ISE-TR-13

PROGRAM SYSTEM SALS FOR NONLINEAR LEAST-SQUARES FITTING: SYSTEM DESIGN

BY

Toru Nakagawa

and

Yoshio Oyanagi

February 1, 1980

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS

UNIVERSITY OF TSUKUBA

Program System SALS for Nonlinear Least-Squares Fitting: System Design.*

Toru Nakagawa[+] and Yoshio Oyanagi[‡]

*Received by the editors February 1, 1980. This work was supported in part by Laboratory of Application Programs, Computer Centre, The University of Tokyo (1975-) and by the Grant of Research Project No. 310228: "Studies on Statistical Program Package", as a part of the Special Project "Formulation Process of Information System and Organization of Scientific Informations" sponsored by the Ministry of Education of Japan (1976-1978).

[+]Department of Chemistry, Faculty of Science, The University of Tokyo, Hongo, Bunkyo-ku, Tokyo, 113 Japan.
[‡]Institute of Information Sciences, University of Tsukuba, Sakura-mura, Niihari-gun, Ibaraki, 305 Japan.

43 pages
2 tables
8 figures

To be submitted to SIAM J. Sci. & Stat. Comp.

ABSTRACT

A program system SALS is developed for statistical data analysis with nonlinear least-squares fitting, featuring:

a) a general-purpose full system containing input/output routines and accessing a user-coded theoretical model subroutine,
b) reliable and fast convergence in nonlinear least-squares algorithms,
c) high precision in five linear least-squares algorithms,
d) robust estimation techniques including Tukey's biweight method,
e) various statistical diagnoses including Akaike's information criterion,
f) easy-to-use commands for input and control, and
g) dynamic allocation of arrays.

The system consists of about 10000 steps of standard Fortran.

Keywords: SALS system, program design, program package, data analysis, model fitting, least squares, nonlinear least squares, singular value decomposition, Marquardt method, statistical analysis, parameter binding, model selection, AIC, robust estimation,

CONTENTS (not for press)

1. INTRODUCTION

In various fields of science and applications, statistical data analyses like least-squares fittings are required to handle with elaborate theoretical models worked out for each problem. Thus, a great number of people have made much efforts to make least-squares programs to solve their own problems. However, they often meet difficulties in getting reliable solutions, in selecting the best models, and in extracting statistical information. These difficulties are mostly originated in

(a) near singularity of normal equations,

(b) instability in the nonlinear refinement algorithms,

(c) contamination of unreliable observations, and

(d) poor guideline to construct and select the models.

Since all these are mainly concerned with numerical analysis and statistics, they should be solved in a way universal to various applieation fields [13, 28,47-49]. Thus, in the present study, we have developed a program system SALS for nonlinear least-squares analysis, of high quality both in numerical and statistical features, and suitably designed for general use in natural sciences.

Even though many preceding works cover one portion or another of the present objectives, none of them have built a general system like the present one:

Subroutines for linear least-squares algorithms are now widely available in various program libraries [17, 23, 40],

mostly for solving the normal equations and some for getting the solutions directly without setting up the normal equations. Subroutines for nonlinear algorithms were also developed by a number of people [23, 40]; some of them take the advantage of minimizing the sum of squares of residuals while others aim to minimize a general nonlinear function. The advantages and disadvantages of these algorithm subroutines depend on the scale and nature of the problems to be solved; hence, it is desirable, but left to be done, to select them and build them up into a unified system of easy control.

Examples of program systems of nonlinear fitting algorithms are MLAB [27], MINUIT [26], and FIT [12]. MLAB is a curve fitting program with interactive operation and is not intended for large-scale data analyses with complex theoretical models. MINUIT is a package of nonlinear minimization algorithms instead of least squares. FIT is a system for statistical model fitting with algorithms for data manipulation, linear least-squares, and nonlinear optimization; the present SALS system has similar objective with FIT and is newly designed with much stress on linear and nonlinear least-squares algorithms, robust estimation, and easy control.

Another class of preceding works are general-purpose 'statistical program packages' [3, 16, 45], such as BMD-BMDP [14, 15], SPSS [39], OMNITAB [24], and SAS [5], which are designed mainly for the fields of social, behavioral, and

biological sciences. Their 'regression', or even 'nonlinear regression', programs expect simple models like polynomials, and do not accept nonlinear models.

The SALS system developed in the present study is a full system for nonlinear least-squares analysis, containing input/output routines and accessing a user-coded subroutine for calculating the theoretical model specific to the user's problem. Selected and reliable algorithms are available for nonlinear as well as linear least-squares calculations. Robust estimation methods in addition to the ordinary least-squares method can be applied to the observed data. After fitting, various statistical diagnoses are output such as error matrix of the estimated parameters, Akaike's information criterion AIC for model selection, and plots of residuals. All these algorithms and statistical options can be easily controlled by the user with a simple set-up of input data.

Table I summarizes the basic specifications of the present system.

2. GENERAL STRUCTURE OF SALS SYSTEM

A. Structure of SALS System as a Black Box

Figure 1 schematically shows the structure of a data fitting program using the present system. In this figure SALS system is shown as a black box accompanying input/output routines. The user's main program simply calls SALS which handles with everything for input/output, data fitting, and statistical treatment. Standard I/O routines are installed in the system, and yet user's own I/O routines are acceptable as options. SALS system also calls a user-coded subroutine for the theoretical model calculation; since the theoretical model depends on the user's problem, the user must supply this subroutine. It is desirable to calculate the Jacobian matrix of the model in this subroutine, but its calculation may be omitted if the analytical form of the derivatives is not obtainable.

One may notice that the subroutines drawn on the right half of Fig. 1 are the essentials of a program for theoretical model calculation, as illustrated in Fig. 2. In general, data analysis usually proceeds in the cycles of (a) theoretical model calculation, (b) data fitting, and (c) diagnosis of the fit and the model; hence, it is a good practice for the user to do theoretical model calculation before the fitting, in order to understand the behavior of the model and get rough estimates of the model parameters.

B. Structure of SALS System

SALS is written in a standard Fortran (JIS 7000 corresponding to ANSI 66) and constructed as a system of about 130 subroutines. It has hiererchical calling structure as illustrated in Fig. 3. Each block in the figure is actually made of one to twenty subroutines. These subroutines are classified into 10 groups, six of them directly related to the fitting algorithms while the rest playing auxilliary roles:

Group 1, SALS: SALS control routine (a) handles several sets of algorithms, parameters, and observations by use of the input routines (b).

Group 2, SALSEX: Execution of a fitting task is prepared: for one set of algorithms, parameters and observations, default values are set for unspecified controls and working areas are dynamically allocated.

Group 3, LSF: For a robust estimation effective weights are adjusted according to the residuals, and then a nonlinear least-squares algorithm is called.

Group 4, NONLIN: Nonlinear least-squares calculation is executed; there are three methods, i.e. (a) Gauss-Newton method with a damping option, (b) Levenberg-Marquardt method with Fletcher's algorithm, and (c) Powell's hybrid method, even though (c) is still under way of development.

Group 5, LINLS: This group of subroutines control the linear least-squares algorithms either (a) without using the

normal equation or (b) with using it. Rank deficiency treatment, iterative improvement of the solution, and switching of linear algorithms are managed.

Group 6, LINEAR: Parameter corrections are calculated with either one of the five selected algorithms for linear least-squares calculation: (a) Modified Gram-Schmidt method, (b) Householder orthogonal transformation method, (c) singlar-value decomposition method, (d) Cholesky decomposition method, and (e) eigenvalue decomposition method. Among these, (a)-(c) obtain the solution directly from the Jacobian matrix without using the normal equation, while (d) and (e) solve the normal equation.

Group 7, STAT: All the statistical information are calculated and outputted: including standard deviation, AIC, error matrix of parameters, and plots of residuals.

Group 8, OUT: Intermediate and final results are outputted.

Group 9, MODL: This group of subroutines are called to get the calculated values and the Jacobian matrix of the user's theoretical model. The user must supply a theoretical model subroutine (b) in either one of the following three forms:

(i) MODELF subroutine for calculating the theoretical values alone,

(ii) MODELD subroutine for calculating the theoretical values and the Jacobian matrix, and

(iii) MODELN subroutine for giving the theoretical values together with the normal-equation matrix.

The control routine (a) of this group serves as an interface between the algorithm and the user-coded subroutine; for instance, if the user supplies a MODELF subroutine, the Jacobian matrix is numerically calculated in this interface by calling MODELF repeatedly.

Group 10, DFCHEK: This is a utility to check the Jacobian matrix calculated by the user's MODELD subroutine. Another Jacobian matrix is obtained by numerically differentiating the theoretical model values supplied by the MODELD subroutine. The two Jacobian matrices are compared and numbers of matching digits are printed out. This checking is done as a separate, preliminary job before fitting.

C. Dynamic Allocation of Arrays

In the SALS system, the arrays of working area are dynamically allocated in a large one-dimensional array WK in the unlabelled COMMON block, and yet they can be used just like the original one- or two-dimensional arrays in the scheme of variable dimensions. The sizes of the working arrays are adjusted according to the number of observations n, the number of variable parameters m, and the algorithms to be used. The arrays are usually allocated separately in WK, but for a smaller size of WK, some of them are overwritten. If WK is too small to store the necessary arrays even with the atmost overwriting, the system prints out an error message

and the necessary size of WK. The size of WK can easily be adjusted by the user declaring the unlabelled COMMON block in his main program (see Fig. 5).

D. Implementation

The present system has been developed on a HITAC 8800/8700 computer at the Computer Centre of the University of Tokyo, which is operated under an operating system OS7 in a virtual memory system. The unlinked object codes of the whole SALS system occupy the memory as large as 180KW, while ordinary runs of them in a dynamic linkage mode need about 150 KW. This memory requirement is much reduced to 60 KW by taking linkage of the object modules in a logical overlay structure without change in the computation time. The longest path in this overlay structure contains only one of the alternative algorithm sets; it essentially consists of the subroutines of groups 4b, 5a, 6c, 7, 8, 9a, and 9b shown in Fig. 3.

