

Site index SDE software Advanced and Programmers Notes

Oscar García

September 2009

Contents

1	Changes from <i>Old Manual</i>	2
2	Implementation	3
2.1	Data	3
2.2	Data preparation	3
2.3	Parameter estimation	4
3	Graphical User Interface	5
4	Model subroutines generation	5
4.1	Example: Fixed asymptote	6
4.2	Example: ALOCBH	8
4.3	Example: Grouping	9
A	TRPARS and TRDERS — Model specification and derivatives	12
B	Old Manual	14

I'll take the easy way out, writing around the old documentation in the *Old Manual* attached at the end of this document as Appendix B. Algorithms are explained in *Oz80.pdf*, which is a longer (older) version of the 1983 article in *Biometrics*. There is also *Report9.pdf* that describes in detail the code in an older version of the software, much of it still relevant. Appendix A contains gory details about derivative calculations, for the record.

Unless stated otherwise, everything here applies equally to the Microsoft Windows and Linux versions.

1 Changes from *Old Manual*

Section 2. The program is now (almost) standard Fortran 77 (actually, mostly Fortran 66). It has been restructured into 4 files: `Driver.f` has the main program and subroutines GET, PUT, and RDATA (these have been changed to use a different data structure, see below). `Guts.f` contains the main routines, unchanged (except for increasing the data array bounds from 20 to 100 in FUNCT and DERS): NEWTON, FUNCT, DERS, CHOL1, CHOL2, and CHINV. `Report.f` contains subroutine REPORT, unchanged. `Models.f` has TRPARS and TRDERS, with a number of model versions selectable through a model number; it may be substituted by a user-supplied model version. In EasySDE version 2.0 array dimensions were increased to handle up to 10 global parameters (was 6).

Section 3.1. The unit numbers are ignored, files are taken from the command line (the monitoring file is hard-wired to standard output). There are three additional items at the end of the control file: model number, and age and height scaling factors. For an example see the *CTRL.tmp* file generated by the GUI.

Section 3.2. For each plot, the structure is now:

- Plot identification (INTEGER*4).
- Number of measurements (integer ≤ 100).
- For each measurement: age and height (double precision), scaled through division by the scaling factors.

Although likely an overkill, the maximum allowed number of measurements for any one plot is now 100, up from the old 20 (memory is cheap these days).

Program *DataPrep* can be used to generate the Data and Parameter files, see Section 2.2.

Section 4. Subroutines TRPARS and TRDERS have some additional arguments, see the source code. If needed, the scaling factors and model number (from the end of the control file) are available through a COMMON block:

```
DOUBLE PRECISION tScale, hScale
INTEGER model
INTEGER*4 plotID
COMMON /ESDE/ tScale, hScale, plotID, model
```

The plot ID number, `plotID`, was added in EasySDE version 1.1. In Section 4 there is an example of when this might be useful.

Program *Generate* can be used to generate automatically these subroutines for any model specification, see Section 4.

2 Implementation

2.1 Data

Data for *DataPrep* must be provided as a text file where each record contains the plot number, age, and top height. All the records from a plot must be contiguous, ordered by increasing age (*DataPrep* checks this). Fields must be separated by white space (any number of spaces, tabs, and/or newlines).

Although not necessary for *DataPrep*, for use with *SitePlot* or gnuplot the plot number, age and height must be together in separate lines, and different plots must be separated by a blank line. See examples in the *Work* folder.

2.2 Data preparation

DataPrep.exe takes the ASCII data file, and optionally a file with initial local parameter values, and generates the intermediate binary data and parameter files for the estimation program *SDEfit*. The command line is:

DataPrep *InputData BinData BinPars InputPars*

InputData (input) is the ASCII input data file, in the format explained in Section 2.1.

InputPars (input) is an optional ASCII file containing some column with initial local parameter estimates, in the same order as *InputData*. Extra header lines are allowed, so that the report file from a previous run could be used. This argument can be omitted.

BinData (output), a binary unformatted Fortran file containing the data, for input to *SDEfit*.

BinPars (output), a binary direct-access file with the local parameter values, for input to *SDEfit*.

The program prompts for the numbers of global and local parameters, initial global estimates, and values and/or location of initial local estimates in *InputPars*, if any. If a local value is entered at the prompt, the same initial value is used for all plots. See the example in Section 4.1.

2.3 Parameter estimation

The main estimation program is *SDEfit.exe*, called as:

SDEfit *Control BinData BinPars Report*

Control (input) is the control data file explained in Section 3.1 of *Old Manual*, with the changes indicated here in Section 1.

BinData and *BinPars* are as described in Section 2.2. *BinData* is not altered. On exit *BinPars* contains the final local parameter estimates and other information (see Section 5.3 in *Old Manual*). It can be used as input to other *SDEfit* runs.

