and ODRPACK's report generation facility is discussed in §1.D. The need for good starting values for the unknown parameters of the model is explained in §1.E, and the use of weights in §1.F. Finally, §1.G describes two features of ODRPACK that simplify the user interface with the package.

Chapter 2 describes the use of ODRPACK in detail. Most experienced users of ODRPACK will only need the information in this chapter. The declaration and call statements for each user callable ODRPACK subroutine are given in §2.A, the subroutine arguments are defined in §2.B, and sample programs are shown in §2.C.

The last three chapters provide auxiliary information. Chapter 3 is generally only needed by users with very time consuming functions; it explains how ODRPACK's features can be exploited in order to minimize cost, and to reduce the possibility of losing results because limits imposed by their computer system were reached. Chapter 4 describes the details of certain ODRPACK computations, and Chapter 5 describes how the user can extract computed results from the work vectors.

1.A. Notation and Problem Definition

The method of least squares is often used to find the parameters $\beta \in \Re^p$ of a mathematical model that defines a relationship between variables that are subject to errors. When errors occur in more than one of the variables, then the relationship between the variables is frequently referred to as a *measurement error model* or an *errors in variables problem.* The computational problem associated with finding the maximum likelihood estimators of the parameters of such models is known as *orthogonal distance regression.* These models are discussed in [Fuller, 1987], and [Seber and Wild, 1989]. The orthogonal distance regression procedure implemented in ODRPACK is reviewed in [Boggs and Rogers, 1990]. That publication also summarizes the results of several simulation studies that compare orthogonal distance regression results to those obtained using ordinary least squares, where the errors are assumed to occur in only one of the variables. (See also [Boggs *et al.*, 1988].)

The model f that defines the relationship between the variables can be either linear or nonlinear in its parameters β. Sometimes one of the observed variables is distinguished as being a *response* that is *dependent* upon the remaining variables, which are commonly called the *explanatory*, *regressor* or *independent* variables. In these cases, the explanatory variables are often used to predict the behavior of the response variable. In other cases, however, there is no such distinction among the variables.

We say that there is an **explicit** relationship f between the variables (x, y) if

$$y \approx f(x;\beta) \ , \tag{1.1}$$

where y denotes the response variable, x denotes the explanatory variables, and y is assumed to be only approximately equal to $f(x;\beta)$ because of the measurement errors in y and possibly x. When both x and y are observed with error, then the parameters β can be found by *orthogonal distance regression.* If only y is subject to measurement error, and x is observed without error, then the parameters of such an explicit model can be obtained using *ordinary least squares* procedures.

Example: The model shown in §2.C.i,

$$y_i \quad \approx \quad f_i(x_i;\beta) \quad \equiv \quad \beta_1 + \beta_2[e^{\beta_3 x_i} - 1]^2$$

for $i = 1, \ldots, n$, which is example 3.2.2 on pages 230-238 of [Fuller, 1987], is an explicit orthogonal distance regression model. In this case, x is the percent saturation of nitrogen gas in a brine solution forced into the pores of sandstone, and y is the observed compressional wave velocity of ultrasonic signals propagated through the sandstone. Here, as in most explicit models, the explanatory variable x is used primarily to predict the behavior of the response variable y. □

If the relationship between the variables is expressed **implicitly**, then

$$f(x;\beta) \approx 0 ,$$

where here we denote *all* of the variables by x. Although one of these variables may still be dependent on the others, when the model is implicit we do not assume that such a dependent variable can be expressed as an explicit function of the other explanatory variables as in (1.1). Thus, in the implicit case the distinction between response and regressor variables is unnecessary. When the relationship is implicit, then the parameters β cannot be estimated by most ordinary least squares procedures, but can be estimated by orthogonal distance regression procedures such as ODRPACK.

Example: One of the simplist examples of an implicit model is that of fitting a circle or ellipse in a plane, as is done in example 3.2.4 on page 244 of [Fuller, 1987]. In this example, shown in §2.C.ii, the data are observations digitized from the x-ray image of a hip prosthesis, where the variables $x_i = (v_i, h_i)$, $i = 1, \ldots, n$, are the vertical and horizontal distances from the origin, respectively, and the implicit model is that of the ellipse

$$f_i(x_i;\beta) \equiv \beta_3(v_i - \beta_1)^2 + 2\beta_4(v_i - \beta_1)(h_i - \beta_2) + \beta_5(h_i - \beta_2)^2 \approx 0$$

for $i = 1, \ldots, n$. □

Finally, sometimes the observations are **multiresponse** and thus must satisfy more than one relationship f. This most commonly arises when the observations are complex variables, although it can occur in other cases as well.

Example: The problem shown in §2.C.iii is an example of multiresponse data that originates because the underlying data are complex. The problem is described in Chapter 4, and on pages 280-281, of [Bates and Watts, 1988]. In this case, the response variable is the pair of values representing the real and imaginary parts of complex-valued impedance measurements, z_i, $i = 1, \ldots, n$, of a polymer, and the explanatory variable, x_i, $i = 1, \ldots, n$, is the (real-valued) frequency. The function is explicit, representing the dielectric constant by a general model proposed in [Havriliak and Negami, 1967],

$$z_i \approx \beta_2 + \frac{\beta_1 - \beta_2}{\left(1 + (\jmath 2\pi x_i e^{-\beta_3})^{\beta_4}\right)^{\beta_5}}$$

for $i = 1, \ldots, n$, where $\jmath = \sqrt{-1}$. For ODRPACK, this must be encoded as a two-term multiresponse problem with $y_i \in \Re^2$ representing the pair of values $[\Re(z_i), \Im(z_i)]$, $i = 1, \ldots, n$. □

The explicit, implicit and multiresponse problem types are described in greater detail in the following sections. They are each easily solved using ODRPACK. The solution procedures required for single and multiresponse data are essentially the same, although, as noted below, ODRPACK does need to know that the data are multiresponse rather than single response in order to find the correct solution. The solution procedures required for explicit and implicit functions, however, are different, with the implicit relationships generally costing more to solve. (See §1.B.) Thus, it is important that users be aware of the differences between explicit and implicit models, and between single and multiresponse data, and invoke ODRPACK appropriately for their particular model and data type.

1.A.i. Explicit Orthogonal Distance Regression

We define the explicit orthogonal distance regression problem as follows. Let (x_i, y_i), $i = 1, \ldots, n$, be an observed set of data, where, for simplicity, we assume that $x_i \in \Re^1$ and $y_i \in \Re^1$. (For a discussion of the orthogonal distance regression problem for higher dimensional data, see §1.A.iii.) Suppose that the values of y_i are a possibly nonlinear function of x_i and a set of unknown parameters $\beta \in \Re^p$, but that both the x_i and the y_i contain actual but unknown errors $\delta_i^* \in \Re^1$ and $\epsilon_i^* \in \Re^1$, respectively, where here, and in the remainder of this document, we use a superscript "$*$" to denote such actual but unknown quantities. Then the observed value of y_i satisfies

$$y_i = f_i(x_i + \delta_i^*; \beta^*) - \epsilon_i^* \qquad i = 1, \ldots, n,$$

for some actual but again unknown value β^*.

The explicit orthogonal distance regression problem is to approximate β^* by finding the β for which the sum of the squares of the n orthogonal distances from the curve $f(x; \beta)$ to the n data points is minimized. This is accomplished by the minimization problem

$$\min_{\beta,\delta,\epsilon} \sum_{i=1}^{n} (\epsilon_i^2 + \delta_i^2) \tag{1.2}$$

subject to the constraints

$$y_i = f_i(x_i + \delta_i; \beta) - \epsilon_i \qquad i = 1, \ldots, n. \tag{1.3}$$

Since the constraints (1.3) are linear in ϵ_i, we can eliminate them and thus the ϵ_i from

(1.2), thereby obtaining

$$\min_{\beta,\delta} \sum_{i=1}^{n} \left([f_i(x_i + \delta_i; \beta) - y_i]^2 + \delta_i^2 \right) \ . \tag{1.4}$$

We then generalize (1.4) to the weighted orthogonal distance regression (ODR) problem

$$\min_{\beta,\delta} \sum_{i=1}^{n} \left(w_{\epsilon_i} \left[f_i(x_i + \delta_i; \beta) - y_i \right]^2 + w_{\delta_i} \delta_i^2 \right) \tag{1.5}$$

by introducing the weights $w_{\epsilon_i} \in \Re^1$ and $w_{\delta_i} \in \Re^1$, $i = 1, \ldots, n$, which are sets of non-negative numbers. The weights w_{ϵ_i} and w_{δ_i} can thus be used to compensate for instances when the y_i and x_i have unequal precision, or when there are observations that should be eliminated from the analysis. (See §1.F.)

1.A.ii. Implicit Orthogonal Distance Regression

The univariate implicit orthogonal distance regression problem arises from the assumption that there is no distinguished variable y and thus that the data must satisfy

$$f_i(x_i + \delta_i^*; \beta^*) = 0 \qquad i = 1, \ldots, n.$$

For $x_i \in \Re^1$, this yields the minimization problem

$$\begin{aligned} &\min_{\beta,\delta} \sum_{i=1}^{n} w_{\delta_i} \delta_i^2 \\ &\text{subject to} \quad f_i(x_i + \delta_i; \beta) = 0 \quad i = 1, \ldots, n, \end{aligned} \tag{1.6}$$

where $w_{\delta_i} \in \Re^1$, $i = 1, \ldots, n$, again denotes a set of positive numbers used for weighting. (See §1.F.) The constraints in (1.6) are not assumed to be linear in δ_i, and therefore cannot be eliminated to create an unconstrained problem as was done above to specify the explicit problem (1.4). The implicit orthogonal distance regression problem must therefore be solved differently than the explicit problem. (See §1.B.)

1.A.iii. Orthogonal Distance Regression for Multidimensional Data

For simplicity, the preceding two sections developed the explicit and implicit orthogonal distance regression problem for univariate data, i.e., for $x_i \in \Re^1$ and $y_i \in \Re^1$ with

$f_i : \Re^{p+1} \to \Re^1$. In this section, we generalize these problems to allow for multivariate explanatory data $x_i \in \Re^m$, $m \geq 1$, and multiresponse dependent variables $y_i \in \Re^q$, $q \geq 1$, which are modeled by $f_i : \Re^{p+m} \to \Re^q$. For such higher dimensional data, the errors $\delta_i \in \Re^m$ associated with x_i, and the errors $\epsilon_i \in \Re^q$ associated with y_i, can be weighted by $w_{\delta_i} \in \Re^{m\times m}$ and $w_{\epsilon_i} \in \Re^{q\times q}$, $i = 1,\ldots,n$, respectively, where the w_{δ_i}must be positive definite matrices and the w_{ϵ_i}must be positive semidefinite matrices.[1]

For the explicit problem, we assume that the q responses of the observed values of y_i satisfies

$$y_i = f_i(x_i + \delta_i^*; \beta^*) - \epsilon_i^* \qquad i = 1,\ldots,n,$$

for some actual, but unknown value β^*. Thus, if we let $X \in \Re^{n\times m}$ and $Y \in \Re^{n\times q}$ denote the arrays with ith row x_i and y_i, respectively, and $\Delta \in \Re^{n\times m}$ and $E \in \Re^{n\times q}$ denote the arrays with ith row δ_i and ϵ_i, respectively, we are then assuming that the observed values X and Y satisfy

$$\begin{aligned} X &= X^* - \Delta^* \\ Y &= Y^* - E^*, \end{aligned}$$

and that the estimated values, which we denote by a "hat," will satisfy

$$\begin{aligned} \hat{X} &= X + \hat{\Delta} \\ \hat{Y} &= Y + \hat{E}\ . \end{aligned}$$

Generalizing the problem presented in §1.A.i, the *explicit multiresponse orthogonal distance regression problem for multivariate explanatory data* is defined as

$$\min_{\beta,\delta,\epsilon} \sum_{i=1}^{n} \left([\epsilon_i^{\mathrm{T}} w_{\epsilon_i} \epsilon_i] + [\delta_i^{\mathrm{T}} w_{\delta_i} \delta_i] \right)$$

$$\text{subject to} \quad \left.\begin{array}{l} Y_{i1} = f_{i1}(x_i + \delta_i; \beta) - E_{i1} \\ Y_{i2} = f_{i2}(x_i + \delta_i; \beta) - E_{i2} \\ \vdots \\ Y_{iq} = f_{iq}(x_i + \delta_i; \beta) - E_{iq} \end{array}\right\} \quad i = 1,\ldots,n\ , \tag{1.7}$$

where subscript il denotes the (i,l)th element of the corresponding array, e.g., Y_{il} denotes the lth response of the ith observation, and $f_{il}(x_i + \delta_i; \beta)$ denotes the model for the lth

[1]The matrix A is *positive definite* if and only if $A^{\mathrm{T}} = A$, and $a^{\mathrm{T}} Aa > 0$ for all nonzero vectors a. It is *positive semidefinite* if and only if $A^{\mathrm{T}} = A$, and $a^{\mathrm{T}} Aa \geq 0$ for all vectors a. When A is positive definite or positive semidefinite, then there is an upper triangular matrix U such that $A = U^{\mathrm{T}} U$. ODRPACK checks for positive (semi)definiteness by attempting to form U.

response of observation i. Because the constraints in (1.7) are linear in E, this is equivalent to the problem

$$\min_{\beta,\delta} \sum_{i=1}^{n} ([f_i(x_i+\delta_i;\beta)-y_i]^{\mathrm{T}} w_{\epsilon_i}[f_i(x_i+\delta_i;\beta)-y_i] + \delta_i^{\mathrm{T}} w_{\delta_i}\delta_i) = \min_{\beta,\delta} S(\beta,\delta)\ , \tag{1.8}$$

where we define the *weighted sum of squares* $S(\beta,\delta)$ by

$$S(\beta,\delta) \equiv S_\epsilon(\beta,\delta) + S_\delta(\beta,\delta)$$

with

$$S_\epsilon(\beta,\delta) \equiv \sum_{i=1}^{n} ([f_i(x_i+\delta_i;\beta)-y_i]^{\mathrm{T}} w_{\epsilon_i}[f_i(x_i+\delta_i;\beta)-y_i])$$

and

$$S_\delta(\beta,\delta) \equiv \sum_{i=1}^{n} (\delta_i^{\mathrm{T}} w_{\delta_i}\delta_i)\ .$$

The *implicit multiresponse orthogonal distance regression problem for multivariate explanatory data* is

$$\min_{\beta,\delta} \sum_{i=1}^{n} [\delta_i^{\mathrm{T}} w_{\delta_i}\delta_i] = \min_{\beta,\delta} S_\delta(\beta,\delta)$$

$$\text{subject to} \qquad \left.\begin{array}{l} f_{i1}(x_i+\delta_i;\beta)=0 \\ f_{i2}(x_i+\delta_i;\beta)=0 \\ \vdots \\ f_{iq}(x_i+\delta_i;\beta)=0 \end{array}\right\} \qquad i=1,\ldots,n\ . \tag{1.9}$$

As was the case in §1.A.ii, the constraints in (1.9) are not assumed to be linear in δ_i, and therefore cannot be eliminated to create an unconstrained problem as was done above to specify the explicit problem (1.8). The implicit orthogonal distance regression problem must therefore be solved differently than the explicit problem. (See §1.B.)

Note that when $q > 1$, the responses of a multiresponse orthogonal distance regression problem cannot simply be treated as q separate observations as can be done for ordinary least squares when the q responses are uncorrelated. This is because ODRPACK would then treat the variables associated with these q observations as unrelated, and thus not constrain the errors δ_i in x_i to be the same for each of the q occurrences of the ith observation. The user must therefore indicate to ODRPACK when the observations are multiresponse, so that ODRPACK can make the appropriate adjustments to the estimation procedure. (See §2.B.ii, subroutine argument `NQ`.)

1.B. Algorithm

The algorithm implemented in ODRPACK is described in [Boggs *et al.*, 1987 and 1989]. Briefly, the solution is found iteratively using a trust region Levenberg-Marquardt method, with scaling used to accommodate problems in which estimated values have widely varying magnitudes. The Jacobian matrices, i.e., the matrices of first partial derivatives of $f_{il}(x_i + \delta_i; \beta)$, $i = 1, \ldots, n$, and $l = 1, \ldots, q$, with respect to each component of β and Δ, are computed at every iteration either by finite differences or by a user supplied subroutine, as specified by subroutine argument `JOB` (see §2.B.ii, and also §4.A). The iterations are stopped when any one of three stopping criteria are met. Two of these indicate the iterations have converged to a solution. These are *sum of squares convergence*, which indicates that the change in the weighted sum of the squared observation errors is sufficiently small, and *parameter convergence*, which indicates the change in the estimated values of β and Δ is sufficiently small. The third stopping criterion is a limit on the number of iterations.

ODRPACK finds the solution of an implicit orthogonal distance regression problem using the classic quadratic penalty function method. The *penalty function* is

$$P(\beta, \delta; r_k) = \sum_{i=1}^{n} \left(r_k \left[f_i(x_i + \delta_i; \beta) \right]^{\mathrm{T}} \left[f_i(x_i + \delta_i; \beta) \right] + \left[\delta_i^{\mathrm{T}} w_{\delta_i} \delta_i \right] \right) \;, \tag{1.10}$$

with *penalty term*

$$\sum_{i=1}^{n} r_k \left[f_i(x_i + \delta_i; \beta) \right]^{\mathrm{T}} \left[f_i(x_i + \delta_i; \beta) \right]$$

and *penalty parameter* r_k. A sequence of unconstrained minimization problems

$$\min_{\beta, \delta} P(\beta, \delta; r_k)$$

is then solved for a sequence of values of the penalty parameter r_k tending to ∞. These problems are equivalent to an explicit orthogonal distance regression problem with $\epsilon_i \equiv f_i(x_i + \delta_i; \beta)$. As $r_k \to \infty$, $\epsilon_i \to 0$, $i = 1, \ldots, n$, and the solution to (1.10) will approach that of the implicit problem defined by (1.9). See, e.g., [Gill *et al.*, 1981] for further discussion of penalty function methods.

1.C. Specifying the Task

The user has the option of specifying several different aspects of the problem:

1. whether the fit is to be by explicit or implicit orthogonal distance regression, or by ordinary least squares (see §1.A);

2. whether the necessary Jacobian matrices should be approximated by ODRPACK using forward or central finite differences, or whether the user has supplied the code to compute them, and, if such code has been provided by the user, whether it should be checked (see §4.A);
3. whether the covariance matrix V_β and standard deviations σ_β should be computed for the estimators of β, and whether the Jacobian matrices should be recalculated at the solution for this computation (see §4.B);
4. whether the errors Δ have been initialized by the user (see §1.E, and §2.B.ii, subroutine arguments JOB and WORK);
5. whether the fit is a "restart," i.e., whether the fit will use information preserved in the vectors BETA, WORK and IWORK to continue from a previously found point (see Chapter 3);
6. whether the data are multiresponse or not;
7. whether subsets of the unknowns β and Δ should be treated as constants with their values held "fixed," allowing the user to examine the results obtained by estimating subsets of the parameters of a general model without rewriting the model subroutine, and allowing the user to specify that some components of X are to be treated as if they are known exactly;
8. whether weighted or unweighted analysis should be performed (see §1.F); and
9. whether the unknowns β and Δ should be scaled to compensate for cases where their values vary widely in magnitude (see §4.D).

The first 5 of these options are specified by ODRPACK subroutine argument JOB; multiresponse data are indicated by subroutine argument NQ; parameter "fixing" is specified by subroutine arguments IFIXB and IFIXX; weighting is controlled by arguments WE and WD; and scaling is controlled by arguments SCLB and SCLD. A detailed discussion of each of these subroutine arguments can be found in §2.B.ii.

1.D. ODRPACK Generated Results

Results generated by ODRPACK are returned to the user in four ways:

- the estimated parameter values $\hat{\beta}$ are returned in subroutine argument BETA;
- the stopping condition is returned in subroutine argument INFO;
- the computed results available at the time the job stopped are returned in the vectors specified by subroutine arguments WORK and IWORK; and
- selected results are summarized in automatically generated reports.

Arguments BETA, INFO, WORK and IWORK are discussed in detail in §2.B.ii. The remainder of this section describes ODRPACK's automatically generated reports.

ODRPACK can generate two different types of reports. The first is an *error report*, and the second is a *computation report*. These are written to the logical units specified by subroutine arguments LUNERR and LUNRPT, respectively. The user can associate these units with a file using a Fortran OPEN statement. If the user sets one or the other of these units to a negative value, then the corresponding report will be generated on unit 6. On most systems, unit 6 is "standard output," which for an "interactive" run will be the screen. If the user sets one or the other of these units to 0, then generation of the corresponding report is suppressed.

ODRPACK's computation reports (see §1.D.ii below) can, at the user's option, be written simultaneously to both the file associated with the unit specified by LUNRPT and to unit 6 (assuming LUNRPT does not specify unit 6). This option, invoked by ODRPACK subroutine argument IPRINT, enables the user to monitor ODRPACK's progress interactively while still preserving the results in a file for future reference. The option is especially useful when the user's model is time consuming and the user is interactively experimenting with starting values and other ODRPACK options: results printed to the screen can be used to terminate ineffective or incorrect runs, while results stored on the logical unit specified by LUNRPT can be used to preserve the successful experiments.

1.D.i. Error Reports

Error reports identify incorrect user supplied information passed to ODRPACK that prevents the computations from beginning. For example, an error report will be generated if the user specifies the number of observations to be an obviously meaningless number, such as a negative value. Error reports are self explanatory, and are not discussed further here. The information written in the error reports is also encoded and returned to the user in subroutine argument INFO. (See §2.B.ii.)

1.D.ii. Computation Reports

Computation reports are divided into three sections:

- **a.** the initial report,
- **b.** the iteration report, and
- **c.** the final report.

Each of these can be either "short" or "long".

1.D.ii.a. Initial Reports

ODRPACK's initial report identifies the output for future reference, and provides information that should enable the user to verify that the problem was specified correctly.

The *short initial report* is shown in §2.C.i.c. It includes

- the values n, m, p and q, the number $\tilde{n}$ of observations with nonzero weights, and the number $\tilde{p}$ of parameters β actually being estimated;
- the values specified by ODRPACK subroutine arguments `JOB`, `NDIGIT`, `TAUFAC`, `SSTOL`, `PARTOL`, and `MAXIT`, that indicate, respectively, the requested task, the number of "good" digits in the user's function results, the stopping criteria for testing for sum of squares convergence, the stopping criteria for testing for parameter convergence, and the maximum number of iterations permitted; and
- the weighted sum of the squared errors $S(\beta,\delta)$, $S_\delta(\beta,\delta)$ and $S_\epsilon(\beta,\delta)$, evaluated at the initial values of β and Δ.

The *long initial report*, shown in §2.C.ii.c and §2.C.iii.c, includes all the information in the short initial report and, in addition, includes a brief summary of the information specified for the function parameters β, and also information about the user supplied values for the X and Y data. This information includes

- the starting parameter values $\beta_k, k = 1,\ldots,p$, whether or not each parameter is to be treated as "fixed", what "scale" value will be used, and, when the derivatives are constructed using finite differences the step size used in that calculation, or when the user supplied derivatives were checked what derivatives were identified as questionable;
- the value of the first and last observation of each column of the explanatory variable X, and when the solution is found by orthogonal distance regression,
  - the starting errors Δ for each of these observations,
  - whether or not these values will be held fixed at their input values,
  - the scale values,
  - the diagonal elements of the weights w_{δ_i} associated with each of these observations, and
  - when the derivatives are constructed using finite differences the step size used in that calculation, or when the user supplied derivatives were checked what derivatives were identified as questionable;
- the value of the first and last observation of the response variable Y, and the value of the corresponding diagonal entries of the error weights w_{ϵ_i}.