SALS program system has been successfully transferred to a number of university computer centers in Japan, where the machines such as HITAC M-170, FACOM M-200, FACOM 230/75, and NEAC ACOS 900 are in operation.

3. ALGORITHMS

A. Fundamental Formulation [10, 21, 35]

Least-squares problems may be expressed symbolically as

$$\mathbf{y} \cong \mathbf{f}(\mathbf{x}) \qquad \text{with uncertainty } \boldsymbol{\sigma}, \tag{1}$$

where $\mathbf{y}$, $\mathbf{f}$, and $\boldsymbol{\sigma}$ are the n-vectors of observed values, theoretical model functions, and uncertainties, respectively, and $\mathbf{x}$ is the m-vector of the model parameters to be determined. The least-squares condition is invoked to determine the best-fit parameters $\hat{\mathbf{x}}$ as

$$S = \mathbf{v}^T\mathbf{W}\mathbf{v} = \Sigma_i w_i (y_i - f_i(\hat{\mathbf{x}}))^2 = \text{minimum}, \tag{2}$$

where

$$\mathbf{v} = \mathbf{y} - \mathbf{f}(\hat{\mathbf{x}}): \text{ residuals}, \tag{3}$$

$$w_i = \text{diag}\{\mathbf{W}\} = \sigma_0^2/\sigma_i^2 : \text{ weights}, \tag{4}$$

σ_0 : standard error.

In case of a linear model, namely

$$\mathbf{f}(\mathbf{x}) = \mathbb{A}\mathbf{x} \tag{5}$$

where

$$\mathbb{A}_{ij} = \partial f_i/\partial x_j : \text{ Jacobian matrix}, \tag{6}$$

the parameters $\hat{\mathbf{x}}$ can be obtained by solving the normal equat

$$(\mathbb{A}^T\mathbf{W}\mathbb{A})\hat{\mathbf{x}} = (\mathbb{A}^T\mathbf{W}\mathbf{y}) \qquad \text{or} \qquad \mathbb{B}\hat{\mathbf{x}} = \mathbb{b}, \tag{7}$$

where

$$\mathbb{B} = \mathbb{A}^T\mathbf{W}\mathbb{A} \qquad \text{and} \qquad \mathbb{b} = \mathbb{A}^T\mathbf{W}\mathbf{y}.$$

Setting up the normal equation (7), however, is not always necessary to get the solution. For instance, if the weighte

Jacobian matrix $\mathbb{A}' = \mathbb{W}^{\frac{1}{2}}\mathbb{A}$ is decomposed into a product of $\mathbb{Q}$ and $\mathbb{R}$, say with the modified Gram-Schmidt method,

$$\mathbb{W}^{\frac{1}{2}}\mathbb{A} = \mathbb{A}' = \mathbb{Q}\mathbb{R}, \tag{8}$$

where $\mathbb{Q}$ is an orthogonal matrix while $\mathbb{R}$ is an upper triangular matrix, then the parameters are directly solved as

$$\hat{\mathbf{x}} = (\mathbb{A}^T\mathbb{W}\mathbb{A})^{-1}(\mathbb{A}^T\mathbb{W}\mathbf{y}) = \mathbb{R}^{-1}\mathbb{Q}^T\mathbb{W}^{\frac{1}{2}}\mathbf{y} = \mathbb{R}^{-1}\mathbb{Q}^T\mathbf{y}', \tag{9}$$

where $\mathbf{y}' = \mathbb{W}^{\frac{1}{2}}\mathbf{y}$ is the weighted observables.

In case of a nonlinear model, iterative refinement methods are used: Around some initial guess of parameters $\hat{\mathbf{x}}^{(0)}$, the model functions $\mathbf{f}$ are expanded into the Taylor series to get the expression

$$\Delta\mathbf{y} \cong \mathbb{A}\Delta\mathbf{x} \qquad \text{with uncertainty} \qquad \sigma \tag{10}$$

in the first-order approximation, where

$$\Delta\mathbf{y} = \mathbf{y} - \mathbf{f}(\hat{\mathbf{x}}^{(0)}) \qquad \text{and} \qquad \Delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}^{(0)}. \tag{11}$$

Then the linear least-squares algorithms are used to obtain the parameter corrections $\Delta\hat{\mathbf{x}}$, and the corrected parameters

$$\hat{\mathbf{x}} = \hat{\mathbf{x}}^{(0)} + \Delta\hat{\mathbf{x}} \tag{12}$$

are now used as the basis for the further refinement.

B. Linear Least-Squares Algorithms [13, 19, 28, 49, 50]

As shown in Fig. 3, SALS system uses the following five selected algorithms for the linear least-squares calculation:

(a) Modified Gram-Schmidt method [6-8],

(b) Householder orthogonal transformation method [9, 11],

(c) Singular-value decomposition method [20, 28],

(d) Cholesky decomposition method, and

(e) Eigenvalue decomposition method [46].

Table II summarizes the outlines of these algorithms in the form of three steps: decomposition, solution, and calculation of the error matrix.

The algorithms (a)-(c) do not set up the normal equation (7) but directly transform the Jacobian matrix $\mathbb{A}'$ to get the solution $\mathbb{x}$. They are relatively new and known to have higher precision even in ill-conditioned cases than the traditional algorithms (d) and (e) which solve the normal equation. Furthermore, their computation times are of the same order as those of the traditional ones. Thus, in SALS system, the modified Gram-Schmidt and the Householder algorithms are used as the standard procedures for ordinary problems. They both contain algorithms for pivoting and for rank deficiency; the rank is judged with a threshold of 16 times the computer precision ε, and for the underterminable variables zeroes are returned. The singular-value decomposition method (c), on the other hand, chooses a proper number of linear combinations of variables to give minimum-norm solution in case of rank deficiency. This feature is useful to obtain reasonable models for linear least-squares problems and to achieve more stable convergence in nonlinear problems. Lawson and Hanson's subroutines [28] are adapted for (c); they use Householder transformation to get a bidiagonal matrix and a QR algorithm to diagonalize it [20].

On the other hand, the algorithms using the normal equation have merits of smaller memory requirement: The normal equation matrix can usually be calculated by accumulating the contributions of each observed datum without storing the whole Jacobian matrix. Thus, SALS includes the options of solving the normal equations for larger-scale problems. In this case, Cholesky method is ordinarily used by virtue of its speed, precision, and small memory requirement. For the purpose of getting a minimum-norm solution in case of rank deficiency, the eigenvalue decomposition method (or diagonalization method) is used. The subroutines developed in the EISPACK system [46] have been adapted after remodeling; they use the Householder tridiagonalization and a QR algorithm with an implicit origin-shift technique.

In order to keep high precision in the linear algorithms, some calculations like vector products are carried out in double precision, even though the whole calculations in SALS are done in single precision. Optional iterative improvement of the solution is also usefull in the linear algorithms (a), (b), and (d).

C. Nonliner Least-Squares Algorithms [13, 44]

For nonlinear least-squares problems, the following two methods are available in SALS system:

(a) Gauss-Newton method with a damping option [22], and

(b) Levenberg-Marquardt method with Fletcher's algorithm [18, 29, 31].

In addition to these, a third method

(c) Powell's hybrid method [42, 43]

is under way of development.

Gauss-Newton method (a) carries out an iterative refinement on the basis of the locally-linearized model approximation (10). This method is useful, in nature, in case of weak nonlinearity and of good initial guess of parameters. In order to extend its applicability to the cases of stronger nonlinearity or of poorer guess of initial parameters, JALS has adapted a damping option: If the residual sum of squares S does not decrease at the refined parameter values of Eq. (12), the parameter corrections $\Delta\hat{\mathbf{x}}$ are reduced into halves stepwise getting a smaller S. The iteration of nonlinear least-squares fitting is terminated either when the refinement has converged, when the number of cycles exceeds the limit, when S fails to decrease more than the specified times, or when S gets smaller than a lower threshold. The convergence of the refinement is approved when S and S^0 of the preceding cycle satisfies the following relation

$$S^0 \cdot (1-\eta) \leqq S \leqq S^0 \cdot (1+\eta), \tag{13}$$

where η stands for the tolerance of order $\varepsilon^{1/2}$, and at the same time when the parameter corrections $\Delta\mathbf{x}$ are small enough.

Levenberg-Marquardt method (b) [29, 31] is similar to the Gauss-Newton method except the way of avoiding the increase in S. It artificially adds some extra terms $\lambda\mathbf{D}$ to the diagonal

elements of the normal-equation coefficient matrix $\mathbb{A}^T\mathbb{W}\mathbb{A}$:

$$(\mathbb{A}^T\mathbb{W}\mathbb{A} + \lambda\mathbb{D})\Delta\hat{\mathbb{x}} = \mathbb{A}^T\mathbb{W}\Delta\mathbb{y}, \tag{14}$$

where $\mathbb{D}$ is taken to be either a unit matrix $\mathbb{I}$, a diagonal matrix diag$\{\mathbb{A}^T\mathbb{W}\mathbb{A}\}$, or $\mathbb{I}$ + diag$\{\mathbb{A}\mathbb{W}\mathbb{A}\}$. If $\mathbb{D} = \mathbb{I}$ and the scale factor λ is large enough, the above solution essentially corresponds to that of the steepest descent method. Hence, by adjusting the λ constant properly, the iterative refinement is expected to converge. Among various empirical ways so far proposed to adjust λ, Fletcher's algorithm [18] is adapted in SALS system. Even though the Levenberg-Marquardt method is usually described in terms of the normal equation as above, the same treatment can be formulated by attaching extra terms to the Jacobian matrix itself [28]; hence, in our SALS program, the Levenberg-Marquardt method may call either one of the five linear least-squares algorithms described in Section 4B. The criteria for terminating the iteration and for checking the convergence are similar to those for (a).

