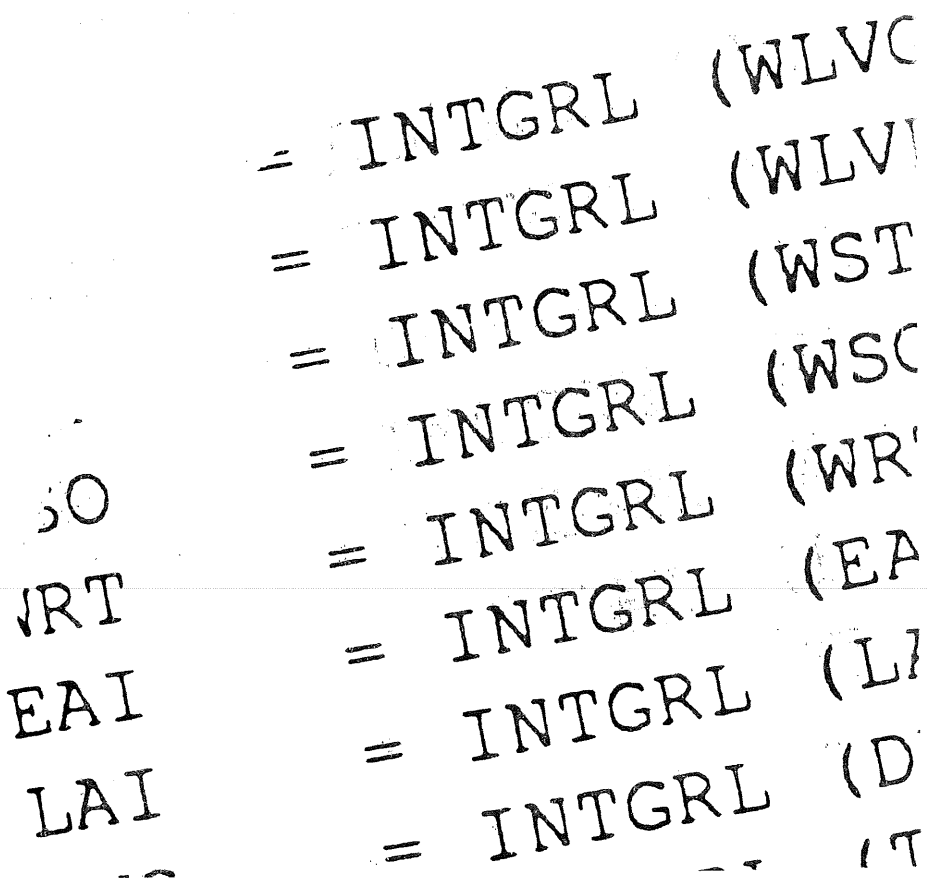
Centre for Agrobiological Research
Wageningen Agricultural University

FSEOPT a FORTRAN Program for Calibration and Uncertainty analysis of Simulation Models

Simulation Report CABO-TT, no. 24

W. Stol, D.I. Rouse, D.W.G. van Kraalingen, O. Klepper

cabo-dlo



May 1992

Simulation Reports CABO-TT

FSEOPT a FORTRAN Program for Calibration and Uncertainty analysis of Simulation Models

W. Stol, D.I. Rouse, D.W.G. van Kraalingen and O. Klepper

Simulation Report CABO-TT no. 24

A joint publication of

Centre for Agrobiological Research (CABO-DLO)

and

Department of Theoretical Production Ecology, Agricultural University

Wageningen 1992

Simulation Reports CABO-TT

Simulation Reports CABO-TT is a series giving supplementary information on agricultural simulation models that have been published elsewhere. Knowledge of those publications will generally be necessary in order to be able to study this material.

Simulation Reports CABO-TT describe improvements of simulation models, new applications or translations of the programs into other computer languages. Manuscripts or suggestions should be submitted to:
H. van Keulen (CABO-DLO) or J. Goudriaan (TPE).

Simulation Reports CABO-TT are issued by CABO-DLO and TPE and they are available on request. Announcements of new reports will be issued regularly. Addresses of those who are interested in the announcements will be put on a mailing list on request.

Address

Simulation Reports CABO-TT
P.O. Box 14
6700 AA Wageningen
Netherlands

Authors affiliation

Willem Stol
Centre for Agrobiological Research - CABO-DLO
P.O. Box 14, 6700 AK Wageningen, The Netherlands

Doug Rouse, visiting scientist,
Department of Theoretical Production Ecology
Present address:
University of Wisconsin, Department of Plant Pathology
1630 Lindendrive, Madison WI 53706,
Wisconsin, USA

Daniel van Kraalingen
Centre for Agrobiological Research - CABO-DLO
P.O. Box 14, 6700 AK Wageningen, The Netherlands

Olivier Klepper
National Institute of Public Health and Environmental Protection
Centre for Mathemathical Methods
P.O. Box 1, 3720 BA Bilthoven, The Netherlands

Contents

Preface

The development of the program presented in this report was initiated by C.T. de Wit who pointed at the work of Olivier Klepper within the Delta Institute. During 1988, a calibration study was carried out on the potato implementation of SUCROS87. The results of this study were presented within the CABO/TPE simulation group and at the First International Workshop on Potato Modelling of the EAPR held in Wageningen, May 29-31,1990.

The FORTRAN program developed in 1988 on the basis of the PRICE algorithm was supplied to various scientists for calibration studies. The program was subsequently used in calibration studies on: photosynthesis-light response curves, combination of radar-backscatter data and crop growth models (SUCROS), soil-water balance, mineralization of nitrogen from organic matter, dry matter partitioning in potatoes and the phenological development of oil-seed rape.

Although most of the studies carried out yielded satisfactory results, the program was difficult to understand for less-experienced FORTRAN-programmers, due to the complex communication between the calibration program and the simulation model. At the suggestion of Peter Kooman, we redefined the program, included suggestions of Michiel Jansen on the Downhill Simplex method of Nelder and Mead and adapted the program to the FORTRAN Simulation Environment (FSE) by van Kraalingen (1991). The result is a program package that will help modellers within the CABO/TPE simulation group and others in executing fast calibration experiments.

We thank Kees Rappoldt for suggestions on the development of this versatile calibration package and critical comments on the report. We hope that this program will give colleagues in the field of modelling research the opportunity to carry out calibration studies and sensitivity analyses to study model behaviour which results in well calibrated, highly applicable models with less programming efforts.

Wageningen, Madison, may 1992

Willem Stol
Doug Rouse
Daniel van Kraalingen
Olivier Klepper

1 Introduction

Explanatory agro-ecological simulation models contain several sources of uncertainty. One of these sources is associated with parameter values used in the models. Some parameter values are obtained from previous research reported in the scientific literature and some are estimated from experimental research conducted locally. Whether a parameter estimate is obtained from the literature or from measurements by the modeller, there may be a range of biologically plausible values to choose from. Subsequent to initial choice of parameter values the final values used in a model are often selected on the basis of comparison between model output and one or more field datasets. The procedure used most often for choosing final parameter values is best described as "trial and error".

The subjective "trial and error" approach is problematic, because there is no way of knowing how good the final choice of parameters is relative to other possible choices. Thus it is impossible to judge how well the model actually corresponds to the important features of the system being modelled. A solution to this problem would be to use a mathematical algorithm for finding the best combinations of model parameters for a given set of field data. There are a number of mathematical approaches to calibration of simulation models (Klepper, 1989, Tarantola, 1987, Press et al., 1986, Van Straten, 1985). The program presented here consists of two of these algorithms. The first one is a controlled random search procedure for global optimization developed by Price (1979). The second one, which is local, is the Downhill-Simplex method from Nelder and Mead (1965) as implemented in Numerical Recipes (Press et al., 1986). The Price algorithm was adapted to the calibration of simulation models by Klepper (1989). This algorithm has been used in a study by Klepper and Rouse (1991) to demonstrate its applicability to crop growth simulation models. It has been used to examine the transportability (Rouse et al., 1991) of a potato growth model based on SUCROS87 (Spitters et al., 1987). The Downhill Simplex method was added later as a faster alternative to the Price-method which is known as a large consumer of computer time, due to its thoroughness in (random) search. Both algorithms will be briefly described. A program called FSEOPT that is an implementation of both algorithms has been written in FORTRAN. A description of FSEOPT will be given along with instructions for its use.

The Price algorithm, as implemented in FSEOPT, generates histograms of parameter values from parametersets found to be acceptable by the algorithm. Both algorithms can either use the sum of absolute residuals or the square root of the sum of squares of the residuals between model output and field data as the basis for determining acceptability of sets of parameters. Other criteria for determining acceptability of parametersets can be introduced by the user. The program also finds the minimum and maximum output values at each sampling moment before and after calibration. A number of statistical programs that can be used to analyse the output of the calibration program to study model behaviour are given in Appendix D. The program FSEOPT is especially suited for calibration of simulation models implemented in the FORTRAN Simulation Environment (FSE). The FSE system for crop growth simulation, recently developed by van Kraalingen (1991), is a set of FORTRAN programs defining a general simulation algorithm in which simulation runs proper can be executed.

To obtain the FSEOPT program, as well as the full TTUTIL library on floppy disk, write to:

Centre for Agrobiological Research,
P.O. Box 14,
6700 AA Wageningen,
The Netherlands.

or:

Department of Theoretical Production Ecology,
Agricultural University,
P.O. Box 430,
6700 AK Wageningen,
The Netherlands.

2 Description of methods

For application of the program it is necessary to identify those model parameters that have a large range of uncertainty and to specify this range. For some parameters there may be insufficient data to specify the range of uncertainty. In that case a guess must be made. However, the algorithms can easily handle parameters with a high degree of uncertainty so that it is possible to give a wide range initially. It is also necessary to specify a goodness of fit function to judge the correspondence between model output and experimental data.

2.1 Criteria for goodness of fit

The choice of a goodness of fit function to judge the degree of correspondence between model output and experimental data depends on the objectives of the researcher. These objectives dictate which state variables will be considered in the study and what goodness of fit function will be chosen.

As an example, the objective may be to determine whether a wheat crop growth model behaves similar to reality with respect to biomass production. In this case the dry weights of stems, leaves and storage organs might be chosen as state variables to be compared with experimental data. It is necessary to express the criterion for goodness of fit ('performance') between model output and experimental data as a single number. In the program FSEOPT the goodness of fit value is calculated using the expression:

$$QT'(i) = \sqrt[IQT]{\sum_{k=1}^{n} \left| \left(\frac{d_{ik} - m_{ik}}{d_{ik} + 10^{-8}} \right) \right|^{IQT}} \quad , (IQT = 1,2) \qquad (1)$$

where d_{ik} and m_{ik} represent the experimental data and the model output respectively, and k is the number of data points over time for the i^{th} state variable. The number of data points k can include: data from different experiments, different harvest dates within experiments and different replicates within harvest dates. The expression $|d_{ik} - m_{ik}|$ denotes the absolute value of $d_{ik} - m_{ik}$. IQT is an optional switch which makes it possible to choose between the L_1 norm, the sum of absolute residuals (IQT=1), or the L_2 norm, the square root of the sum of squares of residuals (IQT=2) (Tarantola, 1987). When the variable IQT has the value 2, outliers have a greater influence on the value of QT.

If several state variables are taken into account in the calibration procedure, the maximum value of QT'(i) is taken as 'over-all' QT using the expression:

$$QT = \max (QT'(i), i = 1, 2,...,n) \qquad (2)$$

where n is the number of different output (state-) variables considered and QT'(i) represents the goodness of fit value of the residuals for the i^{th} state variable.

Alternative formulations for evaluating the degree of correspondence between model output and experimental data have been discussed in the literature (Klepper, 1989, Beck, 1982, Tarantola, 1987, Scholten et al., 1990).

2.2 Optimization algorithms

2.2.1 Controlled Random Search method according to Price

This algorithm for reducing parameter uncertainty was first applied by Klepper (1989) to a large simulation model. He used it for calibration of an ecological model of the Oosterschelde estuary. The algorithm is adapted from that of Price (1976) which is a controlled random search procedure for finding the global minimum of a function with constraints on the independent variables. The algorithm can be visualized as consisting of two parts; the first being non-iterative while the second is iterative. In the first part a number of parametersets are generated consisting of parameter values chosen at random from their biologically plausible ranges. In the second part new parametersets are generated which replace existing sets, if the new set produces model output with a better correspondence to the experimental data than the most unfavourable existing parameterset. The optimization procedure is repeated, either a predetermined number of times or until the range of the QT-values is less then a predefined limit. In FSEOPT the range in QT-values is calculated as the fractional range from best to worst parameterset:

$$Ftol = \frac{2 \cdot |\, QtHigh-QtLow|}{|\, QtHigh+QtLow|} \qquad (3)$$

Noniterative part: Let P(1,j),.....,P(i,j) represent the parameters in the jth parameterset where i denotes the parameters to be calibrated. The first part of the algorithm generates INPS parametersets (INPS >> number of parameters) by choosing values at random from a uniform distribution over the biologically plausible range for each parameter. In the FORTRAN program this is accomplished in subroutine WRRRUN, which writes an input-file that can be read by FSE. For each parameterset generated, a goodness of fit value is calculated in the subroutine COMP as previously described. This requires running the FORTRAN Simulation Environment and submodel(s) for each parameterset to obtain values of the state variables that can be compared to the experimental data. This procedure is repeated for each experimental dataset. The subroutine COMP finally calculates the weighted residuals.

Iterative part: In the second part of the algorithm new parametersets are generated. A newly generated set replaces the worst one if its goodness of fit value is lower then that of any of the current sets of parameters. A new parameterset is obtained by choosing at random q+1 different parametersets from the existing series of INPS ones, where q is the number of parameters in each parameterset. A new parameter value P(i,*) is obtained from the equation:

$$P(i,*) = 2*G(i) - P(i,q+1) \qquad (4)$$

where G(i) is the average value of parameter i from the first q randomly chosen parametersets as represented by the equation:

$$G(i) = \frac{1}{q}\sum_{j=1}^{q} P(i,j) \qquad (5)$$

Equations 4 and 5 are executed for each parameter in the calibration procedure. Call the new parameterset, thus created P(*,*). The value of QT is calculated for P(*,*) and compared to the largest QT from the existing parametersets. If its value is smaller, that QT-value and the corresponding parameterset are replaced with both the newly calculated QT-value and P(*,*). This procedure is repeated for a preset number of iterations or until the range in QT-values meets a predefined criterion.

2.2.2 Downhill-Simplex method according to Nelder and Mead

The Downhill-Simplex method according to Nelder and Mead (adopted from Press et al.,1986) is implemented in FSEOPT as an alternative to the CRS algorithm of Price. The difference between these algorithms is that the CRS algorithm is global, whereas the Downhill simplex method may end up in a local minimum.

Non-iterative part: Is identical to that in the CRS algorithm of Price, except that the number of parametersets in the Price algorithm, INPS, is much larger than the number of parameters in the optimization, INFND, (INPS>>INFND). The number of parametersets in the Downhill Simplex method INPS, equals only INFND+1. The values of the parameters in each set are choosen randomly from within the biologically plausible range. After filling a reruns-file with these parameter values, the subroutine FSE is called to perform model calculations with the generated parametersets. After completion of the simulationruns the model output is compared to the experimental data within the subroutine COMP and QT-values are calculated for each parameterset.

Iterative part: Within a subroutine called AMOEBA the Downhill-Simplex method tries to move in various ways from the position with the highest QT-value to positions with lower QT-values. Imagine that the n points sampled represent the corners of a simplex, the most simple geometrical structure in n-1 dimensional space. The routine has four types of basic steps to 'move' this structure through the parameter space; it can reflect from the high point of the simplex through the centre of gravity of the opposite face, reflect and expand through the centre of gravity of the opposite face, contract from the high point through the centre of gravity along one dimension and contract to the low point from all directions. These search strategies are automatically handled within the subroutine AMOEBA. The algorithm continues to converge the model for a predefined number of iterations or till the calculated range in QT-values meets a predefined criterion. After convergence of the simplex within subroutine

AMOEBA, the simplex is reinitialized with the set with the lowest QT-value as basis. The process of reinitialization is repeated if it is succesful.