1.D.ii.b. Iteration Reports

The ODRPACK iteration reports enable the user to monitor the progress of the fitting procedure, where the user controls at which iterations these reports will be generated.

ODRPACK iteration reports are of the greatest use in cases when ODRPACK fails to find a satisfactory solution. In cases when ODRPACK does reach a satisfactory solution, the *final report* discussed below summarizes the most useful information.

The *short iteration report*, shown in §2.C.i.c, is a one line summary of the results, listing

- the iteration number;
- the cumulative number of function evaluations made by the end of the listed iteration;
- the weighted sum of the squared observation errors $S(\beta, \delta)$ at the current point;
- the actual relative reduction in $S(\beta, \delta)$ at the most recently tried step (used to check for sum of squares convergence);
- the predicted relative reduction in $S(\beta, \delta)$ at the most recently tried step (used to check for sum of squares convergence);
- the ratio of the trust region radius to the scaled norm of β and Δ, which is an upper bound on the relative change in the estimated values possible at the next step (used to check for parameter convergence); and
- whether the step was a Gauss-Newton step.

The *long iteration summary* lists all of the information found in the short iteration summary and, in addition, includes

- the values of β at the end of the current iteration. (At the last iteration, the values listed will be those that produced the actual and predicted relative reductions shown only if the most recently tried step did in fact make the fit better. If not, then the values of β listed are those that produced the best fit.)

The long iteration report requires 125 columns, and uses $\lceil p/3 \rceil$ lines per iteration.

1.D.ii.c. Final Reports

The final report provides information that allows the user to assess the quality of the solution found by ODRPACK.

The *short final report*, shown in §2.C.ii.c and §2.C.iii.c, includes

- the reason the computations stopped;
- the number of iterations;

- the number of function evaluations and, if the Jacobian was supplied by the user, the number of Jacobian evaluations;
- the rank deficiency of the model at the time the computations stopped;
- the inverse of the condition number of the problem at the time the computations stopped (see §4.C);
- the weighted sum of the squared errors, $S(\beta, \delta)$, $S_\delta(\beta, \delta)$ and $S_\epsilon(\beta, \delta)$, evaluated at the final values $\hat{\beta}$ and $\hat{\Delta}$, and if the covariance matrix was computed, the estimated residual variance of the fit and its associated degrees of freedom; and
- the final values $\hat{\beta}$, whether each value was "fixed" using subroutine argument IFIXB or "dropped" by ODRPACK because it induced rank deficiency, and, if the covariance matrix $\hat{V}_\beta$ and standard deviations $\hat{\sigma}_\beta$ were computed, the standard errors and 95% confidence intervals for the estimators of β. (See §4.B).

The *long final report* is shown in §2.C.i.c. It includes the same information as the short final report, and, in addition, provides the values for all $\hat{E}_{il}, i = 1, \ldots, n,\ l = 1, \ldots, q$, and $\hat{\Delta}_{ij}, i = 1, \ldots, n,\ j = 1, \ldots, m$.

1.E. Starting Values

The user must supply starting values for the unknowns being estimated, i.e., for β and Δ. Users familiar with the nonlinear ordinary least squares problem are generally aware of the importance of obtaining good starting values for β. It is equally important here. Good initial approximations can significantly decrease the number of iterations required to find a solution; a poor initial approximation may even prevent a solution from being found. Reasonable initial approximations are often available from previous analysis or experiments. When good starting values are not readily available, the user may have to do some preliminary analysis to obtain them. (See, e.g., [Bates and Watts, 1988], or [Himmelblau, 1970].)

Users who do not provide scale information are strongly encouraged not to use zero as an initial approximation for any of the parameters β since a zero value can result in incorrect scaling. (See §4.D). Setting the initial approximation to the largest magnitude that, for the user's problem, is effectively zero, rather than to zero, will help to eliminate scaling problems and possibly produce faster convergence. For example, if β_1 represents change in a cost measured in millions of dollars, then the value 10 might be considered "effectively zero" and an initial value of $\beta_1^0 = 10$ is preferable to $\beta_1^0 = 0$.

When using orthogonal distance regression it is also important to have good starting values for the estimated errors Δ. The ODRPACK default is to initialize Δ to zero, which is the most obvious initial value. (Note that zero starting values for Δ do not cause scaling problems similar to those discussed above for β.) Initializing Δ to zero, however, is equivalent to initially assigning all of the errors to the dependent variable as is done for ordinary least squares. While this is quite adequate in many cases, in others it is not. A plot of the observed data and of the curve described by the model function for the initial parameters may indicate whether or not zero starting values for Δ are reasonable and may make it possible to visually determine better starting values for some of the Δ_{ij}. For example, if such a plot shows that the vertical distance from a data point (x_i, y_i) to the curve is far larger than the orthogonal distance, then δ_i should probably not be initialized to zero for that point. This may occur near an asymptote or near a local minimum or maximum. In such cases, it is often appropriate to initialize δ_i to the horizontal distance from the data point to the curve. (See §2.B.ii, subroutine arguments `JOB` and `WORK`.)

1.F. Weights

Weights can be used to eliminate observations from the analysis, to compensate for unequal variances or correlations in the variables y and x, or simply to modify the effect of the various errors ϵ and δ on the fit.

The weights $w_{\epsilon_i} \in \Re^{q\times q}$ associated with ϵ_i, $i = 1, \ldots, n$, must be positive semidefinite matrices, and the weights $w_{\delta_i} \in \Re^{m\times m}$ associated with δ_i, $i = 1, \ldots, n$, must be positive definite matrices.[2] If w_{ϵ_i} is diagonal, then the errors in the q responses of y_i are treated as uncorrelated. Similarly, if w_{δ_i} is diagonal, then the errors in the m elements of x_i are treated as uncorrelated. When the weights are not diagonal matrices, then the errors *within* an observation are treated as correlated. In all cases, however, the errors *between* observations are treated as *uncorrelated.*

Observations can be eliminated from the analysis by setting the appropriate weight values $w_{\epsilon_i}$to zero. This will produce the same results as an analysis with the zero-weighted values removed from the data prior to calling ODRPACK. Zero weights can be used to allow for cases of multiresponse data where the number of responses is not constant for all observations. They also allow for easy examination of the effect of outliers and influential data points.

Outliers can often be identified by large values of ϵ and/or δ. Careful checking of the data often leads to confirmation that the data are in error, and sometimes to a correction. When a cause for suspicious data cannot be found, it may be advisable to compare the analysis with and without the questionable data. Caution is in order if the estimates or conclusions are highly sensitive to a small amount of suspicious data. Data that have a very high influence on a fitted curve may not result in large errors, however, even if

[2]See footnote 1 on page 6.

they are in error. In fact, extremely influential observations may force the fitted curve to be very close, leading to very small residuals. It is therefore desirable to identify influential observations and to compare the results obtained with and without these points. Several methods for detecting influential observations for ordinary least squares are discussed in [Belsley *et al.*, 1980], [Bement and Williams, 1969], [Cook, 1977], and [Hoaglin and Welsch, 1978].

Using weights to compensate for unequal error variances is not as straightforward as using zero weights to eliminate observations from the analysis. When the errors e_i, $i = 1, \ldots, n$, and δ_i, $i = 1, \ldots, n$, have covariances $\sigma^2_{e_i} \in \Re^{q \times q}$ and $\sigma^2_{\delta_i} \in \Re^{m \times m}$, respectively, that are *known*, then the weights can be set using

$$w_{\epsilon_i} = c_1 \sigma_{e_i}^{-2} \qquad \text{and} \qquad w_{\delta_i} = c_2 \sigma_{\delta_i}^{-2} ,$$

$i = 1, \ldots, n$, where c_1 and c_2 are constants selected so that the magnitudes of the weighted errors $\epsilon_i^{\mathrm{T}} w_{\epsilon_i} \epsilon_i$ and $\delta_i^{\mathrm{T}} w_{\delta_i} \delta_i$ will be approximately the same.

In practice, weights are often derived from theory or obtained from the data being fit. Users must be aware that incorrectly specified weights can adversely affect the results. (See, e.g., [Boggs and Rogers, 1990a].) Thus, when the need for weights is suspected and the error covariances are not known, it is extremely important to analyze how the weights are affecting the computed results. See [Fuller, 1987] for a more complete discussion of weights and their implications for the estimation of the parameters of measurement error models, and also [Bates and Watts, 1988] for a procedure that can be used to estimate the covariance matrix $\sigma^2_{e_i}$ in the case of multiresponse nonlinear ordinary least squares.

1.G. Default Values and Structured Arguments

ODRPACK uses default values and structured arguments to simplify the user interface. The availability of default values in ODRPACK means that the user does not have to be concerned with determining values for many of the ODRPACK arguments unless the problem being solved requires the use of nondefault values. Structured arguments, described below, can reduce the amount of storage space required for some arguments, and the work required by the user to initialize those arguments.

1.G.i. Default Values

Default values have been specified for ODRPACK subroutine arguments wherever reasonable. These default values are invoked when the user sets the corresponding argument to any negative value. Arrays with default values are invoked by setting the first element of the array to a negative value, in which case only the first value of the array will ever be used. This allows a scalar to be used to invoke the default values of arrays, thus saving space and eliminating the need to declare such arrays.

Users are encouraged to invoke the default values of arguments wherever possible. These values have been found to be reasonable for a wide class of problems. Fine tuning these arguments can then be done later if necessary.

1.G.ii. Structured Arguments

Certain ODRPACK arguments specify attributes of the individual components of X, Δ, and Y. These attribute arrays are frequently either constant for all components or are constant within each column and vary only between the columns. The *structured argument* facility allows a scalar to specify an attribute of an entire column or of the whole array.

For example, ODRPACK argument `WD` specifies the attribute array of weights $W_\Delta \in \Re^{n\times(m\times m)}$ whose ith component specifies the elements of w_{δ_i}, $i = 1,\ldots,n$ (cf. (1.8) and (1.9)). Suppose X contains temperature measurements, where each row indicates a different hour at which the readings were taken, and each column a different day. Then the user might want to weight each component of Δ equally, and thus W_Δ would be constant throughout. If one column of X contained hourly temperature readings and the other hourly humidity readings, however, then the user would probably not want to weight the errors in the temperature measurements the same as those in the humidity measurements, but might want to specify the same weight w_ϵ for each observation. In this case, the components of W_Δ would be constant for each row i, $i = 1,\ldots,n$. Of course, in other cases, the user might want to weight each component of Δ differently, and thus each component of W_Δ would be different.

ODRPACK structured arguments exploit this structure. If each of the elements of an attribute array is the same, then a scalar can be used to specify all elements of the array. If the values of such an array only vary between the columns, then each column of the array can be specified by a single value using a row vector. Thus, it is only necessary for the user to supply all elements of an attribute array when the elements of one or more of the columns must be individually specified. The use of structured arguments in described in detail in §2.B.ii. (See, e.g., subroutine argument `WD`.)

2. Using ODRPACK

2.A. Subroutine Declaration and Call Statements

The declaration and call statements for ODRPACK's user callable routines are given below. `DODR` and `DODRC` invoke the double precision version of the code and `SODR` and `SODRC` invoke the single precision version. `DODR` and `SODR` preset many arguments to their default values and therefore have shorter call statements than `DODRC` and `SODRC`. In contrast, `DODRC` and `SODRC` have expanded call statements that give the user greater control in solving the orthogonal distance regression problem.

The information in this section, §2.A, is provided primarily for reference. Users are directed to §2.B for definitions of the subroutine arguments, and to §2.C for sample programs. The examples, which use Fortran `PARAMETER` statements to dimension ODRPACK arrays, provide a recommended format for creating an ODRPACK driver that will allow future changes to be made easily.

Note that although ODRPACK is distributed in both single precision and double precision versions, both versions may not be available to all users. In addition, even when both versions are available, the single precision version may not be appropriate to use. This is because ODRPACK is sensitive to the machine's precision, and requires approximately 14 decimal places. Somewhat fewer places should still work, but six or seven decimal places are definitely too few for general use, since only the simplest problems could be solved correctly at such reduced precisions. When both versions are available, the user must choose which version of ODRPACK to use based upon which version supplies adequate precision on the target machine. To our knowledge, at present only Cray and CDC machines offer sufficient precision to permit general use of the single precision version of ODRPACK. For other machines, we recommend the double precision version.

If both versions of ODRPACK have sufficient precision on the user's machine, then either may be used. When both the single and double precision versions are available, however, there are generally trade-offs between them. The double precision version will offer greater accuracy in results, while the single precision version will require less storage and possibly less machine time.

DODR: Compute the explicit or implicit weighted orthogonal distance regression, or linear or nonlinear ordinary least squares solution in double precision. Derivatives are either supplied by the user or numerically approximated by ODRPACK. Control variables are preset to their default values, and a three part report of the results can be optionally generated.

```
      PROGRAM MAIN
       |
      EXTERNAL
     +   FCN
      INTEGER
     +   N,M,NP,NQ,
     +   LDY,LDX, LDWE,LD2WE,LDWD,LD2WD,
     +   JOB, IPRINT,LUNERR,LUNRPT,
     +   LWORK,LIWORK, INFO
      INTEGER
     +   IWORK(LIWORK)
      DOUBLE PRECISION
     +   BETA(NP), Y(LDY,NQ),X(LDX,M),
     +   WE(LDWE,LD2WE,NQ),WD(LDWD,LD2WD,M),
     +   WORK(LWORK)
       |
      CALL DODR
     +   (FCN,
     +   N,M,NP,NQ,
     +   BETA,
     +   Y,LDY,X,LDX,
     +   WE,LDWE,LD2WE,WD,LDWD,LD2WD,
     +   JOB,
     +   IPRINT,LUNERR,LUNRPT,
     +   WORK,LWORK,IWORK,LIWORK,
     +   INFO)
       |
      END
```

DODRC: Compute the explicit or implicit weighted orthogonal distance regression, or linear or nonlinear ordinary least squares solution in double precision. Derivatives are either supplied by the user or numerically approximated by ODRPACK. Control values are supplied by the user, and a three part report of the results can be optionally generated.

```
      PROGRAM MAIN
       |
      EXTERNAL
     +   FCN
      INTEGER
     +   N,M,NP,NQ,
     +   LDY,LDX, LDWE,LD2WE,LDWD,LD2WD, LDIFX,
     +   JOB,NDIGIT, MAXIT, IPRINT,LUNERR,LUNRPT,
     +   LDSTPD, LDSCLD, LWORK,LIWORK, INFO
      INTEGER
     +   IFIXB(NP),IFIXX(LDIFX,M), IWORK(LIWORK)
      DOUBLE PRECISION
     +   TAUFAC, SSTOL,PARTOL
      DOUBLE PRECISION
     +   BETA(NP), Y(LDY,NQ),X(LDX,M),
     +   WE(LDWE,LD2WE,NQ),WD(LDWD,LD2WD,M),
     +   STPB(NP),STPD(LDSTPD,M), SCLB(NP),SCLD(LDSCLD,M),
     +   WORK(LWORK)
       |
      CALL DODRC
     +   (FCN,
     +   N,M,NP,NQ,
     +   BETA,
     +   Y,LDY,X,LDX,
     +   WE,LDWE,LD2WE,WD,LDWD,LD2WD,
     +   IFIXB,IFIXX,LDIFX,
     +   JOB,NDIGIT,TAUFAC,
     +   SSTOL,PARTOL,MAXIT,
     +   IPRINT,LUNERR,LUNRPT,
     +   STPB,STPD,LDSTPD,
     +   SCLB,SCLD,LDSCLD,
     +   WORK,LWORK,IWORK,LIWORK,
     +   INFO)
       |
      END
```

SODR: Compute the explicit or implicit weighted orthogonal distance regression, or linear or nonlinear ordinary least squares solution in single precision. (SODR is appropriate for general use only on machines with approximately 14 decimal places of precision for single precision.) Derivatives are either supplied by the user or numerically approximated by ODRPACK. Control variables are preset to their default values, and a three part report of the results can be optionally generated.

```
 PROGRAM MAIN
  |
 EXTERNAL
+   FCN
 INTEGER
+   N,M,NP,NQ,
+   LDY,LDX, LDWE,LD2WE,LDWD,LD2WD,
+   JOB, IPRINT,LUNERR,LUNRPT,
+   LWORK,LIWORK, INFO
 INTEGER
+   IWORK(LIWORK)
 REAL
+   BETA(NP), Y(LDY,NQ),X(LDX,M),
+   WE(LDWE,LD2WE,NQ),WD(LDWD,LD2WD,M),
+   WORK(LWORK)
  |
 CALL SODR
+   (FCN,
+   N,M,NP,NQ,
+   BETA,
+   Y,LDY,X,LDX,
+   WE,LDWE,LD2WE,WD,LDWD,LD2WD,
+   JOB,
+   IPRINT,LUNERR,LUNRPT,
+   WORK,LWORK,IWORK,LIWORK,
+   INFO)
  |
 END
```

SODRC: Compute the explicit or implicit weighted orthogonal distance regression, or linear or nonlinear ordinary least squares solution in single precision. (SODRC is appropriate for general use only on machines with approximately 14 decimal places of precision for single precision.) Derivatives are either supplied by the user or numerically approximated by ODRPACK. Control values are supplied by the user, and a three part report of the results can be optionally generated.

```
      PROGRAM MAIN
       |
      EXTERNAL
     +   FCN
      INTEGER
     +   N,M,NP,NQ,
     +   LDY,LDX, LDWE,LD2WE,LDWD,LD2WD, LDIFX,
     +   JOB,NDIGIT, MAXIT, IPRINT,LUNERR,LUNRPT,
     +   LDSTPD, LDSCLD, LWORK,LIWORK, INFO
      INTEGER
     +   IFIXB(NP),IFIXX(LDIFX,M), IWORK(LIWORK)
      REAL
     +   TAUFAC, SSTOL,PARTOL
      REAL
     +   BETA(NP), Y(LDY,NQ),X(LDX,M),
     +   WE(LDWE,LD2WE,NQ),WD(LDWD,LD2WD,M),
     +   STPB(NP),STPD(LDSTPD,M), SCLB(NP),SCLD(LDSCLD,M),
     +   WORK(LWORK)
       |
      CALL SODRC
     +   (FCN,
     +   N,M,NP,NQ,
     +   BETA,
     +   Y,LDY,X,LDX,
     +   WE,LDWE,LD2WE,WD,LDWD,LD2WD,
     +   IFIXB,IFIXX,LDIFX,
     +   JOB,NDIGIT,TAUFAC,
     +   SSTOL,PARTOL,MAXIT,
     +   IPRINT,LUNERR,LUNRPT,
     +   STPB,STPD,LDSTPD,
     +   SCLB,SCLD,LDSCLD,
     +   WORK,LWORK,IWORK,LIWORK,
     +   INFO)
       |
      END
```

2.B. Subroutine Argument Descriptions

2.B.i. Synopsis

The arguments of the ODRPACK user callable subroutines are logically grouped as shown below. Arguments shown in parenthesis (...) are not included in the DODR and SODR call statements; DODR and SODR automatically preset these variables to the default values given in §2.B.ii. All other arguments are common to all ODRPACK user callable subroutines.

Argument Number	Arguments	Group Description
1	FCN,	Name of user supplied subroutine for function and derivative computation
2 to 5	N,M,NP,NQ,	Problem size specification
6	BETA,	Function parameters
7 to 10	Y,LDY,X,LDX,	Dependent and explanatory variables
11 to 16	WE,LDWE,LD2WE,WD,LDWD,LD2WD,	Weights
17 to 19	(IFIXB,IFIXX,LDIFX,)	Parameter and variable fixing
20 to 22	JOB,(NDIGIT,TAUFAC,)	Computation and initialization control
23 to 25	(SSTOL,PARTOL,MAXIT,)	Stopping criteria
26 to 28	IPRINT,LUNERR,LUNRPT,	Print control
29 to 31	(STPB,STPD,LDSTPD,)	Derivative step sizes
32 to 34	(SCLB,SCLD,LDSCLD,)	Scaling
35 to 38	WORK,LWORK,IWORK,LIWORK,	Work vectors and returned results
39	INFO	Stopping condition

2.B.ii. ODRPACK Subroutine Argument Definitions

The arguments of ODRPACK's user callable subroutines are described below in order of their occurrence in the call statements. Appropriate declaration statements for each argument are shown in brackets [...] following the argument name; the character string `<real>` denotes `REAL` when using single precision subroutines `SODR` and `SODRC` (which should be used only on machines with approximately 14 decimal digits of precision in single precision), and denotes `DOUBLE PRECISION` when using double precision subroutines `DODR` and `DODRC`. Each argument is numbered as shown in §2.B.i, and will be cross referenced by both number and name, e.g., 1–`FCN`, enabling the user to easily find the definition of a specific argument. In addition, a flag indicating whether the argument is passed to or from ODRPACK is placed in the left margin by the argument number. The flags are:

- $\Longrightarrow$ indicating the argument specifies information that must be *input* to ODRPACK, and that the input information is preserved upon return from the ODRPACK subroutine;
- $\Longleftarrow$ indicating the argument specifies information that is *output* by the ODRPACK subroutine; and
- $\Longleftrightarrow$ indicating the argument specifies information that must be *input* to ODRPACK, but that the input values will be *overwritten* by ODRPACK upon return from the subroutine.

NOTE

Substitute `DOUBLE PRECISION` for `<real>` when using `DODR` and `DODRC`.

Substitute `REAL` for `<real>` when using `SODR` and `SODRC`.

****

$\Longrightarrow$1 - FCN [EXTERNAL FCN]

The name of the user supplied subroutine that, given the current values of the explanatory variable, $x_I + \delta_I$, and the current values of the function parameters, β, computes the predicted values

$$\texttt{F(I,L)} = f_{IL}(x_I + \delta_I; \beta) \ ,$$

I $= 1, \ldots, n$, and L $= 1, \ldots, q$, and optionally computes the matrices of first partial derivatives, i.e., the Jacobian matrices

$$\texttt{FJACB(I,K,L)} = \frac{\partial f_{IL}(x_I + \delta_I; \beta)}{\partial \beta_K} \ ,$$

for I $= 1, \ldots, n$, K $= 1, \ldots, p$, and L $= 1, \ldots, q$, and

$$\texttt{FJACD(I,J,L)} = \frac{\partial f_{IL}(x_I + \delta_I; \beta)}{\partial \Delta_{IJ}} \ ,$$

for I $= 1, \ldots, n$, J $= 1, \ldots, m$, and L $= 1, \ldots, q$. The code for evaluating F must always be provided. The code for evaluating FJACB and FJACD is required only when the second digit of argument 20–JOB is greater than or equal to two. (When the second digit of JOB is zero or one, the necessary Jacobian matrices will be approximated by ODRPACK using finite differences. See argument 20–JOB, and also §4.A.) The code for FJACD does not need to be supplied when the fit is by OLS.

