

MiX99

Solving Large Mixed Model Equations

Release IX/2025

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

Last update: Sept 2025
©Copyright 2025


Luke
NATURAL RESOURCES
INSTITUTE FINLAND

Preface

The development of MiX99 was initiated in the late 1990's to allow more sophisticated models for estimating breeding values in dairy cattle. At that time, the focus was on computational efficiency and the target users were experts in dairy genetic evaluations. Therefore, the primary application of this software was in solving large-scale genetic and genomic evaluations for national and international dairy evaluations. However, over the years we have developed the software into a general tool where many models can be used. As a result, MiX99 is used in genetic evaluation of many livestock species, plant breeding and research, in addition to cattle.

About this manual

We explain the command language interface for MiX99, called CLIM, and provide all the basic information required to perform a range of analyses with MiX99. In addition, two supplementary manuals are available: the "Technical Reference Guide for MiX99 Preprocessor" and the "Technical Reference Guide for MiX99 Solver". These guides give additional support for very specific models and for the approximation of reliabilities.

Disclaimer

The MiX99 software is owned by Natural Resources Institute Finland (Luke). When using this program you agree with the following terms. You are not allowed to distribute, copy, give or transfer MiX99, under the same or any other name. Any decisions based on the information provided by MiX99 are made at your own responsibility and risk. Only limited technical support can be provided, but vital questions on its use can be directed to the authors (mix99@luke.fi). Please report any bugs to the authors. MiX99 can be cited by ([MiX99 Development Team, 2025](#)) and ([Pitkänen et al., 2022](#)). If you would like to use MiX99, please contact Natural Resources Institute Finland¹.

MiX99 new (NEW) and development (DEV) features

New MiX99 features are indicated in the documentation by a colored vertical bar and note "NEW" on the right margin.

NEW

Some of the newest MiX99 features currently in development are not yet available in the official MiX99 release. These new MiX99 development features are indicated in the documentation by a colored vertical bar and note "DEV" on the right margin.

DEV

Authors

Martin Lidauer, Matti Taskinen, Kaarina Matilainen, Esa Mäntysaari, Timo Pitkänen, Ismo Strandén

Natural Resources Institute Finland (Luke),
FI-31600 Jokioinen, Finland

<http://www.luke.fi/mix99>

¹MiX99 Development Team, Natural Resources Institute Finland (Luke), FI-31600 Jokioinen, Finland.

Contents

1	Introduction	1
2	Computing Methods	2
2.1	Preconditioned conjugate gradient method	2
2.2	Iteration on data technique	2
2.3	Data work file reduction	2
2.4	Equation family blocks	3
3	How to run MiX99 solver programs	4
3.1	Computing environment	4
3.2	MiX99 solver programs	4
3.3	Multi-threaded MiX99 solvers	4
3.4	Running the solver	5
4	The MiX99 solver option file	7
4.1	Solver option lines	7
4.2	Command line options	17
4.3	Sparse regression matrix (srm)	18
4.4	No residual covariances (noc)	18
4.5	Second level preconditioner (sp)	19
4.6	Determining convergence	19
4.6.1	Choosing a suitable convergence criterion	20
4.6.2	Effect of preconditioning on convergence	21
4.7	External STOP file: stopping iteration	22
4.8	External PEEK file: intermediate solutions during iteration	22
4.9	External ITER file: changing parameters during iteration	23
5	Output files of the MiX99 solvers	25
5.1	Standard output	25
5.2	Successful execution of MiX99 solver	26
5.3	Solution files	28
5.3.1	Formatted solution files	28
5.3.2	Unformatted solution files	28
5.4	Files for model validation purposes	29
6	Reliabilities	30
6.1	Approximate reliabilities using ApaX	30
6.1.1	Approximate reliabilities for single-step	31
6.1.2	Differences of reliability calculation and breeding value estimation	31
6.1.3	ApaX instruction file	33
6.1.4	Guidelines for determining blocking and JFilter	36
6.1.5	ApaX Output files	37

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

6.1.6	Example of ApaX instruction file	38
6.2	Exact reliabilities using exa99	40
6.2.1	Option file for exa99	40
6.2.2	Exa99 output files	42
6.3	Reversed reliability approximation	43
6.3.1	Single-trait Reversed reliability approximation (AccurType 20)	43
6.3.2	Multi-trait Reversed reliability approximation (AccurType 40)	45
7	Daughter Yield Deviations	47
7.1	Calculation of daughter yield deviations	47
7.1.1	Pedigree file	47
7.1.2	MiX99 instruction file	47
7.1.3	MiX99 solver option file	48
7.2	Solution files for daughter yield deviations	48
7.3	Example	48
8	Non-linear models	51
8.1	Threshold-model	51
8.1.1	Instruction file for <code>mix99i</code>	51
8.1.2	Stopping criterion file for <code>mix99s</code>	51
8.1.3	Solution files	52
8.1.4	Example	52
8.2	Gompertz-model	52
8.2.1	Instruction file for <code>mix99i</code>	52
8.2.2	Stopping criterion file for <code>mix99s</code>	53
8.2.3	Solution files	53
8.2.4	Example	53
9	Estimation of variance components	54
9.1	Running MC EM REML	54
9.2	File with starting values of (co)variance components	55
9.3	MiX99 instruction file	55
9.4	MiX99 solver option file	56
9.4.1	Number of data samples	56
9.4.2	Determining convergence of REML parameter estimates	56
9.4.3	Keeping certain variance components fixed	57
9.4.4	MC EM REML for MACE	57
9.5	Standard errors for REML parameter estimates	57
9.6	Solution files for variance components	59
9.7	Example	60
10	Accounting for heterogeneous variance	65
10.1	Computation environment	65
10.2	Models for the heterogeneity of variances	65
10.2.1	Currently supported <i>variance models</i> by MiX99	65
10.3	Input data for the variance model	68
10.3.1	Input data for <code>mix99hv</code>	68
10.3.2	Instruction file for <code>mix99hv</code>	68
10.3.3	Output files from <code>mix99hv</code>	70
10.4	Instruction file for the variance model	71

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

10.5	Variance components for the variance model	74
10.5.1	Files with information for the variance component estimation	74
10.6	Standardization of the multiplicative adjustment factors	75
10.6.1	Files for the standardization procedure	75
10.6.2	Standardization process	75
10.6.3	Multiple residual variance-covariance matrices	76
10.7	Running a model with heterogeneous variance	76
10.7.1	Implementation on shared memory platforms	77
10.8	Workflow and needed files for running multiplicative mixed model	77
10.9	Example	80
10.10	Output files	82
10.10.1	Mi.log and Ms.log	82
10.10.2	HVd.log	83
10.10.3	CYC.log	83
10.10.4	Lambda.log	84
11	Acknowledgement	85
12	References	85
Index		88

1 Introduction

Dairy cattle breeders around the world have moved to so called test-day models from plain animal models for 305d data. Usually this model upgrade leads to a manifold increase in the computing time. This is because test-day models have more effects than traditional models based on 305d data, and also because number of records increase about ten times. In a national evaluation, number of test-day records and number of unknowns in the mixed model equations ([MME](#)) are easily more than 100 million. In the last decade, genomic information has lead to vast increase in dense marker data. Thus, sparse data by animal model is combined with dense genomic data. Consequently, computing times have increased.

Computational techniques and algorithms that were found useful for solving animal models may take weeks to obtain solution to large random regression test day or genomic data model [MME](#). Consequently, faster solving algorithms have been developed. [Strandén and Lidauer \(1999\)](#) and [Lidauer et al. \(1999\)](#) advocated the use of [preconditioned conjugate gradient \(PCG\) method](#). [Lidauer and Strandén \(1998\)](#) and [Strandén \(1999\)](#) showed the usefulness of [parallel computing](#). These techniques have been found to reduce computing time considerably. Genomic data has lead to development of new techniques for including the rapid increase of genomic data into the models.

MiX99 development work has focused on incorporation of new techniques into an iteration on data BLUP-program ([Lidauer and Strandén, 1999](#)). We have also extended the software with programs for different needs: a general program that [approximate reliabilities](#) of bulls' estimated breeding values (EBV), as required by the Interbull; a general program that calculates [exact reliabilities](#) of EBVs via inversion of the coefficient matrix; and programs, which are required when accounting for [heterogeneous variance](#). Although the programs have been designed primarily for genetic evaluation of dairy cattle, the programs can and are used for other species that use other types of statistical models.

The MiX99 package consists of two main programs: pre-processor and solver. The pre-processor program [mix99i](#) reads model instructions, examines input data, and computes data sets for the solver program [mix99s](#), which solves the [MME](#) by [iteration on data](#). The [MME](#) can also be solved by the program [mix99p](#), which is designed to use several CPUs in [parallel](#). In addition, the pre-processed data can be analyzed with additional programs ([apax99](#), [apax99p](#) and [exa99](#)) to compute accuracies of the breeding values. For giving instructions to [mix99i](#) please see the manuals [Command Language Interface for MiX99](#) and [Technical reference guide for MiX99 pre-processor](#). In this reference guide we describe the solver options and instructions for [mix99s](#), [mix99p](#), [apax99](#), [apax99p](#), and [exa99](#).

2 Computing Methods

2.1 Preconditioned conjugate gradient method

The method of conjugate gradient (CG) is an iterative method to solve a linear system $\mathbf{Cs} = \mathbf{r}$. It is based on a geometric approach (Shewchuk, 1994). In breeding value estimation, the $\mathbf{C}$ matrix corresponds to the coefficient matrix of mixed model equations (MME), the $\mathbf{s}$ vector contains the solutions, and the $\mathbf{r}$ vector is the right-hand side of MME.

Preconditioned conjugate gradient method (PCG) uses conjugate gradient method on a transformed problem. Preconditioning is equivalent to solving $\mathbf{M}^{-1}\mathbf{Cs} = \mathbf{M}^{-1}\mathbf{r}$, where $\mathbf{M}$ is a symmetric positive definite **preconditioner matrix** that approximates $\mathbf{C}$. Together with a suitable preconditioner matrix, convergence rate of the CG method is superior to other commonly used algorithms for solving MME (Lidauer et al., 1999).

MiX99 program creates the **preconditioner matrix** $\mathbf{M}$, which comprises of diagonal blocks of the coefficient matrix. Implementation of the PCG algorithm using **iteration on data** (IOD) technique requires keeping four vectors, of size equal to the number of unknowns in the MME, in memory and to read the data and the **preconditioner matrix** once per iteration round. The algorithm does not require any pre-set tuning parameters like relaxation factors.

2.2 Iteration on data technique

The major computational task in PCG is the multiplication of the coefficient matrix with a vector each round of iteration. Therefore, in IOD all data records must be read and processed. IOD technique requires for each record a certain amount (N) of floating point operations to calculate the product coefficient matrix times a vector corresponding to the record. In MiX99, N increases almost linearly with increasing complexity of the statistical model; $N=2(2f+t^2)$, where f is the number of effects in the model, and t is the number of traits in the model.

2.3 Data work file reduction

Complex models with many effects may yield large iteration **work files**, which increases disk input/output (I/O) work. Iteration **work files** were made smaller by a data reduction technique. Data file reduction in MiX99 is based on the concept of avoiding redundant information. The following strategies were considered useful when complex statistical models were used for a typical dairy cattle data:

- 1) Pedigree information and observation data are stored in separate files.
- 2) If several effects in the model have the same class code, only one equation identification number is stored in the iteration data file. This is possible by properly ordering the equations. For example, all random effects on animal like additive genetic and none-additive environment effect have the same class code.
- 3) All regression covariables or a part of them may be placed in a small table rather than read from the data file. The table is accessed by an index. For example, functions of days in milk can be put to a table and the connected covariate values are found by days in milk index.
- 4) When different traits are measured at different time (e.g. different lactation),

observations of the traits may be grouped by the time component to avoid storing large amount of dummy variables for missing information.

2.4 Equation family blocks

In MiX99, equations are ordered by equation families. An equation family block comprises of closely linked equations in the [MME](#). For example, in dairy cattle fixed and random effects that belong to the same herd (herd test-day effects, cows' non-genetic and genetic effects, etc.) are closely linked to each other and therefore form a block of equations. Equations of fixed and random effects which are present in different herds, e.g. age effect or sire effect, are combined into common blocks. The equation family order increases data locality in the computations, which enhances computing speed, and is essential when using [parallel computing](#) ([Strandén and Lidauer, 1999](#)). For more information see [equation family blocks](#) in the *[Technical reference guide for MiX99 pre-processor](#)*.

3 How to run MiX99 solver programs

3.1 Computing environment

MiX99 is written in standard Fortran 90 and is self-contained. It is developed in UNIX and Linux environment. The program has been tested to compile under many UNIX and Linux Fortran 90/95 compilers as well as Windows compilers. When [parallel computing](#) is used, a Message Passing Interface (MPI) library must be available. For the development of the parallel processing using MPI, Open MPI (<http://www.open-mpi.org>) or MPICH (<http://www.mpich.org>) were used.

3.2 MiX99 solver programs

Executing the MiX99 pre-processor program `mix99i` (see [Technical reference guide for MiX99 pre-processor](#)) will create all necessary files for the solver programs. There is a single process and a parallel program available for solving the [MME](#). Both programs use [iteration on data](#) technique and solve the [MME](#) by the [PCG](#) method:

mix99s Uses one process for solving. During the iteration process it reads the work files `Tmp4.pedi0`, `Tmp5.clas0`, `Tmp6.diab0`, and `Tm10.trco0` every round of iteration. After convergence, final solutions are written to solution files.

mix99p Uses several MPI processes in parallel for solving. Number of [parallel](#) processes is defined in the MiX99 [instruction file](#) for `mix99i`. An additional pre-processing program, named `imake99` needs to be executed before running `mix99p`. The `imake99` program makes a file called `Index.bin`. During the iteration, each process (i) reads its own [work files](#) `Tmp4.pedi(i)`, `Tmp5.clas(i)`, `Tmp6.diab(i)`, and `Tm10.trco(i)` every round of iteration. After convergence, one process writes the final solution to the solution files.

3.3 Multi-threaded MiX99 solvers

In addition to the normal MiX99 solver executables, specially compiled **multi-threaded versions of MiX99 solvers** are also included. Multi-threaded MiX99 executables parallelize some of the calculations, especially large dense matrix computations. These multi-threaded MiX99 executables are located in `mp` subdirectory of the **MiX99 binary distribution**. The name of the multi-threaded MiX99 executable is the same as the single-threaded counterpart, only the location (directory) is different.

Number of computational threads (not to be confused with number of MPI processes) used by the multi-threaded MiX99 executable is controlled by one or more environment variables depending on how the dense matrix libraries are compiled in the executables. To cover most of the cases, the following environment variables need to be set:

```
export MKL_NUM_THREADS=10
export OPENBLAS_NUM_THREADS=10
export GOTO_NUM_THREADS=10
export BLIS_NUM_THREADS=10
export OMP_NUM_THREADS=10
```

Both the “single process” (`mix99s`) and the [parallel](#) solvers (`mix99p`) have the multi-threaded version of the executable included and both multi-threaded solvers parallelize some of the dense matrix operations using the multi-threading. The multi-threaded

“single process” solver spawns given number of computational threads (ten in the example above) when calculating the multi-threaded operations.

The multi-threaded parallel MPI solver (`mix99p`) parallelizes calculations, thus, using both MPI and the multi-threading. Depending on the operation, either one or all of the MPI solver processes utilize multi-threading. This means that the overall number of computational threads can be up to number of MPI processes times the given number of threads, but this depends on the used model.

The multiple threads can be taken to use by the above mentioned environmental variables or by solver command “nt”. Effect of the nt command depends on the statistical model and memory option (see 4.1 or 4.2) used:

- 1) no single-step model: all processes to use the same number of threads given by the command.
- 2) `single-step` model with the low memory option “MES” is used: the same as for the no single-step model.
- 3) single-step with memory options “MEL”/“MEB”/“MEM”: only the master process will use the number of threads given, all other processes use only one CPU.

The command line option is like:

```
mpiexec -np 4 mix99p -nt 10 -MEL -s
```

which uses 4 processes in MPI but 10 CPU threads for the master process. When the solver commands are in a file, the “nt” command is on the first line in format like “nt 10”.

The logic behind the differences is that option “MEL”/“MEB”/“MEM” lead to the Master process to have an additional computational work that will benefit from multi-threading. However, instructing all the other processes to have as many threads may lead to use of too many threads and inefficient computations. Because this may be often the case, the optimum number of threads in non single-step or “MES” option can be small.

3.4 Running the solver

Solving mixed model equations using MiX99 involves execution of at least two programs. First, the pre-processing program `mix99i` is executed. Two alternative approach to give the model information are available: either by providing a CLIM command file on the command line or by giving a MiX99 instruction file to the standard input (see [Command Language Interface for MiX99](#); [Technical reference guide for MiX99 pre-processor](#)). After executing the pre-processor program, the solver program `mix99s` can be executed. The solver reads a `solver option file` from the standard input and writes information about the iteration process to standard output.

```
mix99s < solver_option_file
```

Some of the MiX99 solver options can be specified from the command line also (see [Contents](#)). The easiest way to execute solver is to give option -s which uses default values in solving breeding values, and produces standard output files.

```
mix99s -s
```

Note that when `ssSNPBLUP` model is used, the second level preconditioner is recommended to be used. The second level preconditioner affects the preconditioner for the

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

marker effects such that convergence is often improved. An example:

```
mix99s -s -sp 100
```

where the second level preconditioner will use value 100. The best value depends on the data and the convergence criterion.