Powell's hybrid method (c) [42, 43] is under way of adaptation in SALS. According to some preliminary tests, it is expected to be faster and more stable in heavily nonlinear cases without requiring the user to set up the Jacobian matrix.

4. STATISTICAL FEATURES [10, 21, 35,47]

A. Weighted Least-Squares Method

Uncertainties in the observed data may or may not be uniform. Therefore, weighted least-squares analysis is one of the basic functions in SALS. The weights w_i are set as in Eq.(4) by using the input data of experimental uncertainties σ_i and the standard error of observation σ_0 which corresponds to the experimental uncertainty for the unit weight.

B. Binding the Parameters

User's theoretical model often includes a number of variable parameters which are not negligible but difficult to determine from his observations alone. Sometimes such parameters were previously estimated and may be found, say, in literature. In such a case, the parameters may be fixed at the pre-estimated values by inputing the flags in SALS system. It is more reasonable, however, to introduce such pre-estimated values as additional observations accompanying their own uncertainties: by inputing the pre-estimated values $\hat{x}_j^{(0)}$, their uncertainties $\sigma(\hat{x}_j^{(0)})$, and control flags, these parameters are loosely bound around their input values with the weights given by Eq. (4).

C. Standard Deviation

The standard deviation of observation is estimated as

$$\sigma_0^2 = v^T W v/(n - m). \tag{16}$$

According to statistics, the expectation value of $\hat{\sigma}_0$ is σ_0

in Eq. (4). There are two cases concerning the standard deviation:

(a) The experimental uncertainties σ_i are known: In this case, σ_0 is a known arbitrary constant, and $\hat{\sigma}_0$ can be used to check the goodness of the fit, e.g. by use of the chi-square test. If the model passes the test, then we take

$$\hat{\sigma}_0' = \max\{\sigma_0, \hat{\sigma}_0\} \tag{17}$$

as a safer guess of the standard error σ_0.

(b) The experimental uncertainties σ_i are only relatively known: In this case, σ_0 is introduced as an unknown scaling factor in Eq. (4), and the standard deviation $\hat{\sigma}_0$ gives the estimate of σ_0.

D. Uncertainties in Parameters and in Calculated Values

The error matrix, or the variance-covariance matrix, of the parameters are obtained as

$$\Sigma_{\hat{x}} = \sigma_0^2 (A^T W A)^{-1}. \tag{18}$$

The actual ways for calculating $\Sigma_{\hat{x}}$ in various linear least-squares algorithms are summarized in Table II. The $\hat{\sigma}_0'$ in Eq. (17) is safely used in $\Sigma_{\hat{x}}$ in place of σ_0. The uncertainties of the parameters

$$\sigma_{\hat{x}_j} = (\Sigma_{\hat{x}\ jj})^{\frac{1}{2}} \tag{19}$$

and correlation coefficients

$$\rho_{jj'} = (\Sigma_{\hat{x}})_{jj'} / \sigma_{\hat{x}_j} \sigma_{\hat{x}_{j'}} \tag{20}$$

are also outputted. The estimation errors in the calculated

values $\hat{\mathbb{y}} = \mathbb{f}(\hat{\mathbb{x}})$ are also evaluated on the basis of the propagation law of errors as

$$\sigma_{\hat{y}_i} = [(\mathbb{A}\Sigma_{\hat{x}}\mathbb{A}^T)_{ii}]^{\frac{1}{2}} \tag{21}$$

E. Diagnostic Plots of Residuals

For the purpose of diagnosis of the fit, three types of graphs of the residuals are output on the line printer: (a) a histogram, (b) a graph of residuals against the input data number, and (c) a normal probability plot. In these graphs, the normalized residuals v_i/σ_i are plotted after scaling with s in Eq. (24), so as to show the graphs clearly and without scaling out. Histogram (a) is useful to check the existence of outliers. On the graph (b), one can see any systematic pattern of residuals which reveals the incompleteness of the model; if the fitting is good, this graph should appear quite random. This kind of graph can be more powerful if the user plots the residuals against more meaningful abscissa for his problem; such a user subroutine can easily be attached to SALS system. Normal probability plot (c) is useful at the final stages of data analysis to examine the distribution of the residuals in comparison with the normal distribution.

F. Model Selection with AIC [1, 2]

Another statistical feature of SALS system is the introduction of Akaike's information criterion AIC for the purpose of model selection. AIC is defined as

$$\text{AIC} = -2 \log_e (\text{maximum likelihood}) + 2m + \text{const}; \tag{22}$$

hence, in case of normal distribution of errors, it is reduced as

$$AIC = n \log_e S + 2m \tag{23}$$

where S is the residual sum of squares and n and m are the numbers of observations and parameters, respectively. In order to select a proper model, several possible models with different degrees of freedom or of different types should be tested one by one in the least-squares fit. Then the model which gives the minimum AIC is supposed to be the best. It should be noted that AIC is a relative criterion and works well only if the true model is included among the tested models and if $n >> m$.

G. Robust Estimation [4, 25, 47]

In order to protect against erroneous input of observables, several options of robust estimation methods are introduced in SALS system in addition to the ordinary least-squares method. One of the M-estimation methods, namely Tukey's biweight method [47], is chosen as the default procedure in SALS, which proceeds in the following way:

(i) By use of the initial parameter values, the residuals v_i are calculated for the observed values.

(ii) A scale s of the residuals is evaluated by

$$s = \text{median}\{|v_i/\sigma_i|\}, \tag{24}$$

since median is known to be most robust in estimating such a scale.

(iii) According to the residuals, weight adjustment factors w_i^{adj} are calculated by Tukey's biweight method:

$$w_i^{adj} = \begin{cases} [1-(v_i/\sigma_i sc)^2]^2 & \text{if } |v_i/\sigma_i| < sc, \\ 0 & \text{if } |v_i/\sigma_i| \geq sc. \end{cases} \tag{25}$$

The constant c is chosen around 10. The relations (25) are illustrated in Fig. 4.

(iv) In place of the input weights w_i, the effective weights

$$w_i^{eff} = w_i \cdot w_i^{adj} \tag{26}$$

are used in the nonlinear least-squares calculation to get the refined parameters.

(v) With the refined parameter values obtained after one or more cycles, the residuals and the effective weights are calculated again.

The above cyclic process (i)-(v) are repeated until the convergence of the effective weights or until reaching a limit of the weight adjustment cycles.

In robust estimation, a proper choice of the scale sc in Eq. (25) is crucial; the above treatment adjusts it flexibly according to the median of the residuals, so as to prepare for large residuals in the initial stages of fitting and for smaller ones in the final stages. The present methods in SALS have been proved to be effective and robust for several test problems with nonlinear models. It should be noticed

that the statistical treatments described in the preceding sections (4A-F) are based on the hypothesis of the normal distribution of errors. Robust estimation methods, on the other hand, are more practical approaches which prepare against various non-normal distributions of errors. Since data analysis always proceeds on a trial-and-error basis, both the observed data and the models in the initial stages can hardly be ideal as are assumed in the ordinary least-squares theory. Even one outlier in the observed data may destroy the whole least-squares fitting; contrarily, robust estimations are resistant to such outliers, and hence are useful for diagnosis. For this diagnostic purpose, the statistical treatments described in the preceding sections are also applicable in the robust estimation by replacing the input weights w_i with the effective weights w_i^{eff}. These statistical information are useful to check the observations and models before trying a final least-squares fit.

5. CONTROLS

A. Input Job Stream

In order to illustrate the usage and controls of the SALS system, a simple example of an input job stream is shown in Fig. 5.

Following a job card, a system macro command //LSALS is used to access the object program library of SALS system. Then, the user supplies a main program and a model calculation subroutine MODELD. The main program simply calls SALS system and allocates a working area of 4000 words in the unlabelled COMMON block. MODELD calculates the user's model and its Jacobian matrix. In this example, the model is a sum of decaying exponential functions

$$f_i(x) = \Sigma_k a_k \exp(-b_k q_i), \qquad (27)$$

where the coefficients a_1, b_1, a_2, b_2, etc. form a vector of variable parameters **x**.

Then, input data of SALS system are read. SALS uses a command method to allow flexible controls and convenient inputs: every command has a name and six-or-less numerical control data, and is read in a standardized format of (A10, 6F10.0). PROBLEM command initiates the input of a problem and is at the top of the hierarchical control of algorithms, statistics, and outputs. PARAMETER command leads the input of variable parameters; the name and the initial value are read for each parameter in the order as expected in MODELD

subroutine. DATA command leads the input of the observed data and specifies how to handle them; the name (or the running number i in this example), the coordinate q_i, and the observed value y_i are read for each observation. The uncertainties σ_i of these observed data are assumed constant, 0.05, as specified on DATA command. ENDSALS command terminates the SALS input data, while //END terminates the job.

This job works to fit 30 observed values to a nonlinear exponential-decay model. It uses the biweight method for robust estimation, the Gauss-Newton method with damping option for the nonlinear least-squares algorithm, and the modified Gram-Schmidt method for the linear algorithm. It outputs the results of fitting at every cycle of weight adjustment and a complete set of statistical information at the end of the whole fitting. All these functions of SALS system are controlled so easily as shown in Fig. 5. This easy control is achieved by the block structure of inputs and the hierarchical default setting as described in the following sections.

B. Block Structure of SALS Inputs

SALS system has about 30 commands for the control of its inputs, algorithms, statistics, and outputs. Every command is read in a standard format as a Fortran input card having the registered name and six-or-less numerical control data. Figure 6 illustrates the rule of arranging these commands as a set of SALS input data. It should be noticed,

however, that only the four commands and the two kinds of data marked with asteriks in Fig. 6 are always necessary and that all other commands may be omitted as shown in the example in Fig. 5.