At a given call to subroutine FCN, ODRPACK will never request that both the function values (F) and the derivative values (FJACB and FJACD) be computed. While it is generally most cost effective if the user only performs the required computations, it is not an error for both function values and derivatives to be computed each time FCN is invoked. *Note, however, that array* FJACD *must never be altered when the fit is by ordinary least squares, since no space is assigned to that array in that case.*

The user must supply a value for every element of the selected arrays. If some responses of some observations are actually missing, then the user can set the corresponding weights in argument 11–WE to zero in order to remove the effect of the missing observation from the analysis. (See §1.F.)

ODRPACK's parameter fixing arguments IFIXB, IFIXX and LDIFX are passed to subroutine FCN for the user's convenience. When a parameter is fixed, then the derivative with respect to that parameter can be set to zero. ODRPACK will automatically zero these derivative values, however, and thus it is not necessary for the user to be concerned with this unless the derivative computations are especially expensive. (See Chapter 3.)

The argument list and dimension statements for subroutine FCN must be exactly as shown below.

```
      SUBROUTINE FCN(N,M,NP,NQ,
     +               LDN,LDM,LDNP,
     +               BETA,XPLUSD,
     +               IFIXB,IFIXX,LDIFX,
     +               IDEVAL,F,FJACB,FJACD,
     +               ISTOP)

C  INPUT ARGUMENTS
C  (WHICH MUST NOT BE CHANGED BY THIS ROUTINE)
      INTEGER IDEVAL,LDFIX,LDM,LDN,LDNP,M,N,NP,NQ
      INTEGER IFIXB(NP),IFIXX(LDIFX,M),
      <real>  BETA(NP),XPLUSD(LDN,M)

C  OUTPUT ARGUMENTS
      INTEGER ISTOP
      <real>  F(LDN,NQ),FJACB(LDN,LDNP,NQ),FJACD(LDN,LDM,NQ)

      < set ISTOP >

      IF (ISTOP.NE.0) RETURN

C  COMPUTE FUNCTION
      IF (MOD(IDEVAL,10).GE.1) THEN
         < compute F(I,L), I=1,...,N, & L=1,...,NQ >
      END IF

C  COMPUTE DERIVATIVES WITH RESPECT TO BETA
      IF (MOD(IDEVAL/10,10).GE.1) THEN
         < compute FJACB(I,K,L), I=1,...,N, K=1,...,NP, & L=1,...,NQ >
      END IF

C  COMPUTE DERIVATIVES WITH RESPECT TO DELTA
      IF (MOD(IDEVAL/100,10).GE.1) THEN
         < compute FJACD(I,J,L), I=1,...,N, J=1,...,M,  & L=1,...,NQ >
      END IF

      RETURN
      END
```

where

`N` is the number of observations, n.

`M` is the number of elements, m, in each explanatory variable, $x_i \in \Re^m$, i.e., the number of columns of data in $X + \Delta$.

`NP` is the number of function parameters, p.

`NQ` is the number of responses, q, per observation.

`LDN` is an array leading dimension declarator that equals or exceeds n.

`LDM` is an array leading dimension declarator that equals or exceeds m.

`LDNP` is an array leading dimension declarator that equals or exceeds p.

`BETA` is the singly subscripted array that contains the current values of β.

`XPLUSD` is the doubly subscripted array that contains the current value of the explanatory variables, i.e., $X + \Delta$.

`IFIXB` is the singly subscripted array designating whether a given parameter β_{K} is to be treated as fixed. (See argument 17–`IFIXB` below.)

`IFIXX` is the doubly subscripted array designating whether, when the solution is found by orthogonal distance regression, a given explanatory variable X_{IJ} is to be treated as without error. (See argument 18–`IFIXX` below.)

`LDIFX` is the leading dimension of array `IFIXX`. (See argument 19–`LDIFX` below.)

`IDEVAL` is a three digit `INTEGER` variable with decimal expansion $\mathcal{I}_3\mathcal{I}_2\mathcal{I}_1$ passed to subroutine `FCN` by ODRPACK to designate what values are to be computed.

Computation	Digit	
`F`	$\mathcal{I}_1 = 0$	array `F` need not be computed
	$= 1$	function values `F` must be computed (for constructing finite difference derivatives)
	$= 2$	function values `F` must be computed (for evaluating $S(\beta, \delta)$ at new point)
	$= 3$	function values `F` must be computed (for miscellaneous calculations)
`FJACB`	$\mathcal{I}_2 = 0$	array `FJACB` need not be computed
	$= 1$	derivative values `FJACB` must be computed
`FJACD`	$\mathcal{I}_3 = 0$	array `FJACD` must not be altered
	$= 1$	derivative values `FJACD` must be computed (required for ODR fits only)

`F` is the doubly subscripted array in which the $n \times q$ array of predicted values for each response of each observation must be stored.

FJACB is the triply subscripted array in which the $n \times p \times q$ array of partial derivatives with respect to β for each response of each observation must be stored.

FJACD is the triply subscripted array in which the $n \times m \times q$ array of partial derivatives with respect to Δ for each response of each observation must be stored.

ISTOP is a variable that enables the user to instruct ODRPACK to reject the current values in BETA and XPLUSD as unacceptable.

ISTOP	Meaning
< 0	regression procedure should be stopped
$= 0$	current β and Δ are acceptable for use by FCN: requested values were properly computed within FCN and regression procedure should continue
> 0	current β and Δ are unacceptable for use by FCN: if call to FCN was for constructing derivatives regression procedure will be stopped if call to FCN was for evaluating $S(\beta, \delta)$ a new point will be selected closer to the most recently tried acceptable values of β and Δ if call to FCN was for miscellaneous calculations regression procedure will be stopped

Note that even when ISTOP is used to stop the regression procedure, the final summary of the computation report will still be generated if it has been requested. (See argument 26–IPRINT.)

$\Longrightarrow$2 – N [INTEGER N]

The number of observations, n.

$\Longrightarrow$3 – M [INTEGER M]

The number of elements, m, in each explanatory variable $x_i \in \Re^m$, i.e., the number of columns of data in X.

$\Longrightarrow$4 – NP [INTEGER NP]

The number of function parameters, p.

$\Longrightarrow$5 – NQ [INTEGER NQ]

The number of responses, q, per observation.

$\Longleftrightarrow$6 – BETA [<real> BETA(NP)]

The singly subscripted array that contains the (current) values of β.

On input: BETA must contain initial approximations for the function parameters β. Initial approximations should be chosen with care since poor initial approximations can significantly increase the number of iterations required to find a solution and possibly prevent the solution from being found at all. (See §1.E.)

On return: BETA contains the "best" estimate of the parameters, $\hat{\beta}$, at the time the computations stopped.

$\Longrightarrow$7 – Y [<real> Y(LDY,NQ)]

The double subscripted array that contains the values of the response variable $Y_{\mathrm{IL}}, \mathrm{I} = 1, \ldots, n$, $\mathrm{L} = 1, \ldots, q$. When the model is *explicit*, the user must supply a value for each of the $n \times q$ elements of Y; if some responses of some observations are actually missing, then the user can set the corresponding weight in argument 11–WE to zero in order to remove the effect of the missing observation from the analysis. When the model is *implicit*, Y is not referenced. (See argument 20-JOB, and §1.F.)

$\Longrightarrow$8 – LDY [INTEGER LDY]

The leading dimension of array Y. When the model is *explicit*, LDY must equal or exceed n. When the model is *implicit*, LDY must equal or exceed 1.

$\Longrightarrow$9 – X [<real> X(LDX,M)]

The doubly subscripted array that contains the observed values of the explanatory variable X.

$\Longrightarrow$10 – LDX [INTEGER LDX]

The leading dimension of array X. LDX must equal or exceed n.

$\Longrightarrow$11 – WE [<real> WE(LDWE,LD2WE,NQ)]

The triply subscripted array that, when the model is *explicit* specifies how each ϵ_{I} is to be weighted in the weighted orthogonal distance, and when the model is *implicit* specifies the starting penalty parameter value, r_0 (see §1.A, §1.B, §1.F, and argument 20–JOB).

For *explicit* models, WE is a structured argument: only the specific elements of WE identified in the table below are referenced by ODRPACK. (See §1.G.)

WE(1,1,1)	LDWE	LD2WE	For $I = 1, \ldots, n$, w_{ϵ_I}
< 0	—	—	$= -$WE(1,1,1)I_q, I_q a $q \times q$ identity matrix
≥ 0	$= 1$	$= 1$	$= \text{diag}\{$WE(1,1,L2), L2 $= 1, \ldots, q\}$
	$\geq n$	$= 1$	$= \text{diag}\{$WE(I,1,L2), L2 $= 1, \ldots, q\}$
	$= 1$	$\geq q$	$=$ WE(1,L1,L2), L1 $= 1, \ldots, q$, & L2 $= 1, \ldots, q$
	$\geq n$	$\geq q$	$=$ WE(I,L1,L2), L1 $= 1, \ldots, q$, & L2 $= 1, \ldots, q$

For *implicit* models, only the first element of WE is ever referenced, and r_0 is set as follows.

WE(1,1,1)	r_0
≤ 0	$= 10$
> 0	$=$ WE(1,1,1)

⟹12 - LDWE [INTEGER LDWE]

The leading dimension of array WE. LDWE must either exactly equal one, or must equal or exceed n. See argument 11–WE for further details.

⟹13 - LD2WE [INTEGER LD2WE]

The second dimension of array WE. LD2WE must either exactly equal one, or must equal or exceed q. See argument 11–WE for further details.

⟹14 - WD [<real> WD(LDWD,LD2WD,M)]

The triply subscripted array that specifies how each δ_I is to be weighted in the weighted orthogonal distance. (See §1.A and §1.F.) WD is a structured argument: only the specific elements of WD identified in the table below are ever referenced by ODRPACK. (See §1.G.)

WD(1,1,1)	LDWD	LD2WD	For $I = 1, \ldots, n$, w_{δ_I}
< 0	—	—	$= -$WD(1,1,1)I_m, I_m an $m \times m$ identity matrix
$= 0$	—	—	$= I_m$, I_m an $m \times m$ identity matrix
> 0	$= 1$	$= 1$	$= \text{diag}\{$WD(1,1,J2), J2 $= 1, \ldots, m\}$
	$\geq n$	$= 1$	$= \text{diag}\{$WD(I,1,J2), J2 $= 1, \ldots, m\}$
	$= 1$	$\geq m$	$=$ WD(1,J1,J2), J1 $= 1, \ldots, m$, &J2 $= 1, \ldots, m$
	$\geq n$	$\geq m$	$=$ WD(I,J1,J2), J1 $= 1, \ldots, m$, &J2 $= 1, \ldots, m$

⟹15 - LDWD [INTEGER LDWD]

The leading dimension of array WD. LDWD must either exactly equal one, or must equal or exceed n. See argument 14–WD for further details.

$\Longrightarrow$16 – LD2WD [INTEGER LD2WD]

The second dimension of array WD. LD2WD must either exactly equal one, or must equal or exceed m. See argument 14–WD for further details.

$\Longrightarrow$17 – IFIXB [INTEGER IFIXB(NP)]

The singly subscripted array that specifies the indicator variable $\mathcal{F}_{\beta_K}$ designating whether β_K is to be treated as "fixed," i.e., is to be treated as a constant, or is to be "unfixed" and thus is to be estimated.

- If $\mathcal{F}_{\beta_K} = 0$ then β_K is fixed and BETA(K) is not changed.
- If $\mathcal{F}_{\beta_K} \neq 0$ then β_K is unfixed and BETA(K) is overwritten by $\hat{\beta}_K$.

The default values are $\mathcal{F}_{\beta_K} = 1$, $K = 1, \ldots, p$.

IFIXB(1)	For $K = 1, \ldots, p$, $\mathcal{F}_{\beta_K}$
< 0	$= 1$
≥ 0	= IFIXB(K)

$\Longrightarrow$18 – IFIXX [INTEGER IFIXX(LDIFX,M)]

The doubly subscripted array that specifies the indicator variable $\mathcal{F}_{X_{IJ}}$ designating whether, when the solution is found by orthogonal distance regression, the (I, J)th element of the explanatory variable X_{IJ} is to be treated as without error and thus Δ_{IJ} is to be "fixed" at zero, or whether that observation is "unfixed" and therefore the error Δ_{IJ} is to be estimated. (When the solution is found by ordinary least squares, the X_{IJ} are always treated as fixed, and thus $\Delta_{IJ} = 0$ for all $I = 1, \ldots, n$, and $J = 1, \ldots, m$.)

- If $\mathcal{F}_{X_{IJ}} = 0$ then X_{IJ} is fixed and Δ_{IJ} is set to zero.
- If $\mathcal{F}_{X_{IJ}} \neq 0$ then X_{IJ} is unfixed and Δ_{IJ} is estimated.

The default values are $\mathcal{F}_{X_{IJ}} = 1$, $I = 1, \ldots, n$, and $J = 1, \ldots, m$.

IFIXX is a structured argument: only the specific elements of IFIXX identified in the table below are referenced by ODRPACK. (See §1.G.)

IFIXX(1,1)	LDIFX	For $I = 1, \ldots, n$, & $J = 1, \ldots, m$, $\mathcal{F}_{X_{IJ}}$
< 0	—	$= 1$
≥ 0	$= 1$	= IFIXX(1,J)
	$\geq n$	= IFIXX(I,J)

$\Longrightarrow$19 – LDIFX [INTEGER LDIFX]

The leading dimension of array IFIXX. LDIFX must either exactly equal one, or must equal or exceed n. See argument 18–IFIXX for further details.

⟹20 – JOB [INTEGER JOB]

The variable that specifies problem initialization and computational methods. (See §1.C.) The default options, selected when JOB ≤ 0, are that

- the solution will be found by explicit orthogonal distance regression;
- the derivatives will be computed by forward finite differences;
- the covariance matrix will be computed using Jacobian matrices recalculated at the solution;
- Δ will be initialized to zero; and
- the fit will not be a restart.

When JOB ≥ 0, it is assumed to be a 5 digit INTEGER with decimal expansion $\mathcal{I}_5\mathcal{I}_4\mathcal{I}_3\mathcal{I}_2\mathcal{I}_1$, where each digit controls a different option.

Option	Digit	Selection
Computational method (see §1.A)	$\mathcal{I}_1 = 0$	explicit orthogonal distance regression
	$= 1$	implicit orthogonal distance regression
	≥ 2	ordinary least squares
Derivative calculation (see §4.A)	$\mathcal{I}_2 = 0$	forward finite differences
	$= 1$	central finite differences
	$= 2$	user supplied derivative code, checked by ODRPACK
	≥ 3	user supplied derivative code, not checked by ODRPACK
Covariance matrix V_β, & standard deviation σ_β (see §4.B)	$\mathcal{I}_3 = 0$	V_β and σ_β calculated using derivatives recomputed at solution
	$= 1$	V_β and σ_β calculated using derivatives from last iteration
	≥ 2	V_β and σ_β not calculated
Δ Initialization (see §1.E)	$\mathcal{I}_4 = 0$	Δ initialized to zero by ODRPACK
	≥ 1	Δ initialized by user (see argument 36–WORK)
Restart facility (see Chapter 3)	$\mathcal{I}_5 = 0$	fit is not a restart
	≥ 1	fit is a restart

⟹21 – NDIGIT [INTEGER NDIGIT]

The variable that specifies the number of reliable decimal digits ψ in the values computed using subroutine FCN. The value ψ is needed to calculate the default values for the relative step sizes used in calculating finite difference derivatives. (See arguments 29–STPB and 30–STPD, and §4.A.) It is also used to determine when the Jacobian with respect to one or more of the parameters

β appears to be rank deficient. It can not exceed the number of decimal digits Ψ carried by the user's computer for a <real> value. The default value is experimentally determined by ODRPACK for the user's particular model. This determination of ψ requires 4 evaluations of the model function from user supplied subroutine FCN.

NDIGIT	ψ
≤ 1	= default value
≥ 2	$= \min\{\texttt{NDIGIT}, \Psi\}$

$\Longrightarrow$22 – TAUFAC [<real> TAUFAC]

The variable that specifies the factor τ used to initialize the trust region radius. The trust region is the region in which the local approximation to $S(\beta, \delta)$ is considered reliable. The diameter of this region is adaptively chosen at each iteration based on information from the previous iteration. At the first iteration, the initial diameter is set to τ times the length of the full Gauss-Newton step calculated at the initial values of β and Δ. The default value is $\tau = 1$. When $\tau < 1$ the size of the initial step attempted at the first iteration is smaller than the full Gauss-Newton step. This may be appropriate if, at the first iteration, the computed results for the full Gauss-Newton step cause an overflow, or cause β and/or Δ leave the region of interest.

TAUFAC	τ
≤ 0	$= 1$
> 0	$= \min\{\texttt{TAUFAC}, 1\}$

$\Longrightarrow$23 – SSTOL [<real> SSTOL]

The variable that specifies $\mathcal{T}_S$, the stopping tolerance for sum of squares convergence, i.e., for convergence based on the relative change in $S(\beta, \delta)$. The default value is $\mathcal{T}_S = \xi^{1/2}$, where ξ is defined as the smallest value such that $1 + \xi > 1$ for a <real> computation on the machine being used.

SSTOL	$\mathcal{T}_S$
< 0	$= \xi^{1/2}$
≥ 0	$= \min\{\texttt{SSTOL}, 1\}$

$\Longrightarrow$24 – PARTOL [<real> PARTOL]

The variable that specifies $\mathcal{T}_P$, the stopping tolerance for parameter convergence, i.e., for convergence based on relative change in the estimated parameters β and Δ. When the model is *explicit* the default value is $\mathcal{T}_P = \xi^{2/3}$, and when the model is *implicit* the default value is $\mathcal{T}_P = \xi^{1/3}$, where ξ is defined as the smallest value such that $1 + \xi > 1$ for a <real> computation on the machine being used.

PARTOL	$\mathcal{T}_P$
< 0	$=$ default value
≥ 0	$= \min\{$PARTOL$, 1\}$

$\Longrightarrow$25 – MAXIT [INTEGER MAXIT]

The variable that specifies $\mathcal{T}_I$, the maximum number of iterations allowed. The default value depends on whether the fit is a restart or not. (See argument 20–JOB.) If the fit is not a restart, then

$$\mathcal{T}_I = 50 \ .$$

If the fit is a restart, then

$$\mathcal{T}_I = \mathcal{T}_{I^-} + 10 \ ,$$

where $\mathcal{T}_{I^-}$ is the number of iterations completed in the previous run, thus indicating that the procedure will continue for an additional 10 iterations.

MAXIT	Restart	$\mathcal{T}_I$
< 0	no	$= 50$
	yes	$= \mathcal{T}_{I^-} + 10$
≥ 0	no	$=$ MAXIT
	yes	$= \mathcal{T}_{I^-} +$ MAXIT

If MAXIT $= 0$ then no iterations will be taken, but whatever computations are required to complete the final computation report will be made. For example, by setting MAXIT $= 0$ and the third digit of JOB to zero, the user can compute the covariance matrix V_β for the input values β and Δ. (See arguments 20–JOB and 26–IPRINT, and also §1.D.)

$\Longrightarrow$26 – IPRINT [INTEGER IPRINT]

The variable that controls generation of the computation reports described in §1.D. The default computation reports include

- a long initial summary,
- no iteration summary, and
- a short final summary.

When IPRINT < 0, the default reports are generated only on $\mathcal{L}_{CR}$, the logical unit specified by argument 28–LUNRPT. When IPRINT ≥ 0, it is assumed to be a 4 digit INTEGER with decimal expansion $\mathcal{I}_4\mathcal{I}_3\mathcal{I}_2\mathcal{I}_1$, where each digit controls a different part of the computation report and whether that report is to be generated only on $\mathcal{L}_{CR}$ or to both $\mathcal{L}_{CR}$ and unit 6. (See §1.D).

Initial summary			Iteration summary				Final summary		
$\mathcal{I}_4$	- unit -		$\mathcal{I}_3$	$\mathcal{I}_2$	- unit -		$\mathcal{I}_1$	- unit -	
	$\mathcal{L}_{CR}$	6			$\mathcal{L}_{CR}$	6		$\mathcal{L}_{CR}$	6
			≥ 0	$= 0$	none	none			
$= 0$	none	none	$= 0$	≥ 1	none	none	$= 0$	none	none
$= 1$	short	none	$= 1$	≥ 1	short	none	$= 1$	short	none
$= 2$	long	none	$= 2$	≥ 1	long	none	$= 2$	long	none
$= 3$	short	short	$= 3$	≥ 1	short	short	$= 3$	short	short
$= 4$	long	short	$= 4$	≥ 1	long	short	$= 4$	long	short
$= 5$	short	long	$= 5$	≥ 1	short	long	$= 5$	short	long
$= 6$	long	long	$= 6$	≥ 1	long	long	$= 6$	long	long

If $\mathcal{I}_2 = 0$ no iteration summary will be generated, even if the value of $\mathcal{I}_3$ is nonzero.

If $\mathcal{I}_2 \geq 1$ an iteration summary will be generated every $\mathcal{I}_2 th$ iteration beginning with iteration one.

$\Longrightarrow$27 – LUNERR [INTEGER LUNERR]

The variable that specifies $\mathcal{L}_{ER}$, the logical unit number to be used for error messages. (See §1.D.) By default, $\mathcal{L}_{ER} = 6$.

LUNERR	$\mathcal{L}_{ER}$
< 0	$= 6$
$= 0$	error messages suppressed
≥ 0	= LUNERR

$\Longrightarrow$28 – LUNRPT [INTEGER LUNRPT]

The variable that specifies $\mathcal{L}_{CR}$, the logical unit number to be used for computation reports. (See also argument 26–IPRINT, and §1.D.) By default, $\mathcal{L}_{CR} = 6$.

LUNRPT	$\mathcal{L}_{CR}$
< 0	$= 6$
$= 0$	computation reports suppressed,
	even when argument 26–IPRINT is nonzero
≥ 0	= LUNRPT

$\Longrightarrow$29 – STPB [<real> STPB(NP)]

The singly subscripted array that specifies the *relative* step sizes, h_{β_K}, $K = 1, \ldots, p$, used to compute the finite difference derivatives with respect to each of the parameters β as discussed in §4.A. The default value is set as described in §4.A.iii depending on whether forward or central finite difference derivatives are being computed. (See argument 20–JOB.)

STPB(1)	For $K = 1, \ldots, p$, h_{β_K}
≤ 0	= default value
> 0	= STPB(K)

$\Longrightarrow$30 - STPD [<real> STPD(LDSTPD,M)]

The doubly subscripted array that specifies the *relative* step sizes, $h_{\Delta_{IJ}}$, $I = 1, \ldots, n$, and $J = 1, \ldots, m$, used to compute the finite difference derivatives with respect to the errors in each of the elements of X as discussed in §4.A. The default value is set as described in §4.A.iii depending on whether forward or central finite difference derivatives are being computed. (See argument 20–JOB.) STPD is a structured argument: only the specific elements of STPD identified in the table below are referenced by ODRPACK. (See §1.G.)

STPD(1,1)	LDSTPD	For $I = 1, \ldots, n$, & $J = 1, \ldots, m$, $h_{\Delta_{IJ}}$
≤ 0	—	= default value
> 0	$= 1$	= STPD(1,J)
	$\geq n$	= STPD(I,J)

$\Longrightarrow$31 - LDSTPD [INTEGER LDSTPD]

The leading dimension of array STPD. LDSTPD must either exactly equal one, or must equal or exceed n. See argument 30–STPD for further details.