When [parallel computing](#) is used, instead of executing `mix99s` you have to execute `imake99` and `mix99p` after the `mix99i` run has finished:

```
imake99  
mpiexec99 -np 4 mix99p < solver_option_file
```

During the execution of the solver programs, `mix99s` and `mix99p`, they can be instructed to [stop the iteration](#), [store intermediate solutions to files](#), and [change some of the iteration parameters](#) by creating external files `STOP`, `PEEK`, and `ITER`, respectively.

After successful completion of the solver, file `OK_mix99s` or `OK_mix99p` will be created. The file will have the completion time. When this file is missing, the solver was terminated due to some error. When using MiX99 through a script, please check existence of the `OK_mix99s` or `OK_mix99p` file.

4 The MiX99 solver option file

The solvers `mix99s` and `mix99p` can be instructed with different parameters to control memory use, the iteration process, the solving of non-linear models (threshold models, multiplicative models for [heterogeneous variance adjustment](#)); to advise `mix99s` to estimate variance components; or to give instruction to the solver programs to calculate [yield deviations](#), [residuals](#), etc. The solver options must be provided by a solver option file, which is read by the programs from standard input. The **MiX99 solver option files** in the provided examples are named with the suffix `.slv`. However, any name can be given. An alternative approach is to give [option on the command line](#). However, command line options are more limited.

NOTE: When BOTH the solver option file AND the [command line options](#) are given, only the command line options are used by default. With [command line option -i](#) this can be changed so that the solver option file is read AND the options from the command line override the corresponding solver option file values.

Example of MiX99 solver option file:

```
# RAM: RAM use and matrix operations: H=high RAM use, nt 10 = use 10 CPU threads
                                     H          nt 10
# STOP:  max.num.iterations, tolerance, convergence criterion, enforce
          5000                1.0e-5      d                f
# RESID: Residuals calculation (Y,N,H)
          N
# VALID: Model validation
          N
# VAROPT: Variance options for VCE, PEV, HV: (N, E, P1, P2, P3, S, C)
          N
# SOLTYP: Solution file options (Y,N,A,H,D)
          Y
```

4.1 Solver option lines

The MiX99 [solver option file](#) consists of option lines asked by the program. There can be several option lines. The order of the option lines must be the same as given in the following. Option lines are not obligatory. However, if one of the option lines is left away all successive lines must be left out as well. The solver programs will use default values in case an empty MiX99 [solver option file](#) is provided or if part of the option lines are not specified. The solver option file can contain comment lines in the same manner as in the MiX99 [instruction file](#). Information specified after the character `#` are considered as comments. Options specified by characters can be given either in upper case or in low case characters.

RAM A line with at least one character,

```
# RAM demand: X=large (mix99p only), H=high, M=medium, L=low
          H
```

which defines the use of random access memory:

- x** **eXtra**. Like H but some extra in parallel computing (mix99p only)
- h** **High**. All required vectors are kept in memory (fastest execution time). This option is recommended.
- m** **Medium**. Solutions are stored on the disk.
- l** **Low**. Solutions and [residuals](#) are stored on the disk.

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

The option **m** and **l** will reduce memory requirements by 25% and 50%, respectively, with the penalty to increase I/O-operations. However, in case of memory limitations, the option **m** and **l** may yield shortest execution time.

There is an extra memory option, named **x**, for [parallel](#) solver ([mix99p](#)). The option can also be enabled by using parallel solver [command line option](#) `-x`. This memory option uses even more memory than option **h** but can speed up the computations significantly in some cases. The extra memory is used to access the linked list (see `imake99` output) faster. Each process will allocate a vector of size number of unknowns. Thus, when there are 100 million unknowns and 10 processors, this will lead to an extra memory allocation of $10 \times 100 \times 10^6 \times 4$ bytes, i.e., less than 4 giga bytes.

Additional options than can be given after the definition of RAM use:

NO/YES checking of release information (NO/YES), YES is default.

nt n Number CPU in multi-threading is set to be n.

srm m Sparse regress matrix read to memory for matrix number m.

sp v Second level preconditioner of value v for regress matrix.

noc No residual covariances used. Assumes none were given (`mix99s` only).

noQ No genetic groups for A_{gg}^{-1} term, e.g., no $Q'A_{gg}^{-1}Q$, in [single-step](#) models having A_{gg}^{-1} through the PEDIGREE option.

IOP/IM/CHM/PAR in [single-step](#), product of A_{gg}^{-1} times vector can be performed using one of four alternative approaches. Approach IOP uses iteration on pedigree, IM uses iteration in memory, but CHM uses CHOLMOD and PAR uses MKL PARDISO library. Use of memory from lowest to most: IOP, IM, CHM/PAR, where memory need by IOP and IM is quite close but CHM/PAR use much more RAM. Computing time from highest to lowest: IOP, IM, CHM/PAR, where there is substantial difference between all approaches. The CHM and PAR options can use [multi-threading](#).

MEL in [single-step](#), reads G^{-1} or **T** matrix from file to memory. The [MEL](#) option uses efficient matrix multiplication during PCG iteration. The multiplication can use multi-threading by the [multi-threaded versions of MiX99 solvers](#). Note that when the number of genotyped individuals is large the matrix to read is large and consumes a lot of RAM memory.

MEM is like option [MEL](#) but slower and uses slightly less memory, when **T** matrix is used. Same as MEL when G^{-1} .

MES does not read G^{-1} or **T** to memory, memory efficient but slow.

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

MEB b is like option **MEL** but makes the computations in blocks of size **b** in **SSGTBLUP**. Can be faster than MEL for very large matrices.

MEA is like option **MEB** but the block size **b** is computed automatically that leads to a block of 2 GB.

RDS instructs to use a small block size when calculating contributions from regression design matrices. Currently implemented in **mix99s** only.

RDM similar to **RDS** but uses medium block size and keeps SNP marker matrices byte-packed in memory. Currently **mix99s** only.

RDL similar to **RDM** but uses large block size and keeps SNP marker matrices byte-packed in memory. Currently **mix99s** only.

RDX instructs to keep regression design matrices fully in (double precision real) memory. Currently **mix99s** only.

RDB b similar to **RDL** but uses given block size $\text{abs}(b)$ (absolute value of **b**). If **b** is positive, keeps SNP marker matrices byte-packed in memory. Currently **mix99s** only.

RDU m similar to **RDB** but instructs to use given amount of memory (m) when calculating contributions of regression design matrices. Full regression design matrices are kept in memory if possible or at least byte-packed SNP matrices if possible. Block size is also calculated from the given memory limit. Memory size **m** is given as **<amount><unit>** where **<amount>** is an integer and optional **<unit>** is one of K (kilo), M (mega), G (giga, default), T (tera), and P (peta-bytes). Example: **RDU 12G**. Currently **mix99s** only.

ω value for the ω multiplier of matrix A_{gg}^{-1} . In practice, value of ω is typically about 0.6 – 0.8.

Defaults: **h** for high memory, **YES** for check release information, **PAR** for MKL PARDISO library, **MEL** for having G^{-1} or **T** in memory, block size 1 for regression design matrices (except if all are SNP matrices: 1/5/10 for Low/Medium/High memory options), and 1 for value of ω .

Option **MEA** is default for the component-wise **SSGTBLUP** and **ssSNPBLUP** that use 1 byte marker matrix.

The following will change to use **CHM**, **MEB**, and multiplier 0.8:

```
H CHM MEB 1000 0.8
```

In practice, non-single step models ignore the memory option for G^{-1} and the computations for product of A_{gg}^{-1} because these matrices do not exist in these models. However, when these matrices exist as in **single-step**, most memory (option 'MEL') and fastest multiplication approaches (option 'CHM' or 'PAR') gives shortest computing times when there is enough memory. Computations can be faster by using that both of these options can use in multi-threaded version of MiX99. Optimal number of computing threads depends on the computing environment and amount of RAM memory.

STOP

One line with three (**mix99p**) or four (**mix99s**) entries:

```
# maxiter, tolerance, criterion (A/R/M/D), [enforce (F)]:
5000      1.0e-5      D      F
```

The first entry is an integer number that specifies the **maximum number of iterations**. Note that value larger than the number of equations (i.e. the number of the effects) in the model is replaced automatically by this largest allowed number of iterations. The second entry is a real value that specifies the stopping value (i.e., tolerance) for the the **convergence criterion**. The third entry is a character that specifies the **convergence criterion** to which the stopping value will be applied. For example, if a character **a** is specified, then the iteration process will continue until the **convergence criterion CA** will reach a value smaller than the stopping value (i.e, tolerance) that is specified on entry two. The solver programs provide five types of **convergence criteria** (see point 4.6) of which four can be chosen:

- a** **Ca**. Relative difference (residual) between right-hand and left-hand side of the **MME** considering all equations of the additive genetic effects only. If Ca is defined, a suitable stopping value could be 1.0e-4 or 1.0e-5. However, the value is model-dependent and should be at least as small as needed to ensure that the **Cd convergence criterion** reaches a value of 1.0.e-5.
- r** **Cr**. Relative difference between right-hand and left-hand side of the **MME** considering all equations.
- m** **Cm**. Relative difference between **preconditioned** right-hand and left-hand sides of the **MME** considering all equations.
- d** **Cd**. Relative difference between solutions of the last two iteration rounds. If Cd is defined, a suitable stopping value for the majority of analyses would be between 1.0e-4 and 1.0e-5. Note that if Cd is chosen as the **convergence criterion**, the Cd criterion must be met by two consecutive iterations.

The fourth entry is optional and is needed for the **mix99s** solver only. The **mix99s** solver will consider the **STOP** option line only in case an enforcing character **f** is specified for the fourth entry. Otherwise, default values will be used.

Default values for mix99s: 5000 1.0e-5 d

In case of solving a **threshold-model**, a non-linear model will be solved and the iteration process works on two different levels. Therefore, for a threshold-model the **STOP** line must have five entries: an integer, a real value, a character, an enforcing character **f**, and one additional integer. Now the first integer value gives the maximum number of **PCG**-iterations within each NR- or EM-round (default is 100 or number of equations in the **MME**) and the last integer value gives the **maximum number of NR- or EM-rounds** (default is 5000).

RESID

A line with one character specifying the calculation of **residuals**.

```
# RESID: Calculate residuals? (Y/N)
N
```

- y** **Yes.** **Residuals** will be written into the file(s) `eHat.data(i)`. When using `mix99s`, (i) will be zero (0). In case of **parallel processing**, each process writes an own `eHat.data(i)` file with the process number (i) at the end of the filename. The order of the **residuals** corresponds with the order of observations in the input data file. In case of **parallel processing**, the order of the residual files correspond to the order of observations in the input data file beginning with file zero (0) up to number of processes minus one. The `eHat.data(i)` files have as many columns with **residuals** as the maximum number of traits in the largest trait group. This is equal to the `mxntra` parameter given in the `Parlog` file. The `Parlog` file is produced by `mix99i`. The residual columns are ordered in the same sequence as the traits in the trait groups. For missing observations, the corresponding value in the residual files are set to the **missing value -8192.0**.
- n** **No.** No **residuals** are written.
- h** This option is only available in `mix99p` and will create the file(s) `ARsiwi.data(i)`, which contain information about the heterogeneity of variance in the residuals. These files are needed only when accounting for **heterogeneous variance** (see chapter [Contents](#)).

Default value: n

VALID

A line with one entry,

```
# VALID: Model validation (N,P,S,Y,D,I,G,R)
N
```

which instructs the solver to calculate for each observation a corresponding, here specified, sub-quantity of the applied model line, or to instruct the solver to simulate observations based on the specified model. The calculated quantities are written to binary files after the iteration process has ended. Text file is written when the option letter is appended with letter t, e.g. pt. DYDs are always written to a text file. **Missing values** will be specified with **-8192.0**. The structure of the files will be explained in the chapter [5](#):

- n** **None.** None of the options are requested.
- p** **Predictions.** For each observation, the predicted value ($\hat{y}$) is written to the file(s) `yHat.data(i)`.
- s** **Selected Model Factors' Sum.** For each observation the sum of selected model factors is written to the file(s) `sHat.data(i)`. The selected factors must be specified on a following line.
- y** **Yield Deviations (YD).** For each observation the corresponding YD will be written to the file(s) named `YD.data(i)`. The factors included into the YD must be specified on a following line.
- i** **Individual Daughter Deviations (IDD).** For each observation the corresponding IDD will be written into the file(s) named `IDD.data(i)`. The factors included into the IDD must be specified on a following line.

- d Daughter Yield Deviations (DYD).** The solver will calculate for each observation the corresponding IDD and will use it for the calculation of DYDs based on the approach of [Mrode and Swanson \(2004\)](#). For this option the calculated DYD will be written to a formatted file named `Soldyd`. (see chapter [Contents](#)). The factors included into the DYD must be specified on a following line.
- g Generate Observations.** This option is available in `mix99s` only. The `mix99s` solver will not solve the model, but instead will generate for each observation in the data a simulated observation ($\tilde{y}$). Therefore, for all effects in the model true solutions will be simulated based on the provided variance components. Fixed effect solutions will be set to zero. The true solutions are written to the **MiX99** standard solution files. The generated observations will be written into the file named `ySim.data0`. This file can be used in a future **MiX99** run to replace real observation by simulated observations. See the **VAR** instruction line in the [Technical reference guide for MiX99 pre-processor](#) for reading and using of the generated observations instead of the real observations.
When specifying **g** a **SEED** option line must be included after the **VAROPT** option line. The **SEED** option line has one entry, which must be one of those given in the **SEED** option description below (see **VAROPT**).
- r Deregression.** (See chapter: “[Deregression](#)” in [Command Language Interface for MiX99](#)).

The options **y**, **i**, **d** and **g** are not supported when solving non-linear models.

The options **s**, **y**, **i**, or **d** will require adding of a second line, which specifies which factors of the model are included into the calculation of the specified quantity.

FACTOR One line with as many integers as there are factor columns defined in the **REGRESS** instruction line. This is equal to the first integer value of the **REGRESS** instruction line (see [Technical reference guide for MiX99 pre-processor](#)). The order of the integer values on the **FACTOR** line corresponds to the order of the factors specified on the **REGRESS** instruction line. Each integer specifies whether or not the corresponding factor of the model is included into the calculation of the desired quantity.

- 1** The factor will be included into the specified quantity
- 0** The factor will be excluded from the specified quantity

Specification of the factors for the desired quantities will be as following:

Let's assume a model, for which the solver will have the following model terms available after the model has been solving:

$$y = X\hat{b} + Z\hat{p} + Z\hat{a} + \hat{e},$$

where y contains the observations, $\hat{b}$ the estimates for the fixed effect factors, $\hat{p}$ the estimates for the non-genetic animal effect factors, $\hat{a}$ the estimates for the additive genetic animal factors and $\hat{e}$ the residuals.

Selected Model Factors' Sum. Any factor included in $\hat{b}$, $\hat{p}$, or $\hat{a}$ can be included into the sum. All factors included into the sum have to be specified with ones (1); all factors excluded have to be set to zero (0).

Yield Deviations ($YD = y - X\hat{b} - Z\hat{p}$). All factors associated with $\hat{b}$ and $\hat{p}$ have to be set to zero (0); all factors associated with $\hat{a}$ have to be specified with ones (1).

Individual Daughter Deviations (IDD) and **Daughter Yield Deviations (DYD)**. The **IDD** is a quantity which is need also for the calculation of the **DYD**. Thus, for both options the same quantity is needed ($IDD = y - X\hat{b} - Z\hat{p} - 1/2\hat{a}_{dam}$). All factors associated with $\hat{b}$ and $\hat{p}$ have to be set to zero (0); all factors associated with $\hat{a}$ have to be specified with ones (1).

VAROPT A line with one entry that specifies different options related to the adjustment for heterogeneous variance or to the [estimation of variance components](#).

```
# VARPOT: Variance options for VCE, PEV, HV (N,E,P1,P2,P3,S,C)
N
```

n **None.** None of the options are requested.

e <f n> Estimation of Variance Components. The option **e** will instruct **mix99s** to estimate variance components by a stochastic Monte Carlo Expectation Maximization REML (**MC EM REML**) algorithm (for more information please see chapter 9). A special case is option **ei** for estimation of variance components of a MACE model (see [Contents](#)). An additional (optional) instruction can be given after the **e** (or **ei**) character. This instruction has two entries; the character **f** and an integer number n . The optional instructions are needed in case certain variance component parameters are meant to be [fixed](#). This option will be explained in chapter 9.4.3.

When an **e** (or **ei**) is defined on the **VAROPT** option line, three additional instruction lines have to be given.