Report is the main output, see *Old Manual*, Section 5.2.

In addition, iteration information is sent to standard output (*Old Manual*, Section 5.1).

3 Graphical User Interface

The GUI consists of some Tcl/Tk scripts that serve as “glue” to run *DataPrep* and *SDEfit*.

The main program is *EasySDE.tcl*, which calls *modelinfo.ui.tcl* and *RunLog.ui.tcl*. These last two files were generated with SpecTcl, an interactive GUI builder (<http://dev.scriptics.com/software/spectcl/>, now at <http://spectcl/sporceforge.net>). If Tcl/Tk is installed, *EasySDE.tcl* can be run by double-clicking. Otherwise, it can be passed as an argument to *tclkit.exe*, a packaged version of Tcl/Tk (<http://www.equi4.com/tclkit>). In the Windows version this is implemented in the provided shortcuts.

EasySDE.tcl prompts for the data file, parameter estimates, and other information, and executes *DataPrep*, capturing its standard output. It does not use the optional input parameter file. The intermediate binary files are left in DATA.tmp and PARS.tmp. A control file CTRL.tmp is also generated. Then *SDEfit* is executed on these files, capturing the monitoring output, and a slightly edited version of the report file. The standard report file is left in REPORT.tmp. *EasySDE* has also an option for using DATA.tmp, PARS.tmp, and part of CTRL.tmp from a previous run (skipping *DataPrep*).

Also available is *SitePlot.tcl*, which executes gnuplot after preparing a command file in *SitePlot.plt*.

Note: The SpecTcl output tcl files were edited by hand for EasySDE version 2. Therefore, the intermediate SpecTcl *.ui files are out of date.

4 Model subroutines generation

Implementing alternative model versions used to be fairly involved and error-prone (Appendix A). It is now largely automated by *Generate.tcl*. This tcl script takes a model specification as used in *EasySDE*, for instance “G1+G2*L1 L1 G3 0 G4 G5 0 0”, and produces the Fortran code for the appropriate TRPARS and TRDERS subroutines. If parameter expressions are specified, as in this example, it may prompt for derivatives with respect to the global and local variables.

Each basic parameter term must be written without spaces. Use the Fortran/gnuplot exponentiation operator “**”, e.g., `G1*L1**G2`. The script is case-sensitive.

In general, the subroutines are generated complete, in their final form. Only in some more complex situations manual editing is required, see Example 4.3 below.

Save the generated subroutines, e.g. as `MyModel.f`, and compile together with `Driver.f`, `Guts.f`, and `Report.f`.

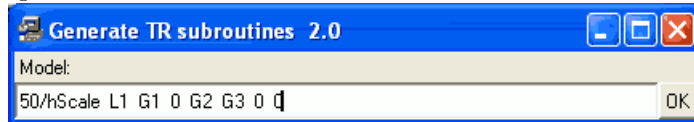
For instance (on Windows), using the gnu G77 Fortran compiler installed in the standard directory, just drag and drop `MyModel.f` on top of the *compile.bat* file provided. This produces the executable `MyModel.f.exe`, that can be used in the same way as *SDEfit.exe*.

4.1 Example: Fixed asymptote

We assume that on Windows the g77 Fortran compiler is available, e.g., after unzipping <http://forestgrowth.unbc.ca/sde/g77win.zip> into *C:*. On Linux the *compile* script checks for *g77*, *gfortran*, or *fort77* (*f2c*).

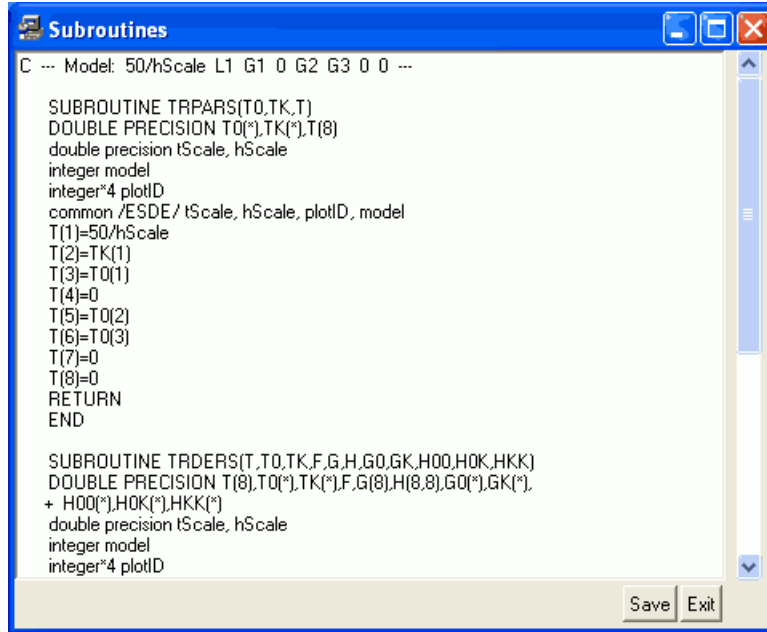
Suppose that our data is too young to provide a reliable estimate for the asymptote *a*, but that we have a reasonable guess from other sources (e.g., *Forest Ecology and Management* 229: 1–6, 2006). Specifically, we want a model like *BLOCAL*, but with *a* fixed, say at 50 m. We may proceed as follows:

1. Run *Generate.tcl*, e.g., on MS Windows, open a command window (Start > All Programs > Accessories > Command Prompt), change directory (folder) by entering `cd C:\EasySDE`, and enter `tclkit Generate.tcl`. On Linux, in the *EasySDE* directory enter `./Generate.tcl`. A small window appears where you enter the model specification:



This means that *a* is 50 divided by the height scale factor, *b* is the first and only local parameter, *c* is global, $\sigma_0 = 0$, σ and σ_m are globals,

and $t_0 = H_0^c = 0$. In this instance, this is all the information needed to generate the *TRPARS* and *TRDERS* subroutines:



```

C --- Model: 50/hScale L1 G1 0 G2 G3 0 0 ---

SUBROUTINE TRPARS(T0,TK,T)
DOUBLE PRECISION T0(*),TK(*),T(8)
double precision tScale, hScale
integer model
integer*4 plotID
common /ESDE/ tScale, hScale, plotID, model
T(1)=50/hScale
T(2)=TK(1)
T(3)=T0(1)
T(4)=0
T(5)=T0(2)
T(6)=T0(3)
T(7)=0
T(8)=0
RETURN
END

SUBROUTINE TRDERS(T,T0,TK,F,G,H,G0,GK,H00,H0K,HKK)
DOUBLE PRECISION T(8),T0(*),TK(*),F,G(8),H(8,8),G0(*),GK(*),
+ H00(*),H0K(*),HKK(*)
double precision tScale, hScale
integer model
integer*4 plotID

```

Save as *MyModel.f*.

2. Now compile, entering
`compile MyModel.f`
(or `./compile MyModel.f` on Linux). This creates an executable named *MyModel.f.exe*. There may be some compiler warnings about possible loss in precision from converting 50 to double-precision. These can be ignored, or avoided by giving the value as a double-precision constant 50.0d0 (or 50d0, or 5d1, etc., perhaps saving a few execution milliseconds).
3. Assuming that your data file is *MyData.dat*, in directory *Work*, create the binary data and parameter files, providing the required info when prompted:

```

C:\EasySDE>DataPrep Work\MyData.dat DATA.tmp PARS.dat
Enter number of locals, minimum required measurements per plot,
age scale, height scale, columns in parameter file, lines to
skip at the start, position of plot ID column:
1 2 10 10 0 0 0

Enter positions of each local in parameter file,

```

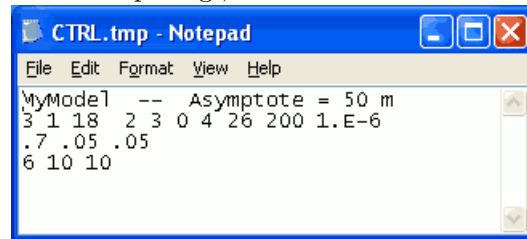
or 0 if given below:
0

Enter initial local estimates (ignored if given in file):
.2

Using 18 plots

(if that is the number of plots in *MyData.dat*). Use *./DataPrep.exe* ... on Linux.

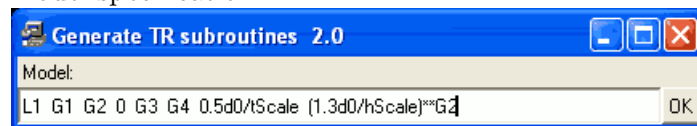
4. Prepare the control file as explained in Section 1, and save it as *CTRL.tmp*, for instance. Perhaps the easiest is to edit an existing *CTRL.tmp*. E.g.,



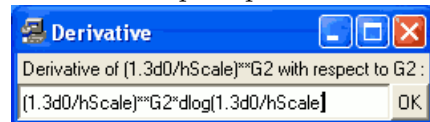
5. Run our custom model fitting program:
MyModel.f.exe CTRL.tmp DATA.tmp PARS.tmp MyModel.rpt
(or *./MyModel.f.exe ...* on Linux). If everything goes well, the iteration log displays on screen, and the results are left in *MyModel.rpt*.