A set of SALS input data is formed by one or more problem blocks and an ENDSALS command. Then, each problem block consists of a PROBLEM command, a specification block, one-or-more method blocks, one-or-more parameter blocks, one-or-more observed data blocks, and an END command; these blocks must be arranged in this order. If two or more blocks of the same kind are read, the fitting job is done for all the combinations of such blocks independently.

PROBLEM command initiates the input of a problem block and also controls the algorithms, statistics, and outputs.

The specification block has four commands which may be read in any order. OUTPUT command specifies the output files for the real-time monitoring and for the bulk result. MODLCHECK command initiates the option to check the Jacobian matrix.

A method block specifies the details of the fitting method to be used. It usually has one algorithm block; if it has two or more algorithm blocks, such fitting algorithms are executed in sequence to get one final result. At the top of an algorithm block, one of the algorithm commands (i.e., SALS, LINEAR, GAUNEW, MARQUARDT, and HYBRID) is specified; LINEAR command actually uses the Gauss-Newton

method with only one cycle of iteration, and SALS command requires the default algorithm. Then, ten more commands for specifying the details of the treatment are read in a free order: LLS controls the linear least-squares algorithm, while TOLERANCE, the convergence and termination criteria in the nonlinear least-squares method. W-ADJUST command specifies the robust estimation method. PRINT, MONITOR, TRACE, and DUMP control various types of outputs in the fitting process, while STATISTICS the output of statistical information.

In the parameter block, PIO command is optionally placed at the top to use user-coded subroutines for the input and output of the parameters. PARAMETER command is always necessary; it shows the heading of the body of parameter inputs and gives the upper limit of the number of parameters to be read. Then, the parameters are read one by one specifying the names, initial values, and the indices whether to handle them as either free, loosely-bound, or rigidly-fixed parameters. ENDPARA command terminates the input of the paremeters.

The data block is similar to the parameter block: DATA command is always necessary and gives the upper limit of the number of input observations, index whether to read the uncertainties or weights, and the standard error σ_0, etc. Then, the observed data are input one by one: they consist of the names (or numberings), coordinates q_i, observed values y_i, and the uncertainties σ_i or the weights w_i. ENDDATA command terminates the input of the observed data. The

observed data are usually read in the standard format as shown in the example in Fig. 5. Moreover, by use of a DIO command more flexible ways are possible in input: For handling a large number of observed data, it is convenient to set up a file of observed data separately from the SALS input file and to use a user subroutine to read in a more suitable format. INITIATE and FINISH commands in the observed data block allow users to attach some extraneous processing before and after the fitting with SALS.

C. Hierarchy in Default Setting

As mentioned above, a large number of commands and their control data are introduced to control various algorithms, statistical treatments, and outputs. It can be too much, however, for a user to remember and specify all these controls. Thus, in SALS system it is allowed for users to omit unimportant commands or their control data in the input: This is achieved by an introduction of a hierarchy for setting default options.

Figure 7 illustrates the hierarchy. Default options are set with reference to the upper-rank control data in the following three cases: (i) when a command is omitted in the input, (ii) when zero (or blank) is read as a control datum of a command, or (iii) when an improper value is read as a control datum. On the other hand, if the lower-rank control data are explicitly specified by the user, the user's inputs are effective.

Six control data of the PROBLEM command are at the top of the hierarchy, as shown in the left-most column in Fig. 7. LMODEL specifies the type or user's model subroutine, (i.e. either MODELF, MODELD, or MODELN) and also controls the MODLCHECK command. LALGO selects the algorithm: it chooses a proper algorithm command and further sets the default options for the LLS and TOLERANCE commands. The third control datum LCYCLE sets the limiting number of iteration cycles of nonlinear least-squares refinement in accordance with the algorithm command. LWADJ0 controls the W-ADJUST command for robust estimation. For adjusting the output, LOUT0 is at the top of the hierarchy; it controls five output commands, and each of them has its main control datum and five-or-less subsiduary control data. The statistical output is controlled by LSTAT0; but if LSTAT0 is not specified explicitly, its default value is set by the output control datum LOUT0.

By virtue of this scheme of default setting, the user may omit most of the commands in Fig. 6; for ordinary use of SALS, he should set only the PROBLEM command as shown in the example in Fig. 5. If he wants some non-standard treatment, he should specify only the related command and its control data.

D. Flow of Execution

The flow of execution in the SALS system is schematically shown in Fig. 8. Main structure is expressed by multi-fold DO loops. As shown in Fig. 6, SALS accept multiple problems,

each of which may have multiple sets of methods, parameters, and observed data; thus, the outermost four-fold DO loops are formed. The observed data set is read at the end of input, and is executed immediately with multiple sets of methods and parameters so that multiple sets of observed data need not be stored in the memory. In each fitting task with a set of method, parameters, and observed data, there are DO loops corresponding to fitting algorithms, weight-adjustment methods, cycles of weight adjustment, and finally cycles of nonlinear least-squares algorithms. In actual uses, most of these DO loops shown in Fig. 8 turn only once; but this DO-loop structure allows flexible use of SALS for wide variety of user's demands.

E. Outputs and Error Messages

Intermediate results as well as the final ones are readily outputted according to the directions of five commands: MESSAGE, PRINT, TRACE, MONITOR, and DUMP. These commands have their own characteristic features. MESSAGE controls the outputs outside the least-squares algorithms. PRINT specifies the outputs of the final results, while TRACE and MONITOR control the intermediate outputs, in linear algorithms in initial cycles and in nonlinear algorithms, respectively. DUMP command dumps out the labelled COMMON blocks and the working area WK. Statistical information described in Section 4 is output according to the directions of the STATISTICS command. Thus, users can tailor the outputs for various purposes: such as

routine runs, tentative fitting runs, watching the behavior of nonlinear algorithms, comparing the performance of linear algorithms, debugging of the system, and so on.

In addition to the error checks of inputs, abnormal behaviors in execution are also checked. On detecting an error, SALS system outputs an error message and either stops or goes ahead after a default fix-up. Error messages can not be suppressed even at the lowest level of output.

F. Optional User-Coded Subroutines

Besides the user's main program and the theoretical model subroutine, the user may optionally attach several subroutines for his own purposes: Subroutines PAPAIN and DATAIN may be coded to read the parameters and observations in suitable formats, on separate files, and/or after some preliminary treatment. Output subroutines PAROUT, DATOUT, and CALOUT are also useful to extend the SALS facilities: Final parameter values and their error matrix may be output by PAROUT onto a separate file for a succeeding job. By coding a CALOUT, a table of observed and calculated values can be printed in a format specialized for the problem, and illustrative graphs of calculated values and residuals can be drawn on a line printer or a display. For these input/output subroutines necessary values are transferred to/from SALS in the arguments of subroutines.

In addition, SALS allows other extraneous processing by users before and after the fitting: Four commands INITIATE,

AINITIATE, AFINISH, and FINISH call user's subroutines at the moments of execution of (a), (b), (c), and (d) shown in Fig. 8, respectively.

6. HISTORY OF DEVELOPMENT

The present project was started in 1975 by a group of volunteer scientists (SALS Group), who are working in the fields of mathematics, statistics, computer science, physics, and chemistry, and belonging to various universities and research institutes in Tokyo area. Basic specifications were discussed by the whole group [32, 33], and the present system has been built up by the present authors. Passing through a pilot system in 1977 and a preliminary version in 1978, the present system (SALS Standard System, Version 2) has been completed in 1979 [36, 41].

A flowcharting method FORTFLOW [34], which uses the flowcharts keeping one-to-one correspondences with Fortran statements, was introduced in the actual programming for the sake of easier initial coding and better maintenance at the same time.

Fortran source programs as well as object modules are available to public use, and User's Reference Mannuals are published in Japanese [37, 38] and under preparation in English.

7. DISCUSSION

As described so far, the present SALS system has extensive features in controls, algorithms, and statistics as a general-purpose program package for the data analysis. The present version is designed as a standard, core system of the family of extended SALS systems to be developed in the future. Thus, several important features are intendedly excluded from the present version. Such points are:

(a) Large-scale problems: The present SALS system assumes that all the necessary information for a fitting can be stored in the main memory. Inside the fitting algorithms no file operation is used, even though data files can be used for the inputs and outputs. For large-scale problems, special algorithms should be introduced to allow the handling of temporary data files while keeping high performance.

(b) Constraints: SALS system allows equality constraints in a few ways: (i) A variable parameter may be fixed at its initial value or bound loosely around it with given uncertainty. (ii) Equality relations of parameters, e.g. $g(\mathbf{x}) = \hat{g}^{(0)} \pm \sigma_g$, can be introduced in SALS by simply regarding them as additional observations. (iii) The user's theoretical model subroutines may be coded so as to eliminate the redundant parameters in the equality constraints. Inequality constraints, however, are not acceptable in SALS; introduction of them would require linear and nonlinear programming algorithms [30].

(c) Double-precision calculation: The present version

of SALS is coded in single precision with partial double-precision calculation in critical parts such as calculation of vector sums. For the problems requiring higher precision, double-precision version of SALS is also available, which is converted with a preprocessor from the standard version. Maintenance and further extension will be done principally on the single-precision version.

(d) General minimization methods: Various methods have recently been developed for minimizing a general nonlinear objective function of parameters $F(\mathbf{x})$ [13]. The nonlinear algorithms installed in the present SALS system all take advantage of the fact that the sum of squares of residuals is to be minimized, because they are effective and efficient in wide range of nonlinear problems. Nevertheless, general minimization methods may be more widely applicable to the problems with stronger nonlinearity.

(e) Correlated observations: In the usual least-squares analysis, the observed data are assumed to be independent, or uncorrelated. There are, however, cases of correlated observations, e.g. data sampled from a continuous measurement and indirect data reduced from some other direct observations. Such correlated observations should be treated with a non-diagonal weight matrix. The present version of SALS does not handle it explicitly, but the user can set up the normal equation with a non-diagonal weight matrix in his model subroutine MODELN.