$\Longrightarrow$32 - SCLB [<real> SCLB(NP)]

The singly subscripted array that specifies the scale values, $\text{SCALE}\{\beta_K\}$, $K = 1, \ldots, p$, of the function parameters, i.e., the reciprocals of the expected magnitudes or typical sizes of $\beta_K, K = 1, \ldots, p$. Scaling is used within the regression procedure in order that the units of the variable space will have approximately the same magnitude. This increases the stability of the procedure, but does not affect the problem specification. Scaling should not be confused with the weighting matrices w_{ϵ_I} and w_{δ_I} specified by arguments 11–WE and 14–WD. (See §1.A, and §1.F.) Except as noted below, the scale values specified for each value of β must be greater than zero. The default values are set as described in §4.D.i.

SCLB(1)	For $K = 1, \ldots, p$, $\text{SCALE}\{\beta_K\}$
≤ 0	= default value
> 0	= SCLB(K)

⟹33 – SCLD [<real> SCLD(LDSCLD,M)]

The doubly subscripted array that specifies the scale values for the errors Δ in the explanatory variable X, i.e., the reciprocals of the expected magnitudes or typical sizes of Δ_{IJ}, $\mathrm{I} = 1, \ldots, n$, and $\mathrm{J} = 1, \ldots, m$. Scaling is used within the regression procedure in order to ensure that the units of the variable space will have approximately the same magnitude. This increases the stability of the procedure, but does not affect the problem specification. Scaling should not be confused with the weighting matrices $w_{\epsilon_{\mathrm{I}}}$ and $w_{\delta_{\mathrm{I}}}$ specified by arguments 11–WE and 14–WD. (See §1.A, and §1.F.) Except as noted below, the scale values specified for each value of Δ must be greater than zero. The default values are set as described in §4.D.ii. SCLD is a structured argument: only the specific elements of SCLD identified in the table below are referenced by ODRPACK. (See §1.G.)

SCLD(1,1)	LDSCLD	For $\mathrm{I} = 1, \ldots, n$, & $\mathrm{J} = 1, \ldots, m$, SCALE$\{\Delta_{\mathrm{IJ}}\}$
≤ 0	—	= default value
> 0	$= 1$	= SCLD(1,J)
	$\geq n$	= SCLD(I,J)

⟹34 – LDSCLD [INTEGER LDSCLD]

The leading dimension of array SCLD. LDSCLD must either exactly equal one, or must equal or exceed n. See argument 33–SCLD for further details.

⟺35 – WORK [<real> WORK(LWORK)]

The singly subscripted array used for <real> work space, and an array in which various computed values are returned. The smallest acceptable dimension of WORK is given below in the definition of argument 36–LWORK. The work area does not need to be initialized by the user unless the user wishes to initialize Δ, which is stored in the first $n \times m$ locations of WORK. An easy way to access these values, either for initialization, as is necessary when the fourth digit of argument 20–JOB is nonzero and the fit is by orthogonal distance regression, or for analysis upon return from ODRPACK, is to include in the user's program the declaration statements

```
<real> DELTA(<N>,<M>)
EQUIVALENCE (WORK(1), DELTA(1,1))
```

where <N> indicates that the first dimension of the array DELTA must be exactly the number of observations, $\mathrm{N} = n$; and <M> indicates that the second dimension of the array DELTA must be exactly the number of columns, $\mathrm{M} = m$, of the explanatory variable X. This allows the error associated with observation X(I,J) to be accessed as DELTA(I,J) rather than as WORK(I+(J-1)*N). The values in DELTA will be over written by the final estimates of the errors in the explanatory variable when this equivalencing method is used. Other values returned in array WORK may also be of interest and can be accessed as described in §5.A.

N.B., if the fit is a "restart," i.e., if the fifth digit of argument 20–JOB is nonzero, then all elements of vector WORK, including the values of DELTA, must be exactly as returned from a previous call to ODRPACK.

$\Longrightarrow$36 - LWORK [INTEGER LWORK]

The length of array WORK.

For orthogonal distance regression LWORK must equal or exceed

$18+11p+p^2+m+m^2+4nq+6nm+2nqp+2nqm+q^2+5q+q(p+m)+(\texttt{LDWE*LD2WE})q.$

For ordinary least squares LWORK must equal or exceed

$18+11p+p^2+m+m^2+4nq+2nm+2nqp+5q+q(p+m)+(\texttt{LDWE*LD2WE})q.$

$\Longleftarrow$37 - IWORK [INTEGER IWORK(LIWORK)]

The singly subscripted array used for INTEGER work space, and an array in which various computed values are returned. The smallest acceptable dimension of IWORK is given below in the definition of argument 38–LIWORK.

Certain values returned in array IWORK are of general interest and can be accessed as described below in §5.B. In particular, the results of the derivative checking procedure are encoded in IWORK. These results may be especially useful if ODRPACK's error reports have been suppressed. (See argument 27–LUNERR.)

N.B., if the fit is a "restart," i.e., if the fifth digit of argument 20–JOB is nonzero, then all elements of vector IWORK must be exactly as returned from a previous call to ODRPACK.

$\Longrightarrow$38 - LIWORK [INTEGER LIWORK]

The length of array IWORK.

For both orthogonal distance regression and ordinary least squares LIWORK must equal or exceed

$20+p+q(p+m).$

$\Longleftarrow$39 - INFO [INTEGER INFO]

The variable used to indicate why the computations stopped.

INFO	Stopping Condition
$= 1$	sum of squares convergence
$= 2$	parameter convergence
$= 3$	both sum of squares and parameter convergence
$= 4$	iteration limit reached
≥ 5	questionable results or fatal errors detected

When `INFO` ≥ 5 the questionable results or fatal errors detected by ODRPACK are reported in the messages generated on the logical units specified by arguments 27–`LUNERR` and 28–`LUNRPT`. In this case, `INFO` is a 5 digit `INTEGER` with decimal expansion $\mathcal{I}_5\mathcal{I}_4\mathcal{I}_3\mathcal{I}_2\mathcal{I}_1$, where $\mathcal{I}_5 = 0$ indicates that questionable conditions were found, and $\mathcal{I}_5 \geq 1$ indicates that fatal errors were detected. The nonzero values of $\mathcal{I}_5$, $\mathcal{I}_4$, $\mathcal{I}_3$, $\mathcal{I}_2$ and $\mathcal{I}_1$ are used to identify what conditions were detected at the time the program stopped.

Questionable Results:		
$\mathcal{I}_5 = 0$ with	$\mathcal{I}_4 \neq 0$	derivatives possibly not correct (see §4.A)
	$\mathcal{I}_3 \neq 0$	ISTOP $\neq 0$ at last call to `FCN` (see argument 1-`FCN`)
	$\mathcal{I}_2 \neq 0$	problem is not full rank at solution
	$\mathcal{I}_1 = 1$	sum of squares convergence
	$= 2$	parameter convergence
	$= 3$	sum of squares and parameter convergence
	$= 4$	iteration limit reached
Falal Errors:		
$\mathcal{I}_5 = 1$ with	$\mathcal{I}_4 \neq 0$	`N` < 1
	$\mathcal{I}_3 \neq 0$	`M` < 1
	$\mathcal{I}_2 \neq 0$	`NP` < 1 or `NP` $>$ `N`
	$\mathcal{I}_1 \neq 0$	`NQ` < 1
$\mathcal{I}_5 = 2$ with	$\mathcal{I}_4 \neq 0$	`LDY` and/or `LDX` incorrect
	$\mathcal{I}_3 \neq 0$	`LDWE`, `LD2WE`, `LDWD` and/or `LD2WD` incorrect
	$\mathcal{I}_2 \neq 0$	`LDIFX`, `LDSTPD` and/or `LDSCLD` incorrect
	$\mathcal{I}_1 \neq 0$	`LWORK` and/or `LIWORK` too small
$\mathcal{I}_5 = 3$ with	$\mathcal{I}_4 \neq 0$	`STPB` and/or `STPD` incorrect
	$\mathcal{I}_3 \neq 0$	`SCLB` and/or `SCLD` incorrect
	$\mathcal{I}_2 \neq 0$	`WE` incorrect
	$\mathcal{I}_1 \neq 0$	`WD` incorrect
$\mathcal{I}_5 = 4$		error in derivatives
$\mathcal{I}_5 = 5$		`ISTOP` $\neq 0$ at last call to `FCN` (see argument 1-`FCN`)
$\mathcal{I}_5 = 6$		numerical error detected

Note that INFO = 60000 indicates an error possibly caused by incorrectly specified user input to ODRPACK, and more commonly by a poor choice of scale or weights, or a discontinuity in the derivatives.

2.C. Examples

The following sample programs invoke DODR and DODRC to solve the examples of explicit, implicit and multiresponse problems shown in §1.A.

The first program invokes DODRC with user supplied derivatives, the second program invokes DODR with the derivatives approximated by ODRPACK using forward finite differences, and the third invokes DODRC with central finite difference derivatives. The use of forward or central difference derivatives generally causes very little change in the results from those obtained using analytic derivatives. (See §4.A.)

Users are encouraged to use these examples, modified as necessary, to form their own ODRPACK drivers. Single precision sample programs can be easily generated from these two programs by changing all DOUBLE PRECISION variables to REAL, and substituting SODR for DODR and SODRC for DODRC. Note especially that by using MAXN, MAXM, MAXNP and MAXNQ to specify the largest problem the program can solve without modification, and by specifying LWORK and LIWORK exactly as shown, the user greatly reduces the number of changes that must be made to the program in order to solve a larger problem.

2.C.i. Example Problem for an Explicit Model

The following sample program invokes DODRC to solve example 3.2.2 on pages 230-238 of [Fuller, 1987]. The data (x_i, y_i) are the percent saturation of nitrogen gas in a brine solution forced into the pores of sandstone, and the observed compressional wave velocity of ultrasonic signals propagated through the sandstone, respectively. These data (listed in §2.C.i.b) are modeled by the explicit function

$$y_i \approx f(x_i;\beta) \equiv \beta_1 + \beta_2[e^{\beta_3 x_i} - 1]^2, \quad i = 1,\ldots,n.$$

The starting values for the model parameters are

$$\beta \;=\; (1500.0, -50.0, -0.1)^{\mathrm{T}}$$

and Δ is initialized to zero.

Fuller notes that it is reasonable to believe that the saturation measurements of 0% and 100% are more precise than the other saturation measurements. We have thus "fixed" x_1, x_2 and x_{12} at their original values. As a consequence, $\delta_1^* = \delta_2^* = \delta_{12}^* = 0$ at the solution. We assume the remaining observed data are all of equal precision, and thus set $w_\epsilon = w_\delta = 1$ using ODRPACK's structured argument feature. The remaining arguments are set to their default values. (See §1.G.)

2.C.i.a. User Supplied Code

```
      PROGRAM sample

c  ODRPACK Argument Definitions
c      ==> fcn      name of the user supplied function subroutine
c      ==> n        number of observations
c      ==> m        columns of data in the explanatory variable
c      ==> np       number of parameters
c      ==> nq       number of responses per observation
c     <==> beta     function parameters
c      ==> y        response variable
c      ==> ldy      leading dimension of array y
c      ==> x        explanatory variable
c      ==> ldx      leading dimension of array x
c      ==> we       "epsilon" weights
c      ==> ldwe     leading dimension of array we
c      ==> ld2we    second dimension of array we
c      ==> wd       "delta" weights
c      ==> ldwd     leading dimension of array wd
c      ==> ld2wd    second dimension of array wd
c      ==> ifixb    indicators for "fixing" parameters (beta)
c      ==> ifixx    indicators for "fixing" explanatory variable (x)
c      ==> ldifx    leading dimension of array ifixx
c      ==> job      task to be performed
c      ==> ndigit   good digits in subroutine function results
c      ==> taufac   trust region initialization factor
c      ==> sstol    sum of squares convergence criterion
c      ==> partol   parameter convergence criterion
c      ==> maxit    maximum number of iterations
c      ==> iprint   print control
c      ==> lunerr   logical unit for error reports
c      ==> lunrpt   logical unit for computation reports
c      ==> stpb     step sizes for finite difference derivatives wrt beta
c      ==> stpd     step sizes for finite difference derivatives wrt delta
c      ==> ldstpd   leading dimension of array stpd
c      ==> sclb     scale values for parameters beta
c      ==> scld     scale values for errors delta in explanatory variable
c      ==> ldscld   leading dimension of array scld
c     <==> work     DOUBLE PRECISION work vector
c      ==> lwork    dimension of vector work
c     <==  iwork    INTEGER work vector
```

```
c      ==> liwork   dimension of vector iwork
c     <==  info     stopping condition

c  Parameters specifying maximum problem sizes handled by this driver
c     maxn          maximum number of observations
c     maxm          maximum number of columns in explanatory variable
c     maxnp         maximum number of function parameters
c     maxnq         maximum number of responses per observation

c  Parameter Declarations and Specifications
      INTEGER    ldifx,ldscld,ldstpd,ldwd,ldwe,ldx,ldy,ld2wd,ld2we,
     +           liwork,lwork,maxm,maxn,maxnp,maxnq
      PARAMETER (maxm=5,maxn=25,maxnp=5,maxnq=1,
     +           ldy=maxn,ldx=maxn,
     +           ldwe=1,ld2we=1,ldwd=1,ld2wd=1,
     +           ldifx=maxn,ldstpd=1,ldscld=1,
     +           lwork=18 + 11*maxnp + maxnp**2 + maxm + maxm**2 +
     +                 4*maxn*maxnq + 6*maxn*maxm + 2*maxn*maxnq*maxnp +
     +                 2*maxn*maxnq*maxm + maxnq**2 +
     +                 5*maxnq + maxnq*(maxnp+maxm) + ldwe*ld2we*maxnq,
     +           liwork=20+maxnp+maxnq*(maxnp+maxm))

c  Variable Declarations
      INTEGER          i,info,iprint,j,job,l,lunerr,lunrpt,m,maxit,n,
     +                 ndigit,np,nq
      INTEGER          ifixb(maxnp),ifixx(ldifx,maxm),iwork(liwork)
      DOUBLE PRECISION partol,sstol,taufac
      DOUBLE PRECISION beta(maxnp),sclb(maxnp),scld(ldscld,maxm),
     +                 stpb(maxnp),stpd(ldstpd,maxm),
     +                 wd(ldwd,ld2wd,maxm),we(ldwe,ld2we,maxnq),
     +                 work(lwork),x(ldx,maxm),y(ldy,maxnq)
      EXTERNAL         fcn


c  Specify default values for dodrc arguments
      we(1,1,1)  = -1.0d0
      wd(1,1,1)  = -1.0d0
      ifixb(1)   = -1
      ifixx(1,1) = -1
      job        = -1
      ndigit     = -1
      taufac     = -1.0d0
      sstol      = -1.0d0
      partol     = -1.0d0
      maxit      = -1
      iprint     = -1
      lunerr     = -1
      lunrpt     = -1
      stpb(1)    = -1.0d0
```

```
      stpd(1,1)  = -1.0d0
      sclb(1)    = -1.0d0
      scld(1,1)  = -1.0d0

c  Set up ODRPACK report files
      lunerr  =   9
      lunrpt  =   9
      OPEN (unit=9,file='report1')

c  Read problem data, and set nondefault value for argument ifixx
      OPEN (unit=5,file='data1')
      READ (5,FMT=*) n,m,np,nq
      READ (5,FMT=*) (beta(i),i=1,np)
      DO 10 i=1,n
         READ (5,FMT=*) (x(i,j),j=1,m),(y(i,l),l=1,nq)
         if (x(i,1).eq.0.0d0 .or. x(i,1).eq.100.0d0) then
            ifixx(i,1) = 0
         else
            ifixx(i,1) = 1
         end if
   10 CONTINUE

c  Specify task: explicit orthogonal distance regression
c                with user supplied derivatives (checked)
c                covariance matrix constructed with recomputed derivatives
c                delta initialized to zero
c                not a restart
c  and indicate short initial report
c               short iteration reports every iteration, and
c               long final report
      job     = 00020
      iprint  = 1112

c  Compute solution
      CALL dodrc(fcn,
     +           n,m,np,nq,
     +           beta,
     +           y,ldy,x,ldx,
     +           we,ldwe,ld2we,wd,ldwd,ld2wd,
     +           ifixb,ifixx,ldifx,
     +           job,ndigit,taufac,
     +           sstol,partol,maxit,
     +           iprint,lunerr,lunrpt,
     +           stpb,stpd,ldstpd,
     +           sclb,scld,ldscld,
     +           work,lwork,iwork,liwork,
     +           info)
      END
```

```
      SUBROUTINE fcn(n,m,np,nq,
     +               ldn,ldm,ldnp,
     +               beta,xplusd,
     +               ifixb,ifixx,ldifx,
     +               ideval,f,fjacb,fjacd,
     +               istop)

c  Subroutine Arguments
c      ==> n        number of observations
c      ==> m        number of columns in explanatory variable
c      ==> np       number of parameters
c      ==> nq       number of responses per observation
c      ==> ldn      leading dimension declarator equal or exceeding n
c      ==> ldm      leading dimension declarator equal or exceeding m
c      ==> ldnp     leading dimension declarator equal or exceeding np
c      ==> beta     current values of parameters
c      ==> xplusd   current value of explanatory variable, i.e., x + delta
c      ==> ifixb    indicators for "fixing" parameters (beta)
c      ==> ifixx    indicators for "fixing" explanatory variable (x)
c      ==> ldifx    leading dimension of array ifixx
c      ==> ideval   indicator for selecting computation to be performed
c     <==  f        predicted function values
c     <==  fjacb    Jacobian with respect to beta
c     <==  fjacd    Jacobian with respect to errors delta
c     <==  istop    stopping condition, where
c                     0 means current beta and x+delta were
c                       acceptable and values were computed successfully
c                     1 means current beta and x+delta are
c                       not acceptable;  ODRPACK should select values
c                       closer to most recently used values if possible
c                    -1 means current beta and x+delta are
c                       not acceptable;  ODRPACK should stop

c  Input arguments, not to be changed by this routine:
      INTEGER          i,ideval,istop,l,ldifx,ldm,ldn,ldnp,m,n,np,nq
      DOUBLE PRECISION beta(np),xplusd(ldn,m)
      INTEGER          ifixb(np),ifixx(ldifx,m)
c  Output arguments:
      DOUBLE PRECISION f(ldn,nq),fjacb(ldn,ldnp,nq),fjacd(ldn,ldm,nq)
c  Local variables
      INTRINSIC        exp


c  Check for unacceptable values for this problem
      IF (beta(1) .LT. 0.0d0) THEN
         istop = 1
         return
      ELSE
```

```
         istop = 0
      END IF

c  Compute predicted values
      IF (MOD(ideval,10).GE.1) THEN
         DO 110 l = 1,nq
            DO 100 i = 1,n
               f(i,l) = beta(1) +
     +                  beta(2)*(exp(beta(3)*xplusd(i,1)) - 1.0d0)**2
  100       CONTINUE
  110    CONTINUE
      END IF

c  Compute derivatives with respect to beta
      IF (MOD(ideval/10,10).GE.1) THEN
         DO 210 l = 1,nq
            DO 200 i = 1,n
               fjacb(i,1,l) = 1.0d0
               fjacb(i,2,l) = (exp(beta(3)*xplusd(i,1)) - 1.0d0)**2
               fjacb(i,3,l) = beta(2)*2*
     +                        (exp(beta(3)*xplusd(i,1)) - 1.0d0)*
     +                        exp(beta(3)*xplusd(i,1))*xplusd(i,1)
  200       CONTINUE
  210    CONTINUE
      END IF

c  Compute derivatives with respect to delta
      IF (MOD(ideval/100,10).GE.1) THEN
         DO 310 l = 1,nq
            DO 300 i = 1,n
               fjacd(i,1,l) = beta(2)*2*
     +                        (exp(beta(3)*xplusd(i,1)) - 1.0d0)*
     +                        exp(beta(3)*xplusd(i,1))*beta(3)
  300       CONTINUE
  310    CONTINUE
      END IF

      RETURN
      END
```

2.C.i.b. User Supplied Data (file `data1`)

```
    12    1    3    1
 1500.0  -50.0  -0.1
     0.0   1265.0
     0.0   1263.6
     5.0   1258.0
     7.0   1254.0
     7.5   1253.0
    10.0   1249.8
    16.0   1237.0
    26.0   1218.0
    30.0   1220.6
    34.0   1213.8
    34.5   1215.5
   100.0   1212.0
```

2.C.i.c. Report Generated by ODRPACK (file report1)

```
*******************************************************
* odrpack version 2.01 of 06-19-92 (double precision) *
*******************************************************



*** derivative checking report for fit by method of odr ***


    for response  1 of observation     3

                           user
                       supplied     relative    derivative
       derivative wrt      value   difference    assessment

            beta(  1)    1.00D+00     0.00D+00    verified
            beta(  2)    1.55D-01     1.66D-06    verified
            beta(  3)    1.19D+02     2.94D-06    verified
          delta( 3, 1)  -2.39D+00     2.96D-06    verified

    number of reliable digits in function results         16
       (estimated by odrpack)

    number of digits of agreement required between
    user supplied and finite difference derivative for
    user supplied derivative to be considered verified     4

    row number at which derivatives were checked           3

      -values of the explanatory variables at this row

         x( 3, 1)   5.00000000D+00
*******************************************************
* odrpack version 2.01 of 06-19-92 (double precision) *
*******************************************************


*** initial summary for fit by method of odr ***

--- problem size:
           n =    12          (number with nonzero weight =    12)
          nq =     1
           m =     1
          np =     3          (number unfixed =     3)

--- control values:
         job = 00020
```

```
                 = abcde, where
                          a=0 ==> fit is not a restart.
                          b=0 ==> deltas are initialized to zero.
                          c=0 ==> covariance matrix will be computed using
                                  derivatives re-evaluated at the solution.
                          d=2 ==> derivatives are supplied by user.
                                  derivatives were checked.
                                  results appear correct.
                          e=0 ==> method is explicit odr.
          ndigit =    16          (estimated by odrpack)
          taufac =     1.00D+00

    --- stopping criteria:
           sstol =     1.49D-08   (sum of squares stopping tolerance)
          partol =     3.67D-11   (parameter stopping tolerance)
           maxit =    50          (maximum number of iterations)

    --- initial weighted sum of squares        =                   6.63720354D+05
            sum of squared weighted deltas     =   0.00000000D+00
            sum of squared weighted epsilons   =   6.63720354D+05

    *** iteration reports for fit by method of odr ***


            cum.                 act. rel.   pred. rel.
     it.  no. fn     weighted   sum-of-sqs   sum-of-sqs              g-n
    num.   evals   sum-of-sqs    reduction    reduction  tau/pnorm  step
    ----  ------  -----------  -----------  -----------  ---------  ----

       1      19  2.51166D+01   9.9996D-01   9.9997D-01  3.499D-01   yes
       2      20  2.14730D+01   1.4507D-01   1.5249D-01  6.900D-02   yes
       3      21  2.14463D+01   1.2418D-03   1.4631D-03  8.573D-03   yes
       4      22  2.14455D+01   3.7676D-05   4.7131D-05  2.016D-03   yes
       5      23  2.14455D+01   1.5116D-06   1.8950D-06  3.927D-04   yes
       6      24  2.14455D+01   6.2002D-08   7.7723D-08  8.017D-05   yes
       7      25  2.14455D+01   2.5359D-09   3.1790D-09  1.617D-05   yes

    *** final summary for fit by method of odr ***

    --- stopping conditions:
            info =     1 ==> sum of squares convergence.
           niter =     7          (number of iterations)
            nfev =    25          (number of function evaluations)
            njev =     8          (number of jacobian evaluations)
           irank =     0          (rank deficiency)
           rcond =     6.12D-03   (inverse condition number)
           istop =     0          (returned by user from subroutine fcn)

    --- final weighted sums of squares       =                    2.14455017D+01
```

```
        sum of squared weighted deltas   =     7.78974669D+00
        sum of squared weighted epsilons =     1.36557550D+01

--- residual standard deviation          =                       1.54364294D+00
        degrees of freedom               =     9

--- estimated beta(j), j = 1, ..., np:

                    beta      s.d. beta    ---- 95%  confidence interval ----

      1   1.26465481D+03     1.0349D+00     1.26231139D+03 to  1.26699822D+03
      2  -5.40184100D+01     1.5840D+00    -5.76050942D+01 to -5.04317257D+01
      3  -8.78497122D-02     6.3322D-03    -1.02187862D-01 to -7.35115621D-02

--- estimated epsilon(i) and delta(i,*), i = 1, ..., n:

        i    epsilon(i,1)      delta(i,1)

        1 -3.45194935D-01  0.00000000D+00
        2  1.05480506D+00  0.00000000D+00
        3 -3.00719286D-02 -6.50838155D-02
        4 -1.13916405D-01 -2.67201445D-01
        5 -1.40250730D-01 -3.31357554D-01
        6 -5.53155555D-01 -1.30641313D+00
        7 -6.99564762D-01 -1.32525687D+00
        8  1.88412530D+00  1.45885497D+00
        9 -1.70916306D+00 -1.18803577D+00
       10  1.80916198D+00  7.71243449D-01
       11  1.90299896D-01  8.24139253D-02
       12 -1.34707485D+00  0.00000000D+00
```

2.C.ii. Example Problem for an Implicit Model

This sample program invokes DODR to solve the implicit problem shown in example 3.2.4 on page 244 of [Fuller, 1987]. In this example, the data (listed in §2.C.ii.b) are observations digitized from the x-ray image of a hip prosthesis, where the variables $x_i = (v_i, h_i)$, $i = 1, \ldots, n$, are the vertical and horizontal distances from the origin, respectively, and the implicit model is that of the ellipse

$$f_i(x_i;\beta) \equiv \beta_3(v_i-\beta_1)^2 + 2\beta_4(v_i-\beta_1)(h_i-\beta_2) + \beta_5(h_i-\beta_2)^2 = 0$$

for $i = 1, \ldots, n$. The starting values for the model parameters are

$$\beta = (-1.0, -3.0, 0.09, 0.02, 0.08)^{\mathrm{T}}$$

and Δ is initialized to zero. Since the observed data are all of equal precision, we set $w_\delta = 1$ using ODRPACK's structured argument feature. The remaining arguments are set to their default values. (See §1.G.)