4.2 Example: ALOCBH

The *ALOCBH* model is already available in the GUI, but let us make it the hard way for illustration purposes. Running *Generate.tcl*, we have the model specification:

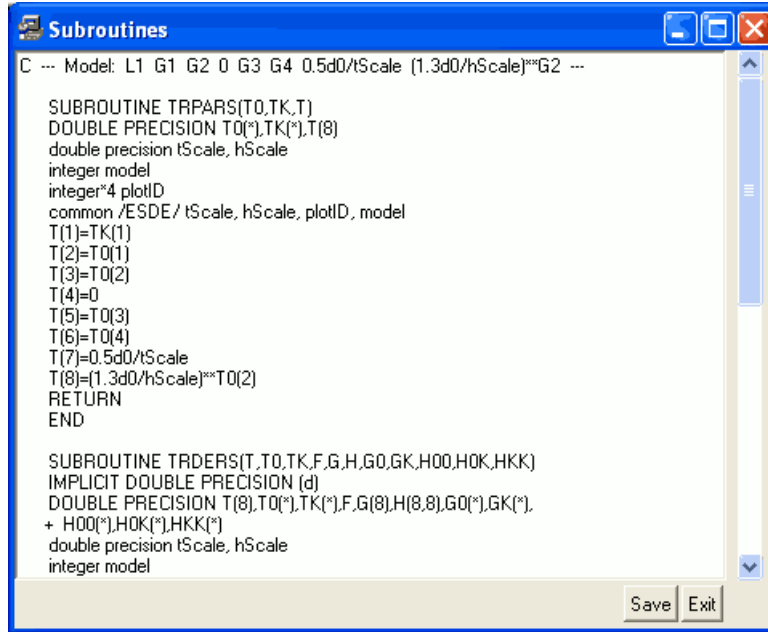


Then we are prompted for some derivative expressions:





Although not strictly necessary, I specified the double-precision *dlog()* instead of just *log()* to keep the compiler happy. The subroutines are generated:



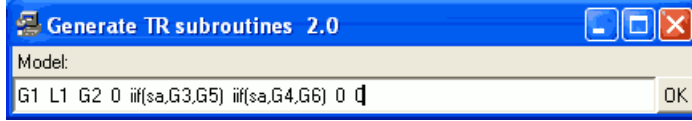
Compare to those in *Models.f*. The rest is similar to example 4.1.

4.3 Example: Grouping

This is a little more ambitious. We have data from both stem analysis (SA) and from permanent sample plots (PSP). We want to fit *BLOCAL* but allowing the variances to differ between the two groups of plots, because the error structures are likely to be different (e.g., *Forest Science* 51: 277–283, 2005). The SA plots can be distinguished by the plot ID being greater than 10000.

One way of handling this is through defining a function, say *iif*(, ,), that returns its second argument if the first (logical) argument is true, or returns its third argument otherwise. This is like the *IIF* function in Excel or OpenOffice Calc, or like the *?:* operator in C.

Assuming that *sa* is a logical variable that is true for SA plots and false for PSP, the model specification can be given to *Generate.tcl* as



The σ and σ_m parameters for SA will be the 3rd and 4th globals, and the 5th and 6th for the PSP. Now some derivatives:



The output from *Generate.tcl* needs to be edited by hand to add the conditional function, set the *sa* logical according to plot number, and include type declarations for *iif* and *sa*. The result looks like this:

```
C --- Model: G1 L1 G2 0 iif(sa,G3,G5) iif(sa,G4,G6) 0 0 ---

double precision function iif(condition, trueValue, falseValue)
logical condition
double precision trueValue, falseValue
iif = falseValue
if (condition) iif = trueValue
end

SUBROUTINE TRPARS(TO,TK,T)
DOUBLE PRECISION TO(*),TK(*),T(8)
double precision tScale, hScale
integer model
integer*4 plotID
common /ESDE/ tScale, hScale, plotID, model
double precision iif
```

```

logical sa
sa = plotID .gt. 10000
T(1)=T0(1)
T(2)=TK(1)
T(3)=T0(2)
T(4)=0
T(5)=iif(sa,T0(3),T0(5))
T(6)=iif(sa,T0(4),T0(6))
T(7)=0
T(8)=0
RETURN
END

SUBROUTINE TRDERS(T,T0,TK,F,G,H,GO,GK,H00,HOK,HKK)
IMPLICIT DOUBLE PRECISION (d)
DOUBLE PRECISION T(8),T0(*),TK(*),F,G(8),H(8,8),GO(*),GK(*),
+ H00(*),HOK(*),HKK(*)
double precision tScale, hScale
integer model
integer*4 plotID
common /ESDE/ tScale, hScale, plotID, model
double precision iif
logical sa
sa = plotID .gt. 10000
d5g3=iif(sa,1d0,0d0)
...