(f) Interactive use: The present version of SALS mostly expects batch runs, even though it has a feature to monitor the processing messages and error messages on real time while writing the bulk results on another file for the line printer. In order to make a version for interactive use, the following modifications are planned: (i) Free formats and/or keyword methods are introduced in the command and data inputs. (ii) Controls, parameters, and observed data are kept on separate files, and then they are combined by an execution command for fitting. (iii) Intermediate as well as final results are outputted on files in order to prepare for user's quit/restart operation. Most of these modifications can be achieved by remodelling only the control/input subroutines of Group 1 mentioned in Section 2B.

ACKNOWLEDGEMENTS

A large number of people have contributed and supported the present project for these four and half years of development. Among others, Prof. Hayato Togawa of Nihon University, Dr. Kunio Tanabe of Institute for Mathematical Statistics, Mr. Masanori Rikihisa of Waseda University, Dr. Tetsuzo Ito of the Institute of Physical and Chemical Research, Dr. Masaaki Sibuya of IBM Japan, Ltd., Prof. Takeo Yamamoto of The University of Tokyo, Dr. Hajime Ohiwa of Toyohashi Technical Science University, Dr. Akio Takenaka of Tokyo Institute of Technology, Prof. Katsunosuke Machida of The University of

Kyoto, and Prof. Isao Suzuki of University of Tsukuba contributed through discussions and coding in parts; Miss Sumie Ueda, Mr. Yukio Yanagisawa, Mr. Mamoru Hirajima, Mr. Hisao Matsuda, and Miss Ayako Kawamata assisted in coding and debugging; and Miss Reiko Toda, Miss Hisako Okamoto, Miss Kumi Takemura, and Mrs. Masako Nakagawa helped in secretarial work. Prof. Masashi Okamoto of Osaka University and Prof. Kozo Kuchitsu of the University of Tokyo gave advices and encouraged the project; Dr. Yukihiko Karaki and Miss Sanae Takahashi of the Computer Centre of the University of Tokyo gave administrative help. The present authors wish to thank all these people who are cited or not cited above.

REFERENCES

[1] H. Akaike. Information theory and an extension of the maximum likelihood principle. Second International Symposium on Information Theory, B. N. Petrov and F. Csáki, eds., Hungarian Academy of Sciences, Budapest, 1973, pp. 267-281.

[2] H. Akaike. A new look at the statistical model identification. IEEE Trans. Automatic Control, AC-19 (1974). pp. 716-723.

[3] Americal Statistical Association. The Index of Available Statistical Software. Proc. Statistical Computing Section, American Statistical Association, Washington, 1977.

[4] D. F. Andrews, P. J. Bickel, F. R. Hampel, P. J. Huber, W. H. Rogers, and J. W. Tukey. Robust Estimation of Location. Princeton Univ. Press, Princeton, 1972.

[5] A. J. Barr, J. H. Goodnight, J. P. Sall, and J. T. Helwig. A User's Guide to SAS 76. SAS Institute Inc., Raleigh, NC, 1976.

[6] Å. Björck. Solving linear least squares problems by Gram-Schmidt orthogonalization. BIT, 7(1967). pp. 1-21.

[7] Å. Björck. Iterative refinement of linear least squares solutions, I. BIT, 7(1967). pp. 257-278.

[8] Å. Björck. Iterative refinement of linear least squares solution, II. BIT, 8(1968). pp. 8-30.

[9] Å. Björck and G. H. Golub. Iterative refinement of linear least squares solutions by Householder transformation. BIT, 7(1967). pp. 322-337.

[10] S. Brandt. Statistical and Computational Methods in Data Analysis, 2nd revised ed. North-Holland, Amsterdam, 1976.

[11] P. Businger and G. H. Golub. Linear least squares solutions by Householder transformations. Num. Math., 7(1965). pp. 269-276.

[12] J. M. Chambers. A computer system for fitting models to data. Appl. Statist., 18(1969). pp. 249-263.

[13] J. M. Chambers. Computational Methods for Data Analysis. Wiley, New York, 1977.

[14] W. J. Dixon, ed. BMD Biomedical Computer Programs. Univ. of California Press, Berkeley, 1973.

[15] W. J. Dixon, ed. BMDP Biomedical Computer Programs. Univ. of California Press, Berkeley, 1975.

[16] W. J. Dixon and R. I. Jennrich. Scope, impact, and status of packaged statistical programs. Ann. Rev. Biophys. & Bioengin., 1(1972). pp. 505-527.

[17] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. LINPACK User's Guide. Society of Industrial and Applied Mathematics, Philadelphia, 1979.

[18] R. Fletcher. A modified Marquardt subroutine for nonlinear least squares. Tech. Report AERE R6799, U. K. Atomic Energy Authority Research Establishment, Harwell, U. K.,

1971.

[19] G. E. Forsythe and C. B. Moler. Computer Solution of Linear Algebraic Systems. Prentice-Hall, Englewood Cliffs, N. J., 1967.

[20] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. Numer. Math., 14(1970). pp. 403-420.

[21] W. C. Hamilton. Statistics in Physical Science. Ronald Press, New York, 1963.

[22] H. O. Hartley. The modified Gauss-Newton method for the fitting of nonlinear regression functions by least squares. Technometrics, 3(1961). pp. 269-280.

[23] HARWELL Subroutine Library. U. K. Atomic Energy Research Establishment, Harwell, U. K.

[24] J. Hilsenrath, G. G. Ziegler, C. G. Messina, P. J. Walsh, and R. Herbold. OMNITAB, A Computer Program for Statistical and Numerical Analysis, NBS Handbook 101. U. S. Government Printing Office, Washington, 1966.

[25] P. J. Huber. Robust statistics; a review. Ann. Math. Statist., 43(1972). pp. 1041-1067.

[26] F. James and M. Roos. MINUIT - A system for function minimization and analysis of the parameter errors and correlations. Comp. Phys. Comm., 10(1975). pp. 343-367.

[27] G. D. Knoff. MLAB: An On-line Modelling Laboratory, Reference Manual. National Institute of Health, Bethesda, MD, 1979.

[28] C. L. Lawson and R. J. Hanson. Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs, N. J., 1974.

[29] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. Quart. appl. Math., 2(1944). pp. 164-168.

[30] D. G. Luenberger. Intruduction to Linear and Nonlinear Programming. Addison-Wesley, Reading, MA, 1974.

[31] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Indust. Appl. Math., 11(1963). pp. 431-441.

[32] T. Nakagawa. Developement of a standard program for least squares. (1), (2) History and goal of the development of SALS system. Univ. Tokyo Comp. Centre News, 8(1976). pp. (5) 68-72, (6) 89-95.

[33] T. Nakagawa, ed. Proceedings of Symposium on Data Analysis for Natural Science. SALS Group, Tokyo, 1977.

[34] T. Nakagawa. Introduction to Fortran Programming for Scientists and Engineers. Tokyo Kagaku Dojin, Tokyo, 1978.

[35] T. Nakagawa and K. Kuchitsu. Data fitting — its ideal and reality. J. Spectrosc. Soc. Japan, 24(1975). pp. 109-126, 165-187.

[36] T. Nakagawa and Y. Oyanagi. Developement of a standard program for least squares. (5) An overview of the developement of SALS system. Univ. Tokyo Comp. Centre News, 11(1979). pp. 63-72.

[37] T. Nakagawa and Y. Oyanagi. Program System for Statistical Analysis with Least Squares Fitting, SALS (Version 2). User's Manual. Part 1, Basic Usage. Univ. of Tokyo Computer Centre, Tokyo, 1979.

[38] T. Nakagawa, Y. Oyanagi, and H. Togawa. Program System for Statistical Analysis with Least-Squares Fitting, SALS (Version 2). User's Manual. Part 2, Controls and Algorithms. Univ. of Tokyo Computer Centre, Tokyo, 1979.

[39] N. H. Nie, C. H. Hull, J. G. Jenkins, K. Steinbrenner, and D. H. Bent. SPSS Statistical Package for the Social Sciences, 2nd ed. McGraw-Hill, New York, 1975.

[40] NPL Program Library, National Physical Laboratory, Teddingron, U. K.

[41] M. Okamoto, C. Asano, and T. Nakagawa. Statistical data processing systems. Progress in Scientific Information Systems in Japan, H. Inose editor, to be published.

[42] M. J. D. Powell. A Hybrid method for nonlinear equations. Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon-Breach, London, 1970, pp. 87-114.

[43] M. J. D. Powell. A Fortran subroutine for solving systems of nonlinear algebraic equations. Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon-Breach, London, 1970, pp. 115-161.

[44] H. Ramsin and P.-A. Wedin. A comparison of some algorithms for the nonlinear least-squares problem. BIT, 17(1977). pp. 72-90.

[45] W. R. Schucany, P. D. Minton, and B. S. Shannon, Jr. A survey of statistical packages. ACM Comp. Surveys, 4(1972). pp. 65-79.

[46] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. Matrix Eigensystem Routines—EISPACK Guide, 2nd ed. Lecture Notes in Computer Science, Vol. 6, G. Goos and J. Hartmanis, eds., Springer-Verlag, Berlin, 1976.

[47] J. W. Tukey. Introduction to today's data analysis. Proceedings of the Conference on Critical Evaluation of chemical and Physical Structural Information, D. R. Lide, Jr. and M. A. Paul, eds., National Academy of Sciences, Washington, 1974, pp. 3-14.

[48] J. W. Tukey. Data analysis for molecular structure: today and tomorrow. Proceedings of the Conference on Critical Evaluation of Chemical and Physical Structural Information, D. R. Lide, Jr. and M. A. Paul, eds., National Academy of Sciences, Washington, 1974, pp. 48-58.