2.C.ii.a. User Supplied Code

```
          PROGRAM sample

    c   ODRPACK Argument Definitions
    c       ==> fcn      name of the user supplied function subroutine
    c       ==> n        number of observations
    c       ==> m        columns of data in the explanatory variable
    c       ==> np       number of parameters
    c       ==> nq       number of responses per observation
    c      <==> beta     function parameters
    c       ==> y        response variable (unused when model is implicit)
    c       ==> ldy      leading dimension of array y
    c       ==> x        explanatory variable
    c       ==> ldx      leading dimension of array x
    c       ==> we       initial penalty parameter for implicit model
    c       ==> ldwe     leading dimension of array we
    c       ==> ld2we    second dimension of array we
    c       ==> wd       "delta" weights
    c       ==> ldwd     leading dimension of array wd
    c       ==> ld2wd    second dimension of array wd
    c       ==> job      task to be performed
    c       ==> iprint   print control
    c       ==> lunerr   logical unit for error reports
    c       ==> lunrpt   logical unit for computation reports
    c      <==> work     DOUBLE PRECISION work vector
```

```
c      ==> lwork    dimension of vector work
c     <==  iwork    INTEGER work vector
c      ==> liwork   dimension of vector iwork
c     <==  info     stopping condition

c  Parameters specifying maximum problem sizes handled by this driver
c     maxn          maximum number of observations
c     maxm          maximum number of columns in explanatory variable
c     maxnp         maximum number of function parameters
c     maxnq         maximum number of responses per observation

c  Parameter Declarations and Specifications
      INTEGER    ldwd,ldwe,ldx,ldy,ld2wd,ld2we,
     +           liwork,lwork,maxm,maxn,maxnp,maxnq
      PARAMETER (maxm=5,maxn=25,maxnp=5,maxnq=2,
     +           ldy=maxn,ldx=maxn,
     +           ldwe=1,ld2we=1,ldwd=1,ld2wd=1,
     +           lwork=18 + 11*maxnp + maxnp**2 + maxm + maxm**2 +
     +                 4*maxn*maxnq + 6*maxn*maxm + 2*maxn*maxnq*maxnp +
     +                 2*maxn*maxnq*maxm + maxnq**2 +
     +                 5*maxnq + maxnq*(maxnp+maxm) + ldwe*ld2we*maxnq,
     +           liwork=20+maxnp+maxnq*(maxnp+maxm))

c  Variable Declarations
      INTEGER          i,info,iprint,j,job,lunerr,lunrpt,m,n,np,nq
      INTEGER          iwork(liwork)
      DOUBLE PRECISION beta(maxnp),
     +                 wd(ldwd,ld2wd,maxm),we(ldwe,ld2we,maxnq),
     +                 work(lwork),x(ldx,maxm),y(ldy,maxnq)
      EXTERNAL         fcn


c  Specify default values for dodr arguments
      we(1,1,1)  = -1.0d0
      wd(1,1,1)  = -1.0d0
      job        = -1
      iprint     = -1
      lunerr     = -1
      lunrpt     = -1

c  Set up ODRPACK report files
      lunerr  =   9
      lunrpt  =   9
      OPEN (unit=9,file='report2')

c  Read problem data
      OPEN (unit=5,file='data2')
      READ (5,FMT=*) n,m,np,nq
      READ (5,FMT=*) (beta(i),i=1,np)
```

```
      DO 10 i=1,n
         READ (5,FMT=*) (x(i,j),j=1,m)
   10 CONTINUE

c  Specify task: implicit orthogonal distance regression
c                with forward finite difference derivatives
c                covariance matrix constructed with recomputed derivatives
c                delta initialized to zero
c                not a restart
      job     = 00001

c  Compute solution
      CALL dodr(fcn,
     +           n,m,np,nq,
     +           beta,
     +           y,ldy,x,ldx,
     +           we,ldwe,ld2we,wd,ldwd,ld2wd,
     +           job,
     +           iprint,lunerr,lunrpt,
     +           work,lwork,iwork,liwork,
     +           info)
      END


      SUBROUTINE fcn(n,m,np,nq,
     +               ldn,ldm,ldnp,
     +               beta,xplusd,
     +               ifixb,ifixx,ldifx,
     +               ideval,f,fjacb,fjacd,
     +               istop)

c  Subroutine Arguments
c      ==> n        number of observations
c      ==> m        number of columns in explanatory variable
c      ==> np       number of parameters
c      ==> nq       number of responses per observation
c      ==> ldn      leading dimension declarator equal or exceeding n
c      ==> ldm      leading dimension declarator equal or exceeding m
c      ==> ldnp     leading dimension declarator equal or exceeding np
c      ==> beta     current values of parameters
c      ==> xplusd   current value of explanatory variable, i.e., x + delta
c      ==> ifixb    indicators for "fixing" parameters (beta)
c      ==> ifixx    indicators for "fixing" explanatory variable (x)
c      ==> ldifx    leading dimension of array ifixx
c      ==> ideval   indicator for selecting computation to be performed
c     <==  f        predicted function values
c     <==  fjacb    Jacobian with respect to beta
c     <==  fjacd    Jacobian with respect to errors delta
c     <==  istop    stopping condition, where
```

```
c                     0 means current beta and x+delta were
c                       acceptable and values were computed successfully
c                     1 means current beta and x+delta are
c                       not acceptable;  ODRPACK should select values
c                       closer to most recently used values if possible
c                    -1 means current beta and x+delta are
c                       not acceptable;  ODRPACK should stop

c  Input arguments, not to be changed by this routine:
      INTEGER          i,ideval,istop,l,ldifx,ldm,ldn,ldnp,m,n,np,nq
      DOUBLE PRECISION beta(np),xplusd(ldn,m)
      INTEGER          ifixb(np),ifixx(ldifx,m)
c  Output arguments:
      DOUBLE PRECISION f(ldn,nq),fjacb(ldn,ldnp,nq),fjacd(ldn,ldm,nq)


c  Check for unacceptable values for this problem
      IF (beta(1) .GT. 0.0d0) THEN
         istop = 1
         return
      ELSE
         istop = 0
      END IF

c  Compute predicted values
      IF (MOD(ideval,10).GE.1) THEN
         DO 110 l = 1,nq
            DO 100 i = 1,n
               f(i,l) = beta(3)*(xplusd(i,1)-beta(1))**2 +
     +                  2*beta(4)*(xplusd(i,1)-beta(1))*
     +                            (xplusd(i,2)-beta(2)) +
     +                  beta(5)*(xplusd(i,2)-beta(2))**2 - 1.0d0
  100       CONTINUE
  110    CONTINUE
      END IF

      RETURN
      END
```

2.C.ii.b. User Supplied Data (file `data2`)

```
 20  2  5  1
 -1.0 -3.0 0.09 0.02 0.08
 0.50  -0.12
 1.20  -0.60
 1.60  -1.00
 1.86  -1.40
 2.12  -2.54
 2.36  -3.36
 2.44  -4.00
 2.36  -4.75
 2.06  -5.25
 1.74  -5.64
 1.34  -5.97
 0.90  -6.32
-0.28  -6.44
-0.78  -6.44
-1.36  -6.41
-1.90  -6.25
-2.50  -5.88
-2.88  -5.50
-3.18  -5.24
-3.44  -4.86
```

2.C.ii.c. Report Generated by ODRPACK (file report2)

```
*******************************************************
* odrpack version 2.01 of 06-19-92 (double precision) *
*******************************************************


*** initial summary for fit by method of odr ***

--- problem size:
          n =    20          (number with nonzero weight =    20)
         nq =     1
          m =     2
         np =     5          (number unfixed =     5)

--- control values:
        job = 00001
            = abcde, where
                     a=0 ==> fit is not a restart.
                     b=0 ==> deltas are initialized to zero.
                     c=0 ==> covariance matrix will be computed using
                             derivatives re-evaluated at the solution.
                     d=0 ==> derivatives are estimated by forward differences.
                     e=1 ==> method is implicit odr.
      ndigit =    15         (estimated by odrpack)
      taufac =     1.00D+00

--- stopping criteria:
       sstol =     1.49D-08  (sum of squares stopping tolerance)
      partol =     6.06D-06  (parameter stopping tolerance)
       maxit =   100         (maximum number of iterations)

--- initial sum of squared weighted deltas =                   0.00000000D+00
        initial penalty function value     =   8.39823392D-01
                penalty term               =   8.39823392D-01
                penalty parameter          =   1.0D+01

--- function parameter summary:

      index        beta(k)    fixed          scale    derivative
                                                       step size
        (k)                  (ifixb)        (sclb)        (stpb)

          1 -1.00000000D+00       no  1.00000000D+00   3.16228D-10
          2 -3.00000000D+00       no  3.33333333D-01   3.16228D-10
          3  9.00000000D-02       no  1.11111111D+01   3.16228D-10
          4  2.00000000D-02       no  5.00000000D+01   3.16228D-10
          5  8.00000000D-02       no  1.25000000D+01   3.16228D-10
```

```
--- explanatory variable and delta weight summary:

      index      x(i,j)  delta(i,j)    fixed     scale    weight    derivative
                                                                     step size
      (i,j)                           (ifixx)    (scld)      (wd)       (stpd)

        1,1   5.000D-01   0.000D+00       no  2.00D+00  1.00D+00   3.16228D-10
        n,1  -3.440D+00   0.000D+00       no  2.91D-01  1.00D+00   3.16228D-10

        1,2  -1.200D-01   0.000D+00       no  8.33D+00  1.00D+00   3.16228D-10
        n,2  -4.860D+00   0.000D+00       no  2.06D-01  1.00D+00   3.16228D-10

*** final summary for fit by method of odr ***

--- stopping conditions:
        info =     2 ==> parameter convergence.
       niter =    18          (number of iterations)
        nfev =   217          (number of function evaluations)
       irank =     0          (rank deficiency)
       rcond =     3.18D-02   (inverse condition number)
       istop =     0          (returned by user from subroutine fcn)

--- final sum of squared weighted deltas =                      8.82420346D-02
        final penalty function value     =    8.82445616D-02
              penalty term               =    2.52700897D-06
              penalty parameter          =    1.0D+05

--- residual standard deviation          =                      7.66994283D-02
        degrees of freedom               =   15

--- estimated beta(j), j = 1, ..., np:

                   beta      s.d. beta    ---- 95%  confidence interval ----

      1  -9.99380972D-01    1.1138D-01    -1.23682206D+00 to -7.61939883D-01
      2  -2.93104848D+00    1.0977D-01    -3.16504351D+00 to -2.69705344D+00
      3   8.75730479D-02    4.1061D-03     7.88199915D-02 to  9.63261044D-02
      4   1.62299739D-02    2.7500D-03     1.03676338D-02 to  2.20923140D-02
      5   7.97538008D-02    3.4963D-03     7.23007073D-02 to  8.72068944D-02
```

2.C.iii. Example Problem for an Explicit Model with Multiresponse Data

The problem shown here is an example of multiresponse data that originates because the underlying data are complex. The problem is described in Chapter 4, and on pages 280-281, of [Bates and Watts, 1988]. In this case, the dependent variable is the pair of values representing the real and imaginary parts of complex-valued impedance measurements of a polymer, z_i, $i = 1, \ldots, n$, and the explanatory variable, x_i, $i = 1, \ldots, n$, is the (real-valued) frequency. The data are shown in §2.C.iii.b. The function form is explicit, representing the dielectric constant by the general model proposed by [Havriliak and Negami, 1967]

$$[\Re(z_i), \Im(z_i)] \approx \beta_2 + \frac{\beta_1 - \beta_2}{\left(1 + (\jmath 2\pi x_i e^{-\beta_3})^{\beta_4}\right)^{\beta_5}}$$

for $i = 1, \ldots, n$, where $\jmath = \sqrt{-1}$. For ODRPACK, this must be encoded as two-term multiresponse data with $y_i \in \Re^2$ representing the pair of values $[\Re(z_i), \Im(z_i)]$, $i = 1, \ldots, n$. Havriliak and Negami (1967) show that the real and imaginary components can be written as

$$\begin{aligned} \Re(z_i) &= \beta_2 + (\beta_1 - \beta_2) R^{\beta_5} \cos(\beta_5 \phi) \\ \Im(z_i) &= (\beta_1 - \beta_2) R^{\beta_5} \sin(\beta_5 \phi) \end{aligned}$$

where

$$R^2 = \left[1 + (2\pi x_i/\beta_3)^{\beta_4} \cos(\pi\beta_4/2)\right]^2 + \left[(2\pi x_i/\beta_3)^{\beta_4} \sin(\pi\beta_4/2)\right]^2$$

and

$$\phi = \arctan\left[\frac{(2\pi x_i/\beta_3)^{\beta_4} \sin(\pi\beta_4/2)}{1 + (2\pi x_i/\beta_3)^{\beta_4} \cos(\pi\beta_4/2)}\right] .$$

The estimation procedure described in [Bates and Watts, 1988] for this multiresponse problem is slightly different from that implemented in ODRPACK. In particular, their procedure provides an estimate of w_ϵ, but does not include estimates of Δ. Thus, we would not assume that the results obtained here using ODRPACK will exactly equal those presented by Bates and Watts.

For our example, we have set the starting values for the model parameters to be the final solution shown on page 152 of [Bates and Watts, 1988], i.e.,

$$\beta = (4.398, 2.451, 8.245, 0.487, 0.571)^{\mathrm{T}}$$

and have initialized Δ to the decade corrections described by them.

Bates and Watts assume that there is no error in the first decade of frequency values, and we have done the same, "fixing" these variables at their input values. Bates and

Watts also identify two outliers in the data set, which are eliminated from our analysis by setting the corresponding weights to zero. The remaining weights w_{ϵ_i} in our example are set to an estimate of the 2×2 covariance matrix of the errors in the responses of the dependent variable,

$$w_{\epsilon_i} = (\check{e}_i^{\mathrm{T}} \check{e}_i)^{-1}, \qquad i = 1, \ldots, n$$

where $\check{e}_i$ is the estimate of ϵ_i, $i = 1, \ldots, n$, obtained using the Bates and Watts solution. The weights w_{δ_i} are set to values proportional to the magnitude of the frequencies. The remaining arguments are set to their default values. (See §1.G.)

2.C.iii.a. User Supplied Code

```
      PROGRAM sample

c  ODRPACK Argument Definitions
c      ==> fcn      name of the user supplied function subroutine
c      ==> n        number of observations
c      ==> m        columns of data in the explanatory variable
c      ==> np       number of parameters
c      ==> nq       number of responses per observation
c     <==> beta     function parameters
c      ==> y        response variable
c      ==> ldy      leading dimension of array y
c      ==> x        explanatory variable
c      ==> ldx      leading dimension of array x
c      ==> we       "epsilon" weights
c      ==> ldwe     leading dimension of array we
c      ==> ld2we    second dimension of array we
c      ==> wd       "delta" weights
c      ==> ldwd     leading dimension of array wd
c      ==> ld2wd    second dimension of array wd
c      ==> ifixb    indicators for "fixing" parameters (beta)
c      ==> ifixx    indicators for "fixing" explanatory variable (x)
c      ==> ldifx    leading dimension of array ifixx
c      ==> job      task to be performed
c      ==> ndigit   good digits in subroutine fcn results
c      ==> taufac   trust region initialization factor
c      ==> sstol    sum of squares convergence criterion
c      ==> partol   parameter convergence criterion
c      ==> maxit    maximum number of iterations
c      ==> iprint   print control
c      ==> lunerr   logical unit for error reports
c      ==> lunrpt   logical unit for computation reports
c      ==> stpb     step sizes for finite difference derivatives wrt beta
c      ==> stpd     step sizes for finite difference derivatives wrt delta
c      ==> ldstpd   leading dimension of array stpd
```

```
c      ==> sclb     scale values for parameters beta
c      ==> scld     scale values for errors delta in explanatory variable
c      ==> ldscld   leading dimension of array scld
c     <==> work     DOUBLE PRECISION work vector
c      ==> lwork    dimension of vector work
c     <==  iwork    INTEGER work vector
c      ==> liwork   dimension of vector iwork
c     <==  info     stopping condition

c  Parameters specifying maximum problem sizes handled by this driver
c     maxn          maximum number of observations
c     maxm          maximum number of columns in explanatory variable
c     maxnp         maximum number of function parameters
c     maxnq         maximum number of responses per observation

c  Parameter Declarations and Specifications
      INTEGER    ldifx,ldscld,ldstpd,ldwd,ldwe,ldx,ldy,ld2wd,ld2we,
     +           liwork,lwork,maxm,maxn,maxnp,maxnq
      PARAMETER (maxm=5,maxn=100,maxnp=25,maxnq=5,
     +           ldy=maxn,ldx=maxn,
     +           ldwe=maxn,ld2we=maxnq,ldwd=maxn,ld2wd=1,
     +           ldifx=maxn,ldscld=1,ldstpd=1,
     +           lwork=18 + 11*maxnp + maxnp**2 + maxm + maxm**2 +
     +                 4*maxn*maxnq + 6*maxn*maxm + 2*maxn*maxnq*maxnp +
     +                 2*maxn*maxnq*maxm + maxnq**2 +
     +                 5*maxnq + maxnq*(maxnp+maxm) + ldwe*ld2we*maxnq,
     +           liwork=20+maxnp+maxnq*(maxnp+maxm))

c  Variable Declarations
      INTEGER          i,info,iprint,j,job,l,lunerr,lunrpt,m,maxit,n,
     +                 ndigit,np,nq
      INTEGER          ifixb(maxnp),ifixx(ldifx,maxm),iwork(liwork)
      DOUBLE PRECISION partol,sstol,taufac
      DOUBLE PRECISION beta(maxnp),sclb(maxnp),scld(ldscld,maxm),
     +                 stpb(maxnp),stpd(ldstpd,maxm),
     +                 wd(ldwd,ld2wd,maxm),we(ldwe,ld2we,maxnq),
     +                 work(lwork),x(ldx,maxm),y(ldy,maxnq)
      EXTERNAL         fcn


c  Specify default values for dodrc arguments
      we(1,1,1)  = -1.0d0
      wd(1,1,1)  = -1.0d0
      ifixb(1)   = -1
      ifixx(1,1) = -1
      job        = -1
      ndigit     = -1
      taufac     = -1.0d0
      sstol      = -1.0d0
```

```
      partol     = -1.0d0
      maxit      = -1
      iprint     = -1
      lunerr     = -1
      lunrpt     = -1
      stpb(1)    = -1.0d0
      stpd(1,1)  = -1.0d0
      sclb(1)    = -1.0d0
      scld(1,1)  = -1.0d0

c  Set up ODRPACK report files
      lunerr  =   9
      lunrpt  =   9
      OPEN (unit=9,file='report3')

c  Read problem data
      OPEN (unit=5,file='data3')
      READ (5,FMT=*) n,m,np,nq
      READ (5,FMT=*) (beta(i),i=1,np)
      DO 10 i=1,n
         READ (5,FMT=*) (x(i,j),j=1,m),(y(i,l),l=1,nq)
   10 CONTINUE

c  Specify task as explicit orthogonal distance regression
c                  with central difference derivatives
c                  covariance matrix constructed with recomputed derivatives
c                  delta initialized by user
c                  not a restart
c  and indicate long initial report
c               no iteration reports
c               long final report
      job     = 01010
      iprint  = 2002

c  Initialize delta, and specify first decade of frequencies as fixed
      DO 20 i=1,n
         if (x(i,1).lt.100.0d0) then
            work(i)    = 0.0d0
            ifixx(i,1) = 0
         else if (x(i,1).le.150.0d0) then
            work(i)    = 0.0d0
            ifixx(i,1) = 1
         else if (x(i,1).le.1000.0d0) then
            work(i)    = 25.0d0
            ifixx(i,1) = 1
         else if (x(i,1).le.10000.0d0) then
            work(i)    = 560.0d0
            ifixx(i,1) = 1
         else if (x(i,1).le.100000.0d0) then
```

```
            work(i)    = 9500.0d0
            ifixx(i,1) = 1
         else
            work(i) = 144000.0d0
            ifixx(i,1) = 1
         end if
   20 CONTINUE

c  Set weights
      DO 30 i=1,n
         if (x(i,1).eq.100.0d0 .or. x(i,1).eq.150.0d0) then
            we(i,1,1) = 0.0d0
            we(i,1,2) = 0.0d0
            we(i,2,1) = 0.0d0
            we(i,2,2) = 0.0d0
         else
            we(i,1,1) =   559.6d0
            we(i,1,2) = -1634.0d0
            we(i,2,1) = -1634.0d0
            we(i,2,2) =  8397.0d0
         end if
         wd(i,1,1)    =  (1.0d-4)/(x(i,1)**2)
   30 CONTINUE

c  Compute solution
      CALL dodrc(fcn,
     +           n,m,np,nq,
     +           beta,
     +           y,ldy,x,ldx,
     +           we,ldwe,ld2we,wd,ldwd,ld2wd,
     +           ifixb,ifixx,ldifx,
     +           job,ndigit,taufac,
     +           sstol,partol,maxit,
     +           iprint,lunerr,lunrpt,
     +           stpb,stpd,ldstpd,
     +           sclb,scld,ldscld,
     +           work,lwork,iwork,liwork,
     +           info)
      END


      SUBROUTINE fcn(n,m,np,nq,
     +               ldn,ldm,ldnp,
     +               beta,xplusd,
     +               ifixb,ifixx,ldifx,
     +               ideval,f,fjacb,fjacd,
     +               istop)

c  Subroutine Arguments
```

```
c      ==> n        number of observations
c      ==> m        number of columns in explanatory variable
c      ==> np       number of parameters
c      ==> nq       number of responses per observation
c      ==> ldn      leading dimension declarator equal or exceeding n
c      ==> ldm      leading dimension declarator equal or exceeding m
c      ==> ldnp     leading dimension declarator equal or exceeding np
c      ==> beta     current values of parameters
c      ==> xplusd   current value of explanatory variable, i.e., x + delta
c      ==> ifixb    indicators for "fixing" parameters (beta)
c      ==> ifixx    indicators for "fixing" explanatory variable (x)
c      ==> ldifx    leading dimension of array ifixx
c      ==> ideval   indicator for selecting computation to be performed
c     <==  f        predicted function values
c     <==  fjacb    Jacobian with respect to beta
c     <==  fjacd    Jacobian with respect to errors delta
c     <==  istop    stopping condition, where
c                     0 means current beta and x+delta were
c                       acceptable and values were computed successfully
c                     1 means current beta and x+delta are
c                       not acceptable;  ODRPACK should select values
c                       closer to most recently used values if possible
c                    -1 means current beta and x+delta are
c                       not acceptable;  ODRPACK should stop

c  Input arguments, not to be changed by this routine:
      INTEGER          i,ideval,istop,ldifx,ldm,ldn,ldnp,m,n,np,nq
      DOUBLE PRECISION beta(np),xplusd(ldn,m)
      INTEGER          ifixb(np),ifixx(ldifx,m)
c  Output arguments:
      DOUBLE PRECISION f(ldn,nq),fjacb(ldn,ldnp,nq),fjacd(ldn,ldm,nq)
c  Local variables
      DOUBLE PRECISION freq,pi,omega,ctheta,stheta,theta,phi,r
      INTRINSIC        atan2,exp,sqrt


c  Check for unacceptable values for this problem
      DO 10 i=1,n
         IF (xplusd(i,1).LT.0.0d0) THEN
            istop = 1
            return
         END IF
   10 CONTINUE
      istop = 0

      pi = 3.141592653589793238462643383279d0

      theta = pi*beta(4)*0.5d0
      ctheta = cos(theta)
```

```
      stheta = sin(theta)

c  Compute predicted values
      IF (MOD(ideval,10).GE.1) THEN
         DO 100 i = 1,n
            freq  = xplusd(i,1)
            omega = (2.0d0*pi*freq*exp(-beta(3)))**beta(4)
            phi   = atan2((omega*stheta),(1+omega*ctheta))
            r     = (beta(1)-beta(2)) *
     +              sqrt((1+omega*ctheta)**2+
     +                       (omega*stheta)**2)**(-beta(5))
            f(i,1) = beta(2) + r*cos(beta(5)*phi)
            f(i,2) =           r*sin(beta(5)*phi)
  100    CONTINUE
      END IF

      RETURN
      END
```