```

The additions in the subroutines are the three lines following the *common* statement.

A TRPARS and TRDERS — Model specification and derivatives

Denote by θ_i the basic model parameters, with

$$\begin{aligned}\theta_1 &= a, & \theta_2 &= b, & \theta_3 &= c, & \theta_4 &= \sigma_0, \\ \theta_5 &= \sigma, & \theta_6 &= \sigma_m, & \theta_7 &= t_0, & \theta_8 &= H_0^c.\end{aligned}$$

Let Gi be the global parameters, and $L1$ the local parameter. We assume here that there is only one local.

A model can be specified by showing how each θ_i depends on the global and local parameters:

$$\theta_i = f(G1, G2, \dots, L1) .$$

This is how the model is given in the GUI and in *Generate.tcl*. Essentially the same thing is done in TRPARS, with $\theta_i \equiv T(i)$, $Gi \equiv T0(i)$, and $L1 \equiv TK(1)$. See Section 4 of the *Old Manual*. $T0()$ and $TK()$ correspond to θ_0 and θ_k in the 1980 paper (*Oz.pdf*).

In TRDERS, first and second derivatives of the log-likelihood with respect to the Gi and $L1$ must be obtained from the derivatives with respect to the θ_i . The first derivatives with respect to θ_i are available in the input vector $G()$, with $G(i) = \partial f / \partial \theta_i$, where f is the log-likelihood. The second derivatives are in the (Hessian) matrix $H(,)$, with $H(i, j) = \partial^2 f / \partial \theta_i \partial \theta_j$.

The first derivatives with respect to the globals and local must be given in the vectors $G0()$ and $GK()$, respectively:

$$\begin{aligned}G0(i) &= \frac{\partial f}{\partial Gi} = \sum_j \frac{\partial f}{\partial \theta_j} \frac{\partial \theta_j}{\partial Gi} = \sum_j G(j) \frac{\partial \theta_j}{\partial Gi} \\ GK(1) &= \frac{\partial f}{\partial L1} = \sum_j \frac{\partial f}{\partial \theta_j} \frac{\partial \theta_j}{\partial L1} = \sum_j G(j) \frac{\partial \theta_j}{\partial L1}\end{aligned}$$

Of course, most terms in the sums will be zero.

The second derivatives are somewhat more complicated, and are output in the vectors $H00()$, $H0K()$, and $HKK()$, for globals, globals \times local, and local, respectively. The vector $H00()$ stores the upper-triangular part of the Hessian with respect to the global parameters (which is symmetric), in

column order. For instance, with 5 global parameters, the indices in $\text{H00}()$ for the elements of the Hessian would be:

$$\begin{bmatrix} 1 & 2 & 4 & 7 & 11 \\ - & 3 & 5 & 8 & 12 \\ - & - & 6 & 9 & 13 \\ - & - & - & 10 & 14 \\ - & - & - & - & 15 \end{bmatrix}$$

In other words, if k is the index in $\text{H00}()$ of the element (i, j) of the Hessian, one has:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i	1	1	2	1	2	3	1	2	3	4	1	2	3	4	5
j	1	2	2	3	3	3	4	4	4	4	5	5	5	5	5

The second derivatives can be calculated as:

$$\begin{aligned} \text{H00}(\mathbf{k}) &= \frac{\partial^2 f}{\partial G_i \partial G_j} = \frac{\partial}{\partial G_j} \text{G0}(\mathbf{i}) \\ &= \frac{\partial}{\partial G_j} \left(\sum_p \text{G}(\mathbf{p}) \frac{\partial \theta_p}{\partial G_i} \right) \\ &= \sum_p \frac{\partial \text{G}(\mathbf{p})}{\partial G_j} \frac{\partial \theta_p}{\partial G_i} + \sum_p \text{G}(\mathbf{p}) \frac{\partial^2 \theta_p}{\partial G_i \partial G_j} \\ &= \sum_p \left(\sum_q \frac{\partial \text{G}(\mathbf{p})}{\partial \theta_q} \frac{\partial \theta_q}{\partial G_j} \right) \frac{\partial \theta_p}{\partial G_i} + \sum_p \text{G}(\mathbf{p}) \frac{\partial^2 \theta_p}{\partial G_i \partial G_j} \\ &= \sum_p \sum_q H_{pq} \frac{\partial \theta_p}{\partial G_i} \frac{\partial \theta_q}{\partial G_j} + \sum_p \text{G}(\mathbf{p}) \frac{\partial^2 \theta_p}{\partial G_i \partial G_j} , \end{aligned}$$