STOPE The first line contains four entries of which the last one is optional; two integers followed by one real value number and one optional real value number. The first integer number specifies the maximum number of **MC EM REML** rounds. If the given number of REML rounds has a negative sign (e.g. -60), previously run **REML estimation is to be continued** using this new total number of rounds. The second integer number specifies the [number of data samples](#) generated and analyzed within a **REML** round. The real valued number is the stopping criterion

for the **REML** analysis. After the **convergence indicator** reaches a value smaller than the specified convergence criterion, the **REML** analysis will perform a sequence of final 30 **MC EM REML** rounds, which will reduce the Monte Carlo error from the parameter estimates. The second, optional real value is the stopping value for solving the BLUP models of the data samples. This stopping value is applied for the same convergence criterion is as defined on the **STOP** option line. This optional value allows to apply different BLUP stopping values for solving BLUPs of the observed data and the sampled data. A less strict stopping criterion for the sampled data may reduce computing time. However, the stopping criterion value needs to be specified with care, to ensure correct estimation of variance components.

Default values: 1000 5 1.0e-9 <same as specified by STOP>

SEED The second line contains one entry, which defines the type of the seed used by the random number generator for generating the data samples.

- d** Default initialization by call to random_seed.
- r** The random number generator is initialized base on the system clock.
- g** The user can specify the seeds for the random number generator. If option **g** is specified *j* integers must be provided in the next line.

Default value: d

MIX99PATH The third line contains the path for the directory where the **mix99i** pre-processor executable is located. In certain intervals the **mix99s** solver will make a system call to **mix99i** to update the preconditioner matrices as explained in chapter 9. This will also cause an update of the **MiX99.lst** file. Variance components listed are no longer the starting values used, but the intermediate estimates that were applied for the most recent **preconditioner matrix** update. If the given directory name is empty (either a pair of quotation marks ("") or minus sign (-)) the pre-processor is assumed to be located in a directory that is included in the search PATH.

- s** **Start-up cycle for heterogeneous variance adjustment.** After **mix99p** has performed a maximum number of 20 iterations (specified on the **STOP** option line) it will write heterogeneity of variance estimates to files named **SiWi.data(i)**. These files will be used by **mix99hv** to create the input data files for the applied **variance model** that describes the heterogeneity of variance in the data.
- c** **Cycle between models for solving the multiplicative mixed model.** The option is needed for the **heterogeneous variance adjustment** and

will instruct `mix99p` to discontinue in certain intervals the iteration process and make system calls for solving the `variance model` by a second, simultaneous **MiX99** analysis. The process will continue until both models have converged.

ADJUST In case **s** or **c** is defined on the `VAROPT` option line, an additional line needs to be specified with as many integers as there are factor columns defined in the `REGRESS` instruction line. This is equal to the first integer value of the `REGRESS` instruction line (see *Technical reference guide for MiX99 pre-processor*). The order of the integer values on the `ADJUST` line corresponds to the order of the factors specified on the `REGRESS` instruction line. Each integer specifies whether or not the corresponding factor of the model is included into the `adjustment of heterogeneous variance`.

- 1** The factor will be included into the HV adjustment.
- 0** The factor will be excluded from the HV adjustment.

Including all factors corresponds to the method by [Meuwissen et al. \(1996\)](#). When excluding some factors from the HV adjustment a restricted `multiplicative mixed model` will be applied. Excluding the fixed effect factors from the example model given in `VALID` will perform a restricted `multiplicative mixed model adjustment for heterogeneous variance` of the form:

$$y_i = X_i \hat{b} + (Z_i \hat{p} + Z_i \hat{a} + \hat{e}_i) \gamma_i,$$

where $\gamma_i = \frac{1}{\lambda_i}$ and λ_i is the `heterogeneous variance adjustment factor` for stratum i .

STOPC In case **c** is defined on the `VAROPT` option line, a second additional line with two entries must be given. The first entry is an integer value giving the maximum number of `heterogeneous variance adjustment` cycles; i.e. the maximal number that the `variance model` will be updated and solved. The second entry is a real value and is the required stopping criterion. The updating and solving of the `variance model` will stop when the `convergence indicator` for the heterogeneity adjustment factors has reached a value smaller than the specified stopping criterion. The `convergence indicator` for the heterogeneity adjustment factors is calculated as:

$$cd_{(k)} = \frac{(\mathbf{l}^{(k)} - \mathbf{l}^{(k-1)})^T (\mathbf{l}^{(k)} - \mathbf{l}^{(k-1)})}{(\mathbf{l}^{(k)})^T (\mathbf{l}^{(k)})},$$

where $cd_{(k)}$ is the value of the convergence indicator in adjustment cycle k and $\mathbf{l}$ is the vector of multiplicative adjustment factors (lambda values).

Default values: 1000 1.0e-7

SOLTYP A line with one character,

```
# SOLTYP: Solution file options (Y,N,A,H)
Y
```

which specifies the way solutions are handled. This option became necessary when modules for solving different non-linear models were implemented in the **MiX99** package. Solving a standard linear model expects **y**. All other options (N,A,H) are related to [adjustment of heterogeneous variance](#) or to [non-linear Gompertz models](#).

- y** **Yes, give standard solution files.** Solution files are written in text format.
- n** **No.** No solutions are written. This is useful when specifying option **s** on the [VAROPT](#) option line.
- d** **DMUINPformat.** Option will produce binary and ASCII files with the pseudo data. See [8.2.2](#). This option is needed only for a simultaneous estimation of variance components for the non-linear Gompertz function model by, for example, using the DMU package for the variance component estimation.

One of the two options (**a**, **h**) must be defined when using **MiX99** for solving the [variance model](#) for [adjustment of heterogeneous variance](#). The option will instruct **mix99p** to accelerate the solutions of the [variance model](#) between consecutive [heterogeneous variance adjustment](#) cycles. The solutions are written to binary files (**SolfixB** for strata of the first effect and **SolaniB** for strata of the second effect in the [variance model](#)).

- a** Accelerated solutions using Aitken acceleration. This option is suitable, if convergence is dominated by a single large eigenvalue. For many models it yields fast convergence (between 40 to 60 cycles).
- h** Accelerated solutions using a Half-Chebyshev golden ratio procedure ([Hesterberg, 2005](#)). This step-lengthening method follows a golden ratio procedure. For large and complex models this was found reliable, but it requires between 80 to 100 cycles. The option is robust and therefore recommend for routine evaluations.

4.2 Command line options

Some of the MiX99 solver options can be alternatively specified from the command line. List of available command line options of the MiX99 programs, such as the (non-parallel) solver `mix99s`, can be obtained with

```
mix99s -h
```

This will print the following instructions for `mix99s`:

Usage:

```
mix99s [-s] [-i] [-nt nt] [-p|-pt|-pb] [-r|-rt|-rb] [-m]
        [-n NITER] [-ncalls NCITER] [-n0 N0ITER] [-c{a,d,r,m} TOL]
        [-peek [-]PITER] [-peek_step PSTEP]
        [-IOP|-IM|-CHM|-PAR] [-o VAL] [-t tau] [-MES|-MEM|-MEB|-MEA|-MEL]
        [-RDS|-RDM|-RDL|-RDX|-RDB <blocksize>|-RDU <mem>]
        [-noQ] [-fQP] [-sp val] [-srm n] [-nocov] [-RHS rhsfile]
        [-MS] [-PA]
        [-h|--help] [-v|--verbose] [-V|--version]
        [--bindir BINDIR] [--datadir DATADIR] [--tmpdir TMPDIR]
```

where

- s: use defaults, solve and produce standard solution files.
- i: use both input file and command line options.
- nt nt: number of threads set to nt.
- p or -pt: use defaults, solve and produce predictions.
- pb: same as -p but produce binary file(s).
- r or -rt: use defaults, solve and produce residuals.
- rb: same as -r but produce binary file(s).
- m: Generate MME.dat and rhs.dat of the problem.
- n NITER: number of iterations.
- ncalls NCITER: number of PCG solver calls (e.g. deregression).
- n0 N0ITER: minimum number of iterations.
Default: 20 if Cd criterion and old solutions, 1 otherwise.
- peek [-]PITER: Write intermediate solutions at iteration PITER.
If negative, file extension `_PEEK` instead of `_<ITER>`.
- peek_step PSTEP: Write solutions every PSTEP iterations.
If PITER negative, file extension `_<ITER>` instead of `_PEEK`.
- peekcr CRTOL Write intermediate solutions at `Cr==CRTOL`
- ca TOL: Ca convergence stopping criterion.
- cd TOL: Cd convergence stopping criterion.
- cr TOL: Cr convergence stopping criterion.
- cm TOL: Cm convergence stopping criterion.
- IOP: use iteration on pedigree in `inv(A11)` of single-step.
- IM: use iteration on matrix in `inv(A11)` of single-step.
- CHM: use Cholmod in `inv(A11)` of single-step.
- PAR: use PARDISO in `inv(A11)` of single-step (default).
- o: coefficient VAL multiplies `inv(A22)` in `ssGBLUP`.
- t: coefficient TAU multiplies `inv(G)/inv(H)` in `ssGBLUP`.
- MES: no `inv(G)/T` in memory.
- MEM: read `inv(G)/T` matrix to memory.
- MEB <bSize>: read `inv(G)/T` matrix to memory and compute by <bSize>.
- MEA: read `inv(G)/T` matrix to memory and compute by automatic block size.
- MEL: read `inv(G)/T` matrix to memory and use efficient multiplication (default).
- RDS: small block size with regression design matrices (RDM). Packed SNP matrix in memory.
- RDM: medium block size with RDM. Packed SNP matrix in memory.
- RDL: large block size with RDM. Packed SNP matrix in memory.
- RDX: full (double precision) RDM kept in memory.
- RDB <blocksize>: given blocksize for RDM.
Packed SNP matrix kept in memory (if blocksize > 0).
- RDU <memory>: given memory usage for RDM. Block size and
packed marker matrix usage depending on <memory>,
where <memory> is <number>[K/M/G/T/P], e.g. 12G.
- MS: Mendelian sampling terms written to SolMS.
- PA: Parent averages written to SolPA.
- noQ: no groups for `inv(A22)` if none in `inv(G)`.
- fQP: include groups for `inv(G)` in componentwise `ssGTABLUP`.
- sp val: second level preconditioner for RegMatrix.

```
-srm n : RegMatrix number n is sparse and read to RAM.
-continue_reml: Continue previous REML iteration.
-PEV: Monte Carlo PEV.
-nocov: No residual covariances assumed between traits.
-RHS rhsfile: name of the file for the external right-hand side.
-nocheck: no checking of data and release time information.
-silent: PCG iteration screen output small.
-h or --help : Show usage.
-v or --verbose: Show additional information.
-V or --version: Show version information.
--bindir BINDIR:
    Directory for MiX99 binaries. Default: (empty)
--datadir DATADIR:
    Data directory. Default: (empty)
--tmpdir TMPDIR:
    Directory for temporary files. Default: (empty)
Corresponding environment variables:
    MIX99_BINDIR, MIX99_DATADIR, MIX99_TMPDIR
Note: Environment variables are used first, then command line options.

Note: If solver command line options given input file is not
      used unless option -i is given in which case command line
      options override the input file values.

Example: -n 100 -cd 1e-5
      will set number iterations to 100, and will
      use Cd stopping criterion with 1e-5 as stopping value.
```

Some of the command line options of **mix99s** may not be available in **mix99p**.

Note that if any solver command line options are given, the **solver option file** (or the standard input) is not read by default and the default values are used for options not specified on the command line.

By specifying command line option **-i** the solver option file is read first AND **options from the command line override the corresponding solver option file values:**

```
mix99s -s -n 200 -cr 1e-5 -i < solver_option_file.slv
```

4.3 Sparse regression matrix (srm)

Solver option srm (sparse regression matrix) allows reading the REGFILE matrix to memory. It is stored as a sparse matrix. Computations use the sparsity of the REGFILE which can lead to reduction in computing time. This option can be useful when the REGFILE matrix has genetic group coefficients, i.e., the file has the so called Q matrix for the phenotype records. The Q matrix can be very sparse.

4.4 No residual covariances (noc)

Solver option noc assumes that there are no residual covariances, i.e., these are zero. Because no residual covariance computations are computed, this can be faster when there are many traits. This option can be useful in **single-step** when many separate traits are analyzed at the same time. Separate analysis of many single-step models can lead to large temporary files to be generated and used at the same time. Simultaneous solving of several such models can be infeasible due to the large RAM memory need. Use of a multiple trait model with zero correlations without noc option will lead to making the zero covariance computations. The noc option leads to not making these computations.

4.5 Second level preconditioner (sp)

Second level preconditioner for regress matrix can be given. This can be useful when the regress matrix describes marker effects and the model has a polygenic effect. The second level preconditioner is applied to all regress matrix effects simultaneously. It is also applied to all marker effects in the `ssSNPBLUP` model. The value given for the second level preconditioner is used to multiply all the regress matrix and marker effect values after the first level (regular) preconditioner has been used.

4.6 Determining convergence

The solver programs `mix99s` and `mix99p` provide five different **convergence indicators**. The convergence indicators are calculated after each round of iteration and are written to the standard output:

Iteration Statistics					
Convergence Indicators					
ROUND	CA	CR	CM	CD	MAX.CHA.
Solution vector will be initialized to be zero					
rhs' * rhs =	734663911244.259				
animal rhs' * rhs =	160746417.003415				
0	1.000	1.000	1.000	0.000	0.000
1	0.1138	0.2698E-01	0.8155E-01	1.000	7.166
2	0.4657E-01	0.1977E-01	0.3428E-01	0.2083	-2.915
3	0.2944E-01	0.1569E-01	0.2513E-01	0.1235	1.821
4	0.2365E-01	0.1201E-01	0.2276E-01	0.1552	2.307
5	0.2375E-01	0.5799E-02	0.2014E-01	0.1615	2.128
6	0.1754E-01	0.4293E-02	0.1565E-01	0.2247	3.150
7	0.1336E-01	0.2695E-02	0.1228E-01	0.1155	-1.904
8	0.9771E-02	0.3307E-02	0.8834E-02	0.8621E-01	1.335
9	0.7163E-02	0.2451E-02	0.6747E-02	0.6375E-01	-1.016
10	0.5710E-02	0.2095E-02	0.5442E-02	0.5017E-01	-1.219

The first four **convergence indicators** are norms that can be selected as the **stopping criterion** of the iteration. For describing these norms, we define that C represents the coefficient matrix of **MME**, $\hat{s}^{(k)}$ the vector of solutions at round k , r the right hand side of the **MME**, and M^{-1} the inverse of the **preconditioner** matrix, which approximates the inverse of C . The four norms are:

$$\begin{aligned}
 ca_{(k)} &= \sqrt{\frac{(\mathbf{r} - C\hat{\mathbf{a}}^{(k)})^T (\mathbf{r} - C\hat{\mathbf{a}}^{(k)})}{(\mathbf{r}_a)^T (\mathbf{r}_a)}}, \\
 cr_{(k)} &= \sqrt{\frac{(\mathbf{r} - C\hat{\mathbf{s}}^{(k)})^T (\mathbf{r} - C\hat{\mathbf{s}}^{(k)})}{(\mathbf{r})^T (\mathbf{r})}}, \\
 cm_{(k)} &= \sqrt{\frac{(\mathbf{r} - C\hat{\mathbf{s}}^{(k)})^T M^{-1} (\mathbf{r} - C\hat{\mathbf{s}}^{(k)})}{(\mathbf{r})^T M^{-1} (\mathbf{r})}}, \\
 cd_{(k)} &= \sqrt{\frac{(\hat{\mathbf{s}}^{(k)} - \hat{\mathbf{s}}^{(k-1)})^T (\hat{\mathbf{s}}^{(k)} - \hat{\mathbf{s}}^{(k-1)})}{(\hat{\mathbf{s}}^{(k)})^T (\hat{\mathbf{s}}^{(k)})}},
 \end{aligned}$$

where $ca_{(k)}$ is the relative difference between left-hand side and right-hand side (residual) of the part of the **MME** which includes the equations of the additive genetic effects; $cr_{(k)}$ is the relative residual of all effects of the **MME**; $cm_{(k)}$ is the **preconditioned** relative residual of the **MME**; and $cd_{(k)}$ is the relative difference between solutions from consecutive iterations.

The norms $ca_{(k)}$ and $cr_{(k)}$ are the most reliable **convergence indicators**. When solving a model by PCG, both of these norms get smaller each iteration which indicates that estimates are converging towards the true solutions of the **MME**. According to the theory of the conjugate gradient methods, every conjugate gradient step/iteration will move the current values to the estimates closer to the true solutions of the **MME** and, thus, both norms will be smaller. However, for some models, especially $cr_{(k)}$ can fluctuate erratically. This is because **preconditioning** M^{-1} is not included into the calculation of the two norms. Moreover, value of the norm varies between models. Because of the latter two characteristics, it is necessary to find for each model the appropriate stopping criterion in case the stopping criterion uses $ca_{(k)}$ or $cr_{(k)}$.

The norm $cm_{(k)}$ is **preconditioned** form of the norm $cr_{(k)}$. It is closer to what the **PCG** iteration is minimizing and therefore, at least in theory, it can be less erratic. The performance, however, depends on the properties of the **preconditioning**. If there is no **preconditioning** (see **PRECON** instruction), $cm_{(k)}$ is equal to $cr_{(k)}$.