3 Description of main program FSEOPT and modules

(A full listing of the program is given in Appendix A)

3.1 Main program FSEOPT

The main program FSEOPT has basically three functions: it specifies the memory requirements for the calibration experiment, selects the optimization method and the output options. Since all arrays are dimensioned with passed-length declaration, array lengths can be set by specifying values of the relevant parameters. This has significant advantages for users. Table 1 provides information on the names of the parameters in use for pass-length declaration, their description and their actual length.

Table 1. Parameters for passed-length declaration in FSEOPT.

Parameter name	Description	Default length
IMXNPS	Max. number of parametersets	100
IMXPAR	Max. number of parameters	20
IMXOUT	Max. number of output-options	4

The main program FSEOPT reads two variables, IMETHD and IOUT, from the optimization-definition file 'OPTIM.DEF'. The variable IMETHD defines the preferred optimization method while the variable array IOUT selects the preferred output options. After reading these values from file, the requested optimization procedure is called to do the calibration experiment. If IMETHD equals 1 subroutine Price is selected, if it equals 2 subroutine SIMPLX is choosen. These underlying optimization routines consist of an optimization algorithm which interacts with FSEOPT-subroutines. These subroutines enable the calibration environment to execute the simulation model with a particular parameterset and compare model output with experimental data to judge model performance. Alternative optimization algorithms can be included in FSEOPT, they would consist of a subroutine with its specific code and calls to the FSEOPT subroutines to perform specified tasks. The structure of FSEOPT is schematically depicted in figure 1.

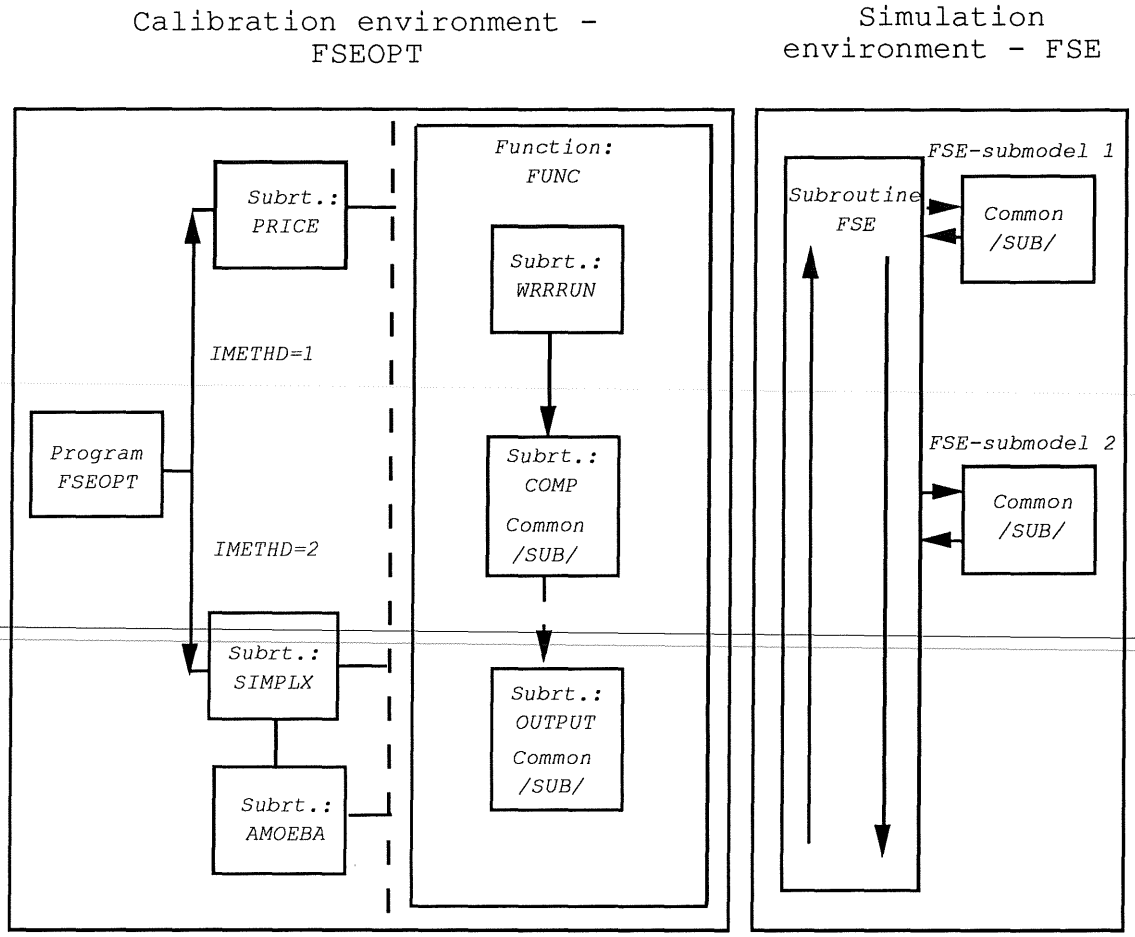


Figure 1. General structure of program FSEOPT

3.2 Subroutine Price

The subroutine Price consists of the Controlled Random Search procedure by Price and calls to the FSEOPT-submodules FUNC, HIGHLO and OUTPUT. The first (non-iterative part) of this routine instructs the function FUNC to generate INPS parametersets and execute the simulation model with each parameterset for each experimental dataset separately. The parametersets and the environmental variables are input to the simulation model, and the simulated values of the statevariables are stored in a matrix. After execution of the FSE-submodel, model output is compared to the experimental data and corresponding QT-values representing model performance are calculated. Then the function FUNC returns to the subroutine PRICE.

The next (iterative) part of the program performs the following operations; INFND+1 (INFND is the number of parameters in the calibration) parametersets are sampled at random from the INPS sets. The average value of each parameter is calculated from INFND of these sets (equation 5). A new parameter value is calculated by reflecting the INFND+1th parameter through the average value of the INFND sets (equation 5). The program has to determine

whether the newly calculated parameter value is within the Biologically Plausible Range (BPR). If a newly calculated parametervalue is outside the BPR, there are two ways to proceed. The user can select the preferred method by setting the variable IBOUND.

If the value of the variable IBOUND equals 0 the procedure restarts and samples INFND+1 sets again. If IBOUND does not equal 0, the parameter value outside the range is set to the limiting value. Next the real function FUNC is called with ITASK=3 (Table 3) to execute the runs for the selected experimental sets with this newly generated parameterset, and calculates QT by comparing with the appropriate experimental datasets. The subroutine HIGHLO is called to find the highest and lowest QT-values within the INPS sets. If the QT-value of the new parameterset is lower than the highest QT-value from the original INPS parametersets, both the QT and the new parametervalues are substituted into that particular set. This whole process proceeds iteratively till the number of iterations equals the preset value of the variable INT, or till the range in QT-values RTOL satisfies the predefined criterion FTOL.

3.3 Subroutine HIGHLO

In the subroutine HIGHLO the QT-values of the parametersets are evaluated. Two pointer-variables are assigned to a value in this subroutine, IPNT1 identifies the parameterset with the highest QT-value ('worst set') and IPNT2 identifies the parameterset with the lowest QT-value ('best set'). This routine is called each time in the iterative part of the algorithm to obtain the highest and lowest QT-values needed to calculate the range in QT-values of the parametersets. Both the values of the pointers IPNT1 and IPNT2 and the minimum and maximum value of QT, QTLOW and QTHGH, respectively, are outputs of this routine.

3.4 Subroutine SIMPLX

The Downhill Simplex method is implemented by means of the subroutines SIMPLX and AMOEBA. The non-iterative part of this optimization procedure is in the module SIMPLX, whereas the iterative part is in the module AMOEBA. Within the non-iterative part the FSEOPT function FUNC is called with ITASK = 2 (Table 3), to generate INFND+1 parametersets, each with INFND parameters, execute the FSE-submodel with these sets and calculate QT-values by comparing the model output with the experimental data. The INFND+1 parametersets and corresponding QT-values are assigned to the variable names used locally in the subroutine AMOEBA. Then a slightly modified version of the subroutine AMOEBA is called with ITASK=3 to perform the optimization. As in PRICE, AMOEBA finishes the optimization process, if the number of iterations reaches the value of INT, or if the tolerance, RTOL, of the QT-values reaches the predefined value of the variable FTOL. The real function FUNC is called from the subroutine SIMPLX with ITASK-values of 3 and 5 after the initial section and with ITASK-values of 4 and 6 just after the iterative part of the subroutine to obtain requested output.

3.5 Subroutine AMOEBA

The subroutine AMOEBA is taken entirely from Numerical Recipes (Press et al., 1986), where the principles and functional description of this routine are given on pages 289-293. It is recommended to read the book and note the restrictions on publication of Numerical Recipes programs when using the Downhill Simplex method for your calibration experiment.

Within AMOEBA the real function FUNC is instructed to generate a new parameterset with the parametervalues suggested by AMOEBA, to perform a run with these parameters, calculate a QT-value and assign these values to their local synonyms YPR and YPRR. The subroutine AMOEBA stops the optimization process and returns to the subroutine SIMPLX when the range in QT values reaches the predefined value FTOL. If the latter is the case, all resulting parametersets have the same performance with respect to the experimental data, and the same parameter values, if the value of FTOL is low enough, for instance 10^{-8}.

3.6 Real function FUNC

The real function FUNC is the main module in FSEOPT, it communicates with the other subroutines WRRRUN, COMP and OUTPUT. In this subroutine five parameters are included from the include block 'DIMENS.INC' that define the lengths of the arrays actually in use. These parameters, their names, description and actual length are listed in Table 2.

Table 2. Parameters for pass-length declaration in real function FUNC

Parameter name	Description	Default length
IMXNDS	Max. number of datasets	5
IMXHVS	Max. number of samplingtimes	13
IMXREP	Max. number of replicates	5
IMXNDP	Max. number of datapoints (model statevariables)	3
IMXNRR	Max. number of reruns	500
IMXDAT	Max. number of experimental observations	4000

The real function FUNC acts as a driver and communication module for WRRRUN, FSE, COMP and OUTPUT. With task-specific calls to these routines this subroutine instructs the other routines to carry out a sequence of tasks as given in table 3.

Table 3. Task-specific calls to and from the real function FUNC

ITASK-value to FUNC	Module called from FUNC	Description of task(s)
1	WRRRUN	Write initial parametersets
	FSE	Execute simulation runs, store data
	COMP	Compare model output with experimental data
2	WRRRUN	Write initial parametersets
	FSE	Execute simulation runs, store data
	COMP	Compare model output with experimental data
3	WRRRUN	Write newly generated parameterset
	FSE	Execute simulation run, store data
	COMP	Compare model output with experimental data
4	OUTPUT	Write (initial) parametersets to file with accompanying QT-values before calibration
5	OUTPUT	Write (initial) 'confidence-intervals' of modelstate variables to file before calibration
6	OUTPUT	Write (final) parametersets to file with accompanying QT-values after calibration
7	OUTPUT	Write (final) 'confidence-intervals' of model-state variables to file after calibration

After performing the sequence of tasks specified in table 3 the function returns to the calling subroutine. The large number of arguments of the real function FUNC is necessary to enable pass-length declaration of arrays. The design of this subroutine is a compromise between a portable model-function and a user-friendly calibration package.

3.7 Subroutine WRRRUN

The task of the subroutine WRRRUN (WRite ReRUN) is to write parameternames and corresponding values to outputfile. This module has an initial section that is evaluated when ITASK equals 1 or 2, when executing the Price or Downhill Simplex method. In this initial

section values for a number of variables are read from the file 'OPTIM.DEF' and supplied to the other modules COMP and OUTPUT. Then the parameternames and corresponding Biologically Plausible Ranges are read from the parameter definition file 'PARAM.DEF'. This is done in a flexible way: when a parameter is added to or deleted from this file, the calibration is executed with the existing parameters in the file. It is checked that each parameter is assigned only two values, and that the lower bound precedes the upper bound.

The number of experimental datasets in the calibration experiment is obtained by counting the number of non-zero values in the array IDSID. Then the observation definition file 'OBSERV.DEF' specifying environmental variables is read. It specifies the number of the weather-stations involved, according to the CABO/TPE Weather System (van Kraalingen, 1991), the various years and the starting dates for all simulation runs. The one-dimensional array HARD stores the harvest dates, in the same units as simulation time (e.g. seconds, minutes, days, years), from the experimental datasets to synchronize model output with experimental data. Harvest data should be entered in ascending order per dataset. The last harvest date of the first experimental set is followed by the first of the second set. Then the number of replicates per dataset is read from file. At last the array DAT stores the experimental data proper in a relational structure. Each observation has an identifier for number of the dataset, number of sampling date, number of replicate and the experimental observation proper.

After the initial section there are three write sections; the first and the second are evaluated when WRRRUN is called with ITASK-values of 1 and 2. These sections write an output file with INPS*NSITES and (INFND+1)*NSITES parametersets for the Price and Downhill-Simplex algorithm, respectively. The third write section generates an output file that exists of the new parameterset derived within the optimization algorithm and environmental variables identifying the selected experimental datasets.

3.8 Subroutine COMP

The main function of subroutine COMP is to make a comparison between model output and experimental data, synchronized on sampling time. In the initial section of the subroutine the experimental data are read from the observation definition file 'OBSERV.DEF'.

The value of the variable IQT, which is read from file 'OPTIM.DEF', determines in which way the differences between model output and experimental results are calculated. Model performance expressed in one single value can be calculated as the sum of the absolute differences (IQT=1) or as the square root of the sum of squares of residuals (IQT=2). The array SUM which stores the sums of the residuals is initialized at 0. The one-dimensional array DAT is transformed into the four-dimensional array OBSERV with the dimension of the datasets: max. number of datasets, max. number of sampling times, max. number of replicates and max. number of statevariables in the calibration experiment. Then the accumulated absolute differences or the square root of the sum of squared differences between model-output and experimental data including replicates are stored in the array SUM(1,M). The 'overall' goodness of fit value for a particular parameterset is the maximum of the goodness of fit values of the individual statevariables. This calculation procedure is repeated for each

parameterset in the initial section of the subroutine COMP. When this subroutine is called with ITASK equals 2, the preceding procedure is repeated, for the model output of each new generated parameterset.

3.9 Subroutine OUTPUT

The function of the subroutine OUTPUT is to write (optional) initial and final results of the calibration experiment to output files. Several output options are available which can be selected in the optimization definition file 'OPTIM.DEF'.

With the outputoptions IOUT(1) and IOUT(2) set to 1, the subroutine OUTPUT is instructed to write two output files, one with parameter values and corresponding QT-values before calibration: the file 'PAR_INIT.DAT', the other, 'PAR_TERM.DAT' with the same variables after calibration.

With output options IOUT(3) and IOUT(4) set to 1, the subroutine OUTPUT is instructed to write for each dataset the initial and the final 'confidence interval' of the model statevariables. That implies that the minimum and maximum values of the relevant statevariables obtained during execution of the runs are written to file each sampling time. With the output options implemented one can observe variance in initial and final input and output in relation to corresponding QT-values. The names of the outputfiles that are created with IOUT(2) and IOUT(4) set to 1 are respectively CF_STIxx.DAT and CF_STTxx.DAT were xx represents the number of the dataset. The names of the state variables involved are represented with the names ST_x_LOW and ST_x_HGH were x represent the number of the variable.