where $H_{pq} = \text{H}(\mathbf{p}, \mathbf{q})$ if $p \leq q$, or $\text{H}(\mathbf{q}, \mathbf{p})$ otherwise,

$$\text{H0K}(\mathbf{i}) = \frac{\partial^2 f}{\partial G_i \partial L1} = \sum_p \sum_q H_{pq} \frac{\partial \theta_p}{\partial G_i} \frac{\partial \theta_q}{\partial L1} + \sum_p \text{G}(\mathbf{p}) \frac{\partial^2 \theta_p}{\partial G_i \partial L1} ,$$

and

$$\text{HKK}(1) = \frac{\partial^2 f}{\partial L1^2} = \sum_p \sum_q H_{pq} \frac{\partial \theta_p}{\partial L1} \frac{\partial \theta_q}{\partial L1} + \sum_p \text{G}(\mathbf{p}) \frac{\partial^2 \theta_p}{\partial L1^2} .$$

Again, most terms will be zero.

B Old Manual

PROGRAM FOR FITTING HEIGHT GROWTH MODELS

1 PRINCIPLES

1.1 Basic model

Height growth is modelled by a differential equation

$$dH^c/dt = b(a^c - H^c) + \sqrt{2b}\sigma\dot{w}(t), \quad (1)$$

with initial condition

$$H^c(t_0) = H_0^c + \epsilon_0 ; \quad \epsilon_0 \sim N(0, \sigma_0^2),$$

where H is the top height, t is age, w is a standardised Wiener process representing “environmental” variation, ϵ_0 is a normal random variable, and $a, b, c, t_0, H_0, \sigma$ and σ_0 are parameters to be estimated. In addition, it is assumed that there are also measurement errors so that the measured heights, h , satisfy

$$h^c = H^c + \sigma_m \epsilon, \quad (2)$$

where the ϵ are independent standard normal variables, and σ_m is another parameter to be estimated.

Integration of (1) gives

$$H = a\{1 - (1 - H_0^c/a^c) \exp[-b(t - t_0)]\}^{1/c} + \eta(t), \quad (3)$$

where $\eta(t)$ is a random error term with a distribution dependent on the parameters (see García 1977,1980,1983 for details).

1.2 Model versions

The model described in 1.1 applies to a particular plot or stand. Some or all of the 8 parameters $a, b, c, t_0, H_0, \sigma, \sigma_m$ and σ_0 may be different for different stands, reflecting differences in site quality. A model for a region or for a forest must specify how the parameters differ between stands.

In a particular version of the model, parameters may be fixed at given values, may be constrained to have the same (unspecified) value for all the plots, or may be free

to take different values for different plots. It is also possible to have relationships between the 8 *basic parameters* fixed or constrained to a common value. This is done by defining new *secondary parameters* which are functions of the old parameters.

A version is specified then by classifying each of eight or more parameters (some of which may be secondary parameters) as *fixed* (given value), *global* (same value for all plots), or *local* (free).

Note: In order for the model to be identifiable, either t_0 or H_0 must be fixed. Also, the number of free parameters is limited by the minimum number of measurements among the plots.

1.2.1 Examples

The usual site index models require that only one parameter other than the σ 's can be local. Parameters σ , σ_m and σ_0 do not affect the form of the height-age curves; they are specific to this program, affecting the estimation procedure through the assumed distribution of the random error terms. Parameters t_0 and H_0 define the origin of the curves and are usually fixed to zero, although in some cases could be used to adjust for the effect of frosts on the initial growth, or for delays due to natural regeneration or different establishment techniques. Some examples of model specification:

- (a) Anamorphic site index curves. Heights on different sites are in the same proportion for all ages. Obtainable with a local, b and c global.
- (b) Common upper asymptote, e.g. site index curves in Elliott and Goulding's Kaingaroa growth model. Have a and c global, b local.
- (c) Burkhardt and Tennent site index equations. Could be specified by setting $b = \beta S$, $a = S/(1 - \exp[20\beta S])^{1/c}$, and making β and c global and S local.
- (d) Southland growth model. The relationship $a = \alpha + \beta S$, with α , β and c global and S local was tested. This includes a) and b) as special cases ($\alpha = 0$ and $\beta = 0$, respectively). Here S , the Site Index, is related to b by $S = a(1 - \exp[-20b])^{1/c}$, that is, $b = -\ln\{1 - [S/(\alpha + \beta S)]^c\}/20$. The measurement standard error σ_m was taken as local and σ as global. Values of t_0 different from zero were tried.

1.3 Estimation procedure

The program computes the likelihood function for the model and searches for parameter values that maximise the function. Specifically, a modified Newton method is used to minimise $-2 \ln L$, where L is the likelihood function (see García 1980, 1983).