The norm $cd_{(k)}$ is widely used because it is easy to calculate and shows very smooth convergence. This norm is almost independent from the applied model and it has been found that a value smaller than 10^{-5} indicates sufficient convergence for most of the models solved by **MiX99**. However, a disadvantage of the norm is that a small value of this norm is no guaranty for a real convergence. This is commonly so for the single-step models. We recommend using $cr_{(k)}$ less than 10^{-7} for single-step models.

The fifth **convergence indicator** gives the **largest change** of an estimate between the last two iterations out of all estimates. A large value at the end of the iteration process usually indicates that some fixed effect classes have very few observations and no unique estimate is found for some effect levels. For many models it has been found that this indicator should converge to a value smaller than 10^{-1} .

In case of solving the **multiplicative mixed model** for the **heterogeneous variance adjustment** (option **c** at the **VAROPT** line) only one **convergence indicator** is provided during the HV cycling process. For computational ease this criterion is $cm_{(k)}$.

4.6.1 Choosing a suitable convergence criterion

Each of the three available norms for indicating progress of convergence has its own characteristics as described above. Based on the experiences which we and the **MiX99** users gained by solving very different models of very different size, we recommend the following alternatives ways to secure sufficiently accurate converged solutions:

Alternative 1: Apply a convergence criterion of 10^{-4} (or between 10^{-4} and 10^{-5}) to the **convergence indicator CD** and check that the **convergence indicators CA** and **CR** show progress of convergence during the whole iteration process.

Alternative 2: Apply the convergence criterion to the **convergence indicator CA** and define a convergence criterion, which will ensure that the convergence criterion **CD** will have reached a value smaller than 10^{-4} at the end of the iteration process.

For some **large routine evaluations** solving time might be critical, and the solver should carry out only the least number of required iteration rounds to achieve sufficient convergence of the solutions. For such evaluations, the most suitable convergence criterion is found by comparing solutions of several test runs, where different strict convergence criteria were applied, with “quasi-true” solutions from a test run with a very strict convergence criterion (e.g. $cd_{(k)}$ norm between 10^{-5} and 10^{-6}). For many models it was found that a correlation of ≥ 0.995 of the genetic animal effect solutions with the “quasi-true” genetic animal effect solutions indicate that sufficient convergence has been achieved (Lidauer and Strandén, 1999).

4.6.2 Effect of preconditioning on convergence

The choice of **preconditioner** matrices can have significant effect on speed of convergence when solving complex models. Generally, the better the inverse of the **preconditioner matrix** approximates the inverse of the coefficient matrix of the **MME**, the faster convergence of the solutions is achieved. However, specifying large **preconditioner matrices** may cause a considerable increase in computations at the cost of total solving time.

The following example demonstrates that the specified **preconditioning** can significantly affect the solving time. For specifying the **preconditioner matrices**, please see **PRECON** instruction line in the *Technical reference guide for MiX99 pre-processor*.

The **example** data included 374007 test-day records from 19709 cows of 130 herds. The data was modeled with a multiple trait random regression model including nine traits. The model including four fixed effects, a random herd-test-day effect and functions for the within-herd lactation curve, the non-genetic animal effect, and the additive genetic animal effect. To make the solving of the model as demanding as possible, no rank reduction was applied. This yielded (co)variance matrices of size 9, 27, 36, 36, and 9 (for the residual), respectively. The **MME** included over 2 million equations.

Here, the effect of three different **preconditioning** alternatives will be demonstrated: Alt.1) a diagonal preconditioner for all effects; this is equal to the diagonal of C . Alt.2) a block diagonal preconditioner for all effects; the block size varied between 9 and 36 depending on the effects. Alt.3) a block diagonal preconditioner for all random effects and one full block preconditioner matrix including all fixed effect equations, which was of size 7401×7401 .

Solving was continued until the **convergence indicator** $cd_{(k)}$ was smaller than 3.16×10^{-5} .

Preconditioner alternative	Number of Iterations	Solving Time (min)	Size of Preconditioner (Mb)
Alt 1) Diagonals	3725	56.3	8
Alt 2) Block diagonal	584	13.6	140
Alt 3) Block diagonal + full block	598	24.0	25

Applying a **block diagonal preconditioner** matrix for all effects yielded the shortest solving time, whereas apply a **diagonal preconditioner** matrix for all effects yielded the smallest **preconditioner matrix**. Experiences showed that a **block diagonal preconditioner** is a good choice for many different models. For very large models the size of the

[preconditioner matrices](#) might be critical. Then, a diagonal preconditioner needs to be applied for some of the effects in the model.

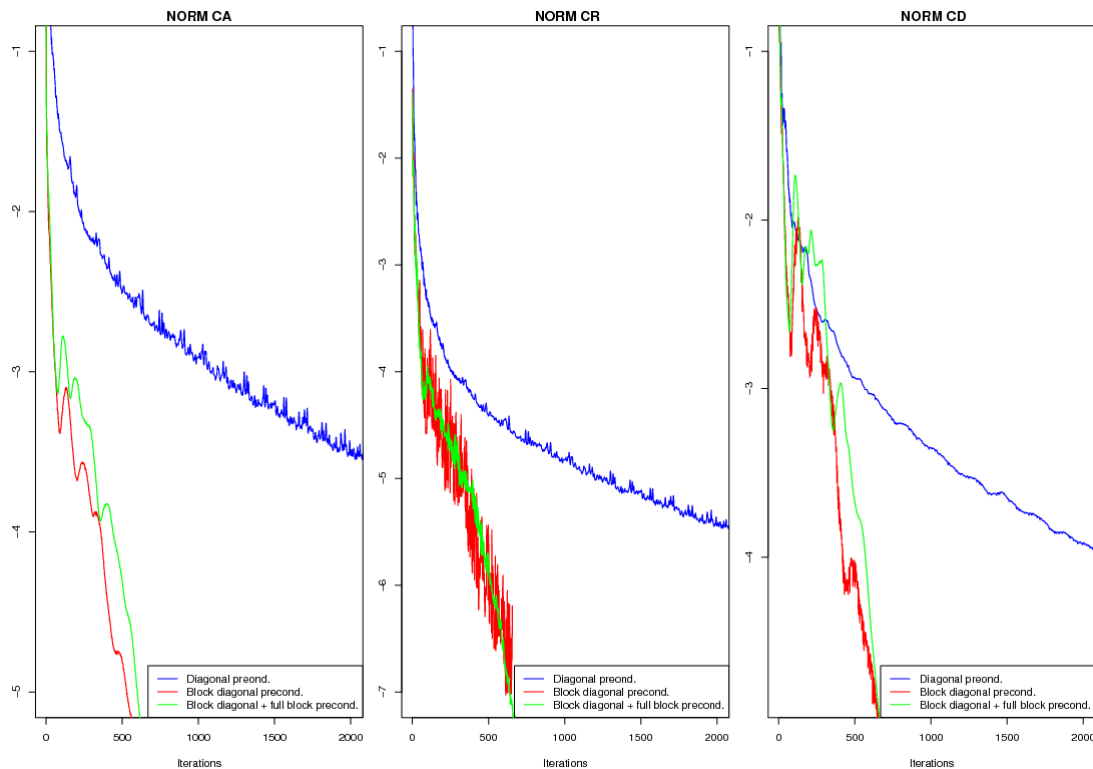


Figure 1: Logarithm of the [convergence indicator](#) norms [CA](#), [CR](#), and [CD](#) by round of iteration, given for different [preconditioning](#) alternatives when solving a complex model.

4.7 External STOP file: stopping iteration

In some situations, it might be useful that the iteration process is stopped in a controlled fashion before one of the specified stopping criterion has been fulfilled. The solver programs can be instructed to stop after the current round of iteration by creating a file named [STOP](#) in the directory where the solver is executed. Then, the solver will write the most recent solutions to the standard solution files. When [parallel computing](#) is applied, the [STOP](#) file has to be accessible by the master process. In case [heterogeneous variance](#) is accounted, a [STOP](#) file can be used to stop the cycling between [mean model](#) and [variance model](#). Then, after the current adjustment cycle is finished, the program will continue with iterations on the [mean model](#) until a stopping criterion is reached or the [STOP](#) file is provided a second time. The solvers will erase the [STOP](#) file from the directory to avoid trouble in future analysis.

4.8 External PEEK file: intermediate solutions during iteration

The MiX99 solver programs can also be instructed to store the intermediate solutions during the iteration by creating a file named [PEEK](#) in the directory where the solver is executed. The existence of the [PEEK](#) file is recognized by the solver at run time, the content of the file is read to memory, and the [PEEK](#) file is then removed.

The [PEEK](#) file may be either empty or contain one or two integers:

```
[ - ] PITER PSTEP
```

If the `PEEK` file is empty, solutions of the current iteration are stored to solution files named with a `_<ITER>` suffix, where `<ITER>` is the iteration number of the current iteration. If the `PEEK` file contains one integer (`PITER`), a target iteration number, the solutions of that iteration is stored to files with a `_<ITER>` suffix. Solutions of the current iteration are also stored if the target iteration has been passed, i.e., current iteration number is larger than the target iteration specified in the `PEEK` file. If the target iteration number is negative (`-PITER`), the file name suffix is constant `_PEEK` instead of the changing iteration number (`_<ITER>`).

If the `PEEK` file contains two integers (`[ - ]PITER PSTEP`), for example

```
20 100
```

the solutions are stored starting from the iteration `PITER` (20 in the example) and repeating after every `PSTEP` iterations (100). The default file suffix is constant `_PEEK` so that the possible large solution files do not fill the file space. With a negative starting iteration number (`-PITER`) the file suffix contains the iteration number (`_<ITER>`) but this must be used carefully.

The starting iteration (`PITER`), iteration step (`PSTEP`), and the choice of the file suffix can also be specified by using the [command line options](#).

4.9 External ITER file: changing parameters during iteration

It may also be useful to be able to modify the parameters of the [iterative method](#) during the iteration. This can happen, for example, if the original [maximum number of iterations](#) is found to be too low or [stopping criterion](#) too tight during the iteration.

The solver programs can be instructed to update some of the iteration parameters by creating a file named `ITER` during the execution in the directory where the solver is executed. When [parallel computing](#) is applied, the `ITER` file has to be accessible by the master process. The `ITER` file is read in the beginning of each `PCG` iteration and affect the iterations henceforth.

Content of `ITER` is either one or two lines similar to [solver option file](#) lines with optional comment lines. The first line contains the same parameters as the `STOP` line of the [solver option file](#) with four:

```
# STOP: maxiter, tolerance, criterion (A/R/M/D), [enforce (F)]:
        6000      1.0e-6      M                      F
```

or five parameters on the line:

```
# STOP: maxiter, tolerance, criterion, [enforce(F), maxiter non-linear]:
        6000      1.0e-6      M                      F          1000
```

The fifth parameter is used by [threshold-models](#) and [deregession](#). Parameters on the `STOP` line control the normal `PCG` iterations of the solvers.

For [estimating variance components](#), optional second line similar to `STOPE` line of the [solver option file](#) can be specified:

```
# STOP: maxiter, tolerance, criterion (A/R/M/D), [enforce (F)]:
        6000      1.0e-6      M                      F
# STOPE: REMLrounds, nSamples, Conv.value VCE
        200       10         1.0e-10
```

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

Parameters on the **STOPE** line control the **MC EM REML** iteration.

Alternatively, in the case of **heterogeneous variance**, the second line of **ITER** file is similar to **STOPC** line of the **solver option file**:

```
# STOP: maxiter, tolerance, criterion(A/R/M/D), [enforce(F)]:
        6000      1.0e-6      M      F
# STOPC: max.HYcycles, conv.value for lambda estimates (Cd^2)
        100      1.0e-9
```

In order to create the **ITER** file safely without the danger of the solver program concurrently reading possibly incomplete file content, a lock file **ITER.LOCK** can be created:

```
touch ITER.LOCK
cp -f ITER.NEW ITER
rm -f ITER.LOCK
```

The solver does not read an existing **ITER** file as long as the lock file **ITER.LOCK** exists. After reading and accepting the **ITER** file successfully, it is renamed to **ITER.OLD**.

Reading of the **ITER** file is notified by a message listing the changed parameters:

```
***** M i X 9 9 s   M e s s a g e   *****
Message:                                     Time: 13:30:10.7  02.07.2019
  Updating iteration parameters from ITER file:
    - Solver STOP line in ITER file:
    >> 6000      1.0e-6      M      F
        - Max. number of iterations changed from 5000 to 6000.
        - Stopping criterion value changed from .5e-6 to .1E-7.
        - Stopping criterion changed from D to M.
        - Renaming ITER file to ITER.OLD.
*****
```

5 Output files of the MiX99 solvers

5.1 Standard output

The solver programs `mix99s` and `mix99p` will write information about the specified solver options, about the iteration process as well as a sample of solutions and the description of the solution files to standard output.

Example of MiX99 solver output:

```
...
_____ Parameters _____

Memory requirements                      : high.

Number of trait patterns                  :          1
Number of residual (co)variance matrices :          1
Number of inv(R) matrices                 :          1

Given number of CPU threads              : 10
Number of CPU threads by MKL             : 10
Number of CPU threads by OpenMP          : 10
Checking of release information in files  : yes

Maximum Number of PCG Iterations ..... 10
Minimum Number of PCG Iterations ..... 1
PCG Stopping Criterion ..... CR < 0.1000E-07

Calculate residuals : No.

Standard solution files.
Preprocessor program git number: 3lae24d7

_____

MiX99_SOLVE: Start of PCG Iteration           Time: 11:12:35.5  30.09.2025
_____

Iteration Statistics
-----

Convergence Indicators
-----
```

ROUND	CA	CR	CM	CD	MAX.CHA.
Solution vector will be initialized to be zero					
	rhs' * rhs =	0.185225002431870			
	animal rhs' * rhs =	5.070000081062319E-002			
0	1.000	1.000	1.000	0.000	0.000
1	0.4842	0.2575	0.2207	1.000	3.430
2	0.1754	0.9276E-01	0.7472E-01	0.1539	0.3692
3	0.1104	0.6328E-01	0.5308E-01	0.8938E-01	-0.3294
4	0.7422E-01	0.4172E-01	0.3741E-01	0.1179	0.3491
5	0.8466E-01	0.4494E-01	0.3465E-01	0.6819E-01	-0.2029
6	0.3030E-01	0.1594E-01	0.1380E-01	0.6376E-01	0.1943
7	0.1739E-01	0.9124E-02	0.7221E-02	0.2382E-01	-0.9714E-01
8	0.6203E-02	0.3774E-02	0.3381E-02	0.8615E-02	-0.2129E-01
9	0.9535E-03	0.4991E-03	0.4089E-03	0.4604E-02	0.1745E-01
10	0.1013E-14	0.5360E-15	0.4344E-15	0.9120E-03	-0.3014E-02

```

Number of iterations reached the number of equations (10)!
Stopping criterion CR < 0.1E-7 achieved in 10 iterations.
_____
```

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

MiX99_SOLVE: End of PCG Iteration

Time: 11:12:35.5 30.09.2025

Solutions for First 20 Levels of Across-Block Fixed Effect: 1 sex

Fact.Trt	Level	N-Obs	Eq-No	Solution	Factor
1 1	1	3	9	4.35850	sex
1 1	2	2	10	3.40443	sex

First 20 Individual Additive Genetic Solutions

Fact.Trt	Animal-ID	N-Desc	N-Obs	Eq-No	Solution	Factor
1 1	1	2	0	1	0.984446E-01	animal
1 1	2	2	0	2	-0.187701E-01	animal
1 1	3	2	0	3	-0.410842E-01	animal
1 1	4	1	1	4	-0.866312E-02	animal
1 1	5	1	1	5	-0.185732	animal
1 1	6	1	1	6	0.176872	animal
1 1	7	0	1	7	-0.249459	animal
1 1	8	0	1	8	0.182615	animal

Description of the Solution Files:

"Solfix"-File: Solutions for Across-Block Fixed Effects

Column	Description
1	Factor Number
2	Trait Number
3	Level Code
4	Number of Observations
5	Solution
6	Name of Factor
7	Name of Trait

"Solani"-File: Solutions for the Individual Genetic Effects

Column	Description
1	Individual ID
2	Number of Descendants
3	Number of Observations
4	Solution for Trait 1 weaningW and Factor animal

MiX99_SOLVE: --- D O N E ---

Time: 11:12:35.5 30.09.2025

5.2 Successful execution of MiX99 solver

Successful execution of MiX99 program is indicated with a line containing

--- D O N E ---

in the end of the output of the program. Before finishing an error-free execution MiX99 program will also create an **OK-file** named **OK_<program>**, i.e. **OK_mix99s** or **OK_mix99p** for the solvers. If this file is missing, the program was terminated due to some

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

error. When using MiX99 through a script, please check existence of the OK-file.

Convergence of the **PCG** iteration process is reported with one or more lines of information depending on the values of the **convergence indicators**, how the iteration was ended, and the number of iterations.