3.10 Interface of FSEOPT with FSE-submodules

To combine an FSE-submodule with FSEOPT some adaptations are necessary. Instead of the FSE main program supplied in van Kraalingen (1991), one should use the modified version supplied with FSEOPT. Each simulation run is terminated at the last sampling date. The logical variable OUTPUT is set to TRUE only when experimental data are available. The data files of the original simulation model implemented within the FSE structure (e.g. 'TIMER.DAT', 'PLANT.DAT' or 'SOIL.DAT') should be combined into one file with the name 'INPUT.DAT', in such a way that it starts with the variables from 'TIMER.DAT'. The file 'INPUT.DAT' is read only once at the beginning of the first simulation run, remains open, and is never opened or closed again to accelerate execution.

The interface (communication) of FSEOPT, the calibration environment with the simulation environment FSE is simple: one common block called /SUB/ which contains two variables, the array SIM and the pointer variable II. After execution of the simulation run(s) the sets of model output are stored in the array SIM. The pointer II is used in the iterative section to start storage of the model output of consecutive runs on a new position in the matrix with model data.

1) In the FSE-submodel: Within the declaration section of the FSE-submodel the declaration of the variables in the common block and the common block itself should be included;

```
*-----Integer declaration for variables in optimization
      INTEGER INHVS

*-----Maximum number of rerunsets in optimization
      INTEGER IMXNRR
      PARAMETER (IMXNRR=500)

*-----Maximum number of observations in time per dataset
      INTEGER IMXHVS
      PARAMETER (IMXHVS=13)

*-----Maximum number of statevariables in the optimization
      INTEGER IMXNDP
      PARAMETER (IMXNDP=3)

*-----Dimension of array in common block with plant-module
      DIMENSION SIM(IMXNRR+IMXNDS,IMXHVS,IMXNDP)

*-----Integer declaration for variables in common block
      INTEGER II
```

```
*-----Common block common with submodule under FSE-driver
*     SIM is a matrix with the simulated data and has the
*     dimension number of datasets, number of harvests,
*     number of replicates, number of datapoints
      COMMON /SUB/ SIM, II
```

2) In the initial section (ITASK=1) of the submodule only one statement should be added:

```
       INHVS = 1
```

this statement will reset the pointer which controls sampling time each run.

3) At the end of the rate calculation section (ITASK=2) the values of the model statevariables are stored within the matrix SIM, if the logical OUTPUT is true. Each time the program enters this section the pointer variable INHVS is assigned to the previous value plus 1. In the FSE-submodule this look likes:

```
*-----fill matrix with simulation output synchronous with
*     harvest dates in experiments
      IF (OUTPUT) THEN
          SIM(II,INHVS,1) = STATE_VARIABLE(1)
          SIM(II,INHVS,2) = STATE_VARIABLE(2)
          SIM(II,INHVS,3) = STATE_VARIABLE(3)
          INHVS = INHVS + 1
      ENDIF
```

4) In the terminal section of the FSE-submodule the pointer II should be assigned to the previous value plus 1, each time the routine enters this section (when terminal conditions occur). Model output of the next run is then started at a new position in the matrix with simulation output.

```
       II = II + 1
```

With these adaptations to the original structure of FSE, the calibration program FSEOPT can be used successfully with this simulation environment.

4 Recommendations for setting up your own calibration experiment

- Identify parameters with uncertainty and determine their range

This is perhaps the most difficult step in calibration. Any reasonably sized simulation model contains a host of parameters. Some are physical constants which are known with a high degree of certainty, while others are obviously only poorly estimated or may not even have known estimates. In between these extremes there will likely be parameters whose selection for inclusion in the calibration of the model is subjective. Initially the safest course is to include all parameters for which there is some doubt with a BPR around their initial value.

Finding the boundaries of the biologically plausible ranges may be difficult as well. In many reports in the literature estimates of parameters are given without any reference to the variability of the estimates. Clearly, it would be helpful if researchers would report the range and variance of their data.

After a first sensitivity analysis on all the parameters, one can redefine the set of parameters to be used in the iterative procedure. To facilitate this, one should use the PRICE algorithm with 100-200 parametersets, execute the initial runs and analyse the output-file 'PAR_INIT.DAT'.

- Implement your simulation model as a FSE-submodule

This step requires to subdivide your simulation model into an initial, a rate calculating, an integrating and a terminal section, compatible with the FORTRAN Simulation Environment FSE. Note that initialization of integrals should be explicit and that the statements within sections should be sorted into an executable algorithm to establish a proper simulation algorithm. Model parameters should be read from data file, at least those that are to be used to use within the calibration procedure. These input-file(s) should be handled with the RD-(read) routines of the TTUTIL library. Extensive information on the use of these routines can be found in Rappoldt and van Kraalingen (1990). More detailed information on the development of FSE-submodules can be found in van Kraalingen (1991).

- Create a parameter definition file

This file should be similar to the one in Appendix C on page C-3. It should contain the name and the lower and upper bounds of the BPR of each modelparameter selected for use within the calibration procedure. As argued before, it will be best to include all parameters with their corresponding biologically plausible range in this file.

- Create an observation definition file

This file should be similar to the one in Appendix C on page C-4. It should contain the specifications of the experiments to be used to evaluate model performance. The file contains for each experiment(-al site): number of weather station, year, starting day of the simulation, sampling date(s) and number of replicates. At last, the observation definition file will contain the experimental data proper. They are encoded in a relational structure with identifiers for experiment, sampling date, replicate and observed characteristic. Missing values in this file are recognized by the value -99.

- Create (optional) files with dataset-specific constants, initial values of statevariables, parameter values or functions

When using FSEOPT it may be necessary to supply dataset-specific constants, initial values of statevariables, parameter values or functions. This can be the case for instance when the aim is to calibrate biological parameters while the modelinputs pertaining to the experimental datasets vary, for example plantdensity or initial biomass in the case of the wheat crop growth model.

In a step-wise calibration procedure where more or less independent biological processes are involved, one can introduce parametervalues resulting of the first calibration as constants into subsequent calibrations, instead of calibrating the various processes simultaneously.

These dataset-specific model variables should be (optionally) written to file(s) called 'SETxx_CN.DEF' where xx identifies the number of the experimental dataset. The specific write statements are ignored, if the file SET01_CN.DEF does not exist on the current directory.

- Determine the goodness of fit function to be used for evaluating the correspondence of model results and experimental data

This is also a subjective step that should be selected on the basis of the objectives of the model. When other criteria than accumulated absolute or square root of the sum of squared differences between model results and experimental state variables are selected, adaptation of subroutine COMP is necessary.

- Redefinition of the capacity of the FSEOPT program

The standard version of FSEOPT can hold observations of 3 state variables from 5 different experiments with 13 intermediate harvests and 5 replicates. It can optimize 20 parameters simultaneously. In the initial part of the PRICE algorithm it can store model output of 500 reruns at maximum; 100 parametersets tested at 5 experimental datasets. When these default values not suit your purposes you should change the parameters, compile both FSEOPT and the FSE-submodule and link the application. The maximum number of parametersets can be

defined with the variable IMXNPS in the program FSEOPT. The number of experimental datasets can be changed with the variable IMXNDS, the number of intermediate harvests with IMXHVS, the number of replicates with IMXREP, the number of statevariables with IMXNDP and the maximum number of reruns with IMXNRR in the include block 'DIMENS.INC'. Changes to the variables IMXNDS, IMXHVS and IMXNDP should be duplicated in the include block 'MINMAX.INC'.

- Adaptations to the TTUTIL library

With the program FSEOPT an adapted version of the subroutine RDDATA is supplied with a different parameter for the variable ILPREP. The current value of this variable is 4000. If the program stops with the message 'In rerun file reruns.dat too many sets occur. Increase the value of ILPREP in subroutine RDDATA. Error in RDINDX: too many sets Press <RETURN>' one should increase the value of ILPREP in RDDATA. Changes to the variable ILNDAT should be applied in RDDATA and in the subroutine WRRRUN. After any change to FSEOPT or TTUTIL-module one should compile the changed module and link the application again.

5 References

Beck, M., 1982
Identifying model of environmental systems behaviour, Mathemathical Modelling 3:467-480

Jansen, M.J.W., 1991
Confidence domains from the Price algorithm ?, GLW-note MJA-91-1, GLW-DLO, Wageningen

Klepper, O., 1989
A model of carbon flows in relation to macrobenthic food supply in the Oosterschelde estuary (S. W. Netherlands), Ph. D. Thesis, Wageningen Agricultural University.

Klepper, O. and D.I. Rouse, 1991
A procedure to reduce parameter uncertainty for complex models by comparison with real system output illustrated on a potato growth model, Agricultural Systems 36:375-395

Kraalingen, D.W.G. van, 1991
The FSE system for crop simulation, Simulation Reports CABO-TT nr. 23, CABO-DLO, Wageningen Agricultural University, 77 pp.

Kraalingen, D.W.G. van, W. Stol, P.W.J. Uithol and M.G.M. Verbeek, 1991
User manual of CABO/TPE Weather System, CABO-DLO-Wageningen Agricultural University, Internal communication, 28 pp.

Nelder, J.A. and R. Mead, 1965
A simple method for function minimization, The Computer Journal 7:308-313

Payne, R.W, P.W. Lane, A.E. Ainsley, K.E. Bicknell, P.G.N. Digby, S.A. Harding, P.K. Leech, H.R. Simpson, A.D. Todd, P.J. Verrier and R.P. White, 1987
Genstat 5: Reference Manual, Clarendon Press, Oxford, 749 pp.

Press, W.H., B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, 1986
Numerical Recipes, the art of scientific computing, Cambridge University Press, 818 pp.

Price, W. L., 1979
A controlled random search procedure for global optimization, The Computer Journal 20:367-370

Rappoldt, C. and D.W.G. van Kraalingen, 1990
Reference manual of the FORTRAN utility library TTUTIL with applications, Simulation Reports CABO-TT nr. 20, CABO-DLO, Wageningen Agricultural University, 122 pp.

Rouse, D.I., O. Klepper and C.J.T. Spitters, 1991
Analysis of a potato crop growth model to determine transportability using an objective calibration procedure, Crop Science, submitted.

Scholten, H., B.J. de Hoop and P.M.J. Herman, 1990
Seneca 1.2: Manual. A Simulation ENvironment for ECological Application, Delta Institute for Hydrobiological Research, Yerseke, The Netherlands

Spitters, C.J.T., H. van Keulen and D. W. G. van Kraalingen, 1989
A simple and universal crop growth simulator: SUCROS87. In Rabbinge, R., S.A. Ward, H.H. van Laar (eds). Simulation and systems management in crop protection, Simulation Monograph 34, Pudoc, Wageningen

Tarantola, A., 1987
Inverse problem theory. Methods for data fitting and model parameter estimation. Elsevier, Amsterdam.

Van Straten, G., 1985
Analytical methods for parameter-space delimitation and application to shallow-lake phytoplankton-dynamics modeling. Appl. Math. and Computation 17:459-482

APPENDIX A - Listing of main program and modules

In this appendix the program FSEOPT and the associated subprograms are listed. These subroutines are PRICE, SIMPLX, AMOEBA, WRRRUN, COMP, OUTPUT, WRSTAT, HIGHLO, INDEXX, the real function FUNC and the include blocks DIMENS.INC and MINMAX.INC.