[49] R. H. Wampler. A report on the accuracy of some widely used least-squares computer programs. J. Amer. Statist. Assn., 65(1970). pp. 549-565.

[50] J. H. Wilkinson and C. Reinsch, eds. Linear Algebra.

Handbook for Automatic Computation, Vol. II, F. L. Bauer, A. Householder, F. W. J. Olver, H. Rutishauser, K. Samelson, and E. Stiefel, eds., Springer-Verlag, Berlin, 1971.

Table I. Basic Specifications of SALS System

Wide Applicability

- Data fitting in experimental/observational sciences.
- Nonlinear as well as linear models coded by users.
- With/without requiring Jacobian matrix.
- Dynamic memory allocation for small/large-scale problems.

Reliable Algorithms

- Five linear least-squares algorithms with/without using the normal equation.
- Singular value analysis in case of rank deficiency.
- Fast and reliable convergence in nonlinear least-squares algorithms.

Statistics and Diagnosis

- Weighted least-squares fitting.
- Fixing and loosely binding some parameters.
- Error matrix and correlation matrix of the parameters.
- Estimation errors in the calculated values.
- Diagnostic plots of residuals.
- Selection of the best-fit model with AIC.
- Robust estimation including Tukey's biweight method.

User-Oriented Input/Output

- Standard I/O routines along with optional user-made I/O.
- Handling with multiple sets of data and fitting methods.
- Command control of algorithms and outputs.
- Error checks, default settings, and error messages.
- Checking user-coded Jacobian matrix.

Maintainabitity

- Hierarchical structure of about 130 subroutines.
- Readable and portable programming in standard Fortran.
- Full documentation.

Table II. Linear Least-Squares Algorithms in SALS

Method \ Stage	Decomposition $A'=$ or $B=$	Solution $\hat{x} =$	Error matrix $\Sigma_{\hat{x}} =$
a. Modified Gram-Schmidt	$A' = QR$	$R^{-1}Q^{T}y'$	$\sigma_0^{2}R^{-1}(R^{-1})^{T}$
b. Householder	$A' = P_{(1)}^{T}..P_{(m)}^{T}\begin{bmatrix}R\\0\end{bmatrix}$ $= QR$	$R^{-1}P_{(m)}..P_{(1)}y'$ $= R^{-1}Q^{T}y'$	$\sigma_0^{2}R^{-1}(R^{-1})^{T}$
c. Singular value decomp.	$A' = T\begin{bmatrix}S\\0\end{bmatrix}U^{T}$	$U[S^{-1}, 0]T^{T}y'$	$\sigma_0^{2}US^{-2}U^{T}$
d. Cholesky	$B = R^{T}R$	$R^{-1}(R^{-1})^{T}b$	$\sigma_0^{2}R^{-1}(R^{-1})^{T}$
e. Eigenvalue decomp.	$B = UEU^{T}$	$UE^{-1}U^{T}b$	$\sigma_0^{2}UE^{-1}U^{T}$

Q : orthogonal matrix (n,m).

R : upper triangular matrix (n,m).

$P_{(i)}$: Elementary orthogonal transformation matrix (n,n).
$P_{(i)} = I - 2w_{(i)}w_{(i)}^{T}$.

T : orthogonal matrix (n,n).

U : orthogonal Matrix (m,m).

S : singular value matrix, diagonal (m,m).

E : eigenvalue matrix, diagonal (m,m), $E = S^{2}$.

Figure Captions

Fig. 1. Structure of a Fitting Program with SALS.

Subroutines with double asterisk should be provided by user. Those with single asterisk can be replaced by user's subroutines.

Fig. 2. Structure of a Model Calculation Programs.

Fig. 3. Structure of SALS System.

Number in each block specifies the subroutine group described in the text.

Fig. 4. Weight Adjustment Factor, Eq. (25), in Tukey's Biweight Method.

Fig. 5. An Example of SALS Input.

Fig. 6. Block Structure of SALS Input.

Fig. 7. Hierarcy for Setting Default Options.

Fig. 8. Flow Diagram of SALS Execution.

Fig. 1. Structure of a fitting program with SALS

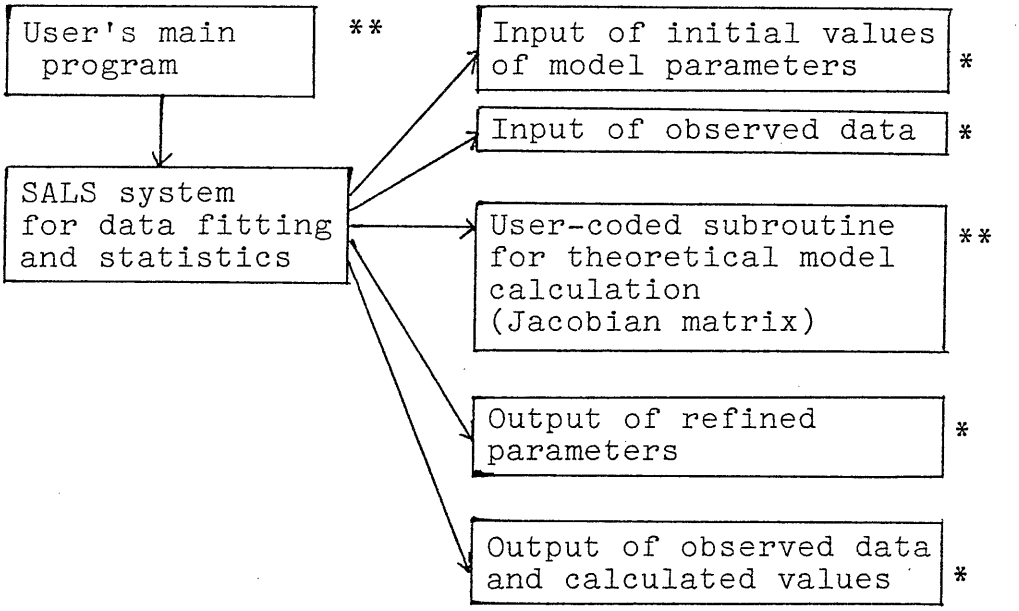
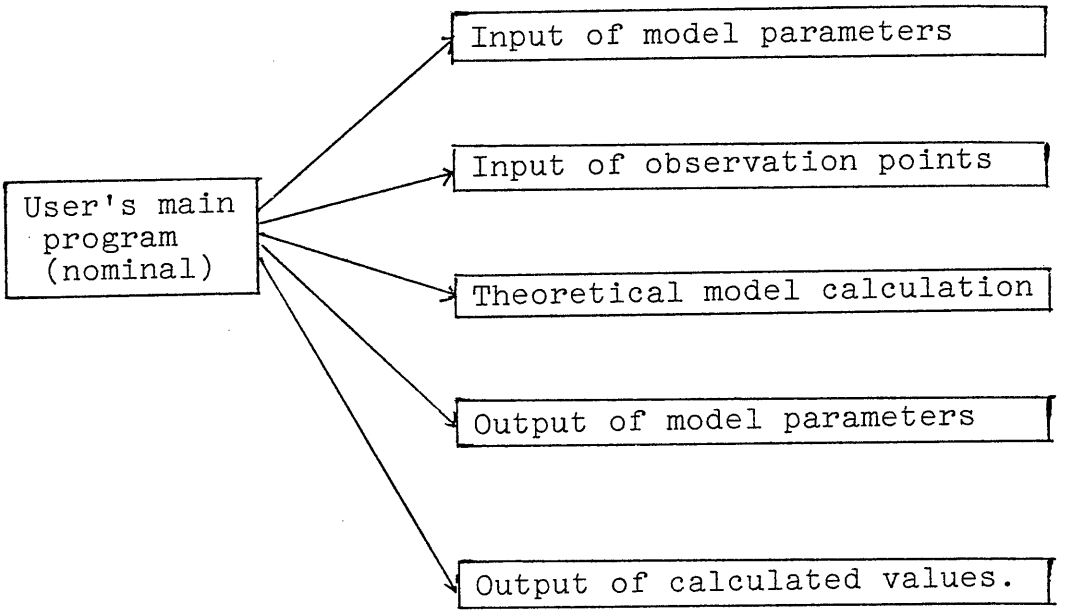


Fig. 2. Structure of a model calculation program.