If the **iteration process was stopped** using an external **STOP** file, this is indicated with:

```
Iteration process has been stopped externally!
```

If the number of iterations reached the number of equations that is the maximum allowed number of iterations, this is indicated with, for example, line:

```
Number of iterations reached the number of equations (10)!
```

As the main **convergence information**, if the user given **stopping criterion** was achieved during the iteration, this is indicated with a line containing, for example:

```
Stopping criterion CR < 0.1E-7 was achieved in 10 iterations.
```

Otherwise, opposite result is indicated with:

```
Stopping criterion CR < 0.1E-7 was _NOT_ achieved in 10 iterations.
```

If **CD convergence indicator** was as the **stopping criterion**, it is reminded that the **cd** indicator needs to be smaller than the given tolerance by two consecutive iterations:

```
NOTE: CD criterion must be met by two consecutive iterations!
```

In addition to the user chosen **stopping criterion**, the convergence of the **cd convergence indicator** is reported depending on its last value with lines:

```
Solutions have converged according to CD criterion of the last iteration.
```

```
Solutions most likely converged according to CD criterion of last iteration.
```

```
Solutions converged poorly according to CD criterion of the last iteration.  
Some solutions may be unreliable.
```

or

```
Solutions converged very poorly according to CD criterion of last iteration.  
Some solutions are unreliable.  
Please check your data, pedigree, (co)variance components, and the model.
```

Additional lines are also shown if the last **largest change** indicator was very large:

```
Very big largest round to round change for solution (MAX.CHA.):  
Size of change: 249849832.9823978  
Equation number: 12998  
Please check fixed effect classes.
```

Lastly, if the number of iterations was very high, this is indicated with:

```
Model showed poor iteration convergence characteristics.
```

or

Model showed very poor iteration convergence characteristics.
Please check your data, model, and pre-conditioning!

5.3 Solution files

5.3.1 Formatted solution files

The structure of the standard solution files depends on the model. Therefore, the solvers write for each solution file an explanation to the standard output after the solving procedure has finished.

- Solani** Solutions for additive genetic effects.
- SolMS** Mendelian sampling terms computed from additive genetic effects.
- SolPA** Parent averages computed from additive genetic effects.
- SolSNP** Solutions for marker effects in the component-wise ssGTABLUP and ssSNPBLUP.
- Solfix** Solutions for all across blocks fixed effects.
- Solfnn** Solutions for the n^{th} within blocks fixed effect. E.g., **Solf02** is the solution file for the 2nd within block fixed effect.
- Solrnn** Solutions for the n^{th} random effect in the model. E.g., **Solr03** is the solution file for the random effects with the random effect number 3. Solution files for the random effects are optional (see [RANSOLFILE](#) instruction line in *Technical reference guide for MiX99 pre-processor*).
- Solreg** Solutions for the regression effects applied across the whole data. (Specified on the [REGRESS](#) instruction line. see *Technical reference guide for MiX99 pre-processor*).
- Soldyd** Solution files with [daughter yield deviation](#) for sires.
- Sol_mn** For some [LS-models](#) only. Solution file with the estimate for the mean.

5.3.2 Unformatted solution files

The **MiX99** solvers write solutions to unformatted files which will allow a restart of the solvers with the solutions given in these files.

- Solvec** The **mix99s** solver writes a copy of the solution vector to this file after the end of the iteration process. At each start of **mix99s** the program will check whether a **Solvec** file is provided, and if so, it will initialize the solution vector with solutions given in **Solvec**. Thus **mix99s** can be restarted without running the pre-processor.

Note, the pre-processor **mix99i** will erase an old **Solvec** file. In case the **mix99i** pre-processor is instructed to include old solutions, it will create a **Solvec**, which will be read by the solver. For instructing MiX99 to read old solutions please see [SOLUNF](#) instruction line in the *Technical reference guide for MiX99 pre-processor*.

- Solpriv(i)**, **Solcommon** These files contain the solutions private to each process and common to each process. These files allow restart of the **mix99p**

solver. The files have the same meaning as the `Solvec` file for the `mix99s` solver.

Solunf Contains all solutions to the `MME` and the original ids of all effect levels. This file is optional (see `SOLUNF` instruction line in the *Technical reference guide for MiX99 pre-processor*) and can be rather large. The file can be used to initialize, in a future evaluation when more data has accumulated, the solution vector with old solutions. For a future evaluation the file must be renamed to `Solold` to be read by the `mix99i` pre-processor.

5.4 Files for model validation purposes

The solver programs can be instructed to provide information useful for model validation purposes or information need for other type of analyses. The specified option on the `RESID` and `VALID` option lines will instruct which of the following unformatted files are created:

- eHat.data (i)** File(s) with `residuals`.
- yHat.data (i)** File(s) with `predicted observations`.
- sHat.data (i)** File(s) with a `sum of selected model factors`.
- YD.data (i)** File(s) with `yield deviations`.
- IDD.data (i)** File(s) with `individual daughter deviations`.

The `mix99s` solver will create one file only with a zero character (0) added at the end of the filename. The `parallel` solver `mix99p` will create as many files as there are processes specified for the parallel run. The files are numbered by (i), where (i) goes from zero to number of processes minus one and the number is added at the end of the filename.

The file(s) contain strictly as many rows as there are data rows in the input data file, regardless whether some input data is missing or not used in an analysis. The order of the rows is consistent with the order of the data rows in the input data file. Each row consists of a fixed number of real values, which depends on the applied model. The number of real values is the same on all rows and is equal to the number of traits in the largest trait group. This number is equal to the `mxntra` parameter in the `Parlog` file. The `Parlog` file is produced by `mix99i` pre-processor. The real values in a particular row correspond to the trait group that is specified on the corresponding data row in the input data file. The order of the values within a row corresponds to the order of the traits within a trait group as specified on the `MODEL` instruction lines (see *Technical reference guide for MiX99 pre-processor*). In case an observation is missing in the data file or it is not used in the analysis, a `missing value` variable will be written for the corresponding real value. This `missing value` variable is `-8192.0`. In case the `SCALE` option is used, all information will be transformed back to the original scale before writing to the files.

All values on a row are stored as single precision real values and simple Fortran programs can be written to transfer the files to text files. However, the **MiX99** packages provides `MiXtools` programs, which allows simple analyses of the information (means and SD by classification), or merging of the files with the input data file. See `MiXtoolmerge.f90` and `MiXtoolms.f90` in the `MiXtools` directory of the package.

6 Reliabilities

6.1 Approximate reliabilities using ApaX

Sometimes reliabilities (accuracy squared) of estimated breeding values are needed, e.g., Interbull requires reliabilities for the evaluated bulls. Reliabilities need elements of inverse of the mixed model equations (MME). Exact inverse of MME cannot be computed in most cases due to computing time or memory limitations. Thus, approximations need to be used. A separate program named `apax99` has been made to calculate approximations to reliabilities. A [parallel computing](#) version has been written as well, named `apax99p`.

In general, ApaX was made to handle linear animal models. However, even here there are restrictions. Reliability calculation accounts only those effects that are defined to be within block. Thus, ApaX requires that both a block code and a relationship code have been defined. Further, some new features are not yet supported. For example, no additional correlation files (needed by MAS BLUP), or regression design matrices (needed by genomic BLUP) are accounted by ApaX.

Four approximation methods to calculate reliabilities have been implemented with some additional ones being variations of these four methods. The approximation methods have two steps ([Strandén et al., 2000](#)). The first step accounts for data design, and the second step accounts for relationship information. The first step is the same for all approximation methods and is computationally most demanding. This step uses [parallel computing](#) in `apax99p`.

The following calculation methods are base to all available methods:

- 1) Interbull reliabilities ([Strandén et al., 2000](#))
- 2) Misztal and Wiggans approach ([Misztal and Wiggans, 1988](#))
- 3) Jamrozik et al. approach ([Jamrozik et al., 2000](#))
- 4) Tier and Meyer approach ([Tier and Meyer, 2004](#))

Reliabilities by method 4, i.e., Tier and Meyer approach, are available in the single processor `apax99` and parallel `apax99p` versions.

Some notes on the base methods:

- Method 1. Calculations are based loosely on the guidelines set by Interbull, and have been accepted by Interbull to be used in the Finnish dairy cattle evaluations. There is a post-processing program called `BR2.f90` that produces the information required by Interbull from the output given by `apax99` or `apax99p`.
- Method 2. Approximation has two steps. The first step calculates information amount due to observations by model design. The second step uses the method of [Misztal and Wiggans \(1988\)](#) to incorporate relationship information.
- Method 3. Similar to method 2, except that the method by [Jamrozik et al. \(2000\)](#) is used in the second step to account for relationship information.
- Method 4. The first step is the same as in the other approximation methods 1-3. However, all subsequent calculations use matrices unlike the other approximations that rely on scalar computations for multiple trait cases as

well. Because of matrix computations, method 4 often uses more memory than the other approximation methods, especially when many traits and/or random regression test day models are analyzed.

6.1.1 Approximate reliabilities for single-step

The [single-step](#) genomic BLUP model includes a genomic relationship matrix. This will lead to a different kind of relationship matrix than having only the pedigree-based relationship matrix A . ApaX assumes that the full $G^{-1} - A_{gg}^{-1}$ has been given and is available to the reliability calculations. However, this information is used only if requested (see below). In general, the computational steps are

- 1) calculate effective record contributions.
In this step, non-genetic effect information is accounted.
- 2) calculate reliabilities.
In this step, effect of the pedigree-based relationship information is accounted
- 3) single-step reliabilities.
In this step, effect of genomic information to the genotyped animals is accounted.

There are three approaches for the single-step models in the last step:

- G0 Do not account genomic information (default)
- G1 Use Misztal approximation method 1
- G2 Use Misztal approximation method 2

Because there are many ways to compute breeding values for single-step, but only one single-step model is allowed, the approach without accounting genomic information is the default. We have developed more accurate and less memory consuming approaches than the G1 and G2 approaches. These require more than just one execution of ApaX. Nevertheless, we recommend using these better approaches instead of the options given here (G1 and G2).

When genomic information is accounted, the Misztal approximation methods ([Misztal et al., 2013](#)) are used (G1 or G2). Method 1 (G1) uses the full genomic relationship matrix as a block, and method 2 (G2) relies only on its diagonal values. Consequently, G1 can be computationally demanding when the number of genotyped individuals is large. Both methods invert matrices having the external matrix, either the full matrix or its diagonal. If the matrix is not positive definite or the resulting reliability is less than that already calculated using pedigree information, the original pedigree-based information reliability is kept. Thus, It is possible that including genomic information using method G1 and G2 does not lead to an increase in reliability compared to pedigree-based EBV reliability.

6.1.2 Differences of reliability calculation and breeding value estimation

Steps for calculation of approximate [reliabilities](#) by the **MiX99** package are similar to solving **MME** by `mix99s/mix99p`. The `apax99` and `apax99p` programs accept data prepared by `mix99i`. There are, however, some restrictions and modifications to the regular directive files:

- 1) Block and relationship codes need to be defined in the data and pedigree files, see command [DATASORT](#). Use of automatic block codes can lead to too low

reliabilities.

- 2) Only effects that are defined to be within block are considered when approximate reliabilities are calculated.
- 3) Animal genetic effect has to be the first effect in the within block equations.
- 4) Inbreeding coefficients are not accounted in the reliability computations.
- 5) No [heterogeneous variance](#) modeling is accounted ([multiple residual variance matrices](#) are, however, used in the first step when calculating cow reliabilities but not in the second step where the residual is taken from the variance component file for all random effects).
- 6) [Preconditioner](#) matrix information is ignored. Thus, computationally lightest preconditioner, i.e., no preconditioner, can be best when no breeding values need to be estimated.
- 7) Genetic groups can exist in the pedigree and data but are not accounted in the reliability calculations. Hence, results are the same with or without genetic groups.
- 8) External correlation matrices in a file for random effects are not accounted except in [single-step](#) model parts of the additional matrix needed by single-step are used.
- 9) Regression coefficient matrices are not accounted.

The use of block and relationship codes in the data file means that corresponding values are in the pedigree file. The first column in the pedigree file has the relationship code and the fourth column has the block code. The data file has to be sorted by the block code as the first key, and the relationship code as the second key. When trait groups are used, the data needs to be sorted by trait group as the third key.

In practice, the (first and) second restriction relates to memory use. Although it is possible to combine large herd blocks in breeding value estimation without problems (especially in single processor case), here large blocks may use too much memory. It is advisable to have blocks with contemporaries in the same block, e.g., herd. If there are many animals that have observations in several blocks (e.g., herd changing cows), memory may become a limiting factor. Some animals do not have a herd number. Then, a dummy herd number needs to be used. Several dummy herd numbers may have to be used in order to keep memory requirements tolerable. Sometimes even animals with observation lack herd number. Then, dam herd number (or dam's dam etc.) may give optimal performance. See [Technical reference guide for MiX99 pre-processor](#) for visual tools on blocks (Visualizing block-to-block dependencies).

After running the preprocessor program `mix99i`, a program called `imake4apax` needs to be executed before running the [parallel](#) version `apax99p`. When breeding values are estimated, number of common blocks need to be defined. Here it does not matter how many common blocks have been given because `imake4apax` will reset this to zero.

6.1.3 ApaX instruction file

The **ApaX** reliability approximation programs `apax99` and `apax99p` require information, which is read by the standard input of the programs.

Example of ApaX instruction file:

```
# Reliability method (AccurType):
2
# Number of non-zeros in sparse matrix (MaxNonZ):
10000
# Original dir file (OriginalDir): "-" = MiX99_IN.DIR and MiX99_IN.OPT:
-
# Absorption level effect (JFilter):
2
```

The information is given on instruction lines in the same order as presented below. Information that is model dependent is given in *italics*:

AccurType Number of approximation method:

- 1** Interbull EDC/ERC and reliabilities (sire model)
- 2** Misztal & Wiggans
- 3** Jamrozik et al. approach
- 4** Interbull Tier and Meyer approach

The following are available only in `apax99`:

- 10** Interbull EDC/ERC/reliabilities, genotyped discounted (sire model)
- 11** Like "10" but animal model
- 20** [Reversed reliability approximation](#)

For [parallel computing](#) there is option **1d** for the Interbull method that is for distributed memory computers. In **1d** the second step of the Interbull method uses [parallel computing](#) as well. The main advantage is distribution of memory where each parallel job uses less memory than a single processor version but combined memory use is greater.

Several additional options can be given on this line:

- G0** No genomic information used for single-step (default)
- G1** Use approximate method 1 for single-step
- G2** Use approximate method 2 for single-step
- O** Write `PEV.bin` file after the 1st approximation step.
- I** Read `PEV.bin`. Thus, no need to perform the 1st step.
- M** Maternal and direct genetic trait assumption in calculation of phenotypic variance. Thus, when additive genetics has both direct and maternal effects, it is necessary to inform the program that correct formula is used in calculation of phenotypic variance:

$$var(g_{direct}) + var(g_{maternal}) + cov(g_{direct}, g_{maternal}) + var(residual).$$

- R** Maternal genetic reliabilities. This option informs that maternal trait reliabilities should be calculated instead of direct genetic reliabilities. Please use option **M** as well to have phenotypic variance calculated properly.

- V** Compute variances of random effects to file `APAX_VAR_COMP.TXT`. The variance components are by the defined test-days in a random test-day regression model.
- L** Long output listing. Default is short output listing.
- C** Cancel checking of pedigree loops (in `apax99`).
- P** Pedigree filename given instead of original directive filename (for [reversed reliability approximation](#)).

Options **O** and **I** allow quicker execution when multiple approximation methods need to be executed. For example, method 1 and 2 reliabilities need to be calculated. First, make method 1 reliabilities and write file `PEV.bin`. Then, `PEV.bin` is read (input) and used in a subsequent run to calculate method 2 reliabilities without the time consuming 1st step. Note that the directive files (for `mix99i` and `apax99`) need to be exactly the same in the two runs except for change in first command line to `apax99` where reliability calculation method and options can be changed.

MaxNonZ Maximum number of non-zeros in the sparse matrix of the largest block. The program will process data block by block, and build coefficient matrix of the mixed model equations ([MME](#)) for the equations within block. [MME](#) of each block is stored as a sparse matrix, and, here, maximum number of non-zeros element in the sparse matrix is given.

It is not necessary to know exactly this value because dynamic memory allocation approach is used to increase size of the sparse matrix. However, it may be good to have a reasonable value for the required size. The implemented dynamic memory allocation approach increases sparse matrix by 50% when the matrix becomes full. Consequently, memory may be reserved more than actually required (at most 50% more).

In shared memory [parallel computers](#) this may be very important. Note that if memory is increased above available memory, less than 50% is added. Eventually, no memory can be added to increase sparse matrix size, and then disk is used through file `Sparse_Matrix.DMPz` where `z` is the process number. Note that this will slow computations considerably. Naturally, the sparse matrix cannot grow beyond size of available disk memory.

StartDIM ***OPTIONAL: asked only when a [covariable table](#) was given in the directive file.*** Starting index value of the first line in the [covariable table](#), if [covariable file](#) is present. The data file has index values (see 7.3) that correspond to lines in the covariable file. Each line of the covariable line is stored internally in **MiX99** compactly such that the starting index value is lost. This information has to be given again here. In dairy cattle, this is often the lowest value for days in milk.