An initial estimate for the parameters is needed for running the program. Good initial estimates will save computer time and reduce the risk of non-convergence or local minima.

There is a possibility of reaching only a local maximum of the likelihood. It is advisable to partially check the estimates (at least for the final model) by running the program from different starting points.

2 PROGRAM

The program is written in Fortran for the ICL 2980. It is structured as a main program and 11 subroutines, two of them supplied by the user.

The main program reads in the problem description and global parameters and calls subroutines NEWTON and REPORT. Subroutine NEWTON is the optimisation procedure. At each iteration it reads and writes current local parameter estimates and derivatives through subroutines GET and PUT, and calls TRPARS and FUNCT for computing the likelihood, and TRPARS, DERS and TRDERS for computing derivatives. It also uses subroutines CHOL1 and CHOL2 to perform Cholesky factorisation operations. Subroutine REPORT prints the final estimates and approximate standard errors and correlations. Uses GET, TRPARS, FUNCT, CHOL2 and CHINV.

Subroutines FUNCT and DERS compute the likelihood function and derivatives given the basic parameters. They use RDATA for reading the data. The user must supply subroutines TRPARS and TRDERS for performing the mapping between basic and global and local parameters and derivatives (see section 4, below).

3 INPUT

3.1 Control and global parameters

The following data is read from logical unit 1, in free format:

- An alphanumeric problem identification (first record, 80 characters).
- Number of global parameters.
- Number of local parameters.
- Number of plots.
- Unit number for data file.
- Unit number for parameter (direct access) file.
- Unit number for monitoring file.
- Unit number for report file (final results).
- Monitoring frequency, i.e., interval between plots for which values are output at each iteration.
- Iteration limit (maximum number of iterations).
- Tolerance for convergence tests (10^{-5} is recommended).
- Initial estimates for global parameters.

3.2 Data file

Unformatted Fortran sequential file. For each plot it must contain, in this order:

- Plot identification (integer).

- Number of measurements (integer).
- 20 values for the ages, in decades (double precision reals).
- 20 values for the heights, in decametres (double precision reals).

This file is not altered by the program. Read by subroutine RDATA. The program could easily be modified for keeping this data in memory.

3.3 Parameter file

Unformatted direct access file. Record length must be sufficient for containing $m_k + m_k(m_0 + 1) + \frac{1}{2}m_k(m_k + 1)$ double precision reals, where m_0 = number of global parameters and m_k = number of local parameters. It is convenient to set the record length to the maximum that might be needed by any model: 39 double precision reals. As many records as there are plots are needed.

On entry to the program, the initial estimates for the local parameters must appear in the first m_k positions of each record. Plots must appear in the same order as in the data file.

On exit, the final estimates for the local parameters appear on the first m_k positions of each record.

Alternatively, an array in memory could be used instead of a file. For doing this, subroutines SETPF, GET, PUT, and SAVEPF need to be altered. Subroutine SETPF can be used to create the array, e.g. by reading from some external file. Subroutines GET and PUT are used for reading and writing record K, respectively, Subroutine SAVEPF can be used for saving the final contents of the array.

4 MODEL SPECIFICATION

A particular model version, i.e. which parameters or functions of the parameters are taken as global, local or fixed, is specified through the user-supplied subroutines TRPARS and TRDERS.

4.1 TRPARS

```
SUBROUTINE TRPARS(T0,TK,T)
      T0 : vector of global parameters (input)
      TK : vector of local parameters (input)
      T  : vector of 8 basic model parameters (output)
```

The subroutine must define the basic parameters given the global and local parameters. The basic model parameters are:

$$\begin{array}{ll} T(1) = a & T(5) = \sigma \\ T(2) = b & T(6) = \sigma_m \\ T(3) = c & T(7) = t_0 \\ T(4) = \sigma_0 & T(8) = H_0^c \end{array}$$

Note that at least one of T(7), T(8) must be fixed (or a functional relationship must be defined between them).

T0, TK and T must be declared as DOUBLE PRECISION.

Example, corresponding to 1.2.1 (b):

```
SUBROUTINE TRPARS(T0,TK,T)
DOUBLE PRECISION T0(4),TK(1),T(8)
T(1) = T0(1)
T(2) = TK(1)
T(3) = T0(2)
T(4) = 0.0D0
T(5) = T0(3)
T(6) = T0(4)
T(7) = 0.0D0
T(8) = 0.0D0
RETURN
END
```

See Appendix 2 for further examples.