Contents

```
*-----          Program FSEOPT - february 1992
*              - Willem Stol and Doug Rouse -
*
*                  FORTRAN - Utility library:
*         - D.W.G. van Kraalingen and C. Rappoldt -

      PROGRAM FSEOPT
      IMPLICIT REAL (A-Z)

*-----Maximum number of parameter-sets in calibration procedure
      INTEGER IMXNPS
      PARAMETER (IMXNPS=100)

*-----Maximum number of parameters in calibration procedure
      INTEGER IMXPAR
      PARAMETER (IMXPAR=20)

*-----Maximum number of output options in calibration procedure
      INTEGER IMXOUT
      PARAMETER (IMXOUT=4)

*-----Dimension of array with QT-values for each parameter-set
      DIMENSION QT(IMXNPS+1)

*-----Dimension of array with optional output switches
      INTEGER IOUT
      DIMENSION IOUT(IMXOUT)

*-----Declaration of array initialized in subroutine SIMPLEX and
*     locally used in subroutine AMOEBA, containing IMXPAR+1
*     parameter-sets with IMXPAR parameter values
      DIMENSION P(IMXPAR+1,IMXPAR)

*-----Character string PARNAM contains the parameter names, array
*     PARVAL the lower and upper bound of each parameter
      CHARACTER*6 PARNAM
      DIMENSION PARNAM(IMXPAR)
      DIMENSION PARVAL(2,IMXPAR)

*-----Dimension of array with pointers to the IMXPAR+1 (at maximum)
*     randomly choosen parameter-sets in the subset
      INTEGER PS(IMXPAR+1)

*-----Array RANGE contains the biological plausible range (upper
*     bound - lower bound) of each parameter
      DIMENSION RANGE(IMXPAR)

*-----Matrix with parameter space, IMXNPS+2 parameter-sets with
*     IMXPAR parameters
      DIMENSION V(IMXNPS+2,IMXPAR)

*-----Declaration of array with QT-values in subroutine AMOEBA
      DIMENSION Y(IMXPAR+1)

*-----Declaration of array with indexes of qt-values
      INTEGER INDX(IMXNPS)

*-----Character string with name of input datafile
      CHARACTER*12 OPTFIL

*-----Variables use as output of routine RDSINT
      INTEGER IMETHD

*-----Variables used as unit number for file I/O or counter in do-loops
      INTEGER IUNIT,IUL,I

*-----End of declarations

*-----Initial section

*-----Unit numbers for file I/O
      IUNIT = 40
      IUL = 20

*-----Filename from file with definitions of optimization
*     algorithm
      OPTFIL = 'OPTIM.DEF'

*-----Read variables which defines the optimization method,
*     tolerance of QT-values and output options
      CALL RDINIT (IUNIT, IUL, OPTFIL)
      CALL RDSINT ('IMETHD',IMETHD)
      CALL RDSREA ('FTOL'  ,FTOL)
      CALL RDAINT ('IOUT'  ,IOUT ,IMXOUT,I)
      CLOSE (IUNIT, STATUS='DELETE')

*-----Optimization methods:
*     1:  Global random search algorithm according to Price
*     2:  Local optimization algorithm downhill-simplex
*         according to Nelder & Mead

      IF (IMETHD.EQ.1) THEN

         CALL PRICE (IMXNPS,IMXOUT,IMXPAR,QT,IOUT,
     $               PARNAM,PARVAL,PS,RANGE,V,FTOL,INDX)

      ELSE IF (IMETHD.EQ.2) THEN

         CALL SIMPLX (IMXNPS,IMXOUT,IMXPAR,QT,IOUT,
     $                P,PARNAM,PARVAL,RANGE,V,Y,FTOL,INDX)

      END IF
      END

      SUBROUTINE PRICE (IMXNPS,IMXOUT,IMXPAR,QT,IOUT,
     $                  PARNAM,PARVAL,PS,RANGE,V,FTOL,INDX)
      IMPLICIT REAL (A-Z)

*-----Declaration of variables in parameter list
      INTEGER   IMXNPS,IMXOUT,IMXPAR
      DIMENSION QT(IMXNPS+1)
      INTEGER   IOUT
      DIMENSION IOUT(IMXOUT)
      CHARACTER*6 PARNAM
      DIMENSION PARNAM(IMXPAR)
      DIMENSION PARVAL(2,IMXPAR)
      INTEGER   PS
      DIMENSION PS(IMXPAR+1)
      DIMENSION RANGE(IMXPAR)
      DIMENSION V(IMXNPS+2,IMXPAR)

*-----Declaration of array with indexes of qt-values
      INTEGER INDX(IMXNPS)

*-----Integer variables output of FUNC
      INTEGER INFND,INPS,INT,IBOUND,INRR

*-----Variables used as pointer in array with QT-values
      INTEGER IPNT1,IPNT2,PST

*-----Variable ITASK specifies tasks to modules
      INTEGER ITASK

*-----Variables used as counter in do-loops
      INTEGER I,ITER,J,K

*-----Logical variables which check whether new parameter-sets
*     are within or outside BPR's (Biological Plausible Range)
      LOGICAL MORE,LESS

*-----End of declarations

      SAVE

*-----Section in which:
*
*     * parameter-sets are generated
*
*     * FSE is instructed to perform model calculations
*       with these parameter-sets
*
*     * Model-output is compared to experimental data

*-----Execute initial runs and compare with experimental data
      ITASK = 1
      DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $              INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)

*-----Write initial parameter-sets and corresponding QT-values to
*     output-file
      IF (IOUT(1).EQ.1) THEN
         ITASK = 4
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

*-----Write initial confidence intervals from state-variables to
*     output-file
      IF (IOUT(3).EQ.1) THEN
         ITASK = 6
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

*-----Section in which iterative new parameter-sets are generated and
*     FSE is instructed to perform model calculations with the new
*     parameter-set.

*-----Write message to screen to monitor program execution
      WRITE (*,'(/,A,I5,A)') ' Price algorithm, execution of ',
     $       INT*(INRR/INPS),' (iterative) runs'

*-----number of iterations of optimization algorithm
      DO 100 ITER = 1,INT

10    CONTINUE
      PS(1) = NINT(UNIFL()*INPS+0.5)
      I = 2
15    IF (I.LE.(INFND+1)) THEN
20       PST = NINT(UNIFL()*INPS+0.5)
         DO 30 J = 1,I-1
            IF (PST.EQ.PS(J)) GO TO 20
30       CONTINUE
         PS(I) = PST
         I = I + 1
      GO TO 15
      END IF

*-----Calculate new parameter set
      MORE = .FALSE.
      LESS = .FALSE.

*-----Calculate average values of each parameter for the INFND
*     randomly choosen parameter sets
      DO 50 I = 1,INFND
         PSUM = 0.
         DO 40 K = 1,INFND
            J = PS(K)
            PSUM = PSUM + V(J,I)
40       CONTINUE
         V(IMXNPS+2,I) = PSUM/INFND
50    CONTINUE

*-----Use formula: V(J,I) = 2.0 * V(IMXNPS+2,I) - V(PS(INFND+1),I),
*     where V(PS(INFND+1),I) is a parameter set choosen from the
*     INPS-INFND remaining unchoosen sets. PS(INFND+1) is the
*     INFND+1 parameter set selected at random in array PS

      DO 60 I = 1,INFND
*-----    value is lower bound value plus range*unifl()
         V(IMXNPS+1,I) = 2.0 * V(IMXNPS+2,I) - V(PS(INFND+1),I)

*-----Set logical names to false if the calculated parameter
*     value falls outside parameter bounds
         LESS = V(IMXNPS+1,I).LT.PARVAL(1,I)
         MORE = V(IMXNPS+1,I).GT.PARVAL(2,I)

*-----Check to see if parameter sets with parameter values
*     out of bounds are going to be discarded (IBOUND=0) or
*     set to the bound (IBOUND<>0)
         IF (IBOUND.EQ.0) THEN
            IF (MORE.OR.LESS) GO TO 10
         ELSE IF (LESS) THEN
            V(IMXNPS+1,I) = PARVAL(1,I)
         ELSE IF (MORE) THEN
            V(IMXNPS+1,I) = PARVAL(2,I)
         END IF
60    CONTINUE

      ITASK = 3
      DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $              INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
```

```
*-----Search all the parameter sets for the set with the highest QT-value
      CALL HIGHLO (IMXNPS,INPS,IPNT1,IPNT2,QT,QTLOW,QTHIGH)

*-----Discard the parameter set with the highest QT-value with the new
*     QT-value, if better, the old parameter-values with the new ones
*     and the accompanying simulation-output in matrix SIM
      IF (QT(IMXNPS+1).LE.QT(IPNT1)) THEN
         QT(IPNT1) = QT(IMXNPS+1)
         DO 70 I = 1,INFND
            V(IPNT1,I) = V(IMXNPS+1,I)
70       CONTINUE
         ITASK = 8
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

      RTOL = 2. * ABS(QTHIGH-QTLOW)/(ABS(QTHIGH)+ABS(QTLOW))

      IF (MOD(ITER,25).EQ.0) THEN
          CALL WRSTAT (ITER,RTOL,QTLOW,QTHIGH)
      ELSE IF (RTOL.LE.FTOL) THEN
          WRITE (*,'(/,1X,A)')
     $       ' Tolerance of QT-values met criterion: Optimization stops'
          CALL WRSTAT (ITER,RTOL,QTLOW,QTHIGH)
          GO TO 110
      END IF

100   CONTINUE

      CALL WRSTAT (ITER,RTOL,QTLOW,QTHIGH)

110   CONTINUE

*-----Write final parameter-sets and corresponding qt-values to
*     output-file
      IF (IOUT(2).EQ.1) THEN
         ITASK = 5
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

*-----Search for new confidence intervals of state-variables
*     within data and write final confidence intervals from
*     state-variables to file
      IF (IOUT(4).EQ.1) THEN
         ITASK = 7
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

      RETURN
      END

      SUBROUTINE SIMPLX (IMXNPS,IMXOUT,IMXPAR,QT,IOUT,
     $                   P,PARNAM,PARVAL,RANGE,V,Y,FTOL,INDX)
      IMPLICIT REAL (A-Z)

*-----Declaration of variables in parameter list
      INTEGER   IMXNPS,IMXOUT,IMXPAR
      DIMENSION QT(IMXNPS+1)
      INTEGER   IOUT
      DIMENSION IOUT(IMXOUT)
      DIMENSION P(IMXPAR+1,IMXPAR)
      CHARACTER*6 PARNAM
      DIMENSION PARNAM(IMXPAR)
      DIMENSION PARVAL(2,IMXPAR)
      DIMENSION RANGE(IMXPAR)
      DIMENSION V(IMXNPS+2,IMXPAR)
      DIMENSION Y(IMXPAR+1)

*-----Declaration of array with indexes of qt-values
      INTEGER INDX(IMXNPS)

*-----Integer variables output of FUNC
      INTEGER INFND,INPS,INT,IBOUND,INRR

*-----Variable ITASK specifies tasks to modules
      INTEGER ITASK

*-----Variables used as counter in do-loops
      INTEGER I,J

*-----Pointer variable in array with parameter-sets
      INTEGER IPNT1,IPNT2

      INTEGER REINIT

*-----End of declarations

      SAVE

*-----Instruct FUNC to do initial runs with simplex parameter-sets
*     and compare model-output against experimental data
      ITASK = 2
      DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $              INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      INPS = INFND+1

*-----Search all the parameter sets for the set with the highest
*     QT-value
      CALL HIGHLO (IMXNPS,INPS,IPNT1,IPNT2,QT,QTLOW,QTHIGH)

*-----Assign lowest qt-value to the variable YOLD, to test succes of
*     iterative part of the optimization procedure
      YOLD = QTLOW

*-----Write initial parameter-sets and corresponding qt-values to
*     output file
      IF (IOUT(1).EQ.1) THEN
         ITASK = 4
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

*-----Write initial confidence intervals from state-variables to
*     output file
      IF (IOUT(3).EQ.1) THEN
         ITASK = 6
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

*-----Assign values of qt and v to the local simplex equivalents
*     y and p
      DO 20 I = 1,INPS
         Y(I) = QT(I)
         DO 10 J = 1,INFND
            P(I,J) = V(I,J)
10       CONTINUE
20    CONTINUE

      WRITE (*,'(/,A)')
     $     ' Simplex algorithm, execution of iterative runs'

*-----Search for minimum of problem within INFND+1th space
      ITASK = 3
      CALL AMOEBA (ITASK,IMXNPS,IMXPAR+1,IMXPAR,IMXPAR,
     $             INFND,INT,QT,FTOL,P,PARNAM,PARVAL,RANGE,
     $             V,Y,IPNT1,IPNT2,INDX)

      DO 40 I = 1,INPS
         QT(I) = Y(I)
         DO 30 J = 1,INFND
            V(I,J) = P(I,J)
30       CONTINUE
40    CONTINUE

      YNEW = QT(IPNT2)
      REINIT = 0

*-----re-initialize simplex (again) if the new iterative
*     procedure was succesfull

45    CONTINUE

      IF (YNEW.LE.YOLD.AND.REINIT.LT.3) THEN

         YOLD = QT(IPNT2)

*-----   Instruct FUNC to re-initialize simplex again, run with the
*        new choosen simplex parameter-sets and compare model-output
*        against experimental data

         WRITE (*,'(/,A)')
     $       ' Simplex algorithm, re-initialization of simplex'

         ITASK = 2
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
         INPS = INFND+1

*-----   Assign values of qt and v to the local simplex equivalents y
*        and p
         DO 60 I = 1,INPS
            Y(I) = QT(I)
            DO 50 J = 1,INFND
               P(I,J) = V(I,J)
50          CONTINUE
60       CONTINUE

         WRITE (*,'(2(/,A))')
     $       ' Simplex algorithm, execution of iterative runs',
     $       ' with re-initialized simplex'

*-----   Search for minimum of problem within INFND+1th space
         ITASK = 3
         CALL AMOEBA (ITASK,IMXNPS,IMXPAR+1,IMXPAR,IMXPAR,
     $                INFND,INT,QT,FTOL,P,PARNAM,PARVAL,RANGE,
     $                V,Y,IPNT1,IPNT2,INDX)

         DO 70 I = 1,INPS
            QT(I) = Y(I)
            DO 80 J = 1,INFND
               V(I,J) = P(I,J)
80          CONTINUE
70       CONTINUE

         YNEW = QT(IPNT2)

         IF (YNEW.LT.YOLD) THEN
            REINIT = 0
         ELSE
            REINIT = REINIT + 1
         END IF

         GO TO 45
      END IF

*-----Write final parameter-sets and corresponding qt-values
*     to output-file
      IF (IOUT(2).EQ.1) THEN
         ITASK = 5
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

*-----Search for new confidence intervals of state-variables
*     within data and write final confidence intervals from
*     state-variables to file
      IF (IOUT(4).EQ.1) THEN
         ITASK = 7
         DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,IBOUND,INT,INPS,INFND,
     $                 INRR,QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      END IF

      RETURN
      END

      SUBROUTINE AMOEBA (ITASK,IMXNPS,MP,NP,IMXPAR,
     $                   NDIM,INT,QT,FTOL,P,PARNAM,PARVAL,RANGE,
     $                   V,Y,IPNT1,IPNT2,INDX)
      IMPLICIT REAL (A-Z)

*-----Declaration of variables in parameter list
      INTEGER IMXNPS,IMXPAR

*-----Amoeba declarations
      INTEGER MP,NP,NMAX

*-----Func declarations
      INTEGER ITASK,INPS,INFND,INT,IBOUND,INRR
```

```
*-----Dimension of array with values of model performance for
*     each rerun
      DIMENSION QT(IMXNPS+1)

*-----Declaration of array with indexes of qt-values
      INTEGER INDX(IMXNPS)

*-----Dimensions of parameter space, IMXNPS+1 parameter-sets
*     with IMXPAR parameters
      DIMENSION V(IMXNPS+2,IMXPAR)
      CHARACTER*6 PARNAM
      DIMENSION PARNAM(IMXPAR)
      DIMENSION PARVAL(2,IMXPAR)
      DIMENSION RANGE(IMXPAR)

*-----Pointer variables in array with parameter-sets
      INTEGER IPNT1,IPNT2

*-----Original amoeba declarations
      PARAMETER (NMAX=20,ALPHA=1.0,BETA=0.5,GAMMA=2.0)
      INTEGER I,J,IHI,ILO,INHI,MPTS,NDIM,ITER
      DIMENSION P(MP,NP),Y(MP),PR(NMAX),PRR(NMAX),PBAR(NMAX)

*-----End of declarations

      SAVE

      INFND = NDIM

      MPTS = NDIM+1
      ITER = 0
1     ILO = 1
      IF (Y(1).GT.Y(2)) THEN
        IHI = 1
        INHI = 2
      ELSE
        IHI = 2
        INHI = 1
      END IF
      DO 11 I = 1,MPTS
        IF( Y(I).LT.Y(ILO)) ILO = I
        IF (Y(I).GT.Y(IHI)) THEN
          INHI = IHI
          IHI = I
        ELSE IF (Y(I).GT.Y(INHI)) THEN
          IF (I.NE.IHI) INHI = I
        END IF
11    CONTINUE

      RTOL = 2.*ABS(Y(IHI)-Y(ILO))/(ABS(Y(IHI))+ABS(Y(ILO)))
      CALL WRSTAT (ITER+1,RTOL,Y(ILO),Y(IHI))

      IF (RTOL.LT.FTOL.OR.ITER.EQ.INT) THEN
         IPNT1 = IHI
         IPNT2 = ILO
         RETURN
      END IF

      ITER = ITER+1
      DO 12 J = 1,NDIM
        PBAR(J) = 0.
12    CONTINUE
      DO 14 I = 1,MPTS
        IF (I.NE.IHI) THEN
          DO 13 J = 1,NDIM
            PBAR(J) = PBAR(J)+P(I,J)
13        CONTINUE
        END IF
14    CONTINUE
      DO 15 J = 1,NDIM
        PBAR(J) = PBAR(J)/NDIM
        PR(J) = (1.+ALPHA)*PBAR(J)-ALPHA*P(IHI,J)
        V(IMXNPS+1,J) = PR(J)
15    CONTINUE
      YPR = FUNC (ITASK,IMXNPS,IMXPAR,
     $            IBOUND,INT,INPS,INFND,INRR,
     $            QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      IF (YPR.LE.Y(ILO)) THEN
        DO 16 J = 1,NDIM
          PRR(J) = GAMMA*PR(J)+(1.-GAMMA)*PBAR(J)
          V(IMXNPS+1,J) = PRR(J)
16      CONTINUE
        YPRR = FUNC (ITASK,IMXNPS,IMXPAR,
     $               IBOUND,INT,INPS,INFND,INRR,
     $               QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
        IF (YPRR.LT.Y(ILO)) THEN
          DO 17 J = 1,NDIM
            P(IHI,J) = PRR(J)
17        CONTINUE
          Y(IHI) = YPRR
        ELSE
          DO 18 J = 1,NDIM
            P(IHI,J) = PR(J)
18        CONTINUE
          Y(IHI) = YPR
        END IF

        IPNT1 = IHI
        ITASK = 8
        DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,
     $                IBOUND,INT,INPS,INFND,INRR,
     $                QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
        ITASK = 3

      ELSE IF (YPR.GE.Y(INHI)) THEN
        IF (YPR.LT.Y(IHI)) THEN
          DO 19 J = 1,NDIM
            P(IHI,J) = PR(J)
19        CONTINUE
          Y(IHI) = YPR
        END IF

        IPNT1 = IHI
        ITASK = 8
        DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,
     $                IBOUND,INT,INPS,INFND,INRR,
     $                QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
        ITASK = 3

        DO 21 J = 1,NDIM
          PRR(J) = BETA*P(IHI,J)+(1.-BETA)*PBAR(J)
          V(IMXNPS+1,J) = PRR(J)
21      CONTINUE
        YPRR = FUNC (ITASK,IMXNPS,IMXPAR,
     $               IBOUND,INT,INPS,INFND,INRR,
     $               QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
        IF (YPRR.LT.Y(IHI)) THEN
          DO 22 J = 1,NDIM
            P(IHI,J) = PRR(J)
22        CONTINUE
          Y(IHI) = YPRR

          IPNT1 = IHI
          ITASK = 8
          DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,
     $                  IBOUND,INT,INPS,INFND,INRR,
     $                  QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
          ITASK = 3

        ELSE
          DO 24 I = 1,MPTS
            IF (I.NE.ILO) THEN
              DO 23 J = 1,NDIM
                PR(J) = 0.5*(P(I,J)+P(ILO,J))
                P(I,J) = PR(J)
                V(IMXNPS+1,J) = PR(J)