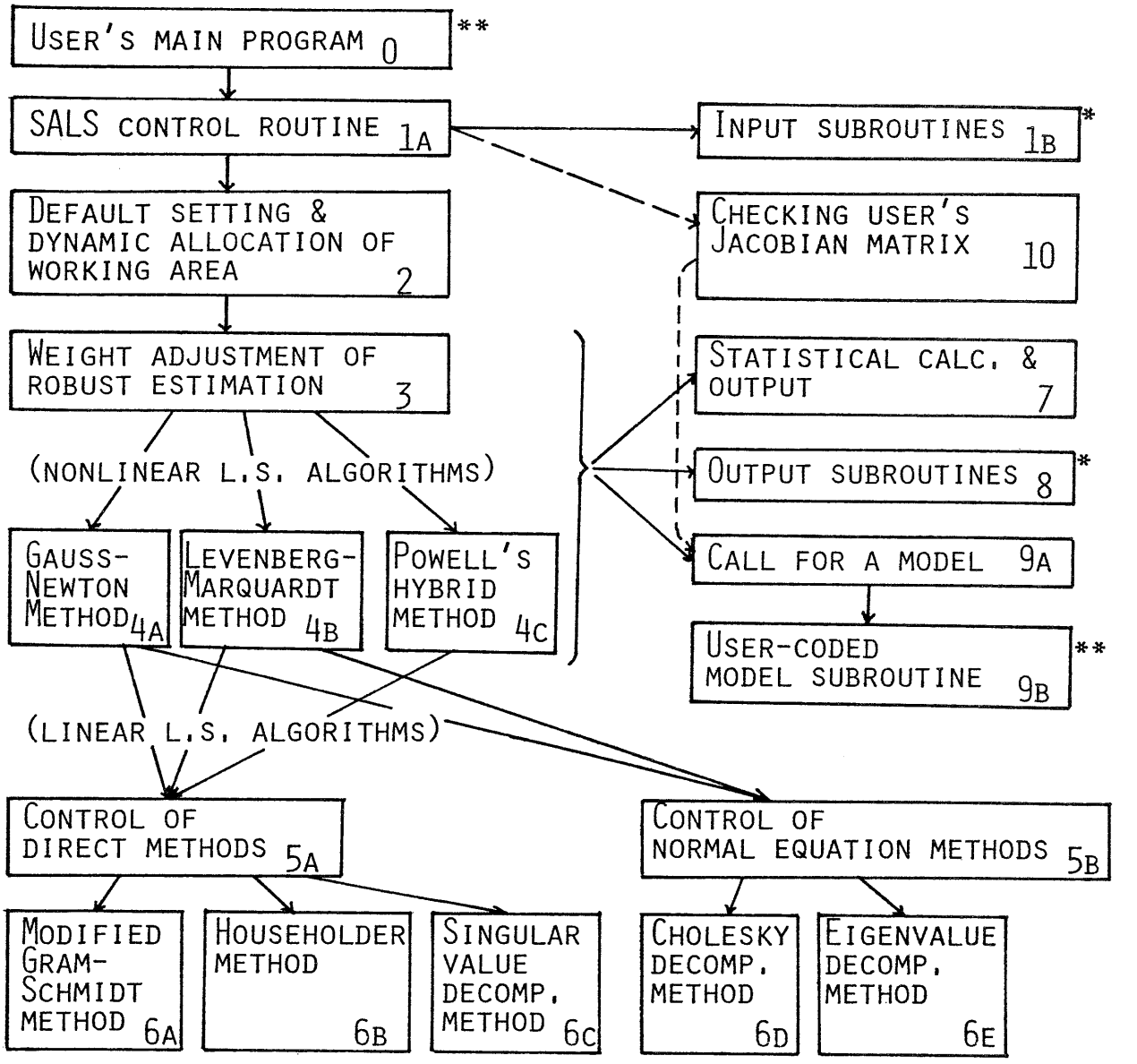


Fig. 3. Structure of SALS system

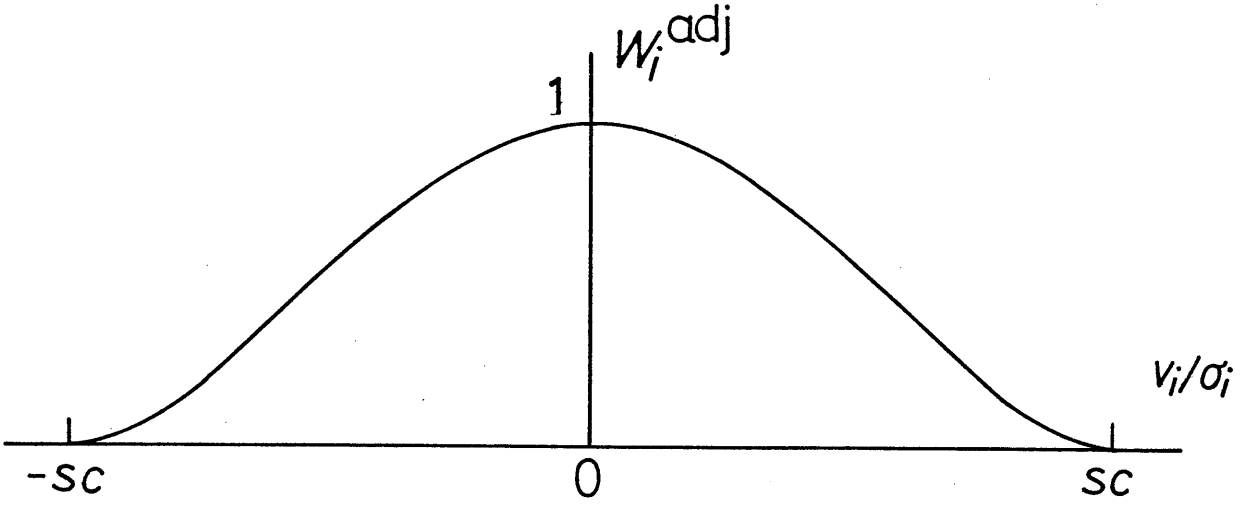
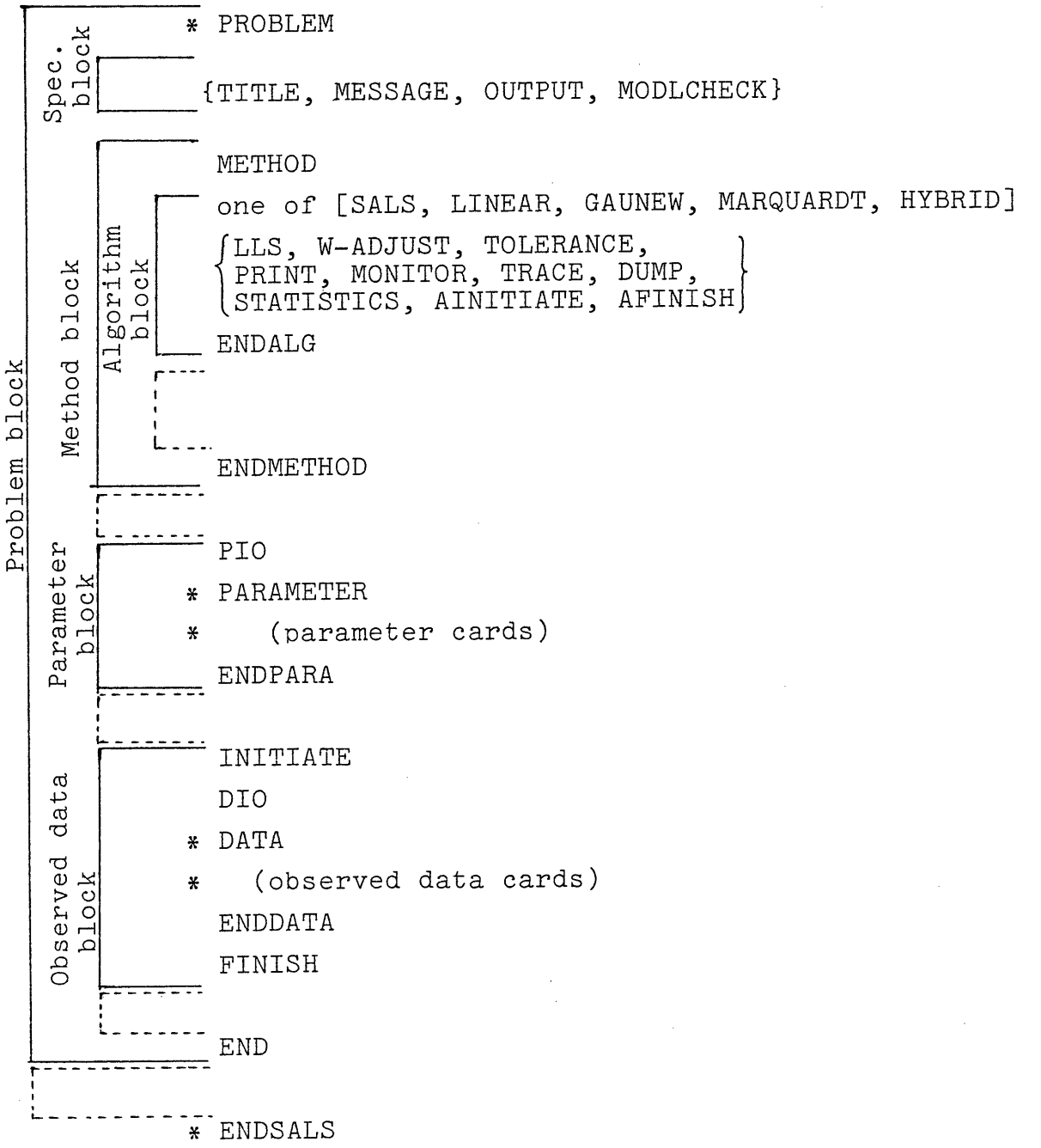


Fig. 4. Weight adjustment factor, Eq. (25), in Tukey's biweight method

```
//EX1:JOB                                                    ... Job card
//LSALS                                                      ... Access to SALS library
//FORTCG
C***  ANALYSIS OF RADIATION LIFETIMES USING SALS
      COMMON MAXWK, NWKMAX, WK(4000)
      CALL SALS(4000)                                        ... User's main program
      STOP
      END
C***  SUPERPOSED EXPONENTIAL DECAY MODEL
      SUBROUTINE MODELD(LC,NP,ND,NC,X,Q,FOBS,
     1                  F,RES,DF,LFIX,MDF)                   ... User's model subroutine
      DIMENSION X(NP),Q(ND),FOBS(ND),F(ND),RES(ND),
     1          DF(MDF,NP),LFIX(NP)
      DO 20 I=1,ND
      F(I)=0.0
      DO 10 J=1,NP,2
      E=EXP(-X(J+1)*Q(J))                                    ... f_i(x) = Σ_k a_k exp(-b_k q_i)
      T=X(J)*E
      F(I)=F(I)+T
      IF(LC.LE.0) GO TO 10                                   ... Calculation of Jacobian
      DF(I,J)=E                                                  matrix DF in case of LC=1
      DF(I,J+1)=-T*Q(I)
   10 CONTINUE
   20 RES(I)=FOBS(I)-F(I)                                    ... Residuals
      RETURN
      END
/*                                                           ... Input data
PROBLEM   2.     5.     2.     4.     2.     3.              ... Control of algorithms and outputs
PARAMETER 4.
  A1      5.0                                                ... Input of initial values
  B1      3.0                                                    of parameters
  A2      1.5
  B2      0.1
DATA      30.    1.     1.     0.05
   1      0.1    9.17                                       ... Input of observations
   2      0.2    7.66
   .       .      .
   .       .      .
   .       .      .
  30      10.0   0.14
ENDSALS                                                      ... End of SALS input
//END                                                        ... End of job
```