4.2 TRDERS

```
SUBROUTINE TRDERS(G,H,G0,GK,H00,HOK,HKK)
  G   : vector of first derivatives for basic parameters
        (input)
  H   : matrix of second ders. for basic parms.(input)
  G0  : vector of first ders. for global parms.(output)
  GK  : vector of first ders. for local parms.(output)
  H00 : vector of second ders. for global parms. (output)
  HOK : vector of cross-ders. between global and local
        parms (output)
  HKK: vector of second ders. for local parms. (output)
```

This subroutine takes the first and second derivatives of the log-likelihood with respect to the basic model parameters, and produces the derivatives with respect to the global and local parameters of a particular model version. Any parameter transformations must be accounted for.

H is a 8×8 matrix, but the derivatives are contained only in its upper-triangular part [i.e. $H(I,J)$ is valid only for $I \leq J$].

H00 and HKK are vectors containing the elements in the upper-triangular parts of the respective second derivative matrices in column order (or the lower-triangular parts in row order).

HOK contains the elements from the matrix of cross-derivatives with respect to global parameters (rows) and local parameters (columns), in row order.

Example, corresponding to 1.2.1 (b):

```
SUBROUTINE TRDERS(G,H,G0,GK,H00,HOK,HKK)
```

```

DOUBLE PRECISION G(8),H(8,8),G0(4),GK(1),
1      H00(10),H0K(4),HKK(1)
G0(1) = G(1)
G0(2) = G(3)
G0(3) = G(5)
G0(4) = G(6)
GK(1) = G(2)
H00(1) = H(1,1)
H00(2) = H(1,3)
H00(3) = H(3,3)
H00(4) = H(1,5)
H00(5) = H(3,5)
H00(6) = H(5,5)
H00(7) = H(1,6)
H00(8) = H(3,6)
H00(9) = H(5,6)
H00(10) = H(6,6)
HKK(1) = H(2,2)
H0K(1) = H(1,2)
H0K(2) = H(2,3)
H0K(3) = H(2,5)
H0K(4) = H(2,6)
RETURN
END

```

5 OUTPUT

5.1 Monitoring file

For each iteration the following is printed.

- For each plot for which the sequence number is a multiple of the monitoring frequency:
 - PLOT: plot number
 - FUNCTION: the contribution of the plot to the function being minimised, i.e., -2 times the log-likelihood
 - LOCAL PARAMETERS
- At the end of the iteration:
 - ITER: iteration number
 - FUNCT: -2 times the log-likelihood
 - NPD: Number of plots for which the Hessians H_{kk} are not positive definite (see García 1980, sections 4.2 and 4.3)

- NPD0: 1 if the matrix with factor L_{00} in equation (36) in García 1980 is not positive-definite, 0 otherwise.
- DMAX: maximum of the absolute values of the elements of δ
- GMAX: maximum of the absolute values of the elements of the gradient
- ALPHA: value of α (García 1980, section 4.2)
- GLOBAL PARMS.: current values of the global parameters

5.2 Report file

This file contains the problem identification, termination condition (“CONVERGED” for a normal run), and the following groups of information.

- For all the plots the following is printed:
 - Plot number
 - Number of measurements
 - Contribution to -2 times the log-likelihood
 - Estimates for the local parameters
 - Approximate standard errors for the estimates (based on the inverse Hessian)
- General info.:
 - Number of plots
 - Total number of measurements
 - Log-likelihood
- Info. on global parameters:
 - Estimates
 - Approx. standard errors for the estimates
 - Approx. correlations between the estimates, based on the inverse Hessian (lower triangular part of the matrix is printed)
- Info. on averages for local parameters, weighted according to number of measurements:
 - Average of estimates
 - Mean standard errors (from average of variances)
 - Mean correlations between local parameters (from average of covariances)
 - Mean correlations between local and global parameters (from average of covariances)
- Same as above but for the absolute values of the parameter estimates. This is useful for the σ 's, where the sign is irrelevant.

5.3 Parameter (direct access) file

The following information is left in each record of the direct access parameter file (one record corresponds to each plot, in the same order as in the input data file):

- Local parameter estimates
- Contribution to -2 times the log-likelihood
- δ_k
- H_{0k} , by rows
- Lower-triangular part of L_{kk}

References

- [1] García, O. (1977). *Height growth in radiata pine. — I. General model and estimation procedure. Progress report.* Forest Mensuration Internal Report No. 9, June 1977.
- [2] García, O. (1980). *A stochastic differential equation model for the height growth of forest stands.* Paper presented at the 5th Australian Statistical Conference, Sydney, August 1980 (unpublished).
- [3] García, O. (1983). *A stochastic differential equation model for the height growth of forest stands.* **Biometrics** **39**, 1059-1072.

APPENDIX 1. Program listing.

APPENDIX 2. TRPARS and TRDERS examples.

APPENDIX 3. Test data and results.

Oscar García
Revised April 1984
Converted to L^AT_EX April 1993