23            CONTINUE
              Y(I) = FUNC (ITASK,IMXNPS,IMXPAR,
     $                     IBOUND,INT,INPS,INFND,INRR,
     $                     QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
            END IF
24        CONTINUE
          END IF
        ELSE
          DO 25 J = 1,NDIM
            P(IHI,J) = PR(J)
25        CONTINUE
          Y(IHI) = YPR

          IPNT1 = IHI
          ITASK = 8
          DUMMY = FUNC (ITASK,IMXNPS,IMXPAR,
     $                  IBOUND,INT,INPS,INFND,INRR,
     $                  QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
          ITASK = 3

        END IF
        GO TO 1
      END

      REAL FUNCTION FUNC (ITASK,IMXNPS,IMXPAR,
     $                    IBOUND,INT,INPS,INFND,INRR,
     $                    QT,PARNAM,PARVAL,RANGE,V,IPNT1,IPNT2,INDX)
      IMPLICIT REAL (A-Z)

*-----Declaration of variables in parameter list
      INTEGER ITASK,IMXNPS,IMXPAR,IBOUND
      INTEGER INT,INPS,INFND,IPNT1,IPNT2

*-----MICROSOFT FORTRAN V 5.1 / VAX FORTRAN V 5.6-199
      INCLUDE 'DIMENS.INC'

*-----MAC FORTRAN/020 V 2.3
*      INCLUDE DIMENS.INC

*-----Array with daynumbers of the observations for each dataset
      DIMENSION HARD((IMXHVS*IMXNDS)+1)

*-----Dimension of array with values of model performance for each
*     rerun
      DIMENSION QT(IMXNPS+1)

*-----Array with number of replicates per measurements
      INTEGER INREP
      DIMENSION INREP(IMXNDS)

*-----Dimensions of parameter space, IMXNPS+1 parameter-sets
*     with IMXPAR parameters
      DIMENSION V(IMXNPS+2,IMXPAR)

*-----Array with names of parameters are stored in the variable
*     PARNAM the corresponding BPR's (Biological Plausible Ranges)
*     in array PARVAL
      CHARACTER*6 PARNAM
      DIMENSION PARNAM(IMXPAR)
      DIMENSION PARVAL(2,IMXPAR)
      DIMENSION RANGE(IMXPAR)

*-----Array with names of files that store dataset-specific
*     constants, parameters, functions or initial values
      CHARACTER*12 CNFIL
      DIMENSION CNFIL(IMXNDS)

*-----Array with data-set identifier
      INTEGER IDSID
      DIMENSION IDSID(IMXNDS)

*-----Arrays with number of weather stations, number of year,
*     start and harvest data and number of harvest per dataset
      INTEGER ISTN,IYEAR,INHVS,INDS,INRR
      DIMENSION ISTN(IMXNDS),IYEAR(IMXNDS)
      DIMENSION DAYB(IMXNDS),INHVS(IMXNDS)
      INTEGER IIHRD
      DIMENSION IIHRD(IMXNDS+1)

*-----Declaration of array with indexes of qt-values
      INTEGER INDX(IMXNPS)

*-----Array with observed data
      DIMENSION OBSERV(IMXNDS,IMXHVS,IMXREP,IMXNDP)

*-----Dimension of array with aggregated values of model
*     performance for each measured variable
      DIMENSION MPF(IMXNDP)

*-----Dimension of array with sums of observed data and
*     residuals between observed and measured data
      DIMENSION SUM(2,IMXNDP)

*-----Variable used as counter in do-loops
      INTEGER I
```

```
*-----Logical variables which check whether new parameter-sets
*     are within or outside BPR's (Biological Plausible Range)
      LOGICAL MORE,LESS

*-----Dimension of array in common block with submodule
      DIMENSION SIM(IMXNRR+IMXNDS,IMXHVS,IMXNDP)

*-----Variable II is pointer in array with simulations
      INTEGER II

*-----Common block common with submodule under FSE-driver SIM is
*     the matrix with the simulated data. It has the dimension
*     number of datasets, number of harvests, number of replicates,
*     number of datapoints
      COMMON /SUB/ SIM, II

*-----End of declarations

      SAVE

      IF (ITASK.EQ.1) THEN

*-----    Write rerun-file with the appropriate parameter-values
         CALL WRRRUN (ITASK,IMXHVS,IMXNDS,IMXNPS,IMXPAR,
     $                INDS,INFND,INHVS,INPS,INRR,
     $                HARD,IBOUND,DAYB,IDSID,IIHRD,INT,
     $                ISTN,IYEAR,INREP,PARNAM,PARVAL,RANGE,
     $                CNFIL,V,IPNT2)

*-----    Write message to screen to monitor program execution
         WRITE (*,'(/,A,I4,A)') ' Price algorithm, execution of ',
     $                          INRR, ' (initial) runs'

*-----    Calculate QT for each parameter set
         II = 1
         CALL FSE

*-----    Instruct COMP to read experimental data from file, to ini-
*        tialize a matrix with them and this calculate the performance
*        of the initial model-runs against the experimental data
         CALL COMP (ITASK,IMXHVS,IMXNDP,IMXNDS,IMXNPS,IMXNRR,IMXREP,
     $              QT,IDSID,II,INDS,INHVS,INPS,MPF,INREP,OBSERV,SIM,
     $              SUM,IPNT1)

      ELSE IF (ITASK.EQ.2) THEN

*-----    Write rerun-file with the appropriate parameter-values
         CALL WRRRUN (ITASK,IMXHVS,IMXNDS,IMXNPS,IMXPAR,
     $                INDS,INFND,INHVS,INPS,INRR,
     $                HARD,IBOUND,DAYB,IDSID,IIHRD,INT,
     $                ISTN,IYEAR,INREP,PARNAM,PARVAL,RANGE,
     $                CNFIL,V,IPNT2)

*-----    Write message to screen to monitor program execution
         WRITE (*,'(/,A,I4,A)') ' Simplex algorithm, execution of ',
     $                          INRR, ' (initial) runs'

*-----    Execute a simulation run with the calculated parameter-sets
*        simulated data are filled on pointers 1:INFND+1
         II = 1
         CALL FSE

*-----    Calculate the model performance against the experimental data
         INPS = INFND+1
         ITASK = 1
         CALL COMP (ITASK,IMXHVS,IMXNDP,IMXNDS,IMXNPS,IMXNRR,IMXREP,
     $              QT,IDSID,II,INDS,INHVS,INPS,MPF,INREP,OBSERV,SIM,
     $              SUM,IPNT1)
         ITASK = 2

      ELSE IF (ITASK.EQ.3) THEN

         DO 10 I = 1,INFND

*-----       Check to see if the calculated parameter value is out of
*           bounds of possible values, if so set logical names to false
            LESS = V(IMXNPS+1,I).LT.PARVAL(1,I)
            MORE = V(IMXNPS+1,I).GT.PARVAL(2,I)

            IF (MORE.OR.LESS) THEN
               FUNC = 1.0E+3
               RETURN
            END IF

10       CONTINUE

*-----    Write rerun-file with the appropriate parameter-values
         II = IMXNPS+1
         CALL WRRRUN (ITASK,IMXHVS,IMXNDS,IMXNPS,IMXPAR,
     $                INDS,INFND,INHVS,INPS,INRR,
     $                HARD,IBOUND,DAYB,IDSID,IIHRD,INT,
     $                ISTN,IYEAR,INREP,PARNAM,PARVAL,RANGE,
     $                CNFIL,V,IPNT2)

*-----    Execute a simulation run with the calculated parameter-set
*        simulated data are filled on pointer IMXNPS+1
         CALL FSE

*-----    Calculate the model performance against the experimental data
         ITASK = 2

*-----    After the run of the model the pointer II is reset to IMXNPS+1
         II = IMXNPS+1
         CALL COMP (ITASK,IMXHVS,IMXNDP,IMXNDS,IMXNPS,IMXNRR,IMXREP,
     $              QT,IDSID,II,INDS,INHVS,INPS,MPF,INREP,OBSERV,SIM,
     $              SUM,IPNT1)
         ITASK = 3
         FUNC = QT(IMXNPS+1)

      ELSE IF (ITASK.EQ.4) THEN

         ITASK = 1
         CALL OUTPUT (ITASK,INFND,INPS,QT,PARNAM,V,IMXNPS,IMXPAR,HARD,
     $                IMXHVS,IMXNDS,IDSID,IIHRD,IMXNDP,INDX)

      ELSE IF (ITASK.EQ.5) THEN

         ITASK = 2
         CALL OUTPUT (ITASK,INFND,INPS,QT,PARNAM,V,IMXNPS,IMXPAR,HARD,
     $                IMXHVS,IMXNDS,IDSID,IIHRD,IMXNDP,INDX)

      ELSE IF (ITASK.EQ.6) THEN

         ITASK = 4
         CALL COMP (ITASK,IMXHVS,IMXNDP,IMXNDS,IMXNPS,IMXNRR,IMXREP,
     $              QT,IDSID,II,INDS,INHVS,INPS,MPF,INREP,OBSERV,SIM,
     $              SUM,IPNT1)
         ITASK = 3
         CALL OUTPUT (ITASK,INFND,INPS,QT,PARNAM,V,IMXNPS,IMXPAR,HARD,
     $                IMXHVS,IMXNDS,IDSID,IIHRD,IMXNDP,INDX)

      ELSE IF (ITASK.EQ.7) THEN

         ITASK = 4
         CALL COMP (ITASK,IMXHVS,IMXNDP,IMXNDS,IMXNPS,IMXNRR,IMXREP,
     $              QT,IDSID,II,INDS,INHVS,INPS,MPF,INREP,OBSERV,SIM,
     $              SUM,IPNT1)
         CALL OUTPUT (ITASK,INFND,INPS,QT,PARNAM,V,IMXNPS,IMXPAR,HARD,
     $                IMXHVS,IMXNDS,IDSID,IIHRD,IMXNDP,INDX)

      ELSE IF (ITASK.EQ.8) THEN

         II = IMXNPS+1
         ITASK = 3
         CALL COMP (ITASK,IMXHVS,IMXNDP,IMXNDS,IMXNPS,IMXNRR,IMXREP,
     $              QT,IDSID,II,INDS,INHVS,INPS,MPF,INREP,OBSERV,SIM,
     $              SUM,IPNT1)

      END IF

      RETURN
      END

      SUBROUTINE WRRRUN (ITASK,IMXHVS,IMXNDS,IMXNPS,IMXPAR,
     $                   INDS,INFND,INHVS,INPS,INRR,
     $                   HARD,IBOUND,DAYB,IDSID,IIHRD,INT,
     $                   ISTN,IYEAR,INREP,PARNAM,PARVAL,RANGE,
     $                   CNFIL,V,IPNT2)
      IMPLICIT REAL (A-Z)

*-----Declaration of variables in parameter list
      INTEGER ITASK,IMXHVS,IMXNDS,IMXNPS,IMXPAR,INDS,INFND
      INTEGER INHVS,INPS,INRR,IBOUND,INT,IPNT2

*-----Variables with names of datafiles
      CHARACTER*12 OPTFIL, PARFIL, OBSFIL, CNFIL(IMXNDS)

*-----Variable used as parameter in RDSINT
      INTEGER IMETHD

*-----Variable used as parameter in RDAREA
      INTEGER IIDSID

*-----Variable used as counter in do-loops
      INTEGER NSITES

*-----Dimensions of parameter space, IMXNPS+1 parameter-sets
*     with IMXPAR parameters
      DIMENSION V(IMXNPS+2,IMXPAR)

*-----Character string PARNAM contains the parameter names, array
*     PARVAL the lower and upper bound of each parameter
      CHARACTER*6 PARNAM
      DIMENSION PARNAM(IMXPAR)
      DIMENSION PARVAL(2,IMXPAR)
      DIMENSION RANGE(IMXPAR)

*-----Array with data-set identifier
      INTEGER IDSID
      DIMENSION IDSID(IMXNDS)

*-----Arrays with the number of samples in time for each dataset
      INTEGER IHRD,IIHRD

*-----Arrays with number of weather stations, number of year,
*     start and harvest data and number of harvest per dataset
      INTEGER ISTN,IYEAR,INREP
      DIMENSION ISTN(IMXNDS),IYEAR(IMXNDS)
      DIMENSION DAYB(IMXNDS),INHVS(IMXNDS)
      DIMENSION IIHRD(IMXNDS+1)
      DIMENSION INREP(IMXNDS)

*-----Array with daynumbers of the observations for each dataset
      DIMENSION HARD((IMXHVS*IMXNDS)+1)

*-----Name of internal file
      CHARACTER*2 STRING

*-----Variables used as unit number
      INTEGER IUNIT,IUL,IUNITR

*-----Variables used as counter in do-loops
      INTEGER I,J,K,L

*-----Variables used as argument to subroutine RDFREA
      INTEGER IREQ

*-----Logical variable to check existence of file
      LOGICAL THERE

*-----Logical variable to check status of subroutine
      LOGICAL INIT, FIRST, SECOND

*-----Common variables with RD-routines
*     index data file ; index availabale after CALL RDINIT
      INTEGER ILNDAT,IARDAT,IPTDAT,INFDAT
      CHARACTER DATLIS*6,DATTYP*1
      PARAMETER (ILNDAT=400)
      DIMENSION DATLIS(ILNDAT),DATTYP(ILNDAT)
      DIMENSION IARDAT(ILNDAT),IPTDAT(ILNDAT)
      COMMON /TTUDD1/ DATLIS,DATTYP
      COMMON /TTUDD2/ IARDAT,IPTDAT,INFDAT

*-----End of declarations

      SAVE

      DATA INIT /.FALSE./


      IF (.NOT.INIT) THEN
```

```
      FIRST = .TRUE.
      SECOND = .FALSE.

*-----Unit numbers for file I/O
      IUNIT = 40
      IUL = 20

*-----Filename from file which defines the optimization
*     characteristics
      OPTFIL = 'OPTIM.DEF'

*-----Read variables with optimization characteristics
      CALL RDINIT (IUNIT, IUL, OPTFIL)
      CALL RDSINT ('IMETHD' ,IMETHD)
      CALL RDSINT ('IBOUND' ,IBOUND)
      CALL RDSINT ('INPS'   ,INPS)
      CALL RDSINT ('INT'    ,INT)
      CALL RDSINT ('INDS'   ,INDS)
      CALL RDAINT ('IDSID'  ,IDSID ,IMXNDS ,IIDSID)
      CLOSE (IUNIT, STATUS='DELETE')

*-----Filename from parameter definition file
      PARFIL = 'PARAM.DEF'

*-----Analyse file with parameter-names and BPR's, get
*     names and number of parameters in the file
      CALL RDINIT (IUNIT, IUL, PARFIL)

      IREQ = 2
      INFND = INFDAT
      DO 10 I = 1,INFND
         CALL RDFREA (DATLIS(I),PARVAL(1,I),IREQ,IREQ)
         IF (PARVAL(1,I).GT.PARVAL(2,I)) CALL ERROR ('WRRRUN',
     $        'Lower bound exceeds upper bound in PARAM.DEF.')
         PARNAM(I) = DATLIS(I)
10    CONTINUE
      CLOSE (IUNIT, STATUS='DELETE')

*-----Create array containing the multiplication factor used to change
*     the uniform random deviate on the interval 0-1 into a value on
*     the range of the parameter
      DO 20 I = 1,INFND
         RANGE(I) = PARVAL(2,I)-PARVAL(1,I)
20    CONTINUE

      NSITES = 0
      DO 25 K = 1,INDS
         IF (IDSID(K).EQ.1) NSITES = NSITES + 1
25    CONTINUE

*-----Filename from observation definition file
      OBSFIL = 'OBSERV.DEF'

*-----Input section ; analyse input file
      CALL RDINIT (IUNIT, IUL, OBSFIL)

*-----Get values from file
      CALL RDAINT ('ISTN'  , ISTN  , IMXNDS, I)
      CALL RDAINT ('IYEAR' , IYEAR , IMXNDS, I)
      CALL RDAINT ('INREP' , INREP , IMXNDS, I)
      CALL RDAREA ('DAYB'  , DAYB  , IMXNDS, I)
      CALL RDAREA ('HARD'  , HARD  , (IMXHVS*IMXNDS)+1, IHRD)
      CLOSE (IUNIT,STATUS='DELETE')

      J = 1
      IIHRD(1) = J
      DO 30 I = 1,IHRD
         IF (HARD(I+1).LE.HARD(I)) THEN
            J = J + 1
            IIHRD(J) = I + 1
            INHVS(J-1) = IIHRD(J) - IIHRD(J-1)
         END IF
30    CONTINUE

*-----check if file with data-set specific constants
*     of first data-set does exist
      CNFIL(1) = 'SET01_CN.DEF'
      THERE = .FALSE.
      INQUIRE (FILE=CNFIL(1),EXIST=THERE)

      IF (THERE) THEN
         DO 35 I = 1,IMXNDS
            WRITE (STRING(1:2),'(I2.2)') I
            CNFIL(I) = 'SET'//STRING(1:2)//'_CN.DEF'
35       CONTINUE
      END IF

*-----Set logical variable init to true
      INIT = .TRUE.

   END IF

*-----Write reruns file to enable multiple runs with FSE
   IUNITR = 21
   CALL FOPEN (IUNITR,'RERUNS.DAT','NEW','DEL')

*-----Price initial runs
   IF (ITASK.EQ.1) THEN

*-----Number of rerun-sets is n experimental data-sets times
*     n parameter-sets
      INRR = NSITES * INPS

*-----Write headerfile in reruns file
      WRITE (21,'(A,I4,A)') '*-----Reruns-file: ',INRR,
     $     ' parameter-sets generated by FSEOPT'

*-----Create INRR parameter-sets with INFND parameters
      DO 50 J = 1,INPS

         DO 40 I = 1,INFND
*-----       value is lower bound value plus range*unifl()
            V(J,I) = PARVAL(1,I)+RANGE(I)*UNIFL()
40       CONTINUE

         DO 45 K = 1,INDS
            IF (IDSID(K).EQ.1) THEN
*-----          write section to identify location and year
               WRITE (21,
     $        '(/,1X,A,I2,'' ; '',A,I4,'' ; '',A,F4.0)')
     $        'ISTN  =',ISTN(K),'IYEAR = ',IYEAR(K),'DAYB =',DAYB(K)

*-----       copy dataset specific constants, initial values or
*           functions from file with constants to reruns file
            IF (THERE) CALL COPFIL (90,CNFIL(K),21)

*-----       write section with parameter names and values
            WRITE (21,'(6(1X,4(A6,''='',F9.5,'' ; ''),/))')
     $      (PARNAM(I),V(J,I),I=1,INFND)

*-----       write section to synchronize model- and experimental data
            WRITE (21,'(1X,A)') 'HARDAY ='
            WRITE (21,'(10(F4.0,A))') (HARD(L),',',
     $           L=IIHRD(K),IIHRD(K+1)-2), HARD(IIHRD(K+1)-1)
            END IF
45       CONTINUE

50    CONTINUE

      CLOSE (21)

*-----Simplex initial runs
      ELSE IF (ITASK.EQ.2) THEN

*-----Number of rerun-sets is n experimental data-sets
*     times n parameters + 1
      INRR = NSITES * (INFND+1)

*-----Write headerfile in reruns file
      WRITE (21,'(A,I3,A)') '*-----Reruns-file: ',INRR,
     $     ' parameter-sets generated by FSEOPT'

*-----Simplex first optimization

      IF (FIRST) THEN

*-----The parameter-values of the first parameter-set are random
*     choosen, by adding range*unifl() to the lower bound, this
*     this is the initial simplex side, these parameter-values
*     are applied to the other sets
         DO 60 I = 1,INFND

* start somewhere in the biological plausible range
*           V(1,I) = PARVAL(1,I)+RANGE(I)*UNIFL()

* start at initial value of parameter
            V(1,I) = PARVAL(1,I)+RANGE(I)*0.5

            DO 55 J = 2,INFND+1
               V(J,I) = V(1,I)
55          CONTINUE
60       CONTINUE

*-----   The other vertices of the simplex are set-up by adding
*        or subtracting 0.1 of the (biological plausible) range
*        to the value of the first parameterset. The choice between
*        adding or subtraction will depend on the placement of the
*        parameter-set with respect to the parameter-bounds
         DO 70 I = 2,INFND+1
            DO 65 J = 1,INFND
               IF (J.EQ.I-1) THEN
                  HALF = PARVAL(1,J)+0.5*RANGE(J)
                  IF (V(I,J).LT.HALF) THEN
                     V(I,J) = V(I,J)*1.2
                  ELSE
                     V(I,J) = V(I,J)*0.8
                  END IF
               END IF
65          CONTINUE
70       CONTINUE

         FIRST = .FALSE.
         SECOND = .TRUE.

*-----Simplex next iterative loops
      ELSE IF (SECOND) THEN

         DO 75 I = 1,INFND
            V(1,I) = V(IPNT2,I)
75       CONTINUE

         DO 85 I = 1,INFND
            DO 80 J = 2,INFND+1
               V(J,I) = V(1,I)
80          CONTINUE
85       CONTINUE

*-----   The other vertices of the simplex are set-up with the
*        same method as the original simplex, taking the best set
*        of the first iterative section as initial simplex side
         TMP = UNIFL()*0.2
         DO 95 I = 2,INFND+1
            DO 90 J = 1,INFND
               IF (J.EQ.I-1) THEN
                  HALF = PARVAL(1,J)+0.5*RANGE(J)
                  IF (V(I,J).LT.HALF) THEN
                     V(I,J) = V(I,J)*(1.+TMP)
                  ELSE
                     V(I,J) = V(I,J)*(1.-TMP)
                  END IF
               END IF
90          CONTINUE
95       CONTINUE

      END IF

*-----Create INFND+1 parameter-sets with INFND parameters
      DO 105 J = 1,INFND+1

         DO 100 K = 1,INDS
            IF (IDSID(K).EQ.1) THEN
*-----          write section to identify location and year