CovarInfo ***OPTIONAL: asked only when a [covariable table](#) was given in the directive file.*** One line for each model line in the directive file for `mix99i`, if [covariable file](#) is present. Each line has 3 numbers: Smallest [covariable table](#) index value used for the breeding value index using random regressions, number of [covariable table](#) calculation points, and distance between

the points. For example, if breeding value is defined by calculating random regressions from days in milk every day from 8 to 312 (to get 305 day breeding values), then a triplet 8 305 1 is given (StartDIM above must be 8 or lower). If the days are from day 15 every 30th day for 10 points, then the triplet is 15 10 30.

OriginalDir Name of the original directive file given to the pre-processor program `mix99i`. If dash (-), uses the input directives and command line options stored in files `MiX99_IN.DIR` and `MiX99_IN.OPT` by the last run of the pre-processor.

NumBVs **OPTIONAL: Asked only when there are more than one trait in the model.** Number of breeding values to be calculated. Zero (0) indicates breeding values for all traits with default [Weights](#). Zero is allowed only for models for which the default weights can be deduced. NOTE: the value can be at most number of traits.

Weights **OPTIONAL: Asked only when [NumBVs](#) is greater than 1.** A line for each breeding value index ([NumBVs](#)). Each line gives weights according to the breeding value index wanted. Number of weight values on each line is equal to the number of traits in the model. For example, there are three traits in the model. Now, an index weight line has three values, each informing how the traits are weighted in a breeding value index. A valid line would be like '1 1 0' where the first two traits are weighted equally but last one is not accounted.

H2calc **OPTIONAL: Asked only when there are more than 2 random effects, i.e., more than just genetic and residual random effects.** Random effect numbers included in the calculation of phenotypic variance in heritability, if more than 2 random effects (i.e. additive genetics and residual) in the model. The two last random effects (genetic and residual) are automatically included in the calculation of phenotypic variance of heritability. However, when there are other random effects as well, their inclusion is asked here. The random effects are numbered as in the parameter file for variance components. For example, there are five random effects: herd test month (number 1), first permanent environment (2), second permanent environment (3), additive genetic (4), and residual (5). If all but herd test day is wanted in the heritability then a line '2 3' is given indicating that second and third random effect are included in the phenotypic variance.

JFilter Calculation of reliabilities is programmed such that all within block effects are absorbed to the animal genetic effects. They can be absorbed either exactly or approximately. JFilter describes which effects are approximately absorbed. If the exact absorption is done, the sparse matrix may be filled and exhaust available memory. In addition, the computations may be slow. Therefore, some effects can be only approximately absorbed with no additional fill-in to the sparse matrix. In order to minimize memory use in absorption, ordering of the effects within block should be such that the smallest number is given to an effect with observations from a single animal and larger numbers to the effects with observations from several

animals. Effects are ordered as given in the [within block ordering](#). Exact absorption is commonly done for the effects that are within animals such as permanent environmental effects. For example, let's consider a case in which we have three within block effects: 1) genetic, 2) permanent environment, and 3) herd year. If we want to absorb the herd effect approximately, the JFilter value of 3 is given. Thus, only the permanent environment effect (effect 2) is exactly absorbed to the genetic effects. In a situation, in which we want to exactly absorb all other within block effects to the genetic effects (n within block effects), a value of n+1 should be given to the JFilter. In the example given above, JFilter would be 4.

6.1.4 Guidelines for determining blocking and JFilter

Two of the most difficult variables affecting reliability calculations are blocking ([SORT_R](#)) and absorption level ([JFilter](#)). Some guidelines for calculation of [reliabilities](#):

1) Start model building for reliability calculations from a simple to a more complex model.

The simplest model for reliability calculations should have the genetic effects and non-genetic animal effects (e.g., permanent environment). Then, some kind of management effect can be tested in the model. Management effects are commonly approximately absorbed (see next comment) and can create problems in absorption process. Look at the output. Perhaps, there are many management effects or alternatives. Test them separately. The less there are messages of type 'ABSORB: Singularity in row' the better from numerical calculation point of view.

2) Have only one effect per trait absorbed approximately.

When several effects are absorbed approximately, they tend to double count information, and may lead to negative information which is seen as messages of singularity given by the absorption procedure. Approximate absorption does not account possible co-linearity or other correlations in design between effects. As an extreme case, assume an effect being twice in the model and both effects are absorbed approximately. Then, the other effect will lead to double accounting of the effect, because approximate absorption does not notice that the same effect is twice in the model. This means that prediction error variance left for an individual can become negative. In general, this is the more likely the less there are number of observations in an effect class, e.g., small herd and herd-year classes.

The approximately absorbed effect from the full model should be the one with most levels for the model to account best management effect from modelling point of view. This effect is typically a herd management effect by time effect like herd-year. Unfortunately, this can be numerically the worst effect. Sometimes it is possible to have more than one effect per trait if all approximately absorbed effects are random. However, this is very rare due to likely increase in numerical problems.

3) All effects close to animal (e.g., permanent environment) should be absorbed exactly. These effects do not result in additional fill-in, if order of observations for an animal can be such that they are close to each other. If observations of an animal are in different blocks, then memory is used more.

4) Blocking should be used to group animals by the herd management effect in the model (management without time) for efficient memory management.

- If there are too many blocks, then memory is used too much because many animals tend to become block/herd changers.
- If there are not enough blocks, memory is again used too much, because all information for a block is read to memory.

5) If there is a maternal effect in the model, try to have dam in the same block as its offspring with observations. When dam is in a different block than any of its offspring, the two blocks are connected, and there are block changer equations. **ApaX** will make a separate block from these block changers. The block should be kept as small as possible for memory reasons by ordering as many dams as possible to the same blocks as their offspring with observations.

If there are problems with reliability calculations, it is best to start with a simple model which has only animal genetic effect. Then increase level of complexity in the model. Problems in reliability calculations involve: long computing time, unusual reliabilities (close to one without reason), diminishing reliability with increased data. Changes in blocking strategy, absorption level or effects in the reliability calculations model are most likely to cure any problems.

Finally, one useful strategy in understanding reliability calculations is to have a dummy pedigree where all animals have unknown parents. Reliabilities using such pedigree can be compared with reliabilities by complete pedigree information (using methods 2 and 3). Comparison shows if the problems are due to pedigree, and not due to above mentioned variables. In addition, it gives information on which animals are the ones with most problems.

6.1.5 ApaX Output files

Once **apax99** or **apax99p** has been executed, the reliabilities will be written to a solution file, called **PEVani**. The solution file has similar structure to the solution file **Solani** produced by the solver programs. However, the **PEVani** file has a different format depending on the computing method used. When the Interbull method is used each line has both effective daughter contributions (EDC) for sires (effective record contributions (**ERC**) for others) and reliabilities. When any other method is used, only reliabilities are given and format of the **PEVani** file is the same as the **Solani** file.

The **PEVani** file produced by the Interbull method can be made more accessible by program **BR2.f90**. This program expects that files **PEVani** and **BR2.dat** written by **apax99/apax99p** are available. In addition, names of two files are asked:

SireFile This file has id numbers of all the sires for which the sire reliabilities were calculated. The file should be a regular text file with each line having a bull id number as the first number.

OutFile This is the output file that will be generated.

Each line of the output file has the following information:

- 1) Id number of a bull.
- 2) Effective daughter contributions (**EDC**) for each sire breeding value (second pedigree file column) and effective record contributions (**ERC**) for all the others.
- 3) Reliability for each breeding value according to EDC/ERC.

6.1.6 Example of ApaX instruction file

ApaX99 instruction file:

```
# Type of analysis: 1= Interbull accuracies
1
# Maximum number of non-zeros in the sparse matrix
600000
# Start DIM in covariable table file,
1
# For each model line: First DIM, Number of DIMS, DIM step
8 305 1
8 305 1
8 305 1
# Original directive file given to mix99i (or -):
miniT.mix
# Number of breeding values
3
# Weights for breeding values
1 0 0
0 1 0
0 0 1
# random effects accounted in h2 calculations
2
# Absorption level effect
3
```

MiX99 instruction file for mix99i (miniT.mix):

```
# Estimation of breeding values for milk, protein kg and fat kg for the
# Finnish dairy cattle using a multiple trait random regression test day
# model based on covariance functions. Reduced data and model: only first
# lactation included.
# -----
# title
#   Finnish RRTD-Model; first 0.1% of data from 1988 to Feb. 2000
# INTEGER
#   herd animal trgrp cowxlac hy htd mg ym yr_sea age dcc dim
# REAL
#   milk protein fat
# traits
#   3
# trait-groups, input column
#   1 3
# input column of block code and relationship code
#   1 2
# number of fixed- and random factors columns in the model lines
#   8 13
# MODEL:
#   | fixed effects |ran|non-hereditary | add. genet.
#   | | season | |dom| across 1&2 L | animal eff.
#s t wt|s1 s2 s3 s4 s5|age dcc ym|htm|n1 n2 n3 n4 n5 n6|a1 a2 a3 a4 a5 a6
1 1 - 9 9 9 9 9 10 11 8 6 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 2 - 9 9 9 9 9 10 11 8 6 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 3 - 9 9 9 9 9 10 11 8 6 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
# order of effects within block
#   - - - - - - - - 3 2 2 2 2 2 1 1 1 1 1 1
#htm n1 n2 n3 n4 n5 n6 a1 a2 a3 a4 a5 a6
1 2 2 2 2 2 2 3 3 3 3 3 3
1 - - - - - - - - - - - -
1 - - - - - - - - - - - -
# pedigree for animal effects: num: a1 a2 a3 a4 a5 a6
#   6 1 1 1 1 1 1
#n s1 s2 s3 s4 s5| age dcc ym|htm|n1 n2 n3 n4 n5 n6| a1 a2 a3 a4 a5 a6
21 t1 t2 t3 t4 t96 cl cl cl cl t5 t6 t7 t8 t9 t10 t59 t60 t61 t62 t63 t64
21 t1 t2 t3 t95 t97 cl cl cl cl t11 t12 t13 t14 t15 t16 t65 t66 t67 t68 t69 t70
21 t1 t2 t3 t95 t98 cl cl cl cl t17 t18 t19 t20 t21 t22 t71 t72 t73 t74 t75 t76
# combining of traits
```

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

```

y
# |      fixed effects      |ran| non-hereditary | add. genet.
# | |      season      | |dom| across 1&2 L | animal eff.
# s1 s2 s3 s4 s5|age dcc ym|htm|n1 n2 n3 n4 n5 n6|a1 a2 a3 a4 a5 a6
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1
3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1

# covariable filename:
suomiTDMpara.cov
# number of covariable columns
100
# integer input column in data file with covariable index (DIM)
12
# method used for relationship
am
# input file
Ter.dat
# int-col. real-col. form
12      3      f
# code for missing real values
0.0
# scaling (y/n)
n
# pedigree file
miniTDM.pedi
# parameter file
suomiTDMpara.in
# directory for the temporary files
.
# solution files for random effects: htm non-ac animal
n      n      y
# binary solution file
n
# block preconditioner WpW, XpX (=D_M)
d d d d

```

6.2 Exact reliabilities using exa99

Approximate reliabilities can be calculated using `apax99`. However, **MiX99** contains another program, which allows the exact calculation of accuracies via inversion of the coefficient matrix. The program is named `exa99`, has dynamic memory allocation, and is executed after running the pre-processor `mix99i`. `exa99` is only useful for small problems (up to 200 000 equations). When specifying the model in the **MODEL** line(s) of the MiX99 **instruction file** for `mix99i`, it is important to order the effects by number of levels. For within block effects, the effect with the most levels should get the lowest block ordering number and the effect with the least levels the highest. Similarly, for across block fixed effects, the effect with most levels should be specified first and the effect with the least levels should be specified last. This is important to keep the memory requirements as low as possible when inverting the coefficient matrix.

6.2.1 Option file for exa99

Execution of `exa99` requires an option file, which is read by standard input. The first information asked by `exa99` are internal storage format used for making coefficient matrix of the mixed model equations. There are three formats:

F Full dense storage

F

Internally the program stores the coefficient matrix in a rectangular double precision matrix. In practice, this storage format will give the fastest computations but may require a lot of computer RAM memory.

A number can be given after the 'F' letter. This number is used to multiple diagonal elements of the coefficient matrix to make it full rank. Default value is 1.0001. The format is

F 1.0001

V Packed vector storage

V

Internally the program stores the coefficient matrix in a packed upper triangle vector format. Thus, the RAM memory need is about half of the full dense storage. However, computationally this storage form can be much slower.

A number can be given after the 'V' letter. This number is used to multiple diagonal elements of the coefficient matrix to make it full rank. Default value is 1.0001. The format is

V 1.0001

S Sparse matrix storage

S

Internally the program stores the coefficient matrix as a sparse matrix. Number of non-zeros is set to be 15 times the size of the coefficient matrix. A number can be given after the 'S' letter which is initial guess for the number of non-zeros in the coefficient matrix. Thus, then the format is

S 1234567

Only the lower triangle of the matrix is stored. Thus, memory need can be much less than for the packed vector storage. If the number non-zeros given is not enough, the program will increase sparse matrix size. However, this will use more memory than is optimal. Computationally the sparse matrix storage form is the slowest. In addition, total memory need depends on the order of equations and can be even higher than for the full dense storage.

The sparse matrix storage allows numerical filtering of low values in order to preserve some sparsity. After the format line, a line having three numbers (these numbers are examples) is given:

1E-5 1E-6 1E-7

where the first number (1E-5) is for the operational zero of diagonal values, the second number (1E-6) is for the operational zero of off-diagonal values, and the last value (1E-7) is matrix sparsity value. Often these values are the same.

The three values have two purposes. First, these values are used to detect singularities in the coefficient matrix, i.e., dependencies. Singularity detection can help making the computations more reliable because rounding errors due to almost zero values are not propagated. Second, the values allow preserving some sparsity in the matrix. The operational values mean that when during inversion calculations absolute value of added element is less than operational value, the value is not added to the coefficient matrix. Too high operational value leads to too high approximation of computations. The matrix sparsity value is used to make a matrix element value zero when absolute value of an element is less than the sparsity value. Thus, the operational zero values are used to effectively zero values calculated during inversion calculations but the sparsity value is used to neglect elements for these computations even before these computations are made.

The operational zero and sparsity values are ways to increase sparsity in the coefficient matrix during computations. The higher the values given the more approximations are used in the computations. Note that none of these values affect during making of the coefficient matrix.

After the matrix format information has been given, name of the file having diagonal values of the genetic relationship matrix is asked. The file has format:

<ID code> <Diagonal value>

For example, diagonal of the pedigree-based relationship matrix for animal i is $1 + F_i$ where F_i is inbreeding coefficient of animal i .

An example of an **exa99** option file:

F
Adiag.dat

Full dense storage format and diagonal of relationship matrix is in file Adiag.dat.

For sparse matrix storage the command lines can be:

```
S 1234567
1E-5 1E-6 1E-7
Adiag.dat
```

6.2.2 Exa99 output files

The output files have the same setup as when solving the mixed model equations. Structure of the output files depends on the model. Therefore, explanation of the content of those files is given in the printout of the particular run of **exa99**.

- ACCanl** PEV and accuracies for animal effects.
- SEfix** Standard errors for all across blocks fixed effects.
- SEfnn** Standard errors for the n^{th} within blocks fixed effect. E.g., **SEf02** is the solution file for the 2nd within block fixed effect.
- PEVrnn** Prediction error variance for the n^{th} random effect in the model. E.g., **PEVr03** is the solution file for the random effects with the random effect number 3.
- SEreg** Standard errors for the across whole data regression effects.

6.3 Reversed reliability approximation

The reliability calculation (of animal model) can be reversed so that from given reliabilities it is possible to approximate **weights** that would lead back to the same reliabilities. These weights, considered as **effective record contributions** (ERC), can be derived from reliabilities, pedigree, and heritability of the considered trait using reversed Harris and Johnson algorithm (Harris and Johnson, 1998) in single-trait case, and using reversed Tier and Meyer (Tier and Meyer, 2004) in multi-trait case.

The reversed reliability approximation can be calculated with **apax99** (but not **apax99p**) using **AccurType** 20 (single-trait) or **AccurType** 40 (multi-trait). The main input parameters, the reliabilities, are given to **apax99** in a separate file containing two or more columns: identification numbers and one or more reliabilities:

File containing reliabilities for reversed reliability approximation:

Id ₁	Reliability ₁
2	0.15809E-01
4	0.14464
5	0.14379
6	0.11543
8	0.15527
⋮	⋮

Note that the ERCs will be calculated just for those animals that are listed in the file. ERCs of other animals in the pedigree will be set to zero.

A slightly modified **ApaX instruction file** is used to calculate the ERCs from the given reliabilities. Other input information, i.e. pedigree and heritability, are either given using the original MiX99 preprocessor directive file or given directly without the need to use the MiX99 preprocessor.

Both the single-trait and the multi-trait reversed reliability estimation are **parallelized using OpenMP**. Number of parallel computing threads can be controlled with **apax99** command line option **-nt**. Note that the multi-threaded MiX99 executables located in the **mp** subdirectory must then be used.