2.C.iii.b. User Supplied Data (file data3)

```
  23     1     5    2
4.0 2.0 7.0 0.40 0.50
    30.0  4.220  0.136
    50.0  4.167  0.167
    70.0  4.132  0.188
   100.0  4.038  0.212
   150.0  4.019  0.236
   200.0  3.956  0.257
   300.0  3.884  0.276
   500.0  3.784  0.297
   700.0  3.713  0.309
  1000.0  3.633  0.311
  1500.0  3.540  0.314
  2000.0  3.433  0.311
  3000.0  3.358  0.305
  5000.0  3.258  0.289
  7000.0  3.193  0.277
 10000.0  3.128  0.255
 15000.0  3.059  0.240
 20000.0  2.984  0.218
 30000.0  2.934  0.202
 50000.0  2.876  0.182
 70000.0  2.838  0.168
100000.0  2.798  0.153
150000.0  2.759  0.139
```

2.C.iii.c. Report Generated by ODRPACK (file report3)

```
******************************************************
* odrpack version 2.01 of 06-19-92 (double precision) *
******************************************************


*** initial summary for fit by method of odr ***

--- problem size:
         n =    23          (number with nonzero weight =    21)
        nq =     2
         m =     1
        np =     5          (number unfixed =     5)

--- control values:
       job = 01010
           = abcde, where
                    a=0 ==> fit is not a restart.
                    b=1 ==> deltas are initialized by user.
                    c=0 ==> covariance matrix will be computed using
                            derivatives re-evaluated at the solution.
                    d=1 ==> derivatives are estimated by central differences.
                    e=0 ==> method is explicit odr.
    ndigit =    15          (estimated by odrpack)
    taufac =     1.00D+00

--- stopping criteria:
     sstol =     1.49D-08   (sum of squares stopping tolerance)
    partol =     3.67D-11   (parameter stopping tolerance)
     maxit =    50          (maximum number of iterations)

--- initial weighted sum of squares        =                 1.71064070D+03
        sum of squared weighted deltas     =   2.01382943D-04
        sum of squared weighted epsilons   =   1.71064050D+03

--- function parameter summary:

      index         beta(k)    fixed           scale    derivative
                                                         step size
        (k)                   (ifixb)          (sclb)        (stpb)

          1  4.00000000D+00       no  2.50000000D-01   1.00000D-05
          2  2.00000000D+00       no  5.00000000D-01   1.00000D-05
          3  7.00000000D+00       no  1.42857143D-01   1.00000D-05
          4  4.00000000D-01       no  2.50000000D+00   1.00000D-05
          5  5.00000000D-01       no  2.00000000D+00   1.00000D-05

--- explanatory variable and delta weight summary:
```

```
      index      x(i,j)  delta(i,j)    fixed     scale    weight    derivative
                                                                     step size
      (i,j)                           (ifixx)    (scld)      (wd)       (stpd)

        1,1   3.000D+01   0.000D+00      yes  3.33D-02  1.11D-07   1.00000D-05
        n,1   1.500D+05   1.440D+05       no  6.67D-06  4.44D-15   1.00000D-05

--- response variable and epsilon error weight summary:

      index      y(i,l)      weight
      (i,l)                    (we)

        1,1   4.220D+00   5.596D+02
        n,1   2.759D+00   5.596D+02

        1,2   1.360D-01   8.397D+03
        n,2   1.390D-01   8.397D+03

*** final summary for fit by method of odr ***

--- stopping conditions:
        info =     1 ==> sum of squares convergence.
       niter =     8          (number of iterations)
        nfev =   121          (number of function evaluations)
       irank =     0          (rank deficiency)
       rcond =     8.15D-03   (inverse condition number)
       istop =     0          (returned by user from subroutine fcn)

--- final weighted sums of squares       =                    4.20538922D-01
        sum of squared weighted deltas   =    5.54021897D-04
        sum of squared weighted epsilons =    4.19984900D-01

--- residual standard deviation          =                    1.62122431D-01
        degrees of freedom               =   16

--- estimated beta(j), j = 1, ..., np:

                   beta      s.d. beta    ---- 95%  confidence interval ----

      1   4.37998803D+00     1.3063D-02     4.35229388D+00 to  4.40768218D+00
      2   2.43330576D+00     1.3050D-02     2.40563820D+00 to  2.46097332D+00
      3   8.00288459D+00     1.1671D-01     7.75544803D+00 to  8.25032115D+00
      4   5.10114716D-01     1.3264D-02     4.81992824D-01 to  5.38236609D-01
      5   5.17390233D-01     2.8853D-02     4.56218498D-01 to  5.78561968D-01

--- estimated epsilon(i) and delta(i,*), i = 1, ..., n:

        i    epsilon(i,1)    epsilon(i,2)      delta(i,1)
```

```
 1 -7.38558795D-03  1.25939187D-03  0.00000000D+00
 2 -1.05614733D-03 -1.22846292D-03  0.00000000D+00
 3 -2.70863920D-03 -2.14347329D-03  0.00000000D+00
 4  4.68593517D-02 -4.25940138D-03  0.00000000D+00
 5  8.08102389D-03 -3.47539194D-03  0.00000000D+00
 6  1.53882522D-03  3.85293713D-04  3.03694400D+01
 7  4.60535703D-03  1.19118896D-03  3.78986750D+01
 8  4.50906164D-03  1.23570892D-03  6.22630487D+01
 9 -1.00621895D-03 -2.91865043D-04  1.11186980D+02
10  1.05810802D-02  3.27284194D-03  1.15709877D+02
11  6.93622739D-03  2.43482106D-03  2.41436591D+02
12  3.95828011D-05  1.75905014D-05  9.61344532D+02
13 -3.77617796D-03 -2.42907814D-03  1.33029845D+03
14 -5.56734978D-04 -1.70123784D-03  2.07511566D+03
15  2.08263807D-03 -2.23723233D-03  2.90289532D+03
16 -7.50689916D-03  2.16462893D-03  5.21815818D+03
17 -1.56731844D-03  2.03367085D-04  7.54564636D+03
18 -5.93223183D-04  2.72069171D-05  1.74201021D+04
19  1.15260099D-04 -2.42126131D-07  2.42745472D+04
20  2.63641111D-04  5.18510319D-06  3.78492052D+04
21 -3.81011180D-04 -1.03963850D-05  5.53493280D+04
22 -3.36822611D-04 -1.26141391D-05  8.75791432D+04
23  2.87173883D-03  1.41199841D-04  1.29496300D+05
```

3. When the Model Is Very Time Consuming

ODRPACK executes user supplied subroutine `FCN` not only to compute the initial sum of the squared errors $S(\beta, \delta)$ and to obtain function and derivative values within its main iterative procedure, but also when setting the default value for the number of good digits in the function results, when performing derivative checking, and when constructing the covariance matrix and standard deviations of the estimated parameters $\hat{\beta}$. When the time required for finding the solution is dominated by the evaluation of `FCN`, the user will want to make judicious use of these options in light of their "cost." Let $\tilde{p}$ be the number of unfixed parameters β, and let $\varphi = 1$ if the fit is by orthogonal distance regression and $\varphi = 0$ if the fit is by ordinary least squares. Then the number of times the function and derivatives are evaluated in each of these instances is summarized as follows:

Computation	Function Evaluations	Derivative Evaluations[1]	Controlling Variable
• Initial $S(\beta, \delta)$:	1	0	—
• Per Iteration:	≥ 1	1	—
• Default number of good digits in function results:	4	0	`NDIGIT`
• Derivative checking:	$\geq q(\tilde{p} + \varphi m)$	1	`JOB`
• Default covariance matrix:	0	1	`JOB`

Users with a very time consuming subroutine `FCN` should also be aware of two of ODRPACK's options that are specifically designed for such problems. The most important of these is the restart facility. The other is the option of constructing the covariance matrix without recomputing the derivative matrices at the solution. This second option is discussed in §2.B.ii, under the description of subroutine argument `JOB`, and also in §4.B. The remainder of this section describes how ODRPACK's restart facility can be used to minimize the risk of losing important results because system imposed time limits are reached before the solution is found.

[1] A forward finite difference approximation to the derivative requires $q(\tilde{p} + \varphi m)$ function evaluations, and a central finite difference approximation requires $2q(\tilde{p} + \varphi m)$ function evaluations. (See §4.A.)

The restart facility enables the user to step through the solution procedure one or more iterations at a time without incurring any additional function or derivative evaluations over what would be required if the procedure were allowed to run to convergence. Figure 3.1 shows an example of how the restart facility can be employed. This example allows up to a total of 30 iterations, and writes the contents of arrays `BETA`, `WORK` and `IWORK` to a file between every iteration, alternating between two files. This minimizes the chance of losing significant amounts of important information due to system imposed limits: if such a limit is reached before convergence, arrays `BETA`, `WORK` and `IWORK` can be restored using the saved data and the computations restarted. In this example, the initial computation report is only generated at the first iteration, while the final computation report is generated after every iteration. The options selected include constructing the covariance matrix using the derivative matrices from the last iteration, and thus no additional calls to subroutine `FCN` are incurred in order to provide the standard deviations of the parameters printed in each of the reports.

Users with very time consuming problems should be aware that, depending on which options are selected, ODRPACK will make at least one and possibly more calls to subroutine `FCN` before attempting to generate any computation reports. Also, on many systems the output generated by a program is not written directly in a file but rather is stored in a "buffer" until the buffer is full, and is only written to the file at that point. If a run is aborted prematurely, either by the user or because a system imposed limit is reached, then the content of these buffers might not be emptied into the files associated with them. Thus, the files associated with the logical units specified by arguments `LUNERR` and `LUNRPT` might not include all information actually generated by ODRPACK at the time it stopped. When the user is not getting the expected reports from ODRPACK, it may be necessary to have ODRPACK generate reports *directly* to "standard output," which is usually not buffered, in order to determine exactly where the computations are stopping. (See §2.B.ii, subroutine arguments `IPRINT`, `LUNERR` and `LUNRPT`.)

Figure 3.1: Using ODRPACK's Restart Facility

```
|
c  set up files to save computations for future restarts
      lun1 = 11
      OPEN (unit=lun1,file='save1.dat')
      lun2 = 12
      OPEN (unit=lun2,file='save2.dat')

c  set the maximum number of iterations for each call to ODRPACK to one
c  so results can be stored between iterations
      maxit = 1

c  set argument appropriately, making sure for first iteration that
c        fit is not a restart and
c        covariance matrix is constructed without recomputing derivatives
      job = 00100

c  set iprint to indicate a long initial report,
c                         a short iteration report, and
c                         a long final report
      iprint = 2112

c  step through up to 30 iterations
      DO 100 niter = 1, 30

         CALL dodrc(fcn, ... ,info)

c  save the contents of beta, work and iwork for future reference
         IF (mod(niter,2).eq.1) THEN
            lun = lun1
         ELSE
            lun = lun2
         END IF
         OPEN  (UNIT=lun)
         WRITE (lun,*) (beta(k),k=1,np)
         WRITE (lun,*) (work(i),i=1,lwork)
         WRITE (lun,*) (iwork(i),i=1,liwork)
         CLOSE (UNIT=lun)

         IF (info.ge.10000 .or. mod(info,10).le.3) then
c  stop because either a fatal error was detected, or the problem converged
            stop
         ELSE
c  set job to indicate the next iteration is a restart and
c  set iprint to suppress the initial report for future iterations.
            job = 10100
            iprint = 0112
         END IF
  100 CONTINUE
c
|
```

4. Computational Details

4.A. Computing the Jacobian Matrices

As was noted in §1.A, the matrices of first partial derivatives, i.e., the Jacobian matrices

$$\begin{aligned} &\tfrac{\partial f_{il}(x_i+\delta_i;\beta)}{\partial \beta_k}, \quad i=1,\ldots,n, \quad k=1,\ldots,p, \quad \& \; l=1,\ldots,q, \\ &\tfrac{\partial f_{il}(x_i+\delta_i;\beta)}{\partial \Delta_{ij}}, \quad i=1,\ldots,n, \quad j=1,\ldots,m, \quad \& \; l=1,\ldots,q, \end{aligned} \tag{4.1}$$

are required at every iteration. These can be provided by the user as described for subroutine argument FCN in §2.B.ii, or can be approximated automatically by ODRPACK. User supplied derivatives are generally either "hand coded" as is done in the example program shown in §2.C.i, or are the product of an "automatic differentiation" tool. ODRPACK's approximations are formed using either forward or central finite differences.

4.A.i. "Hand Coded" Derivatives

Hand coded derivatives are those produced by the user without the aid of a differentiation tool. Because coding errors are a common problem with hand coded derivatives, ODRPACK has an option to check the validity of the user supplied derivative code by comparing its results to finite difference values for the derivative. The derivative checking procedure examines the unfixed variables at only one row of the Jacobian matrix, and is therefore quite efficient. Checking only one row is reasonable for regression models since the same code is frequently used to compute the model function and derivatives for each row, as is the case for each of the examples shown in §2.C.

When the value of the user supplied derivative disagrees with the corresponding finite difference value, the checking procedure attempts to determine whether the disagreement is due to an error in the user's code, or is due to the inaccuracy of the finite difference approximation. The checking procedure generates an error report when one or more of the derivatives are found to be questionable. This information is also returned to the user in subroutine argument IWORK. (See §5.B.)

Questionable derivatives can occur when the derivative is exactly zero or when the numerical derivative used in the checking procedure is believed to be inaccurate because of the properties of the function. Zero valued derivatives are questionable because they could indicate that the initial values of the function parameters β might be hiding an error in the derivative, such as could occur if the initial value of one of the parameters was zero. Users should examine the ODRPACK error reports, or the encoded values in subroutine argument `IWORK` as described in §5.B, to determine the cause of the questionable results, and then examine subroutine `FCN` to insure that there is not an error in the user supplied derivatives that could be adversely affecting the least squares results.

4.A.ii. Automatic Differentiation

Automatic differentiation tools produce code to calculate a function's derivatives directly from the code used to compute the function values. Such tools enable the user to generate the derivatives required by ODRPACK without the tedium and errors associated with hand coded derivatives. An overview of automatic differentiation is presented in [Griewank, 1989], and a survey of automatic differentiation software is provided in [Juedes, 1991]. We have found tools such as ADIFOR [Bischof *et al.*, 1991] and DAPRE [Stephens and Pryce, 1991], which are precompilers that transform a Fortran subroutine that evaluates the function into a Fortran subroutine that evaluates both the function and its derivatives, especially suitable for use with ODRPACK.

Currently, most differentiation tools, including ADIFOR and DAPRE, generate code to evaluate the function and derivative values simultaneously. Least squares procedures, however, need the derivatives only after determining that a satisfactory new point has been found, and this determination requires at least one and possibly more function evaluations. Thus, when using a differentiation tool, one has two choices: either the function and derivatives can be always evaluated together using the code generated by the differentiation tool; or the hand coded function can be evaluated by itself until a satisfactory new point has been found, at which time the derivative code generated by the differentiation tool code can be evaluated. The first choice has the drawback that sometimes the computed derivative values will not be used. If the second option is selected, the function values produced by the differentiation tool generated code are used only for the evaluation of the Jacobian matrices and not within the least squares procedure.

For the differentiation tools and problems examined by the authors, the time required to evaluate the derivative and function together using code generated by automatic differentiation is frequently significantly more than that required to evaluate only the hand coded function. We thus believe that in most cases it will be more cost effective for users to employ the second of the two choices mentioned above. Users must also be aware that ODRPACK never asks that both the function and its derivatives be computed in a single call to subroutine `FCN`, and that it is possible that an invocation of `FCN` for evaluating the derivative will not be *immediately* preceded with a call to `FCN` to evaluate the function for the same parameter values. Thus, if the first of the two options is employed, the user will need to construct a mechanism for saving the computed derivatives until ODRPACK actually requests them, possibly using the Fortran `COMMON` facility, and also a mechanism for determining whether the saved derivative values were in fact those evaluated at the selected parameter values.

4.A.iii. Finite Difference Derivatives

Finite difference derivatives are automatically constructed by ODRPACK to approximate the Jacobian matrices when the user does not supply code within subroutine `FCN` to compute them. Either forward or central finite differences can be employed for the approximation. (See §2.B.ii, subroutine argument JOB.) A central finite difference derivative gives a more accurate approximation than the corresponding forward finite difference derivative, but at the expense of an additional call to subroutine `FCN` for each partial derivative computed. The interested reader is referred to [Dennis and Schnabel, 1983] and [Gill *et al.*, 1981] for a more complete discussion of forward and central finite difference approximations.

4.A.iii.a. Forward Finite Difference Derivatives

The forward finite difference derivative with respect to β for response l of observation i is computed using

$$\frac{f_{il}(x_i + \delta_i;\ \beta + \tilde{h}_{\beta_k} u_k) - f_{il}(x_i + \delta_i; \beta)}{\tilde{h}_{\beta_k}} \ ,$$

$i = 1, \ldots, n$, $k = 1, \ldots, p$, and $l = 1, \ldots, q$, where u_k is the kth unit vector, i.e., the kth column of a $p \times p$ identity matrix, and where $\tilde{h}_{\beta_k}$ is the finite difference step size,

$$\tilde{h}_{\beta_k} = h_{\beta_k} |\beta_k|, \qquad k = 1, \ldots, p, \tag{4.2}$$

with h_{β_k} the *relative* step size for parameter β_k specified by subroutine argument `STPB`. (See §2.B.ii.) The default value for the relative step for a forward finite difference derivative is

$$h = \tfrac{1}{100} 10^{-\psi/2} \ ,$$

where ψ indicates the number of good digits in the results of the user supplied subroutine `FCN`. (See §2.B.ii, subroutine argument `NDIGIT`.) This default value is selected based on empirical evidence that indicates it generally outperforms the commonly recommended value $h = 10^{(-\psi/2)}$. (See, e.g., [Dennis and Schnabel, 1983].) Procedures for selecting near optimal relative step sizes are discussed in [Gill *et al.*, 1981] and [Schnabel, 1982].)

The step $\tilde{h}_{\beta_k}$ must be large enough so that approximately half of the good digits of $f_{il}(x_i + \delta_i;\ \beta + \tilde{h}_{\beta_k} u_k)$ and $f_{il}(x_i + \delta_i; \beta)$ will be the same. The forward finite difference approximation to the derivative can then be expected to have roughly half the number of good digits as are in the computed value of $f_{il}(x_i + \delta_i; \beta)$. When the computation of $f_{il}(x_i + \delta_i; \beta)$ has sufficient precision, then forward finite difference derivatives will cause very little change in the results from those that would be obtained using hand coded or automatic differentiation derivatives.

The forward finite difference derivatives with respect to Δ are formed analogously to those with respect to β. (See §2.B.ii, subroutine argument STPD, for specification of the relative stepsize.)

4.A.iii.b. Central Finite Difference Derivatives

When the user suspects that the forward finite difference approximation will not provide sufficient precision, then a central finite difference approximation can be used. (See §2.B.ii, subroutine argument JOB.) The central finite difference approximation to the partial derivative with respect to β_k for response l of observation i is given by

$$\frac{f_{il}(x_i + \delta_i; \beta + \tilde{h}_k u_k) - f_{il}(x_i + \delta_i; \beta - \tilde{h}_k u_k)}{\tilde{h}_k} ,$$

$i = 1, \ldots, n$, $k = 1, \ldots, p$, and $l = 1, \ldots, q$. The step $\tilde{h}_k$ is formed as in (4.2), where the default value for the relative step for a central finite difference derivative is

$$h = 10^{-\psi/3} .$$

The central finite difference derivatives with respect to Δ are again formed analogously.