               WRITE (21,
     $        '(/,1X,A,I2,'' ; '',A,I4,'' ; '',A,F4.0)')
     $        'ISTN  =',ISTN(K),'IYEAR = ',IYEAR(K),'DAYB =',DAYB(K)

*-----       copy dataset specific constants, initial values or
*           functions from file with constants to reruns file
            IF (THERE) CALL COPFIL (90,CNFIL(K),21)

*-----       write section with parameter names and values
```

```
            WRITE (21,'(6(1X,4(A6,''='',F9.5,'' ; ''),/))')
     $             (PARNAM(I),V(J,I),I=1,INFND)

*-----      write section to synchronize model- and experimental data
            WRITE (21,'(1X,A)') 'HARDAY ='
            WRITE (21,'(10(F4.0,A))') (HARD(L),',',
     $           L=IIHRD(K),IIHRD(K+1)-2), HARD(IIHRD(K+1)-1)
         END IF
100      CONTINUE

105   CONTINUE

*-----Rerun with model within iterative optimization loop
      ELSE IF (ITASK.EQ.3) THEN

*-----Write headerfile in reruns file
      WRITE (21,'(2A)')  '*-----Reruns-file: 1',
     $       ' parameter-set generated by FSEOPT'

*-----Create for each experimental dataset in the optimization
*     1 parameterset with INFND parameters, get parameter-values
*     at pointer IMXNPS+1 from matrix with parameter-values
      J = IMXNPS+1

      DO 110 K = 1,INDS
         IF (IDSID(K).EQ.1) THEN
*-----      write section to identify location and year
           WRITE (21,
     $     '(/,1X,A,I2,'' ; '',A,I4,'' ; '',A,F4.0)')
     $     'ISTN  =',ISTN(K),'IYEAR = ',IYEAR(K),'DAYB =',DAYB(K)

*-----      copy dataset specific constants, initial values or
*          functions from file with constants to reruns file
           IF (THERE) CALL COPFIL (90,CNFIL(K),21)

*-----      write section with parameter names and values
           WRITE (21,'(6(1X,4(A6,''='',F9.5,'' ; ''),/))')
     $     (PARNAM(I),V(J,I),I=1,INFND)

*-----      write section to synchronize model- and experimental data
           WRITE (21,'(1X,A)') 'HARDAY ='
           WRITE (21,'(10(F4.0,A))') (HARD(L),',',
     $          L=IIHRD(K),IIHRD(K+1)-2), HARD(IIHRD(K+1)-1)
        END IF
110   CONTINUE

      END IF
      CLOSE (21)

      RETURN
      END

      SUBROUTINE COMP (ITASK,IMXHVS,IMXNDP,IMXNDS,IMXNPS,IMXNRR,IMXREP,
     $                 QT,IDSID,II,INDS,INHVS,INPS,MPF,INREP,OBSERV,SIM,
     $                 SUM,IPNT1)
      IMPLICIT REAL (A-Z)

*-----Declaration of variables in parameter list
      INTEGER ITASK,IMXHVS,IMXNDP,IMXNDS,IMXNPS,IMXNRR,IMXREP,IPNT1
      INTEGER II,INDS,INHVS,INREP
      DIMENSION INREP(IMXNDS)
      DIMENSION INHVS(IMXNDS)
      DIMENSION QT(IMXNPS+1)

*-----Dimension of array with sums of observed data and
*     residuals between observed and measured data
      DIMENSION SUM(2,IMXNDP)

*-----Dimension of array with aggregated values of model
*     performance for each measured variable
      DIMENSION MPF(IMXNDP)

*-----Character string with name of input datafile
      CHARACTER*12 OPTFIL

*-----Maximum number of data-points per observation in matrix
*     DAT (dummy matrix which is read in OBSERV)
      INTEGER IIDAT
      INTEGER IMXDAT
      PARAMETER (IMXDAT=4000)
      DIMENSION DAT(IMXDAT)

*-----Array with observed data
      DIMENSION OBSERV(IMXNDS,IMXHVS,IMXREP,IMXNDP)

*-----Variable INPS contains number of parameter runs
      INTEGER INPS

*-----Variable ANDS contains actual number of datasets
      INTEGER ANDS

*-----Variable ITMP scratch integer
      INTEGER ITMP

*-----Parameter in calculation method model performance
      INTEGER IQT

*-----Array with data-set identifier
      INTEGER IDSID
      DIMENSION IDSID(IMXNDS)

*-----Logical name used in initial section
      LOGICAL INIT

*-----Dimension of array in common block with submodule
      DIMENSION SIM(IMXNRR+IMXNDS,IMXHVS,IMXNDP)

*-----Variables used as unit number for file I/O or counter in do-loops
      INTEGER IUNIT,IUL,I,J,K,L,M,N

*-----MICROSOFT FORTRAN V 5.1 / VAX FORTRAN V 5.6-199
      INCLUDE 'MINMAX.INC'

*-----MAC FORTRAN/020 V 2.3
*      INCLUDE MINMAX.INC

*-----End of declarations

      SAVE

      DATA INIT /.FALSE./

      IF (.NOT.INIT) THEN

*-----   Unit numbers for file I/O
        IUNIT = 70
        IUL = 50

*-----   Filename from optimization definition file (OPTIM.DEF)
        OPTFIL = 'OPTIM.DEF'
        CALL RDINIT (IUNIT, IUL, OPTFIL)

*-----   Get value from file
        CALL RDSINT ('IQT', IQT)
        CLOSE (IUNIT, STATUS='DELETE')

*-----   Filename from file which defines the experimental datasets
        OPTFIL = 'OBSERV.DEF'
        CALL RDINIT (IUNIT, IUL, OPTFIL)

*-----   Get values from file
        CALL RDAREA ('DAT'   , DAT   , IMXDAT, IIDAT)
        CLOSE (IUNIT, STATUS='DELETE')

*-----   End of input section

*-----   Initialize array SUM
        DO 10 I = 1,IMXNDP
           SUM(1,I) = 0.
           SUM(2,I) = 0.
10      CONTINUE

*-----   Change one-dimensional array in a four-dimensional array
        M = 0
15      IF (M.LT.IIDAT/5) THEN
           N = M * 5
           I = NINT(DAT(N+1))
           J = NINT(DAT(N+2))
           K = NINT(DAT(N+3))
           L = NINT(DAT(N+4))
           OBSERV (I,J,K,L) = DAT(N+5)
           M = M + 1
        GOTO 15
        END IF

        ANDS = 0
        DO 45 M = 1,IMXNDS
           ANDS = ANDS + IDSID(M)
45      CONTINUE

        INIT = .TRUE.
      END IF

      IF (ITASK.EQ.1) THEN

        DO 100 I = 1,INPS

        DO 50 M = 1,IMXNDP
           SUM(1,M) = 0.
50      CONTINUE

        ITMP = 0
        DO 55 J = 1,INDS
           IF (IDSID(J).EQ.1) THEN
              ITMP = ITMP+1
              DO 80 K = 1,INHVS(J)
                 DO 70 L = 1,INREP(J)
                    DO 60 M = 1,IMXNDP
*-----               calculate sum of differences between experimental
*                   and simulated data according to the method choosen
*                   in optimization definition file.
                    DK = OBSERV(J,K,L,M)
                    MK = SIM((I-1)*ANDS+ITMP,K,M)
                    SUM(1,M) = SUM(1,M)+ABS((DK-MK)/(DK+1E-8))**IQT
60                  CONTINUE
70                CONTINUE
80             CONTINUE
           END IF
55      CONTINUE

        QT(I) = 0.
        DO 90 M = 1,IMXNDP

*-----      apply the appropriate factor to normalize residuals
           IF (IQT.EQ.1) THEN
              MPF(M) = SUM(1,M)
           ELSE IF (IQT.EQ.2) THEN
              MPF(M) = SQRT(SUM(1,M))
           END IF

*-----      find the maximum of residuals
           QT(I) = AMAX1 (MPF(M),QT(I))

90      CONTINUE

100     CONTINUE

      ELSE IF (ITASK.EQ.2) THEN

        DO 110 M = 1,IMXNDP
           SUM(1,M) = 0.
110     CONTINUE

        ITMP = 0
        DO 115 J = 1,INDS
           IF (IDSID(J).EQ.1) THEN
              ITMP = ITMP + 1
              DO 140 K = 1,INHVS(J)
                 DO 130 L = 1,INREP(J)
                    DO 120 M = 1,IMXNDP
*-----               calculate sum of differences between experimen-
*                   tal and simulated data according to the method
*                   choosen in optimization definition file.
                    DK = OBSERV(J,K,L,M)
                    MK = SIM(II-1+ITMP,K,M)
                    SUM(1,M) = SUM(1,M)+ABS((DK-MK)/(DK+1E-8))**IQT
120                 CONTINUE
130              CONTINUE
140           CONTINUE
           END IF
115     CONTINUE
```

```
         QT(IMXNPS+1) = 0.
         DO 150 M = 1,IMXNDP

*-----      apply the appropriate factor to normalize residuals
            IF (IQT.EQ.1) THEN
               MPF(M) = SUM(1,M)
            ELSE IF (IQT.EQ.2) THEN
               MPF(M) = SQRT(SUM(1,M))
            END IF

*-----      find the maximum of residuals
            QT(IMXNPS+1) = AMAX1 (MPF(M),QT(IMXNPS+1))

150      CONTINUE

      ELSE IF (ITASK.EQ.3) THEN

         ITMP = 0
         DO 160 J = 1,INDS
            IF (IDSID(J).EQ.1) THEN
               ITMP = ITMP + 1
               DO 190 K = 1,INHVS(J)
                  DO 180 L = 1,INREP(J)
                     DO 170 M = 1,IMXNDP
*-----                   replace the existing values of state variables
*                       with the new values with closer correspondence
                        SIM((IPNT1-1)*ANDS+ITMP,K,M) =
     $                  SIM(II-1+ITMP,K,M)
170                  CONTINUE
180               CONTINUE
190            CONTINUE
            END IF
160      CONTINUE

      ELSE IF (ITASK.EQ.4) THEN

         DO 220 J = 1,IMXNDS
            DO 210 K = 1,IMXHVS
               DO 200 M = 1,IMXNDP
                  LOW (J,K,M) = 1.0E+8
                  HIGH(J,K,M) = -1.0E+8
200            CONTINUE
210         CONTINUE
220      CONTINUE

         DO 260 I = 1,INPS
            ITMP = 0
            DO 230 J = 1,INDS
               IF (IDSID(J).EQ.1) THEN
                  ITMP = ITMP+1
                  DO 250 K = 1,INHVS(J)
                     DO 240 M = 1,IMXNDP
*-----                   calculate maxima and minima of state variables
*                       within model-simulations
                        LOW(J,K,M) = AMIN1(SIM((I-1)*ANDS+ITMP,K,M),
     $                                     LOW(J,K,M))
                        HIGH(J,K,M) = AMAX1(SIM((I-1)*ANDS+ITMP,K,M)
     $                                     ,HIGH(J,K,M))
240                  CONTINUE
250               CONTINUE
               END IF
230         CONTINUE
260      CONTINUE

      END IF

      RETURN
      END

      SUBROUTINE OUTPUT (ITASK,INFND,INPS,QT,PARNAM,V,IMXNPS,IMXPAR,
     $                   HARD,IMXHVS,IMXNDS,IDSID,IIHRD,IMXNDP,INDX)
      IMPLICIT REAL (A-Z)

      INTEGER ITASK,INFND,INPS,IMXNPS,IMXPAR,IMXHVS,IMXNDS,IIHRD
      INTEGER IMXNDP,IDSID
      DIMENSION QT(IMXNPS+1)
      CHARACTER*6 PARNAM
      DIMENSION PARNAM(IMXPAR)
      DIMENSION V(IMXNPS+2,IMXPAR)
      DIMENSION HARD(IMXHVS*IMXNDS)
      DIMENSION IDSID(IMXNDS)
      DIMENSION IIHRD(IMXNDS+1)

*-----Character string with name of input datafile
      CHARACTER*14 OPTFIL
      CHARACTER*6 STRING

*-----Variables used as unit number for file I/O or counter
*     in do-loops
      INTEGER IUNIT,I,J,K,L,M

*-----Name of internal file
      CHARACTER*14 INTERN

*-----Name of character string
      CHARACTER*80 TEXT

*-----Declaration of array with indexes of qt-values
      INTEGER INDX(IMXNPS)

*-----MICROSOFT FORTRAN V 5.1 / VAX FORTRAN V 5.6-199
      INCLUDE 'MINMAX.INC'

*-----MAC FORTRAN/020 V 2.3
*      INCLUDE MINMAX.INC

*-----End of declarations

      SAVE

*-----Unit number for file I/O
      IUNIT = 70

      IF (ITASK.EQ.1) THEN

*-----   Write output-file with initial parameter sets and QT-values
         OPTFIL = 'PAR_INIT.DAT'
         CALL FOPEN (IUNIT,OPTFIL,'NEW','DEL')
         CALL OUTDAT (1,IUNIT,'SET-i',0.)

         DO 20 I = 1, INPS
            CALL OUTDAT (2,IUNIT,'SET-i',FLOAT(I))
            DO 10 J = 1, INFND
               CALL OUTDAT (2,IUNIT,PARNAM(J),V(I,J))
10          CONTINUE
            CALL OUTDAT (2,IUNIT,'QT',QT(I))
20       CONTINUE

         TEXT =
     $   'FSEOPT - calibration: (unsorted) initial sets and QT-values'
         CALL OUTDAT (4,IUNIT,TEXT,0.)

         CALL INDEXX (INPS,QT,INDX)
         CALL OUTDAT (1,IUNIT,'SET-i',0.)

         DO 22 I = 1, INPS
            CALL OUTDAT (2,IUNIT,'SET-i',FLOAT(INDX(I)))
            DO 21 J = 1, INFND
               CALL OUTDAT (2,IUNIT,PARNAM(J),V(INDX(I),J))
21          CONTINUE
            CALL OUTDAT (2,IUNIT,'QT',QT(INDX(I)))
22       CONTINUE

         TEXT =
     $   'FSEOPT - calibration: (sorted) initial sets and QT-values'
         CALL OUTDAT (4,IUNIT,TEXT,0.)
         CALL OUTDAT (99, 0, ' ', 0.)
         CLOSE (IUNIT)

      ELSE IF (ITASK.EQ.2) THEN

*-----   Write final parameter-sets and QT-values to output file
         OPTFIL = 'PAR_TERM.DAT'
         CALL FOPEN (IUNIT,OPTFIL,'NEW','DEL')
         CALL OUTDAT (1,IUNIT,'SET-i',0.)

         DO 40 I = 1, INPS
            CALL OUTDAT (2,IUNIT,'SET-i',FLOAT(I))
            DO 30 J = 1, INFND
               CALL OUTDAT (2,IUNIT,PARNAM(J),V(I,J))
30          CONTINUE
            CALL OUTDAT (2,IUNIT,'QT',QT(I))
40       CONTINUE

         TEXT =
     $  'FSEOPT - calibration: (unsorted) final sets and QT-values'
         CALL OUTDAT (4,IUNIT,TEXT,0.)

         CALL INDEXX (INPS,QT,INDX)
         CALL OUTDAT (1,IUNIT,'SET-i',0.)

         DO 42 I = 1, INPS
            CALL OUTDAT (2,IUNIT,'SET-i',FLOAT(INDX(I)))
            DO 41 J = 1, INFND
               CALL OUTDAT (2,IUNIT,PARNAM(J),V(INDX(I),J))
41          CONTINUE
            CALL OUTDAT (2,IUNIT,'QT',QT(INDX(I)))
42       CONTINUE

         TEXT =
     $   'FSEOPT - calibration: (sorted) final sets and QT-values'
         CALL OUTDAT (4,IUNIT,TEXT,0.)
         CALL OUTDAT (99, 0, ' ', 0.)
         CLOSE (IUNIT)

      ELSE IF (ITASK.EQ.3) THEN

*-----   Write initial confidence interval of state-var's to output-file
         STRING = 'ST_CFI'
         DO 50 K = 1,IMXNDS
            IF (IDSID(K).EQ.1) THEN
               WRITE (INTERN(1:8),'(A,I2.2)') STRING,K
               OPTFIL = INTERN(1:8)//'.DAT'
               CALL FOPEN (IUNIT,OPTFIL,'NEW','DEL')
               CALL OUTDAT (1,IUNIT,'HARDAY',0.)
               DO 70 I = IIHRD(K), IIHRD(K+1)-1
                  CALL OUTDAT (2,IUNIT,'HARDAY',HARD(I))
                  DO 60 L = 1, IMXNDP
                     WRITE (INTERN(1:8),'(A,I1,A)') 'ST_',L,'_LOW'
                     M = 1 + I - IIHRD(K)
                     CALL OUTDAT (2,IUNIT,INTERN(1:8),LOW(K,M,L))
                     WRITE (INTERN(1:8),'(A,I1,A)') 'ST_',L,'_HGH'
                     CALL OUTDAT (2,IUNIT,INTERN(1:8),HIGH(K,M,L))
60                CONTINUE
70             CONTINUE
               CALL OUTDAT (4,IUNIT,
     $         'FSEOPT - confidence intervals before calibration',0.)
               CALL OUTDAT (99, 0, ' ', 0.)
               CLOSE (IUNIT)
            END IF
50       CONTINUE

      ELSE IF (ITASK.EQ.4) THEN

*-----   Write final confidence interval of state-var's to output-file
         STRING = 'ST_CFT'
         DO 80 K = 1,IMXNDS
             IF (IDSID(K).EQ.1) THEN
                WRITE (INTERN(1:8),'(A,I2.2)') STRING,K
                OPTFIL = INTERN(1:8)//'.DAT'
                CALL FOPEN (IUNIT,OPTFIL,'NEW','DEL')
                CALL OUTDAT (1,IUNIT,'HARDAY',0.)
                DO 100 I = IIHRD(K), IIHRD(K+1)-1
                   CALL OUTDAT (2,IUNIT,'HARDAY',HARD(I))
                   DO 90 L = 1, IMXNDP
                      WRITE (INTERN(1:8),'(A,I1,A)') 'ST_',L,'_LOW'
                      M = 1 + I - IIHRD(K)
                      CALL OUTDAT (2,IUNIT,INTERN(1:8),LOW(K,M,L))
                      WRITE (INTERN(1:8),'(A,I1,A)') 'ST_',L,'_HGH'
                      CALL OUTDAT (2,IUNIT,INTERN(1:8),HIGH(K,M,L))
90                 CONTINUE
100             CONTINUE
                CALL OUTDAT (4,IUNIT,
     $          'FSEOPT - confidence intervals after calibration',0.)
                CALL OUTDAT (99, 0, ' ', 0.)
                CLOSE (IUNIT)
             END IF
80       CONTINUE
      END IF

      RETURN
      END
```

```
      SUBROUTINE HIGHLO (IMXNPS,INPS,IPNT1,IPNT2,QT,QTLOW,QTHIGH)
      IMPLICIT REAL (A-Z)

*-----Variables in parameter-list
      INTEGER IMXNPS,INPS,IPNT1,IPNT2
      DIMENSION QT(IMXNPS+1)

*-----Integer variable used in do-loops
      INTEGER I

*-----End of declarations

      SAVE

      QTLOW  = 1.0E+8
      QTHIGH = -1.0E+8

      DO 10 I = 1, INPS
         IF (QT(I).LT.QTLOW) THEN
            IPNT2 = I
            QTLOW = QT(I)
         END IF
         IF (QT(I).GT.QTHIGH) THEN
            IPNT1 = I
            QTHIGH = QT(I)
         END IF
10    CONTINUE

      RETURN
      END

      SUBROUTINE WRSTAT (ITER,RTOL,QTLOW,QTHIGH)
      IMPLICIT REAL (A-Z)

      INTEGER ITER

      SAVE

      WRITE (*,'(/,1X,A,I10,3(/,1X,A,G10.5))')
     $  '        Iteration-n: ',ITER,
     $  '          Tolerance: ',RTOL,
     $  '  Qt-value best-set: ',QTLOW,
     $  ' Qt-value worst-set: ',QTHIGH

      RETURN
      END
```

```
      SUBROUTINE INDEXX (N,ARRIN,INDX)
      IMPLICIT REAL(A-Z)

      INTEGER I,J,N,IR,L
      DIMENSION ARRIN(N)
      INTEGER INDXT,INDX(N)

      DO 11 J=1,N
        INDX(J)=J
11    CONTINUE
      L=N/2+1
      IR=N
10    CONTINUE
        IF (L.GT.1) THEN
          L=L-1
          INDXT=INDX(L)
          Q=ARRIN(INDXT)
        ELSE
          INDXT=INDX(IR)
          Q=ARRIN(INDXT)
          INDX(IR)=INDX(1)
          IR=IR-1
          IF (IR.EQ.1) THEN
            INDX(1)=INDXT
            RETURN
          ENDIF
        ENDIF
        I=L
        J=L+L
20      IF (J.LE.IR) THEN
          IF (J.LT.IR) THEN
            IF (ARRIN(INDX(J)).LT.ARRIN(INDX(J+1)))J=J+1
          ENDIF
          IF (Q.LT.ARRIN(INDX(J))) THEN
            INDX(I)=INDX(J)
            I=J
            J=J+J
          ELSE
            J=IR+1
          ENDIF
        GO TO 20
        ENDIF
        INDX(I)=INDXT
      GO TO 10
      END

*-----Include block 'DIMENS.INC'

*-----Maximum number of harvests (sampling times)
      INTEGER IMXHVS
      PARAMETER (IMXHVS=13)

*-----Maximum number of state-variables in the optimization
      INTEGER IMXNDP
      PARAMETER (IMXNDP=3)

*-----Maximum number of experimental data-sets
      INTEGER IMXNDS
      PARAMETER (IMXNDS=5)

*-----Maximum number of rerun-sets in optimization
      INTEGER IMXNRR
      PARAMETER (IMXNRR=500)

*-----Maximum number of replicates per observation
      INTEGER IMXREP
      PARAMETER (IMXREP=5)

*-----Include block 'MINMAX.INC'

*-----Dimension of array with minimum and maximum values
*     of simulation-runs, these values are interpreted as
*     confidence intervals. When you will change the size
*     of the FSEOPT program the order is the number of
*     datasets (5), number of harvests (13) and the number
*     of datapoints (3)
      DIMENSION LOW(5,13,3)
      DIMENSION HIGH(5,13,3)

      COMMON /CONF1/ LOW,HIGH
```