NEW

6.3.1 Single-trait Reversed reliability approximation (AccurType 20)

- **AccurType** 20 without option **p** indicates that the original preprocessor directive file name is given as **OriginalDir**:

Example **apax99** input file of Reversed reliability approximation:

```
# Reliability method (AccurType): Reversed reliability approximation
20

# Filename of the reliability information (id and r2s):
id_r2s.dat

# Original dir file (OriginalDir): "-" = MiX99_IN.DIR and MiX99_IN.OPT:
-

# ERC parameters: tol maxit smallest [h2]
1e-8 100 0.0001
```

whereas option **p** indicates that the pedigree filename is given instead of the original directive filename:

Example of Reversed reliability approximation using pedigree file directly:

```
# Reliability method (AccurType): Reversed reliability approximation,
# p = pedigree file given, no mix99i needed:
20 p

# Filename of the reliability information (id and r2s):
id_r2s.dat

# Pedigree file:
data/AM.ped

# Number of ERC values (and r2s):
1

# ERC parameters: tol    maxit  smallest  [h2]
                  0      0      0          0.33333333333333
```

In both cases the second line of the input file ([MaxNonZ](#)) is replaced by the name of the file containing the reliabilities.

- If [AccurType](#) 20 p is given, number of reliability columns in the reliability file must be given as the [NumBVs](#) line. This also indicates how many ERCs are calculated for each animal. [NumBVs](#) line must be given also for [AccurType](#) 20 (without option p) if there are more than one trait in the model. Note that the traits are assumed to be independent when using [AccurType](#) 20. Use [AccurType](#) 40 for true multi-trait ERCs.
- After the [NumBVs](#) line there must be a line containing at least three numbers for the ERC calculation specific parameters:
 - **Tolerance:** The reversed Harris and Johnson algorithm calculates the ERC values iteratively. After each step convergence of the iteration is determined by comparing the two last iterations. Convergence is achieved if the relative 2-norm of the difference is smaller than the given tolerance. Value zero indicates **default value** 10^{-8} .
 - **Maximum number of iterations:** If the reversed Harris and Johnson iteration is not converged within the given maximum number of iterations, an error message is printed to screen, and the program is stopped. Value zero indicates **default value** 100.
 - **Smallest allowed ERC value:** Occasionally ERC values tend to become negative during the reversed Harris and Johnson iteration. In such cases the ERC value is replaced by the smallest allowed ERC value. Value zero indicates **default value** 0.0001.
 - **Heritabilities (optional):** Heritability value for each [NumBVs](#) ERC trait or a single value for all traits. Must be given for [AccurType](#) 20 p but can also be used to replace the heritabilities calculated from the original preprocessor directive file ([AccurType](#) 20 without option p). Zero value indicates **default value** 0.5 (with option p) or value calculated from original dir file (without option p).
- [JFilter](#) line can be omitted.

Calculated ERC values can be obtained from the fourth column of the `PEVani` file.

6.3.2 Multi-trait Reversed reliability approximation (AccurType 40)

As mentioned, `AccurType 20` can be used to calculate ERCs for more than one traits, i.e. columns in the `r2` file. `AccurType 20`, however, assumes that the traits are independent. `AccurType 40`, on the other hand, calculates true multi-trait ERCs.

Currently, only `AccurType 40 p` is implemented. It is meant to be used independently without the need to run `mix99i` first.

For multi-trait reversed reliability approximation `ApaX instruction file` needs some additional information:

Example of multi-trait Reversed reliability approximation:

```
# Reliability method (AccurType): Multi-trait reversed reliability approximation
# p = pedigree file given, no mix99i needed:
40 p

# Filename of the reliability information (id and r2s):
id_r2s.dat

# Pedigree file:
data/AM.ped

# Number of ERC values (and r2s):
3

# G0 file name:
G0.txt

# R0 file name:
R0.txt

# ERC: tolerc maxit smallest tol_newton maxit_newton
      1e-6   100   1e-10   1e-4       20
# ERC: tol_r2 minit lastit belowit
      1e-7    10    0.5    10
```

- **Genetic and residual covariance matrices need to be given in separate files** containing number of traits lines each having number of traits matrix values from each matrix row. File names of these files are given after the line containing then number of traits (or ERC values).
- The multi-trait reversed reliability estimation has **two nested iterations**. The outer iteration is similar to the single-trait case whereas the inner iteration is based on Newton's method.
- As the traits can be of different scales, the multi-trait reversed reliability estimation has **two different convergence tolerances**. The first tolerance (`tolerc`) compares the 2-norm of difference of two last iterations similarly as in the single-trait case. The second tolerance (`tol_r2`) operates on "reliability" values calculated from the ercs so that the values are better scaled for different traits. **Convergence is achieved when either of the tolerances is exceeded for all traits.**
- The estimation of accuracy on the second tolerance (`tol_r2`) is controlled by additional parameters. The accuracy is estimated after atleast `minit` (outer) iterations are calculated, proportion `lastit` of the last iterations are used to

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

estimate the accuracy, and last `belowit` accuracy estimates need to be below the tolerance (`tolr2`).

- The ***inner Newton iteration*** has its own tolerance (`tol_newton`) and maximum number of iteration (`maxit_newton`).
- ***Zeros as parameter values use the default values*** shown in the example above.

7 Daughter Yield Deviations

The request of daughter yield deviation (DYD) for different dairy cattle studies and for [model validation](#) purposes made it necessary to implement adequate calculation procedures into MiX99. A general approach for the calculation of DYD is implemented into **MiX99**. The calculations follow the method presented in [Mrode and Swanson \(2004\)](#). For simple models, like single trait animal models, the approach will yield DYD, which are calculated in the same manner as given in [VanRaden and Wiggans \(1991\)](#). In case of random regression models, DYD coefficients will be given. These coefficients can be used in a post-processing procedure to obtain DYD for certain intervals of the time trajectory, like 305-day yield DYD.

DYDs can be obtained for each bull that has daughters with records. Furthermore, it is possible to classify within each bull the DYDs by a classification variable. This is useful for [model validation](#) purposes.

Sometimes there is need for yield deviations (YD) or individual daughter deviations (IDD). Here, an IDD is defined as the YD minus half of the dam's additive genetic animal effect. MiX99 provides an option to calculate for each observation that is included in the analysis the corresponding YD or IDD. The YDs and IDDs will be written to unformatted files. For instructing **MiX99** to calculate YDs or IDDs please see the explanations given for the [VALID](#) option in chapter 4, and about [Contents](#) in chapter 5.

7.1 Calculation of daughter yield deviations

This chapter will explain how **MiX99** can be instructed to calculate [DYD](#). Instructions have to be given for the pre-processing and for the solver programs. In case it is desired to classify the DYDs within sires, additional information will be needed in the pedigree file.

7.1.1 Pedigree file

The additional information in the pedigree file is required only if [DYD](#) should be classified within sires (optional). A column with the classification variable has to be added to the pedigree file. This column must be given after the pedigree information (i.e., column 4 or higher) or after the blocking variable code in case blocking of the data is desired (i.e., in this case column 5 or higher). The classification code will indicate to which within-sire class the daughter will be grouped, e.g., the classification code could correspond with the calving years of the daughters. The classification codes need to be numbered from one (1) to n within a sire. Some classes may be missing within a sire. For animals without observations in the data file a zero (0) is given.

7.1.2 MiX99 instruction file

Instructions specific to the calculation of [DYD](#) have to be given on two instruction lines. First, on the [PEDFILE](#) instruction line an integer value has to be provided after the pedigree filename. There are two alternatives available. Alternative 1: specifying a zero (0) will instruct **MiX99** to calculate DYDs for each bull with daughters that have observations. Alternative 2: Specifying the column number of the pedigree file that contains a within-sire classification will instruct **MiX99** to calculate for DYD for each class within sire. Second, on the [PRECON](#) instruction line a [block diagonal preconditioner](#) matrix (option **b**) must be defined for the additive genetic animal effect.

7.1.3 MiX99 solver option file

Instruction specific to the calculation of **DYD** have to be given on two option lines. The option **d** must be specified on the **VALID** option line. This is followed by a **FACTOR** option line which specifies the model factors that are included into the DYD. Please see [Daughter Yield Deviations](#) in chapter 4.1, for specifying the **FACTOR** option line.

7.2 Solution files for daughter yield deviations

Daughter yield deviations are written to a file named **Soldyd**. The structure of the file is similar to that one of the file **Solani**. The structure of the file is model-specific and the solver will write an explanation to **standard output**. In case of a random regression model, **DYD** regression coefficients are given. Number and order of the coefficients is the same as for the animal effect coefficients in **Solani**. The DYD coefficients are followed by as many integers as there are coefficients for a particular DYD function. The order of these integers is the same as the order of the DYD coefficients, and the integers may have a value of one or zero. Ones indicate that the corresponding coefficients were estimable. This is important for multi-trait or random regression models, where it might be not possible that some DYD functions are not defined. For instance, given a model specifies that first and second lactation observations are different traits and all daughters of a sire have first lactation observations only. Then, there is no DYD function available for the second lactation.

For random regression models, the same covariables as applied for the additive genetic animal effect are also applied to the DYDs. Note: A bull's DYD function is only defined for the time interval in which the bull's daughters have observations. An extrapolation of the function beyond this time interval may yield absurd results.

7.3 Example

The example explained in chapter 7.3 of the [Technical reference guide for MiX99 pre-processor](#) will be modified for the calculation of **DYD**. Modifications are in bold.

CLIM command file:

```
TITLE " RANDOM REGRESSION, L.Schaeffer & J.Dekkers (1994) "

DATAFILE example3.dat
INTEGER HTD Animal
REAL Covar_1 Covar_2 Milk

DATASORT BLOCK=HTD PEDIGREECODE=Animal

PEDFILE DYD example3.ped
PEDIGREE G am

PARFILE variance_comp.ex3

PRECON b d f
WITHINBLOCKORDER G HTD

MODEL
Milk = Lact_curve(Covar_1 Covar_2) HTD G(1 Covar_1 Covar_2 | Animal)
```

MiX99 instruction file:

```
# TITLE:
RANDOM REGRESSION, L.Schaeffer & J.Dekkers (1994)

# INTEGER:
HTD Animal
```

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

```
# REAL:
Covar_1 Covar_2 Milk
# TRAITS:
1
# TRGRP:
1 -
# DATASORT: block_code, relationship_code (single residual var.)
1 2
# FIXRAN: number of fixed and random factors in the model
1 3
# MODEL: trait_group trait weight herd-test-day gamma0 gamma1 gamma2
1 3 - 1 2 2 2
# WITHINBLOCKORDER: order of effects within blocks
2 1 1 1
# RANDOM: gamma0 gamma1 gamma2
1 1 1
# RELATIONSHIPS: number: gamma0 gamma1 gamma2
3 1 1 1
# REGRESS: number beta1 beta2 herd-test-day gamma0 gamma1 gamma2
6 1 2 cl cl 1 2
# COMBINE:
n
# PEDIGREE:
am
# DATAFILE:
example3.dat
# VAR:
2 3 f
# MISSVA:
0.0
# SCALE:
n
# PEDFILE: filename, a zero for calculation of DYD
example3.ped 0
# PARFILE:
variance_comp.ex3
# TMPDIR:
.
# RANSOLFILE: animal effect
y
# SOLUNF:
n
# PRECON: block diagonal preconditioner for the animal effect in WpW
b d f
# PARALLEL: number of processors used by the solver program
1
# COMMONBLOCKS:
0
```

MiX99 solver option file:

```
# RAM RAM demand: H=high
H
# STOP Max.iter, Tolerance, Convergence criterion (Cr), Force
1000 1.0e-7 r f
# RESID Residuals calculation
N
# VALID D for calculation of DYDs
D
# FACTOR: beta1 beta2 herd-test-day gamma0 gamma1 gamma2
0 0 0 1 1 1
# VAROPT Variance options for VCE, PEV, HV
N
# SOLTYP Solution file options (N,Y,A,H)
Y
```

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

Description of the *Soldyd* file written by the solver to standard output:

```
.
.
  Daughter Yield Deviations
  -----

  First 50 Solutions
  -----
    Sire   Class  N-Daug.  N-Rec.  Effect  Trait  Solution      Def.
    9       1      2        9        1       1    -2.98764       1
    9       1      2        9        2       1     0.488173E-01  1
    9       1      2        9        3       1     0.284341       1
    10      1      2        9        1       1      3.74280       1
    10      1      2        9        2       1    -0.433969E-01  1
    10      1      2        9        3       1    -0.516160       1
    11      1      2        5        1       1    -1.03777       1
    11      1      2        5        2       1     0.841567E-02  1
    11      1      2        5        3       1     0.498087E-01  1

  "Soldyd"-File: Daughter Yield Deviations for Sires

  Column | Description
  -----
    1      Code of Sire
    2      Within Sire Classification
    3      Number of Daughters
    4      Total Number of Records
    5      DYD Solution / Coefficient : Factor 1 Trait 1
    6      DYD Solution / Coefficient : Factor 2 Trait 1
    7      DYD Solution / Coefficient : Factor 3 Trait 1
    8      DYD is defined (yes=1,no=0): Factor 1 Trait 1
    9      DYD is defined (yes=1,no=0): Factor 2 Trait 1
    10     DYD is defined (yes=1,no=0): Factor 3 Trait 1
.
.
```

8 Non-linear models

Two non-linear models are implemented into MiX99. Estimation of categorical variables is implemented by the generalized linear mixed model with the probit link function, and estimation of the growth curve models is implemented by linearization of non-linear [Gompertz function model](#) using second order Taylor series expansion. Use of these models is still exiguous and extensions of the MiX99 program for these models are therefore considered as test-versions. There is no possibility to give the [CLIM command file](#) for these models yet. Instead, the MiX99 [instruction file](#) should be made.

8.1 Threshold-model

Prediction of breeding values is possible for models with one categorical and several linear traits. Models are allowed to have [missing traits](#) and unequal design matrices for traits. Thresholds can be estimated or set to be known. (Co)variance components have to be known and residual variance of the categorical trait should be set to one.

8.1.1 Instruction file for **mix99i**

The categorical trait is defined by giving **Tn**, where n is a number of thresholds, on its own column between first and second parameters in the [MODEL](#) line. For example, for a binary trait (recorded as 1 and 2) option **T1** needs to be marked. In case of a multiple trait model, the linear traits are defined first (see [example 7.9](#)). When the threshold model is defined, one or two additional instruction lines must be given right after the [MODEL](#) line.

First additional line defines the method that is used to analyze the threshold models. There are two options: **em** option defines the Expectation Maximization algorithm (EM) ([Gilmour and Thompson, 1998](#)) and **nr** option defines the Newton-Raphson algorithm (NR) ([Janss and Foulley, 1993](#); [Hoeschele, Tier, and Graser, 1995](#)). By default, thresholds are estimated simultaneously. Optionally, additional characters “**ft**” can be specified to indicate fixed threshold values. Thus, a new line must follow, where the threshold values for categorical trait are defined. This line should contain as many real numbers as defined for the categorical trait in the [MODEL](#) line.

8.1.2 Stopping criterion file for **mix99s**

Solving of the [threshold-model](#) is a non-linear problem and iterative in two levels. Therefore, the [STOP](#) line changes to have five entries: an integer, a real value, a character, an enforcing character “**f**” and an integer. Now the first integer value gives the maximum number of [PCG](#)-iterations within each NR- or EM-round (default is 100 or number of equations in the [MME](#)) and the last integer value gives the maximum number of NR- or EM-rounds (default is 5000). Analyze is set to be converged when only one [PCG](#) iteration round is needed within the NR round, or less than 10 [PCG](#) iteration rounds are needed within the EM round.

There should be insignificant differences in solutions between two algorithms, but the EM algorithm is generally slower to converge than NR algorithm. NR algorithm is critical to attain good solutions within the rounds. Increasing the maximum number of [PCG](#)-iterations within each round may lead to fewer NR rounds and in that way faster convergence finally. Instead, EM algorithm will need reasonable solutions to certain extent within each EM round, after which increase in accuracy will not improve the total convergence.

8.1.3 Solution files

Solution files are equal to the linear mixed model case. When thresholds are estimated, these are printed in the output and in the end of the `Solfix`-file (with factor name Threshold and factor number 0).

8.1.4 Example

[Example 7.9](#) in *Technical reference guide for MiX99 pre-processor* contains the instruction file for bivariate model with one binary trait.

8.2 Gompertz-model

A multiplicative Gompertz model (i.e. $\ln(y) = \ln(a) - b * \exp(-k * t) + e$) as demonstrated in [Vuori et al. \(2006a\)](#); [Vuori et al. \(2006b\)](#) is implemented in MiX99. Thus, log-transformation is needed for original observations y , although solutions are for three parameters a , b and k . All traits introduced in the analysis must currently have the non-linear Gompertz form. Models are allowed to have [missing traits](#) and unequal design matrices for traits. (Co)variance components could be known or estimated simultaneously between each round by the covariance component estimation program which allows linear random regression models.

8.2.1 Instruction file for `mix99i`

Model specification for non-linear model reminds the specification of regression models, although few additional features and restrictions compared to model specification for linear random regression traits exist. The two most important lines in the [instruction file](#) therefore are `MODEL` and `REGRESS` lines which will be discussed below.