4.B. Covariance Matrix

The linearized confidence regions and intervals for the unknowns β and Δ estimated by orthogonal distance regression are the same as the linearized regions and intervals that would be obtained if the orthogonal distance regression problem were solved as a $p+nm$ parameter nonlinear ordinary least squares problem. If we express the orthogonal distance regression problem defined by (1.8) or (1.9) as such a nonlinear ordinary least squares problem with $nq + nm$ observations and $p + nm$ unknowns, and we designate the unknowns of this ordinary least squares problem as $\eta^{\mathrm{T}} = (\beta^{\mathrm{T}}, \delta_1^{\mathrm{T}}, \ldots, \delta_n^{\mathrm{T}})$, then the sum of squares to be minimized is

$$S(\eta) \equiv G(\eta)^{\mathrm{T}} \Omega G(\eta)$$

where $G(\eta)$ is the vector valued function whose ith "element" is defined by

$$g_i(\eta) = \begin{cases} f_i(x_i + \delta_i; \beta) - y_i & i = 1, \ldots, n, \\ \delta_{i-n} & i = n+1, \ldots, 2n, \end{cases}$$

and $\Omega \in \Re^{(nq+nm)\times(nq+nm)}$ is the block diagonal weighting matrix given by

$$\Omega = \begin{bmatrix} w_{\epsilon_1} & & & & & \\ & \ddots & & & & \\ & & w_{\epsilon_n} & & & \\ & & & w_{\delta_1} & & \\ & & & & \ddots & \\ & & & & & w_{\delta_n} \end{bmatrix}.$$

The ordinary least squares representation of (1.8) or (1.9) is thus

$$\min_{\eta} S(\eta) = \min_{\eta} \sum_{i=1}^{2n} g_i(\eta)^{\mathrm{T}} \Omega_{ii} g_i(\eta) \tag{4.3}$$

where Ω_{ii} denotes the (i,i)th "element" of Ω.

Let $G'(\hat{\eta}) \in \Re^{(nq+nm)\times(p+nm)}$ denote the Jacobian matrix with (ν, k)th element equal to $\partial g_\nu(\eta)/\partial \eta_k$ evaluated at $\hat{\eta}$. If we assume that $G'(\hat{\eta})$ and Ω are full rank, so that $[G'(\hat{\eta})^{\mathrm{T}} \Omega G'(\hat{\eta})]$ is nonsingular, then the linearized covariance matrix for the estimators $\hat{\eta}$ is the $(p+nm) \times (p+nm)$ matrix

$$\hat{V} \quad = \quad \hat{\sigma}^2 [G'(\hat{\eta})^{\mathrm{T}} \Omega G'(\hat{\eta})]^{-1},$$

where $\hat{\sigma} = S(\hat{\eta})/\mu = S(\hat{\beta}, \hat{\delta})/\mu$ is the estimated residual variance with μ degrees of freedom. (The degrees of freedom is the number of observations with nonzero weights minus the number of parameters actually being estimated, i.e., $\mu = \tilde{n} - \tilde{p}$.) This covariance matrix $\hat{V}$ can be partitioned

$$\hat{V} = \begin{bmatrix} \hat{V}_\beta & \hat{V}_{\beta\delta} \\ \hat{V}_{\delta\beta} & \hat{V}_\delta \end{bmatrix}$$

where $\hat{V}_\beta \in \Re^{p \times p}$ is the covariance matrix for the estimators $\hat{\beta}$, $\hat{V}_\delta \in \Re^{nm \times nm}$ is the covariance matrix for the estimators $\hat{\Delta}$, and $\hat{V}_{\beta\delta} = \hat{V}_{\delta\beta}^{\mathrm{T}} \in \Re^{p \times nm}$ gives covariances between β and Δ. It is the covariance matrix $\hat{V}_\beta$ of the estimators $\hat{\beta}$ that is automatically provided by ODRPACK. The actual computational technique used by ODRPACK to compute $\hat{V}_\beta$ is described in detail in [Boggs and Rogers, 1990b].

By default, ODRPACK will recompute the Jacobian matrices at the final solution before constructing the covariance matrix. However, ODRPACK also provides the option of constructing the covariance matrix using the Jacobian matrices from the last iteration. (See §2.B.ii, subroutine argument `JOB`.) The option of using the Jacobian matrices from the last iteration to construct the covariance matrix is especially useful when the evaluation of user supplied subroutine `FCN` is very time consuming. Assuming that the algorithm has actually converged, using the Jacobian matrices from the last iteration should give essentially the same covariance matrix as that which would be obtained using the Jacobian matrices recomputed at the solution. Once the user confirms that the solution is satisfactory, the covariance matrix can easily be computed at the actual solution by calling ODRPACK with the final values of β and Δ as input, and with argument `MAXIT` $= 0$ and the third digit of argument `JOB` set to 0.

The standard deviations, $\hat{\sigma}_\beta$, of the function parameters $\hat{\beta}$ listed in the ODRPACK final report are the square roots of the diagonal elements of $\hat{V}_\beta$, i.e.,

$$\hat{\sigma}_{\beta_k} = \hat{V}_\beta^{1/2}(k,k) \ .$$

The 95% confidence intervals are computed using

$$\hat{\beta}_k \pm t_{.975,\mu}\hat{\sigma}_{\beta_k}$$

where $t_{.975,\mu}$ is the appropriate value for constructing a two-sided 95% confidence interval using the Student's t value for μ degrees of freedom. When $\mu > 20$, $t_{.975,\mu} \approx 2$; when $\mu < 5$, $t_{.975,\mu} > 2.5$.

If necessary, the full covariance matrix $\hat{V}$ for all of the estimators $\hat{\eta}$ can be computed using the equations given in [Boggs and Rogers, 1990b], or can be "automatically" obtained from most ordinary least squares software (including ODRPACK) by solving the orthogonal distance regression problem as the ordinary least squares problem defined by (4.3).

Note that for nonlinear ordinary least squares, the linearized confidence regions and intervals are asymptotically correct as $n \to \infty$ [Jennrich, 1969]. For the orthogonal distance regression problem, they have been shown to be asymptotically correct as $\sigma^* \to 0$ [Fuller, 1987]. The difference between the conditions of asymptotic correctness can be explained by the fact that, as the number of observations increases in the orthogonal distance regression problem one does not obtain additional information for Δ. Note also that $\hat{V}$ is dependent upon the weight matrix Ω, which must be assumed to be correct, and cannot be confirmed from the orthogonal distance regression results. Errors in the values of w_{ϵ_i} and w_{δ_i} that form Ω will have an adverse affect on the accuracy of $\hat{V}$ and its component parts. The results of a Monte Carlo experiment examining the accuracy of the linearized confidence intervals for four different measurement error models is presented in [Boggs and Rogers, 1990b]. Those results indicate that the confidence regions and intervals for Δ are not as accurate as those for β.

Despite its potential inaccuracy, the covariance matrix is frequently used to construct confidence regions and intervals for both nonlinear ordinary least squares and measurement error models because the resulting regions and intervals are inexpensive to compute, often adequate, and familiar to practitioners. *Caution must be exercised when using such regions and intervals, however, since the validity of the approximation will depend on the nonlinearity of the model, the variance and distribution of the errors, and the data itself.* When more reliable intervals and regions are required, other more accurate methods should be used. (See, e.g., [Bates and Watts, 1988], [Donaldson and Schnabel, 1987], and [Efron, 1985].)

4.C. Condition Number

For a *linear* least squares system of equations

$$A\beta =_2 Y \ , \tag{4.4}$$

with $A \in \Re^{n\times p}$ assumed to have full column rank and $=_2$ meaning "equals in the least squares sense," the *condition number* of A is defined as

$$\kappa(A) \equiv \|A\| \left\|A^\dagger\right\| \ ,$$

where $A^\dagger \equiv (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$ is known as the pseudo inverse of A. From this definition, we can show that $\kappa(A) \geq 1$, and that $\kappa(A) \to \infty$ as the columns of A become dependent.

Using $\kappa(A)$, bounds can be constructed on the relative error in the true least squares solution $\beta^* = A^\dagger Y$ due to a perturbation E_Y of Y, or to a perturbation E_A of A. While the actual bounds, discussed in detail in [Stewart, 1973], are quite complicated, we can roughly approximate them as follows. Let $\bar{\beta}_Y$ and $\bar{\beta}_A$ denote the solutions to the perturbed systems, $\bar{\beta}_Y = A^\dagger(Y + E_Y)$ and $\bar{\beta}_A = (A + E_A)^\dagger Y$, respectively, and let $R = A\beta^* - Y$ denote the residual at β^*. Then

$$\frac{\|\beta^* - \bar{\beta}_Y\|}{\|\beta^*\|} \lessapprox \kappa(A)\,\frac{\|E_Y\|}{\|Y\|} \tag{4.5}$$

and

$$\frac{\|\beta^* - \bar{\beta}_A\|}{\|\beta^*\|} \lessapprox \left(\kappa(A) + \kappa^2(A)\,\frac{\|R\|}{\|Y\|}\right)\frac{\|E_A\|}{\|A\|} \ . \tag{4.6}$$

For (4.5) we thus observe that the relative error in the solution could be as much as $\kappa(A)$ larger than the relative error in Y. Similarly, for (4.6) we observe that if the residual R is small, then the relative error in A is multiplied by $\kappa(A)$, whereas if the residual is not small, then the second term in (4.6) will dominate and the relative error in A could be multiplied by as much as $\kappa^2(A)$.

If we express the condition number $\kappa(A)$ as a power of 10, i.e., $\kappa(A) = 10^{\mathcal{K}}$, then (4.5) implies that the elements of the least squares solution could have $\mathcal{K}$ fewer significant digits of accuracy than the elements of Y, while (4.6) implies that the least squares solution could have as many as $2\mathcal{K}$ fewer significant digits of accuracy than the elements of A. *Therefore, $\kappa(A)$ sufficiently large could indicate that the least squares solution has no significant digits.* For the condition number to provide a meaningful estimate of ill-conditioning, however, the user's problem must be formulated so that the errors in the columns of A are equilibrated. This requires an intimate knowledge of the problem, and cannot be done as part of ODRPACK's automatic scaling procedure. If this equilibration has not been done by the user, the condition number returned by ODRPACK may not reflect the true conditioning of the problem.

ODRPACK returns an approximation to the *inverse* of the condition number $\kappa(M_\delta J_\beta)$, where J_β is the Jacobian matrix of partial derivatives with respect to β, and M_δ is a block diagonal matrix formed using the partial derivatives with respect to Δ as described in [Boggs *et al.*, 1987]. The matrix $M_\delta J_\beta$ is used by ODRPACK to form a linearization of the user's problem at the solution, and can thus be substituted for the matrix A in in the above discussions. The approximate inverse condition number is calculated as described in the *Linpack Users' Guide* [Dongarra *et al.*, 1980].

4.D. Scaling Algorithms

Poorly scaled problems, i.e., problems in which the unknowns β and Δ vary over several orders of magnitude, can cause difficulty for least squares procedures. ODRPACK's scaling algorithms (discussed below) attempt to overcome these difficulties automatically, although it is preferable for the user to choose the units of β and Δ so that their estimated values will have roughly the same magnitude. (See, e.g., [Dennis and Schnabel, 1983].) When the variables have roughly the same magnitude, the ODRPACK scaling algorithm will select scale values that are roughly equal, and the resulting computations will be the same (except for the effect of finite precision arithmetic) as an unscaled analysis, i.e., an analysis in which all of the scale values are set to one. If the user does not do this, the ODRPACK scaling algorithm will select varying scale values. This will not change the optimal solution, but it may affect the number of iterations required, or, in some cases, whether the algorithm is or is not successful.

The scale value times the corresponding absolute value of the expected solution should be approximately one. For example, if β_K is expected to be of order 10^{10} then $\text{SCALE}\{\beta_K\}$ should be set to 10^{-10}, while if β_K is expected to lie between -10^{-2} and -10^{-4} then $\text{SCALE}\{\beta_K\}$ should be set to 10^3. Scaling should not be confused with the weighting matrices w_{ϵ_i} and w_{δ_i} specified by subroutine arguments `WE` and `WD`. (See also §1.A and §1.F.)

4.D.i. Scaling β

ODRPACK chooses the default scale values for the estimated values of β as follows.

If some of the values of β are nonzero then
    Let $\beta_{\max}$ = the largest nonzero absolute value in β, and
    Let $\beta_{\min}$ = the smallest nonzero absolute value in β.
    For $\mathtt{K} = 1, \ldots, p$ do
        If $\beta_{\mathtt{K}} = 0$ then
            $\text{SCALE}\{\beta_{\mathtt{K}}\} = 10/\beta_{\min}$
        Else if $\log(\beta_{\max}) - \log(\beta_{\min}) > 1$ then
            $\text{SCALE}\{\beta_{\mathtt{K}}\} = 1/|\beta_{\mathtt{K}}|$
        Else
            $\text{SCALE}\{\beta_{\mathtt{K}}\} = 1/\beta_{\max}$
Else if all of the values of β are zero then
    For $\mathtt{K} = 1, \ldots, p$ do
        $\text{SCALE}\{\beta_{\mathtt{K}}\} = 1$

Users may also substitute their own scaling values for β. (See §2.B.ii, subroutine argument SCLB.)

4.D.ii. Scaling Δ

ODRPACK chooses default scale values for the estimated errors Δ_{IJ} in the explanatory variables as follows.

For $\texttt{J} = 1, \ldots, m$ do
    If some values in column J of X are nonzero then
        Let X_{max} = the largest nonzero absolute value in column J
        Let X_{min} = the smallest nonzero absolute value in column J
        For $\texttt{I} = 1, \ldots, n$ do
            If $X_{\text{IJ}} = 0$ then
                $\text{SCALE}\{\Delta_{\text{IJ}}\} = 10/X_{\text{min}}$
            Else if $\log(X_{\text{max}}) - \log(X_{\text{min}}) > 1$ then
                $\text{SCALE}\{\Delta_{\text{IJ}}\} = 1/|X_{\text{IJ}}|$
            Else
                $\text{SCALE}\{\Delta_{\text{IJ}}\} = 1/X_{\text{max}}$
    Else if all values in column J of X are zero then
        For $\texttt{I} = 1, \ldots, n$ do
            $\text{SCALE}\{\Delta_{\text{IJ}}\} = 1$

Users may also substitute their own scaling values for Δ. (See §2.B.ii, subroutine argument SCLD.)

5. Work Vectors

5.A. Extracting Information from Vector WORK

Upon return from a call to ODRPACK, array WORK contains values that may be of interest to the user. To extract information from WORK, the following declaration statement must be added to the user's program:

```
       LOGICAL
     +    ISODR
       INTEGER
     +    DELTAI,EPSI,XPLUSI,FNI,SDI,VCVI,
     +    RVARI,WSSI,WSSDEI,WSSEPI,RCONDI,ETAI,
     +    OLMAVI,TAUI,ALPHAI,ACTRSI,PNORMI,RNORSI,PRERSI,
     +    PARTLI,SSTOLI,TAUFCI,EPSMAI,
     +    BETAOI,BETACI,BETASI,BETANI,SI,SSI,SSFI,QRAUXI,UI,
     +    FSI,FJACBI,WE1I,DIFFI,
     +    DELTSI,DELTNI,TI,TTI,OMEGAI,FJACDI,
     +    WRK1I,WRK2I,WRK3I,WRK4I,WRK5I,WRK6I,WRK7I,
     +    LWKMN
```

where DELTAI through WRK7I are variables that indicate the starting locations within WORK of the stored values, and LWKMN is the minimum acceptable length of array WORK.

The appropriate values of DELTAI through WRK7I are obtained by invoking subroutine SWINF when using either of the single precision ODRPACK subroutines SODR or SODRC, and by invoking DWINF when using either of the double precision subroutines DODR or DODRC. The call statements for SWINF and DWINF have the same argument lists. To invoke either subroutine, use

```
       CALL <winf>
     +    (N,M,NP,NQ,LDWE,LD2WE,ISODR,
     +    DELTAI,EPSI,XPLUSI,FNI,SDI,VCVI,
     +    RVARI,WSSI,WSSDEI,WSSEPI,RCONDI,ETAI,
```

```
+     OLMAVI,TAUI,ALPHAI,ACTRSI,PNORMI,RNORSI,PRERSI,
+     PARTLI,SSTOLI,TAUFCI,EPSMAI,
+     BETAOI,BETACI,BETASI,BETANI,SI,SSI,SSFI,QRAUXI,UI,
+     FSI,FJACBI,WE1I,DIFFI,
+     DELTSI,DELTNI,TI,TTI,OMEGAI,FJACDI,
+     WRK1I,WRK2I,WRK3I,WRK4I,WRK5I,WRK6I,WRK7I,
+     LWKMN)
```

where SWINF should be substituted for <winf> when using SODR and SODRC, and DWINF should be substituted for <winf> when using DODR and DODRC. The variables N, M NP, NQ, LDWE, and LD2WE must be input to SIWINF and DIWINF with exactly the same values as were used in the original call to ODRPACK, and ISODR must be input set to *true* if the fit was by orthogonal distance regression, and be input set to *false* if the fit was by ordinary least squares. *Note that when* ISODR *is* false*, the locations that are specified by* DELTASI *through* WRK1I *are the same as the location specified by* DELTAI.

In the following descriptions of the information returned in WORK, $\triangleright$ indicates values that are most likely to be of interest.

$\triangleright$ WORK(DELTAI) is the first element of an $n \times m$ array DELTA containing the estimated errors $\hat{\Delta}$ in the explanatory variables at the solution, where

WORK(DELTAI-1+I+(J-1)*N) = DELTA(I,J) = $\hat{\Delta}_{\text{IJ}}$

for I $= 1, \ldots, n$, and J $= 1, \ldots, m$.

DELTAI $= 1$.

$\triangleright$ WORK(EPSI) is the first element of an $n \times q$ array EPS containing the estimated errors $\hat{E}$ in the response variables at the solution, where

WORK(EPSI-1+I+(L-1)*N) = EPS(I,L) = $\hat{E}_{\text{IL}}$

for I $= 1, \ldots, n$, and L $= 1, \ldots, q$.

EPSI $= nm + 1$.

$\triangleright$ WORK(XPLUSI) is the first element of an $n \times m$ array XPLUSD containing the final estimates of the explanatory variable $\hat{X}$, where

WORK(XPLUSI-1+I+(J-1)*N) = XPLUSD(I,J) = $\hat{X}_{\text{IJ}} = X_{\text{IJ}} + \hat{\Delta}_{\text{IJ}}$

for I $= 1, \ldots, n$, and J $= 1, \ldots, m$.

XPLUSI $= nm + nq + 1$.

▷ WORK(FNI) is the first element of an $n \times q$ array FN containing the final estimates of the response variable $\hat{Y}$, where

WORK(FNI-1+I+(L-1)*N) $=$ FN(I,L) $= \hat{Y}_{\mathrm{IL}} = f_{\mathrm{IL}}(x_{\mathrm{I}} + \hat{\delta}_{\mathrm{I}}; \hat{\beta})$

for $\mathtt{I} = 1, \ldots, n$, and $\mathtt{L} = 1, \ldots, q$.

FNI $= 2nm + nq + 1$.

▷ WORK(SDI) is the first element of a $p \times 1$ array SD containing the standard deviations $\hat{\sigma}_{\beta_{\mathrm{K}}}$ of the function parameters β, i.e., the square roots of the diagonal entries of the covariance matrix, where

WORK(SDI-1+K) $=$ SD(K) $= \hat{V}_{\beta}^{1/2}(\mathtt{K},\mathtt{K}) = \hat{\sigma}_{\beta_{\mathrm{K}}}$

for $\mathtt{K} = 1, \ldots, p$. The standard deviations are only computed when the third digit of JOB is less than or equal to 1. (See §2.B.ii, subroutine argument JOB, and §4.B.) Rows of SD corresponding to fixed elements of β, and to elements dropped because they induced rank deficiency, are set to zero.

SDI $= 2nm + 2nq + 1$.

▷ WORK(VCVI) is the first element of a $p \times p$ array VCV containing the values of the covariance matrix of the parameters β prior to scaling by the residual variance, where

WORK(VCVI-1+I+(J-1)*(NP)) $=$ VCV(I,J) $= \hat{\sigma}^{-2} \hat{V}_{\beta}(\mathtt{I},\mathtt{J})$

for $\mathtt{I} = 1, \ldots, p$ and $\mathtt{J} = 1, \ldots, p$. The covariance matrix is only computed when the third digit of JOB is less than or equal to 1. (See §2.B.ii, subroutine argument JOB, and §4.B.) Rows and columns of VCV corresponding to fixed elements of β, and to elements dropped because they induced rank deficiency, are set to zero.

VCVI $= 2nm + 2nq + p + 1$.

▷ WORK(RVARI) is the estimated residual variance $\hat{\sigma}^2 = S(\hat{\beta}, \hat{\delta})/\mu$. (See §4.B.)

RVARI $= 2nm + 2nq + p + p^2 + 1$.

▷ WORK(WSSI) is $S(\hat{\beta}, \hat{\delta})$. (See §1.A.)

WSSI $= 2nm + 2nq + p + p^2 + 2$.

▷ WORK(WSSDEI) is $S_{\delta}(\hat{\beta}, \hat{\delta})$. (See §1.A.)

WSSDEI $= 2nm + 2nq + p + p^2 + 3$.

▷ `WORK(WSSEPI)` is $S_\epsilon(\hat{\beta},\hat{\delta})$. (See §1.A.)
`WSSEPI` $= 2nm + 2nq + p + p^2 + 4$.

▷ `WORK(RCONDI)` is $\kappa^{-1}(M_\delta J_\beta)$ at the solution. (See §4.C.)
`RCONDI` $= 2nm + 2nq + p + p^2 + 5$.

▷ `WORK(ETAI)` is the relative error in the function values computed within `FCN`.
`ETAI` $= 2nm + 2nq + p + p^2 + 6$.

`WORK(OLMAVI)` is the average number of steps required to obtain the Levenberg-Marquardt parameter.

`WORK(TAUI)` is the trust region radius at the time the computations stopped.

`WORK(ALPHAI)` is the Levenberg-Marquardt parameter at the time the computations stopped.

`WORK(ACTRSI)` is the actual relative reduction in $S(\beta,\delta)$ from the last iteration.

`WORK(PNROMI)` is the norm of the scaled values of $\hat{\beta}$ and $\hat{\delta}$.

`WORK(RNORSI)` is $S(\hat{\beta},\hat{\delta})^{1/2}$.

`WORK(PRERSI)` is the predicted relative reduction in $S(\beta,\delta)$ from the last iteration.

`WORK(PARTLI)` is the stopping tolerance used to detect parameter convergence.

`WORK(SSTOLI)` is the stopping tolerance used to detect sum of squares convergence.

`WORK(TAUFCI)` is the factor used to compute the initial trust region radius.

`WORK(EPSMAI)` is machine precision, i.e., the smallest value ξ such that $1+\xi > 1$.

`WORK(BETAOI)` is the first element of a $p \times 1$ array `BETAO` containing the *initial* estimates of the function parameters β^0, where
`WORK(BETAIO-1+K)` = `BETAO(K)` $= \beta^0_{\mathrm{K}}$

for K $= 1, \ldots, p$. For implicit models, BETA0 is the initial value used for the last value of the penalty parameter.