APPENDIX B - Interface of FSEOPT and user-defined submodules

In this appendix the interface of FSEOPT and user-defined submodules and the example of the file RERUNS.DAT are listed.

Contents Page

Include these declarations in the declaration section of your FSE-(sub)model:

```
*--------------------------------------
*----- Begin of declarations for FSEOPT

      INCLUDE 'DIMENS.INC'

*-----Integer declaration for variables in optimization
      INTEGER INHVS

*-----Dimension of array in common block with plant-module
      DIMENSION SIM(IMXNRR+IMXNDS,IMXHVS,IMXNDP)

*-----Integer declaration for variables in common block
      INTEGER II

*-----Common block common with submodule under FSE-driver
*     SIM is a matrix with the simulated data and has the
*     dimension number of datasets, number of harvests,
*     number of replicates, number of datapoints
      COMMON /SUB/ SIM, II

*----- End of declarations for FSEOPT
*--------------------------------------
```

Include this assignment in the initial section (ITASK=1) of your FSE-submodel:

```
      INHVS = 1
```

Include this section in the integral section (ITASK=3) of your FSE-submodel:

```
*-----fill matrix with simulation output synchronous with
*     harvest dates in experiments
      IF (OUTPUT) THEN
          SIM(II,INHVS,1) = STATE_VARIABLE(1)
          SIM(II,INHVS,2) = STATE_VARIABLE(2)
          SIM(II,INHVS,3) = STATE_VARIABLE(3)
          INHVS = INHVS + 1
      ENDIF
```

Include this assignment in the terminal section (ITASK=4) of your FSE-submodel:

```
      II = II + 1
```

```
**-----Reruns-file:    4    parameter-sets generated by FSEOPT

ISTN  = 1 ; IYEAR = 1985 ; DAYB = 70.

*-------------------------------------------------------------------------*
* Contents of input file: SET01_CN.DEF                                    *
*-------------------------------------------------------------------------*

NPL = 100.

LA0   =   0.5470 ; TMBJUV=   1.3550 ; RGRL  =   0.0170 ;
HARDAY =
71., 77., 84., 91., 98.,105.,112.,119.,126.,133.,
140.,147.,154.

ISTN  = 1 ; IYEAR = 1985 ; DAYB = 70.

*-------------------------------------------------------------------------*
* Contents of input file: SET01_CN.DEF                                    *
*-------------------------------------------------------------------------*

NPL = 100.

LA0   =   0.5926 ; TMBJUV=   0.4417 ; RGRL  =   0.0190 ;
HARDAY =
71., 77., 84., 91., 98.,105.,112.,119.,126.,133.,
140.,147.,154.

ISTN  = 1 ; IYEAR = 1985 ; DAYB = 70.

*-------------------------------------------------------------------------*
* Contents of input file: SET01_CN.DEF                                    *
*-------------------------------------------------------------------------*

NPL = 100.

LA0   =   0.5934 ; TMBJUV=   2.0391 ; RGRL  =   0.0130 ;
HARDAY =
71., 77., 84., 91., 98.,105.,112.,119.,126.,133.,
140.,147.,154.

ISTN  = 1 ; IYEAR = 1985 ; DAYB = 70.

*-------------------------------------------------------------------------*
* Contents of input file: SET01_CN.DEF                                    *
*-------------------------------------------------------------------------*

NPL = 100.

LA0   =   0.6124 ; TMBJUV=   1.2635 ; RGRL  =   0.0133 ;
HARDAY =
71., 77., 84., 91., 98.,105.,112.,119.,126.,133.,
140.,147.,154.
```

APPENDIX C - Listing of definition files

In this appendix the input datafiles of FSEOPT are listed. There is one file containing the definitions of the optimization procedure, called 'OPTIM.DEF'. A second file defines the parameternames and their Biologically Plausible Ranges, the name of this file is 'PARAM.DEF'. The third file, 'OBSERV.DEF', contains variables which define the experimental environment, such as number of datasets, sampling dates, replicates and the observed data.