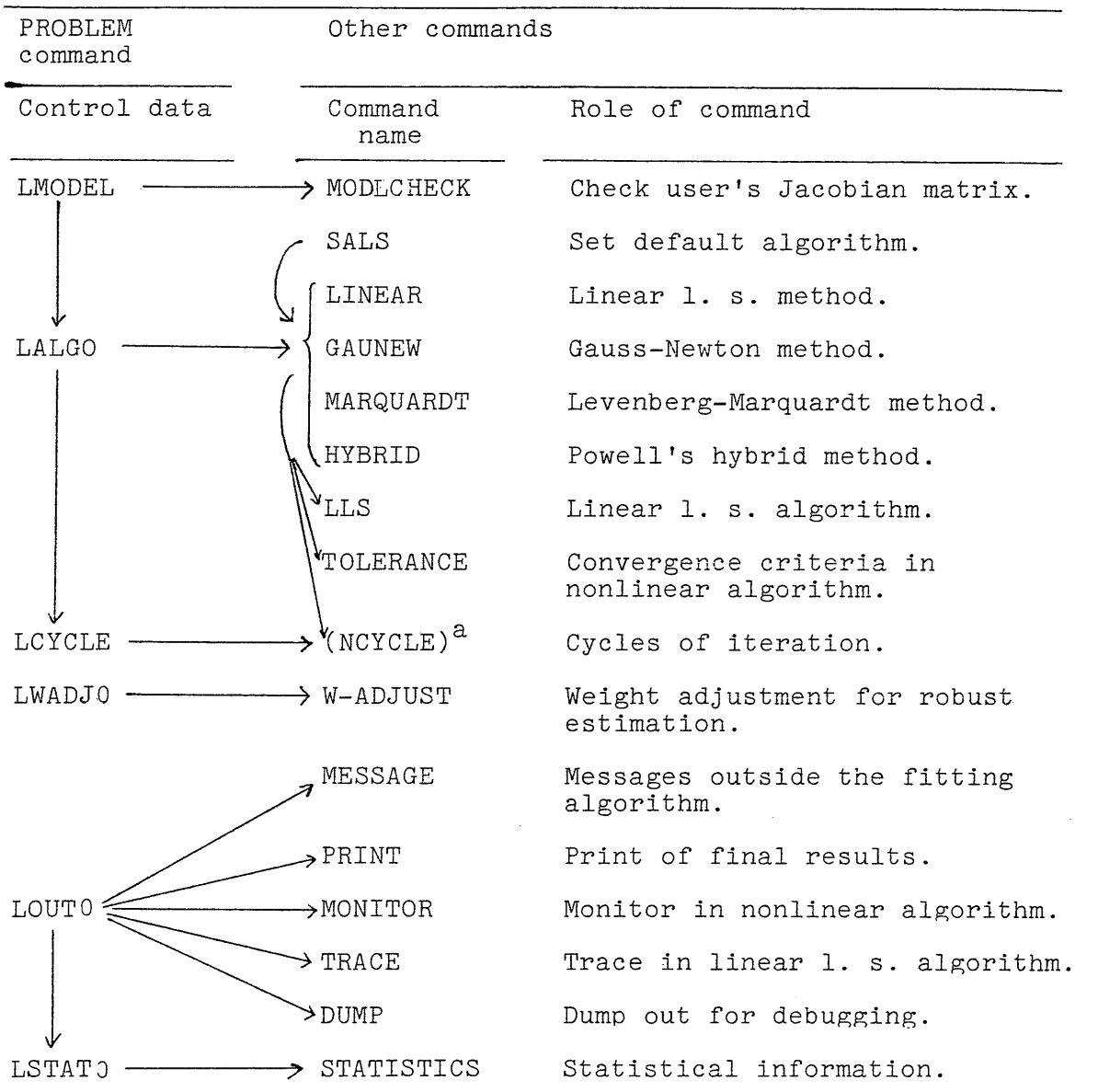
Fig. 5. An Example of SALS Input

Fig. 6. Block Structure of SALS Input



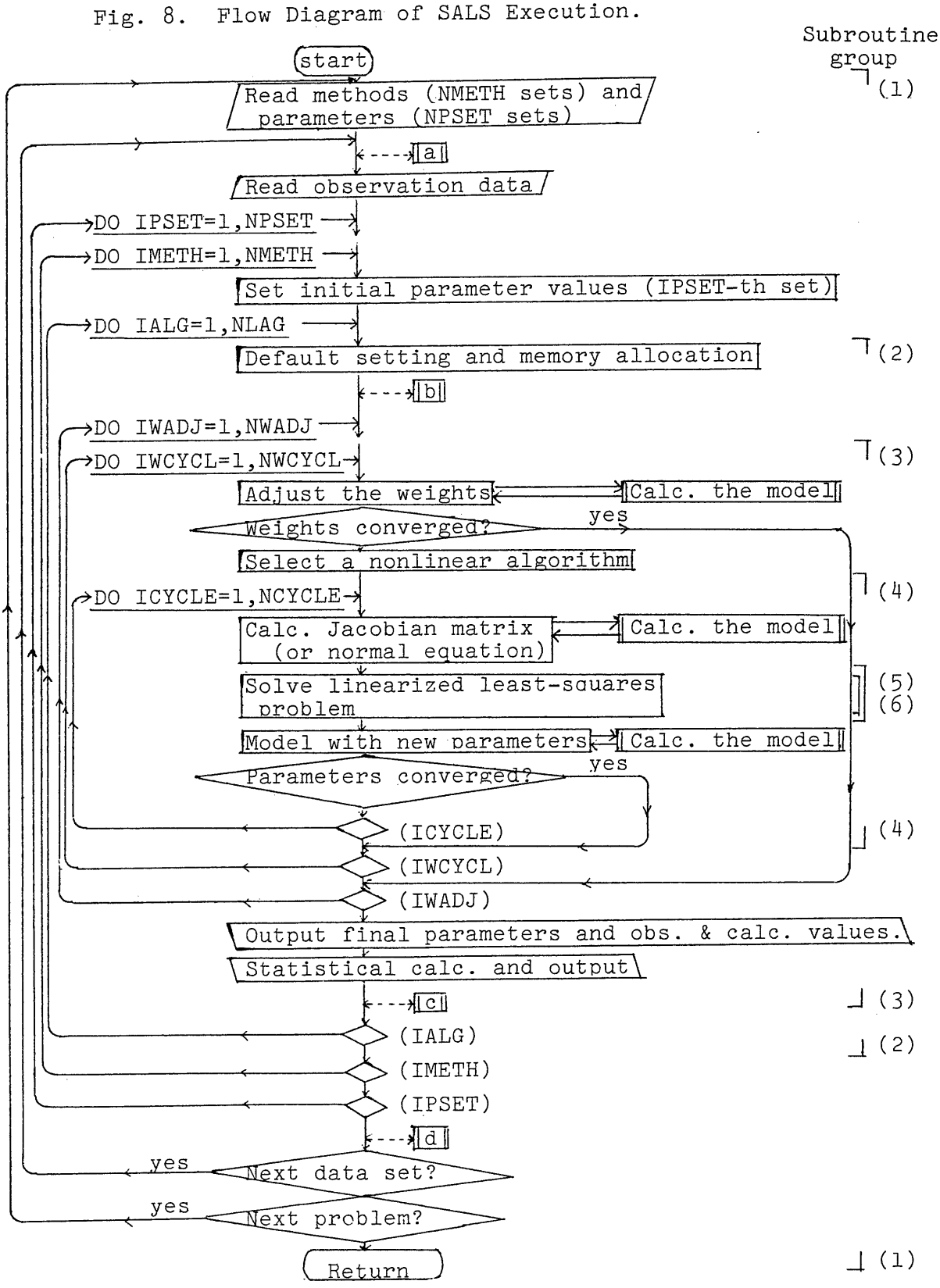
Note: *: always necessary (all others may be omitted).
: repetition of the preceding block, if necessary.
{ }: free order inside.

Fig. 7. Hierarchy for Setting Default Options



a) The first control datum of the algorithm command (GAUNEW, MARQURDT, or HYBRID).

Fig. 8. Flow Diagram of SALS Execution.



INSTITUTE OF INFORMATION SCIENCES AND ELECTRONICS
UNIVERSITY OF TSUKUBA
SAKURA-MURA, NIIHARI-GUN, IBARAKI 305 JAPAN

REPORT DOCUMENTATION PAGE

REPORT NUMBER
ISE-TR-80-13

TITLE

Program System SALS for Nonlinear Least-Squares Fitting: System Design

AUTHOR(s)

Toru Nakagawa and

Yoshio Oyanagi

REPORT DATE
February 1, 1980

NUMBER OF PAGES
54

MAIN CATEGORY
Mathematics of Computation

CR CATEGORIES
5.

KEY WORDS
SALS system, program design, program package, data analysis, model fitting, least squares, nonlinear least squares, singular value decomposition, Marquardt method, statistical analysis, parameter binding, model selection, AIC, robust estimation

ABSTRACT

A program system SALS is developed for statistical data analysis with nonlinear least-squares fitting, featuring:

a) a general-purpose full system containing input/output routines and accessing a user-coded theoretical model subroutine,
b) reliable and fast convergence in nonlinear least-squares algorithms,
c) high precision in five linear least-squares algorithms,
d) robust estimation techniques including Tukey's biweight method,
e) various statistical diagnoses including Akaike's information criterion,
f) easy-to-use commands for input and control, and
g) dynamic allocation of arrays.

The system consists about 10000 steps of standard Fortran.

SUPPLEMENTARY NOTES