WORK(BETACI) is the first element of a $p \times 1$ array BETAC containing the *current* working estimates of the $\tilde{p}$ unfixed function parameters $\tilde{\beta}^c$, where

WORK(BETACI-1+K) $=$ BETAC(K) $= \tilde{\beta}^c_{\mathrm{K}}$

for K $= 1, \ldots, \tilde{p}$.

WORK(BETASI) is the first element of a $p \times 1$ array BETAS containing the *saved* working estimates of the $\tilde{p}$ unfixed function parameters $\tilde{\beta}^s$, where

WORK(BETASI-1+K) $=$ BETAS(K) $= \tilde{\beta}^s_{\mathrm{K}}$

for K $= 1, \ldots, \tilde{p}$.

WORK(BETANI) is the first element of a $p \times 1$ array BETAN containing the *new* working estimates of the $\tilde{p}$ unfixed function parameters $\tilde{\beta}^n$, where

WORK(BETANI-1+K) $=$ BETAN(K) $= \tilde{\beta}^n_{\mathrm{K}}$

for K $= 1, \ldots, \tilde{p}$.

WORK(SI) is the first element of a $p \times 1$ array S containing the step in the $\tilde{p}$ unfixed function parameters at the last iteration, where

WORK(SI-1+K) $=$ S(K)

for K $= 1, \ldots, \tilde{p}$.

WORK(SSI) is the first element of a $p \times 1$ array SS containing the scale of the $\tilde{p}$ unfixed function parameters $\tilde{\beta}$, where

WORK(SSI-1+K) $=$ SS(K) $= \textsc{scale}\{\tilde{\beta}_{\mathrm{K}}\}$

for K $= 1, \ldots, \tilde{p}$.

WORK(SSFI) is the first element of a $p \times 1$ array SSF containing the scale of all of the function parameters β, where

WORK(SSFI-1+K) $=$ SSF(K) $= \textsc{scale}\{\beta_{\mathrm{K}}\}$

for K $= 1, \ldots, p$.

WORK(QRAUXI) is the first element of a $p \times 1$ array QRAUX used during the computations, where

WORK(QRAUXI-1+I) $=$ QRAUX(I)

for $\mathtt{I} = 1, \ldots, p$.

`WORK(U)` is the first element of a $p \times 1$ array `U` used during the computations, where

`WORK(UI-1+I)` $=$ `U(I)`

for $\mathtt{I} = 1, \ldots, p$.

`WORK(FSI)` is the first element of an $n \times q$ array `FS` containing the *saved* estimated errors E^s in the response variable, where

`WORK(FSI-1+I+L*N)` $=$ `FS(I,L)` $= E^s_{\mathrm{IJ}}$

for $\mathtt{I} = 1, \ldots, n$, and $\mathtt{L} = 1, \ldots, q$.

`WORK(FJACBI)` is the first element of an $n \times p \times q$ array `FJACB` containing the weighted partial derivative with respect to the $\tilde{p}$ unfixed function parameters $\tilde{\beta}$, where

`WORK(FJACBI-1+I+(K-1)*N+(L-1)*N*NP)`

$\quad = $ `FJACB(I,K,L)` $= \mathcal{W}_{\mathrm{I}} \frac{\partial f_{\mathrm{IL}}(x_{\mathrm{I}}+\delta_{\mathrm{I}};\beta)}{\partial \tilde{\beta}_{\mathrm{K}}}$

for $\mathtt{I} = 1, \ldots, n$, $\mathtt{K} = 1, \ldots, \tilde{p}$, $\mathtt{L} = 1, \ldots, q$, and ${\mathcal{W}_{\mathrm{I}}}^{\mathrm{T}} \mathcal{W}_{\mathrm{I}} = w_{\epsilon_{\mathrm{I}}}$. The derivatives are the values evaluated at the beginning of the last iteration unless the user requested that the covariance matrix be computed using the final solution, in which case they are the values obtained at the final solution. (See §2.B.ii, subroutine argument `JOB`.)

`WORK(WE1I)` is the first element of an `LDWE` $\times$ `LD2WE` $\times\, q$ array `WE1` containing the Cholesky factorizations for the weights w_ϵ specified in `WE`, where

`WORK(WE1I-1+I+(L1-1)*LDWE+(L2-1)*LDWE*LD2WE)`

$\quad = $ `WE1I(I,L1,L2)`

for $\mathtt{I} = 1, \ldots,$ `LDWE`, $\mathtt{L1} = 1, \ldots,$ `LD2WE`, and $\mathtt{L2} = 1, \ldots, q$. `WE1I` specifies the factorization in the same manner that subroutine argument `WE` is used to specify w_ϵ. (See §2.B.ii.)

`WORK(DIFFI)` is the first element of an $q \times (p + m)$ array `DIFF` containing the relative differences between the user supplied derivatives and the finite difference values they were checked against, where

`WORK(DIFFI-1+L+(J-1)*NQ)` $=$ `DIFF(L,J)` for L $= 1, \ldots, q$, and J $= 1, \ldots, (p + m)$.

WORK(DELTSI) is the first element of an $n \times m$ array DELTAS containing the *saved* working estimates of the errors Δ^s in the explanatory variables, where

WORK(DELTSI-1+I+(J-1)*N) = DELTAS(I,J) $= \Delta^s_{\mathrm{IJ}}$

for I $= 1, \ldots, n$, and J $= 1, \ldots, m$. If ISODR is *false*, then this array is equivalenced to the array DELTA beginning in WORK(DELTAI).

WORK(DELTNI) is the first element of an $n \times m$ array DELTAN containing the *new* working estimates of the errors Δ^n in the explanatory variables, where

WORK(DELTNI-1+I+(J-1)*N) = DELTAN(I,J) $= \Delta^n_{\mathrm{IJ}}$

for I $= 1, \ldots, n$, and J $= 1, \ldots, m$. If ISODR is *false*, then this array is equivalenced to the array DELTA beginning in WORK(DELTAI).

WORK(TI) is the first element of an $n \times m$ array T used in the computations, where

WORK(TI-1+I+(J-1)*N) = T(I,J)

for I $= 1, \ldots, n$, and J $= 1, \ldots, m$. If ISODR is *false*, then this array is equivalenced to the array DELTA beginning in WORK(DELTAI).

WORK(TTI) is the first element of an $n \times m$ array TT containing the scale of each the estimated errors Δ in the explanatory variable, where

WORK(TTI-1+I+(J-1)*N) = TT(I,J) $= \textsc{scale}\{\Delta_{\mathrm{IJ}}\}$

for I $= 1, \ldots, n$, and J $= 1, \ldots, m$. If ISODR is *false*, then this array is equivalenced to the array DELTA beginning in WORK(DELTAI).

WORK(OMEGAI) is the first element of a $q \times q$ array OMEGA used during the computations, where

WORK(OMEGAI-1+L1+(L2-1)*NQ) = OMEGA(L1,L2)

for L1 $= 1, \ldots, q$, and L2 $= 1, \ldots, q$. If ISODR is *false*, then this array is equivalenced to the array DELTA beginning in WORK(DELTAI).

WORK(FJACDI) is the first element of an $n \times m \times q$ array FJACD containing the weighted partial derivative with respect to Δ, where

WORK(FJACDI-1+I+(J-1)*N+(L-1)*N*M)
$= \text{FJACD(I,J,L)} = \mathcal{W}_{\mathrm{I}} \frac{\partial f_{\mathrm{IL}}(x_{\mathrm{I}}+\delta_{\mathrm{I}};\beta)}{\partial \Delta_{\mathrm{IJ}}}$
for I $= 1,\ldots,n$, J $= 1,\ldots,m$, L $= 1,\ldots,q$, and ${\mathcal{W}_{\mathrm{I}}}^{\mathrm{T}}\mathcal{W}_{\mathrm{I}} = w_{\epsilon_{\mathrm{I}}}$. The derivatives are the values evaluated at the beginning of the last iteration unless the user requested that the covariance matrix be computed using the final solution, in which case they are the values obtained at the final solution. (See §2.B.ii, subroutine argument JOB.) If ISODR is *false*, then this array is equivalenced to the array DELTA beginning in WORK(DELTAI).

WORK(WRK1I) is the first element of an $n \times m \times q$ array WRK1 required for work space, where

WORK(WRK1I-1+I+(J-1)*N+(L-1)*N*NQ) = WRK1(I,J,L)

for I $= 1,\ldots,n$, J $= 1,\ldots,m$, and L $= 1,\ldots,q$. If ISODR is *false*, then this array is equivalenced to the array DELTA beginning in WORK(DELTAI).

WORK(WRK2I) is the first element of an $n \times q$ array WRK2 required for work space, where

WORK(WRK2I-1+I+(L-1)*N) = WRK2(I,L)

for I $= 1,\ldots,n$, and L $= 1,\ldots,q$.

WORK(WRK3) is the first element of a $p \times 1$ array WRK3 required for work space, where

WORK(WRK3-1+K) = WRK3(K)

for K $= 1,\ldots,p$.

WORK(WRK4I) is the first element of an $m \times m$ array WRK4 required for work space, where

WORK(WRK4I-1+J1+(J2-1)*M) = WRK4(J1,J2)

for J1 $= 1,\ldots,m$, and J2 $= 1,\ldots,m$.

WORK(WRK5I) is the first element of an m array WRK5 required for work space, where

WORK(WRK5I-1+J) = WRK5(J)

for J $= 1,\ldots,m$.

`WORK(WRK6I)` is the first element of an $n \times p \times q$ array `WRK6` required for work space, where
`WORK(WRK6I-1+I+(K-1)*N+(L-1)*N*NQ)` = `WRK6(I,K,L)`
for I $= 1, \ldots, n$, K $= 1, \ldots, p$, and L $= 1, \ldots, q$.

`WORK(WRK7I)` is the first element of an $5q$ array `WRK7` required for work space, where
`WORK(WRK7I-1+J)` = `WRK7(J)`
for J $= 1, \ldots, 5q$.

5.B. Extracting Information from Vector IWORK

Upon return from a call to ODRPACK, array IWORK contains values that may be of interest to the user. To extract information from IWORK, the following declaration statement must be added to the user's program

```
      INTEGER
     +   MSGBI,MSGDI,IFIX2I,ISTOPI,
     +   NNZWI,NPPI,IDFI,
     +   JOBI,IPRINI,LUNERI,LUNRPI,
     +   NROWI,NTOLI,NETAI,
     +   MAXITI,NITERI,NFEVI,NJEVI,INT2I,IRANKI,LDTTI,
     +   LIWKMN
```

where MSGBI through LDTTI are variables that indicate the starting locations within IWORK of the stored values, and LIWKMN is the minimum acceptable length of array IWORK. The appropriate values of MSGBI through LDTTI are obtained by invoking subroutine SIWINF when using either of the single precision ODRPACK subroutines SODR or SODRC, and by invoking DIWINF when using either of the double precision subroutines DODR or DODRC. The call statements for SIWINF and DIWINF have the same argument lists. To invoke either subroutine, use

```
      CALL <iwinf>
     +   (M,NP,NQ,
     +   MSGBI,MSGDI,IFIX2I,ISTOPI,
     +   NNZWI,NPPI,IDFI,
     +   JOBI,IPRINI,LUNERI,LUNRPI,
     +   NROWI,NTOLI,NETAI,
```

```
     +    MAXITI,NITERI,NFEVI,NJEVI,INT2I,IRANKI,LDTTI,
     +    LIWKMN)
```

where SIWINF should be substituted for <iwinf> when using SODR and SODRC, and DIWINF should be substituted for <iwinf> when using DODR and DODRC. Note that the values of M, NP, and NQ must be input to SIWINF and DIWINF with *exactly* the same values as were used in the original call to ODRPACK.

In the following descriptions of the information returned in IWORK, $\triangleright$ indicates values that are most likely to be of interest.

$\triangleright$ IWORK(MSGBI) is the first element of a $1 + (q \times p)$ array MSGB used to indicate the results of checking the partial derivatives with respect to β at observation NROW. (See IWORK(NROWI) below.)

The value of IWORK(MSGBI) summarizes the results over all β.

- If IWORK(MSGBI) < 0 then
  the partial derivatives with respect to β were not checked.
- If IWORK(MSGBI) $= 0$ then
  the partial derivatives with respect to each of the β_{K}, K $= 1, \ldots, p$, for each of the q responses appear to be correct.
- If IWORK(MSGBI) $= 1$ then
  the partial derivative with respect to at least one of the β_{K}, K $= 1, \ldots, p$, appears to be questionable for at least one of the q responses.
- If IWORK(MSGBI) $= 2$ then
  the partial derivative with respect to at least one of the β_{K}, K $= 1, \ldots, p$, appears to be seriously questionable for at least one of the q responses.

The value of IWORK(MSGBI+L+(K-1)*NQ), L $= 1, \ldots, q$, K $= 1, \ldots, p$, indicates the individual results of checking the partial derivative of the Lth response with respect to each β_{K}, where for L $= 1, \ldots, q$, and K $= 1, \ldots, p$:

- If IWORK(MSGBI+L+(K-1)*NQ) $= -1$ then
  the partial derivative of the Lth response with respect to β_{K} was not checked because β_{K} was fixed.
- If IWORK(MSGBI+L+(K-1)*NQ) $= 0$ then
  the partial derivative of the Lth response with respect to β_{K} appears to be correct, i.e., the relative difference between its value and the finite difference approximation it is checked against is

within the required tolerance.

- If IWORK(MSGBI+L+(K-1)*NQ) = 1 then
  the partial derivative of the Lth response with respect to β_K is questionable because the user supplied derivative and the finite difference value it is checked against are both zero.
- If IWORK(MSGBI+L+(K-1)*NQ) = 2 then
  the partial derivative of the Lth response with respect to β_K is questionable because either the user supplied derivative is exactly zero and the finite difference value it is checked against is only approximately zero, or the user supplied derivative only approximately zero and and the finite difference value it is checked against is exactly zero.
- If IWORK(MSGBI+L+(K-1)*NQ) = 3 then
  the partial derivative of the Lth response with respect to β_K is questionable because either the user supplied derivative is exactly zero and the finite difference value it is checked against is not even approximately zero, or the user supplied derivative not even approximately zero and and the finite difference value it is checked against is exactly zero.
- If IWORK(MSGBI+L+(K-1)*NQ) = 4 then
  the partial derivative of the Lth response with respect to β_K is questionable because the finite difference value it is being checked against is questionable due to a high ratio of relative curvature to relative slope or to an incorrect scale value.
- If IWORK(MSGBI+L+(K-1)*NQ) = 5 then
  the partial derivative of the Lth response with respect to β_K is questionable because the finite difference value it is being checked against is questionable due to a high ratio of relative curvature to relative slope.
- If IWORK(MSGBI+L+(K-1)*NQ) = 6 then
  the partial derivative of the Lth response with respect to β_K is questionable because it does not agree with the finite difference value it is being checked against to the required tolerance, although the values do agree in their first two digits. (See IWORK(NTOLI) below.)
- If IWORK(MSGBI+L+(K-1)*NQ) = 7 then
  the partial derivative of the Lth response with respect to β_K is seriously questionable because it has fewer than two digits agree-

ment with the finite difference value it is being checked against.

MSGBI $= 1$.

▷ IWORK(MSGDI) is the first element of a $1 + (q \times m)$ array MSGD used to indicate the results of checking the partial derivatives with respect to Δ, analagous to MSGB described above. The values in MSGD have the same meanings as those used to indicate the results of checking the partial derivatives with respect to β, except that the value of IWORK(MSGDI) summarizes the results over all columns of Δ, and the values of IWORK(MSGDI+L+(J-1)*NQ), L $= 1, \ldots, q$, J $= 1, \ldots, m$, indicates the individual results for checking the partial derivative of the Lth response with respect to the Jth column of Δ at observation NROW. (See IWORK(NROWI) below.)

MSGDI $= qp + 2$.

▷ IWORK(IFIX2I) is the first element of a $p \times 1$ array IFIX2 containing values used to indicate whether a given parameter is unfixed, fixed or dropped because it induced rank deficiency, where

WORK(IFIX2I-1+K) = IFIX2(K)

for K $= 1, \ldots, p$. If

- IFIX2(K) = 1 the parameter was unfixed,
- IFIX2(K) = 0 the parameter was fixed,
- IFIX2(K) = -1 the parameter was dropped, and
- IFIX2(K) = -2 no parameters were estimated.

IFIX2I $= qp + qm + 3$.

▷ IWORK(ISTOPI) is value of ISTOP returned from the last call to subroutine FCN.

ISTOPI $= qp + qm + p + 3$.

IWORK(NNZWI) is the number of nonzero ϵ error weights, w_{ϵ_I}, I $= 1, \ldots, n$.

IWORK(NPPI) is the number $\tilde{p}$ of function parameters actually being estimated, i.e., the number of unfixed parameters.

IWORK(IDFI) is the degrees of freedom, μ, of the fit, i.e., the number of observations with nonzero weights minus the number of parameters actually being estimated.

IWORK(JOBI) is the value used to specify problem initialization and computational methods.

IWORK(IPRINI) is the print control value used.

IWORK(LUNERI) is the logical unit number used for error reports.

IWORK(LUNRPI) is the logical unit number used for computation reports.

IWORK(NROWI) is the observation NROW at which the derivatives were checked.

IWORK(NTOLI) is the number of digits of agreement required between the numerical derivatives and the user supplied derivatives for the user supplied derivatives to be considered correct.

IWORK(NETAI) is the number of good digits in the function results returned by user supplied subroutine FCN.

IWORK(MAXITI) is the maximum number of iterations allowed.

IWORK(NITERI) is the number of iterations taken.

IWORK(NFEVI) is the number of function evaluations made.

IWORK(NJEVI) is the number of Jacobian matrix evaluations made.

IWORK(INT2I) is the number of internal doubling steps taken at the time the computations stopped.

IWORK(IRANKI) is the rank deficiency at the solution.

IWORK(LDTTI) is the leading dimension of the $n \times m$ array TT. (See §5.A.)

Bibliography

[1] American National Standards Institute (1977), *ANS FORTRAN X3.9-1977*, American National Standards Institute, New York, NY.

[2] Bates, D. M., and D. G. Watts (1988), *Nonlinear Regression Analysis and its Applications*, John Wiley and Sons, New York, NY.

[3] Belsley, D. A., E. Kuh, and R. E. Welsch (1980), *Regression diagnostics,* John Wiley and Sons, New York, NY.

[4] Bement, T. R., and J. S. Williams (1969), "Variance of weighted regression estimators when sampling errors are independent and heteroscedastic," *J. Amer. Statists. Assoc.*, 64:1369-1382.

[5] Bischof, C., A. Carle, G. Corliss, A. Griewank, P. Hovland (1991), "ADIFOR — generating derivative codes from Fortran programs," ADIFOR Working Note #1, MCS-P263-0991, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL; also available as CRPC-TR91185, Center for Research on Parallel Computation, Rice University, Houston, TX.

[6] Boggs, P. T., R. H. Byrd, and R. B. Schnabel (1987), "A stable and efficient algorithm for nonlinear orthogonal distance regression," *SIAM J. Sci. Stat. Comput.*, 8(6):1052-1078.

[7] Boggs, P. T., R. H. Byrd, J. R. Donaldson, and R. B. Schnabel (1989), "Algorithm 676 — ODRPACK: Software for Weighted Orthogonal Distance Regression," *ACM Trans. Math. Software*, 15(4):348-364.

[8] Boggs, P. T., J. R. Donaldson, R. B. Schnabel, and C. H. Spiegelman (1988), "A computational examination of orthogonal distance regression," *J. Econometrics*, 38(1/2):169-201.

[9] Boggs, P. T., and J. E. Rogers (1990a), "Orthogonal Distance Regression," *Contemporary Mathematics*, 112:183-194.

[10] Boggs, P. T., and J. E. Rogers (1990b), "The Computation and Use of the Asymptotic Covariance Matrix for Measurement Error Models," Internal Report 89-4102, Applied and Computational Mathematics Division, National Institute of Standards and Technology, Gaithersburg, MD.

[11] Cook, D. R. (1977), "Detection of influential observations in linear regression," *Technometrics,* 19:15-18.

[12] Dennis, J. E., and R. B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ.

[13] Donaldson, J. R., and Schnabel, R. B. (1987), "Computational Experience with Confidence Regions and Confidence Intervals for Nonlinear Least Squares," *Technometrics*, 29(1):67-82.

[14] Donaldson, J. R., and P. V. Tryon (1986), "STARPAC - the Standards Time Series and Regression Package," Internal Report 86-3448, Computing and Applied Mathematics Laboratory, National Institute of Standards and Technology, Boulder, CO.

[15] Dongarra, J. J., C. B. Moler, J. R. Bunch, and G. W. Stewart (1979), *LINPACK Users' Guide*, SIAM, Philadelphia, PA.

[16] Efron, B. (1985), *The Jackknife, the Bootstrap and Other Resampling Plans,* SIAM, Philadelphia, PA.

[17] Filliben, J. J. (1977), "User's Guide to the DATAPAC Data Analysis Package," (unpublished: available from NIST Statistical Engineering Division, Gaithersburg, MD).

[18] Fox, P. A., A. D. Hall, and N. L. Schryer (1978), "Algorithm 528 — Framework for a Portable Library [z]," *ACM Trans. Math. Software*, 4(2):177-188.

[19] Fuller, W. A. (1987), *Measurement Error Models*, John Wiley and Sons, New York, NY.

[20] Gill, P. E., W. Murray, and M. H. Wright (1981), *Practical Optimization*, Academic Press, New York, NY.

[21] Griewank, A. (1989), "On automatic differentiation," in *Mathematical Programming: Recent Developments and Applications*, M. Iri and K. Tanabe, editors, Kluwer Academic Publishers, Amsterdam, Holland.

[22] Havriliak, S. Jr., and S. Negami (1967), "A Complex Plane Representation of Dielectric and Mechanical Relaxation Processes in Some Polymers," *Polymer*, 8: 161-205.

[23] Himmelblau, D. M. (1970), *Process Analysis by Statistical Methods*, John Wiley and Sons, New York, NY.

[24] Hoaglin, D. C., and R. E. Welsch (1978), "The hat matrix in regression and ANOVA," *American Statistician,* 32:17-22.

[25] Jennrich, R. I. (1969), "Asymptotic Properties of Non-linear Least Squares Estimators," *Annals of Mathematical Statistics*, 40:633-643.

[26] Juedes, D. (1991) "A taxonomy of automatic differentiation tools," in *Proceedings of the Workshop on Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, A. Griewank and P. L. Toint, editors, SIAM, Philidelphia, PA.

[27] Lawson, C., R. Hanson, D. Kincaid, and F. Krogh (1979), "Basic linear algebra subprograms for FORTRAN usage", *ACM Trans. Math. Software*, 5(3):308-371.

[28] Schnabel, R. B. (1982), "Finite difference derivatives - theory and practice," (unpublished: available from author).

[29] Seber, G. A. F., and C. J. Wild (1989), *Nonlinear Regression*, John Wiley and Sons, New York, NY.

[30] Stephens, B. R., and J. D. Pryce (1991), "DAPRE: a differentiation arithmetic system for Fortran," Technical Report ACM-91-3, Royal Military College of Science, Shrivenham, UK.

[31] Stewart, G. W. (1973), *Introduction to Matrix Computations*, Academic Press, New York, NY.