Contents Page

```
*-------------------------------------------------------------*
* Input datafile 'OPTIM.DEF' : contains the user-definable
* data for the calibration procedure.
*-------------------------------------------------------------*

*-----preferred optimization algorithm; IMETHD, units: -
*     1: Price-method, controlled random search method
*        for global optimization
*     2: Simplex-method, local optimization method
IMETHD = 2

*-----Preferred method for calculation of Qt value, IQT,
*     1: Sums of absolute differences observed-simulated
*     2: Sums of squares of differences observed-simulated
*     units: -
IQT = 1

*-----option switch which tackles parameter values outside
*     predefined Biological Plausible Range (BPR), IBOUND, units = -
*     0 ; new generated parametervalues outside bounds
*         are discarded
*     1 ; new generated parametervalues outside bounds
*         are set to bound
IBOUND = 1

*-----number of parameter sets; INPS, units: -
*     Rule of thumb: at least 10 times the number of
*     modelparameters within the calibrationexperiment
INPS = 30

*-----number of iterations of algorithm; INT, units: -
*     Rule of thumb: at least 6 times the number of
*     parametersets INPS.
INT = 300

*-----tolerance of CT-values of parametersets, stop
*     criteria for optimization; FTOL, units: -
FTOL = 1.0E-8

*-----the number of datasets (meaning number of year/
*     locations of data are entered); NDS, units: -
INDS = 5

*-----the datasets which will be run; IDSID, units: -
IDSID = 1, 0, 0, 0, 0

*-----output switches are entered below, 0 value skips output
* IOUT(1) : Output of initial parametersets and corresponding
*           Ct-values before calibration.
* IOUT(2) : Output of final parametersets and corresponding
*           Ct-values after calibration.
* IOUT(3) : Output of initial confidence intervals of model-
*           statevariables.
* IOUT(4) : Output of final confidence intervals of model-
*           statevariables.
IOUT = 1, 1, 1, 1
```

```
*-------------------------------------------------------
* File: 'PARAM.DEF', this file contains the parameters
* which are used in the calibration experiment
*-------------------------------------------------------

*-----Extrapolated leaf area at field emergence,
*     units = cm**2/plant
LA0 = 0., 1.0

*-----Base temperature for juvenile growth, units = gr. C
TMBJUV = 0., 3.

*-----Relative growth rate during exponential leaf area growth
*     units = cm**2/cm**2/gr.C/d
RGRL = 0.01, 0.02
```

```
*-----------------------------------------------------
* File: 'OBSERV.DEF', this file contains the experimental
* data which are used in the calibration experiment.
*-----------------------------------------------------

*-----numbers of weatherstations in the subsequent
*     dataset are entered, ISTN, units = -
ISTN = 1, 1, 1, 1, 1

*-----numbers of weatherstations in the subsequent
*     dataset are entered, IYEAR, units = -
IYEAR = 1985, 1986, 1987, 1988, 1989

*-----the starting date (julian daynumber) for each
*     data set, DAYB, units: julian daynumber
DAYB = 70., 70., 70., 70., 70.

*-----the sampling dates of the subsequent datasets
*     are entered, HARD, units: julian daynumber
HARD =  71.,  77.,  84.,  91.,  98., 105., 112., 119., 126., 133.,
       140., 147., 154.,
        71.,  77.,  84.,  91.,  98., 105., 112., 119., 126., 133.,
       140., 147., 154.,
        71.,  77.,  84.,  91.,  98., 105., 112., 119., 126., 133.,
       140., 147., 154.,
        71.,  77.,  84.,  91.,  98., 105., 112., 119., 126., 133.,
       140., 147., 154.,
        71.,  77.,  84.,  91.,  98., 105., 112., 119., 126., 133.,
       140., 147., 154.

*-----number of replicates in the subsequent datasets,
*     NREP, units: -
INREP = 1, 1, 1, 1, 1

*-----the observations are grouped by number of dataset,
*     sample number, number of replicate, number of
*     measurements, DAT, units: variable
DAT =

1.,   1.,   1.,   1.,   0.,
1.,   2.,   1.,   1.,   0.,
1.,   3.,   1.,   1.,   0.,
1.,   4.,   1.,   1.,   0.,
1.,   5.,   1.,   1.,   0.,
1.,   6.,   1.,   1.,   0.01,
1.,   7.,   1.,   1.,   0.04,
1.,   8.,   1.,   1.,   0.69,
1.,   9.,   1.,   1.,   0.13,
1.,  10.,   1.,   1.,   0.40,
1.,  11.,   1.,   1.,   1.00,
1.,  12.,   1.,   1.,   1.75,
1.,  13.,   1.,   1.,   2.54,
```

```
2.,   1.,   1.,   1.,   0.,
2.,   2.,   1.,   1.,   0.,
2.,   3.,   1.,   1.,   0.,
2.,   4.,   1.,   1.,   0.,
2.,   5.,   1.,   1.,   0.,
2.,   6.,   1.,   1.,   0.,
2.,   7.,   1.,   1.,   0.,
2.,   8.,   1.,   1.,   0.02,
2.,   9.,   1.,   1.,   0.07,
2.,  10.,   1.,   1.,   0.26,
2.,  11.,   1.,   1.,   0.74,
2.,  12.,   1.,   1.,   1.77,
2.,  13.,   1.,   1.,   2.83,

3.,   1.,   1.,   1.,   0.,
3.,   2.,   1.,   1.,   0.,
3.,   3.,   1.,   1.,   0.,
3.,   4.,   1.,   1.,   0.,
3.,   5.,   1.,   1.,   0.,
3.,   6.,   1.,   1.,   0.,
3.,   7.,   1.,   1.,   0.02,
3.,   8.,   1.,   1.,   0.08,
3.,   9.,   1.,   1.,   0.25,
3.,  10.,   1.,   1.,   0.58,
3.,  11.,   1.,   1.,   1.21,
3.,  12.,   1.,   1.,   2.57,
3.,  13.,   1.,   1.,   3.50,

4.,   1.,   1.,   1.,   0.,
4.,   2.,   1.,   1.,   0.,
4.,   3.,   1.,   1.,   0.,
4.,   4.,   1.,   1.,   0.,
4.,   5.,   1.,   1.,   0.,
4.,   6.,   1.,   1.,   0.,
4.,   7.,   1.,   1.,   0.02,
4.,   8.,   1.,   1.,   0.05,
4.,   9.,   1.,   1.,   0.19,
4.,  10.,   1.,   1.,   0.58,
4.,  11.,   1.,   1.,   1.45,
4.,  12.,   1.,   1.,   2.44,
4.,  13.,   1.,   1.,   2.87,

5.,   1.,   1.,   1.,   0.,
5.,   2.,   1.,   1.,   0.,
5.,   3.,   1.,   1.,   0.,
5.,   4.,   1.,   1.,   0.,
5.,   5.,   1.,   1.,   0.01,
5.,   6.,   1.,   1.,   0.02,
5.,   7.,   1.,   1.,   0.05,
5.,   8.,   1.,   1.,   0.09,
5.,   9.,   1.,   1.,   0.27,
5.,  10.,   1.,   1.,   0.69,
5.,  11.,   1.,   1.,   1.77,
5.,  12.,   1.,   1.,   2.97,
5.,  13.,   1.,   1.,   3.51
```

APPENDIX D - Listing of statistical programs

In this appendix two statistical programs are listed which can be used to analyse output of the calibration procedure to become familiar with model behaviour.

Contents Page

```
" ************************  HISTOGRAM.GEN  ***************************** "
" * This genstat program produces a correlation and covariance matrix    * "
" * and histograms of all the values of parameters within the parameter- * "
" * sets. The data are read from the outputfiles par_init.dat, which     * "
" * can be requested optionally from the calibration procedure.          * "
" ************************************************************************ "

JOB  'CORRELATION MATRIX AND HISTOGRAMS AFTER CALIBRATION'

UNIT [4]

POIN [VALUES=LA0_I,TMBJUV_I,RGRL_I] PRM1
POIN [VALUES=LA0_T,TMBJUV_T,RGRL_T] PRM2
VARI PRM1[],PRM2[]

OPEN NAME='PAR_TERM.DAT';CHAN=2;FILE=INPUT;WIDTH=132
SKIP [CHAN=2] 7
READ [CHAN=2;END=*] ISET,PRM2[],QT_T
CLOSE CHAN=2;FILE=INPUT

" CORRELATION MATRIX "

MODEL PRM2[]
TERM [PRIN=C] PRM2[]

" COVARIANCE MATRIX "

SSPM [TERMS=PRM2[]] SSP
FSSPM [PRIN=SSPM] SSP

OPEN NAME='PAR_INIT.DAT';CHAN=2;FILE=INPUT;WIDTH=132
SKIP [CHAN=2] 7
READ [CHAN=2;END=*] ISET,PRM1[],QT_I
CLOSE CHAN=2;FILE=INPUT

PEN NUMBER=1;LINESTYLE=1;SYMBOL=0;METHOD=LINE;SIZE=1.5
PEN NUMBER=2;METHOD=POINT

AXES 1...2; PENAXES=1

FRAME WINDOW=1...4;YLOW=0.3,0.3,0.,0.;YUPP=1.,1.,0.3,0.3;\
XLOW=0.,0.48,0.,0.48;XUPP=0.52,1.0,0.52,1.0

OPEN 'HISTO_1.GRD';CHAN=1;FILETYPE=GRAPHICS
VARI [VALUES=0.,0.1...1.] PAR1
DHIS [TITLE='Histogram 1';LIMITS=PAR1;SCREEN=K;WIND=1;KEYW=3] PRM1[1]
DHIS [TITLE='Histogram 2';LIMITS=PAR1;SCREEN=K;WIND=2;KEYW=4] PRM2[1]

OPEN 'HISTO_2.GRD';CHAN=1;FILETYPE=GRAPHICS
VARI [VALUES=0.,0.3...3.] PAR2
DHIS [TITLE='Histogram 3';LIMITS=PAR2;SCREEN=K;WIND=1;KEYW=3] PRM1[2]
DHIS [TITLE='Histogram 4';LIMITS=PAR2;SCREEN=K;WIND=2;KEYW=4] PRM2[2]

OPEN 'HISTO_3.GRD';CHAN=1;FILETYPE=GRAPHICS
VARI [VALUES=0.01,0.011...0.02] PAR3
DHIS [TITLE='Histogram 5';LIMITS=PAR3;SCREEN=K;WIND=1;KEYW=3] PRM1[3]
DHIS [TITLE='Histogram 6';LIMITS=PAR3;SCREEN=K;WIND=2;KEYW=4] PRM2[3]

OPEN 'QT.GRD';CHAN=1;FILETYPE=GRAPHICS
VARI [VALUES=0.,0.1...1.0] QTV
DHIS [TITLE='Histogram 7';LIMITS=QTV;SCREEN=K;WIND=1;KEYW=3] QT_I
DHIS [TITLE='Histogram 8';LIMITS=QTV;SCREEN=K;WIND=2;KEYW=4] QT_T

STOP
```

```
" ************************** CONFID.GEN ************************ "
" * This genstat program produces graphs of the final values of    * "
" * measured field data and worst model outcome before calibration * "
" * The data are read from FSEOPT outputfiles: st_cfixx.dat and    * "
" * st_cftxx.dat.                                                  * "
" ****************************************************************** "

JOB 'CONFIDENCE INTERVALS - BEFORE AND AFTER CALIBRATION'

UNIT [13]
VARI DAY,VAR[1...2],VAR2[1...2]

OPEN NAME='ST_CFI01.DAT';CHAN=2;FILE=INPUT;WIDTH=132
SKIP [CHAN=2] 6
READ [CHAN=2;END=*] DAY,VAR[]
CLOSE CHAN=2;FILE=INPUT

VARI TIMEEXP,LAI
READ TIMEEXP,LAI

71.   0.00000
77.   0.00000
84.   0.00000
91.   0.00000
98.   0.00000
105.  0.18672E-01
112.  0.44154E-01
119.  0.69925E-01
126.  0.13226
133.  0.40995
140.  1.0019
147.  1.7523
154.  2.5483:

" This part of the GENSTAT program produces graphs of the final "
" values of measured field data (line) and worst model outcome  "
" (points) after calibration. The data are read from OPTI       "

OPEN NAME='ST_CFT01.DAT';CHAN=2;FILE=INPUT;WIDTH=132
SKIP [CHAN=2] 6
READ [CHAN=2;END=*] DAY,VAR2[]
CLOSE CHAN=2;FILE=INPUT

PEN NUMBER=1;LINESTYLE=1;SYMBOL=0;METHOD=MONO;SIZE=1.
PEN NUMBER=2;METHOD=POINT
AXES 1...2; PENAXES=1
FRAME WINDOW=1...4;YLOW=0.3,0.3,0.,0.;YUPP=1.,1.,0.3,0.3;\
XLOW=0.,0.48,0.,0.48;XUPP=0.52,1.0,0.52,1.0

OPEN 'CONFID_1.GRD';CHAN=1;FILETYPE=GRAPHICS
DGRAP [SCREEN=C;WIND=1;KEYW=3] VAR[1],VAR[2],LAI;DAY,DAY,TIMEEXP;PEN=1,1,2
DGRAP [SCREEN=K;WIND=2;KEYW=4] VAR2[1],VAR2[2],LAI;DAY,DAY,TIMEEXP;PEN=1,1,2

STOP
```

APPENDIX E - List of names and definitions

Variable name	Subroutine/Function Am	Co	Fs	Fu	Ou	Pr	Se	Si	Wr	Description
ALPHA	+									Variable that defines expansion and contraction
BETA	+									Variable that defines expansion and contraction
QT	+	+	+	+		+	+	+		Array with values of model performance
QTHIGH						+	+			Maximum value of model performance ('worst set')
QTLOW						+	+			Minimum value of model performance ('best set')
DAT		+								Array with experimental data
DUMMY						+		+		Dummy variable in real function FUNC
FTOL	+		+			+		+		Tolerance of values in array QT
GAMMA	+									Variable that defines expansion and contraction
HARD				+					+	Array with data on sampling time
HIGH		+								Array with maximum values of modeloutput per dataset
IBND	+			+		+		+	+	Variable which allows parametervalues in or outside bounds
IQT		+								Variable which defines the method of calculation of QT
IDATEB				+					+	Array with day numbers at start of simulation
IDSID		+		+					+	Array with dataset identifiers (0/1)
IDUM			+						+	Dummy array to store optional output switches
IHI	+									Pointer for set with highest function value
IHRD				+	+				+	Array with number of sampling dates per dataset
IIDAT		+								Actual number of data in array DAT
IIDSID									+	Actual number of datasetidentifiers
IIHRD				+					+	Array with pointers to array elements
ILBUF									+	Array length of variable XBUF
ILO	+									Pointer for set with lowest function value
IMETHD			+						+	Variable which defines the optimization algorithm
IMXDAT		+								Maximum number of data points
IMXHVS		+		+					+	Maximum number of sampling dates
IMXNDP		+		+						Maximum number of datapoints
IMXNDS		+		+					+	Maximum number of datasets
IMXNPS	+	+	+	+		+	+			Maximum number of parametersets
IMXNRR		+		+						Maximum number of reruns
IMXOUT			+			+		+		Maximum number of output-options
IMXPAR	+		+	+		+		+	+	Maximum number of parameters