Non-linear Gompertz-model traits are defined in the `MODEL` line by giving the character **G** on its own column between first and second parameters. Effects in the `MODEL` lines are defined as many times as number of parameters of the Gompertz function the effect is related to (i.e. 1-3 times for Gompertz function, see [example 7.8](#)). However, for all traits, at least one common fixed effect needs to be related to all three parameters. This effect is defined first in the `MODEL` line, and further, is defined to be across-block effect (see `WITHINBLOCKORDER`). I.e., dash (-) must be specified for the first effect in `WITHINBLOCKORDER` line. If many such effects occur, you may define effect with smallest number of levels first.

All factors in the `MODEL` line must be considered as covariables read from the separate file, i.e., covariable columns have to be specified with a preceding "t" on the `REGRESS` line(s). Real input column for covariable indicates the parameter of the Gompertz function the effect is related to: **t1** for mature weight, **t2** for relative initial weight and **t3** for maturation rate. An additional column for time is needed at the end of each `REGRESS` line. This is not counted in the number of regression effects specified first in the `REGRESS` line.

In consequence, a covariable file is always needed. The index connecting an observation and a set of covariables is usually time of measurement but the rest of the file differs from those to linear random regression models. Because the Gompertz function has three parameters, first three covariable columns should contain only ones. After these, a column with the time variable is set. This can also be a scaled time if needed. For example, select time scaling so that estimate for maturation rate is approximately one. The number of columns needed in covariable table is therefore at least 4, i.e., number of parameters in the non-linear Gompertz function plus one for the time dependent.

Addition to **MODEL** and **REGRESS** lines, restrictions on **PRECON** and **DATASORT** lines should be mentioned: (1) Diagonal (d) must be defined for all **preconditioners** and (2) multiple residual (co)variance matrices are not possible for non-linear Gompertz function models.

Note: The iterative algorithm for non-linear Gompertz function model is implemented so that a restart of analysis is needed. For this reason the user must define the convergence of the iterative process by itself. However, this enables the possible update of new variance components for each iteration round by calling **mix99i** first. Therefore, both options in the **SOLUNF** line are possible, but restart of **mix99s** is possible with option **n** as well in case the covariance components are known and won't change from round to round. The case when estimation of variance components is done simultaneously is covered in the section of stopping criterion file.

8.2.2 Stopping criterion file for **mix99s**

Stopping criterion file is defined as for linear traits. Now the stopping criterion is for BLUP solutions within each round. To decide the final convergence of the iterative process, user must define the convergence by itself over repeated BLUP analysis. One option would be to append the solutions after each round to another file which is studied for converge of solutions.

It is possible to estimate variance components simultaneously between each round by another analysis. This is allowed by the option **d** in **SOLTYP** line, which will produce binary and ascii files named **DMUINP** and **DMUINP.dat**, respectively. There is a historical reason for naming files according to DMU, but any other suitable program can be used also. These files include the pseudo data, i.e., linearized observations and covariables, among others. **Missing observations** and **covariables** in **DMUINP** and **DMUINP.dat** files are coded as -99999. First within these files are integer columns and then real columns. Integer columns contain within and across block fixed effects and random non-genetic and genetic effects, in this order. After these there is the column with zeros. First real columns contain linearized observations for each trait, after which linearized covariables for each three Gompertz function parameters by traits follow. Last real column contain ones. Note: Good knowledge and carefulness is needed to estimate the covariance components accurately.

8.2.3 Solution files

Program gives solutions for Gompertz-model parameters (i.e. a , b and k) defined for each effect. Because model mimics linear random regression model, solutions files are equal to the linear random regression model case.

8.2.4 Example

Example 7.8 in *Technical reference guide for MiX99 pre-processor* contains the instruction file for Gompertz function.

9 Estimation of variance components

For prediction of breeding values, variance components need to be known. An implementation of the Monte Carlo (MC) Expectation Maximization (EM) Restricted Maximum Likelihood (REML) (MC EM REML) algorithm for the estimation of variance components (Matilainen et al., 2012) is available in the `mix99s` solver. The algorithm applies a resampling procedure to estimate prediction error variances (PEV) needed in the EM REML equations. Estimates of location parameters are obtained from the real data within each REML round, whereas PEV is obtained within each REML round by repeatedly simulating data and estimating the location parameters of the simulated data. This enables calculation of PEV without inversion of the coefficient matrix, leading to memory requirements equal to the solving of the mixed model equations. Although EM algorithm is known to be slow in convergence, the MC EM REML makes REML feasible for large data sets and complex models for which the inversion of the coefficient matrix would be too memory and time consuming.

The implementation for the variance component estimation supports the majority of models possible in *MiX99* including GBLUP and SNP-BLUP models. However, single-step models and models that include an effect with an autoregressive correlation structure as well as [threshold models](#) and [Gompertz models](#) are not supported yet.

9.1 Running MC EM REML

Estimation of variance components requires execution of two programs. First, the pre-processor program `mix99i` is executed either by using [CLIM command file](#) or *MiX99* [instruction file](#) with initial values for the variance components. Then, the solver program `mix99s` is executed with specific instructions in the [solver option file](#) (see [VAROPT](#) option line in chapter 4.1).

Because models used in MC EM REML estimation are usually complex and analyses time consuming, it is also possible to [change some of the iteration parameters](#) to be more suitable during the execution of the solver `mix99s`. This can be done by using external file `ITER`. Both the parameters regarding breeding value estimation and the parameters regarding variance component estimation can be updated.

If the estimation is wanted to be stopped beforehand, external `STOP` file will do it in a controlled way.

In case the estimation has stopped or ended, but more REML rounds is needed, the estimation can be continued by starting solver `mix99s` without running the pre-processor `mix99i` first. In that case maximum number of iterations for variance component estimation is marked as negative value in the `STOPE` line of the [solver option file](#). This value is the new total number of REML rounds, i.e., value should be larger than the REML rounds in the previous run. For example, when estimation has stopped on the REML round 1000 but you want to make another 1000 rounds, then the new maximum number of iterations for estimation of variance components is -2000. The estimation, including REML rounds, will start from what they ended (last row in the `REMLlog` file) and estimates from the new REML rounds will be appended to `REMLlog` file as well. Other [PCG](#) and REML iteration parameters can also be changed for the continued estimation.

The variance component estimation can also be restarted by performing another

estimation using the previously estimated variance components as the initial values. The estimated variance components from the `parfile` need to be first copied to the file specified in `PARFILE` line of the instruction file (and possibly `resfile` to `RESFILE`). Remember also to save the old `REMLlog` file to another name to keep old estimates available. Other estimation parameters can be modified at this stage, too. Then, the pre-processor program `mix99i` is run to restart the estimation from the new initial values of the variance components before running the solver `mix99s` for the restarted estimation.

How **MiX99** works to estimate variance components will be explained in more detail in the following.

9.2 File with starting values of (co)variance components

The file with the starting values for the (co)variance components must be in the same format as described in the chapter [File with \(co\)variance components](#) of the *Technical reference guide for MiX99 pre-processor*. The file will be specified in the `PARFILE` instruction line of the `CLIM command file` or `MiX99 instruction file`. The same rules apply also for a file with starting values for the multiple residual (co)variance matrices in the case that a model with multiple residual (co)variances is applied (optional).

9.3 MiX99 instruction file

There is no need to give a specific instruction regarding variance component estimation neither in the `MiX99 instruction file`, nor in the `CLIM command file`. Because pre-processor `mix99i` will at the beginning form `preconditioner` matrices with the initial values for the variance components, updating of the matrices with the most current variance component estimates was found crucial to enhance convergence. In the currently implemented version of the variance estimation module the `mix99s` solver will automatically, in certain `REML` round intervals, make a system call to start a `mix99i` pre-processing run. Therefore `MiX99` pre-processor stores its instructions directives (`.DIR`) and possible command line options (`.OPT`) automatically to files `MiX99_IN.DIR` and `MiX99_IN.OPT`, respectively. So, during the preconditioner update these files need to be available and contain the information from the original pre-processor run. Alternatively, `MiX99 instruction file` named as `MiX99_DIR.DIR` can be used to specify the pre-processor directives for the preconditioner update. This file is automatically created by the pre-processor when a `CLIM command file` is used for the model. File `MiX99_DIR.DIR` contains the same information in `instruction file` format.

Updating of the `preconditioner` matrices is done every 10th `REML` round during the first 100 `REML` rounds and on every 100th `REML` round thereafter. If the updating was unsuccessful, the text

```
Updating of preconditioner failed!
```

is printed to the standard output and further instructions are given to check for additional error information:

```
See files MiX99_DIR.LOG and WARNING.log.
```

If pre-processor directives for the preconditioner update could not be found, this is indicated by error message:

Either MiX99_IN.DIR or MiX99_DIR.DIR file is needed as directive file for mix99i.

9.4 MiX99 solver option file

The solver `mix99s` is instructed to estimate variance components by specifying option `e` on the `VAROPT` option line and by giving information on three additional option lines `STOPE`, `SEED` and `MIX99PATH` thereafter. Specifying the information on three additional option lines is explained in chapter 4.1 regarding the option `e`. If some of the given parameters for the `STOPE` line are found to be poor during the execution of `MC EM REML`, see [Contents](#) for different possibilities to change these.

9.4.1 Number of data samples

The second parameter given on the `STOPE` line is ***number of data samples***. This is a number of data samples generated and analyzed within a one round of stochastic `MC EM REML`. The number of analyzed data samples within a `REML` round will affect the accuracy of the prediction error variance estimates. Increasing the number of samples will reduce the Monte Carlo error associated with the prediction error variances. However, the size of Monte Carlo error depends also on the model specified and on the amount of data and animals included in the analysis.

We observed that convergence of the `MC EM REML` algorithm is not affected by the number of samples specified and for many models even one sample per `REML` round is sufficient. The number of specified data samples is critical, because each additional data sample will require one additional BLUP model to be solved within a `REML` round, which increases the total time of the `REML` analysis.

Our experiences so far suggest that analysis with large amount of data and sufficient number of animals (e.g. test-day data with observation from over 10 000 animals) needs only one Monte Carlo sample per `REML` round. When amount of data is rather small in relation to the number of parameters to be estimated, a higher number of samples (5, 10 or 20) might be more appropriate. For some analyses with small amount of data, software using non Monte Carlo `REML` implementation can be more suitable.

9.4.2 Determining convergence of REML parameter estimates

There is a need for a **convergence indicator** which accounts for the characteristics that parameter estimates are associated with Monte Carlo noise. The currently implemented convergence indicator is calculated from the vectors containing predicted variance component estimates at two points $x - 1$ and x ($\hat{s}(x - 1)$ and $\hat{s}(x)$), where the prediction is based on estimated variance components obtained during the latest x EM rounds ($\hat{\theta}^{(k-x+1)}, \dots, \hat{\theta}^{(k)}$), and where a predicted estimate for each variance parameter is calculated as $\hat{s}_i(x) = \hat{\alpha}_i + \hat{\beta}_i x$. The size of x is chosen to be large enough to minimize the Monte Carlo noise in the **convergence indicator**, which is calculated for `REML` round k :

$$cc_E^{(k)} = \frac{(\hat{s}^{(k)}(x) - \hat{s}^{(k)}(x - 1))^T (\hat{s}^{(k)}(x) - \hat{s}^{(k)}(x - 1))}{(\hat{s}^{(k)}(x))^T (\hat{s}^{(k)}(x))}$$

After $cc_E^{(k)}$ has reached a value smaller than the specified convergence criterion (see `STOPE` option line), the `REML` analysis will perform a sequence of 30 additional `MC EM REML` rounds, which will reduce the Monte-Carlo error from the parameter estimates by using weighted average with decreasing weights for latest solutions. Depending on

the analysis, we have found that values between 10^{-8} to 10^{-9} are suitable convergence limits.

9.4.3 Keeping certain variance components fixed

For some analyses it might be desired that certain pre-defined variance components (starting values) remain unchanged during the [MC EM REML](#) analysis. The instructions about which variance component parameters are kept fixed are given in the [MiX99 solver option file](#).

For this option the three entries **e f n** have to be specified on the [VAROPT](#) option line, where **f** instructs [mix99s](#) to keep some parameters unchanged and the third entry **n** is an integer value which tells how many parameters should remain unchanged. This option will require the inserting of **n** additional lines right after the [VAROPT](#) line. Each line specifies one parameter that should remain unchanged. A line consists of three integers, where the first integer is the random effect number followed by the row-column combination. In practice, you can copy the corresponding line from the parameter file excluding the variance component parameter itself.

In case [multiple residual variance matrices](#) are applied, four integers need to be defined for residual variance component parameters to be unchanged. The first integer number is equal to the random effect number of the residual effect. The second integer gives the residual variance class number. This is equal to the first number on the corresponding parameter line in the file with multiple residual (co)variances (see chapter 3.4 in [Technical reference guide for MiX99 pre-processor](#)). The last two integers specify the row-column combination.

9.4.4 MC EM REML for MACE

As a special case is the estimation of variance components for a MACE model. Variance component estimation of the MACE model can be done by keeping the residual variance fixed at unity and applying weights, $w_{ij} = EDC_{ij}/(\lambda_j \sigma_{g_j}^2)$ with $\lambda_j = (4 - h_j^2)/h_j^2$, for deregressed breeding values for bull *i* in different countries *j* ([Tyrisevä et al., 2011](#)). However, estimation of variance components will change genetic variances $\sigma_{g_j}^2$ that were originally used in the calculation of the weights. Therefore, it will be more accurate to update the weights after new estimates of genetic variances are available. MiX99 can do this updating of weights for the MACE model automatically after each [REML](#) round when option **ei** instead of just **e** is specified for estimation of variance components on the [VAROPT](#) option line.

9.5 Standard errors for REML parameter estimates

MiX99 is capable to calculate standard errors for the variance component estimates at the last REML round. Approximated standard errors are based on variances over sampled gradients as explained for NR-method in [Matilainen et al. \(2013\)](#). They are calculated automatically when the [number of data samples](#) is at least 10. Adequate number of samples depends on the data and model, but 50 or more samples is recommended.

As using number of data samples less than 10, even a single one, is much faster, one can first start the estimation using the smaller number of samples. Then, after the convergence of the variance components, one additional REML round can be executed afterwards. This can be done in different ways. Either the [estimation of variance components can be continued](#) or another variance component estimation can

be started [using the estimated variance components as the initial values](#). In both cases the number of samples can be set to larger than 10, for example 50, in the [solver option file](#) so that the standard errors are calculated.

Approximated standard errors, as well as all covariances of REML estimates in the information matrix, are printed after the last REML round. Standard errors are printed to the `vceSE`, and information matrix is printed to the `vceI`. The `vceSE` has four columns. It resembles the `parfile`, but has approximated standard errors in the fourth column. If model has [multiple residual variance matrices](#), standard errors for all residual classes are printed after the first residual class (like in `REMLlog`). Example about `vceSE` is for one trait with two random effects and residual:

`vceSE:`

1	1	1	208.134
2	1	1	272.508
3	1	1	59.4737

The information matrix `vceI` has seven columns. The first three integers indicate the first parameter and the next three integers indicate the second parameter. The seventh column has the covariance between the two parameters. For example, for the case above `vceI` is

`vceI:`

1	1	1	1	1	1	43319.7
2	1	1	1	1	1	-44735.8
2	1	1	2	1	1	74260.4
3	1	1	1	1	1	-1222.08
3	1	1	2	1	1	485.008
3	1	1	3	1	1	3537.12

where the first line has value for the first random effect, the second line has value between the first and second random effect, the third line has value for the second random effect, and so on. Because the information matrix contains a value for each combination of the (co)variance parameters, it can be become very large.

Consider a model with six (co)variance parameters. The file for standard errors has six lines and is

`vceSE:`

1	1	1	98919.7
1	2	1	3534.95
1	2	2	155.895
2	1	1	82881.3
2	2	1	2914.91
2	2	2	126.006

The information matrix contains 21 elements:

`vceI:`

1	1	1	1	1	1	0.978512E+10
1	2	1	1	1	1	0.310664E+09
1	2	1	1	2	1	0.124959E+08
1	2	2	1	1	1	0.987140E+07
1	2	2	1	2	1	483421.
1	2	2	1	2	2	24303.1
2	1	1	1	1	1	-0.755926E+10

2	1	1	1	2	1	-0.238666E+09
2	1	1	1	2	2	-0.741774E+07
2	1	1	2	1	1	0.686931E+10
2	2	1	1	1	1	-0.236382E+09
2	2	1	1	2	1	-0.937649E+07
2	2	1	1	2	2	-355953.
2	2	1	2	1	1	0.216383E+09
2	2	1	2	2	1	0.849670E+07
2	2	2	1	1	1	-0.737152E+07
2	2	2	1	2	1	-356791.
2	2	2	1	2	2	-17917.6
2	2	2	2	1	1	0.667658E+07
2	2	2	2	2	1	319118.
2	2	2	2	2	2	15877.5

For model with 42 (co)variance parameters, the information matrix contains as much as 903 elements.

9.6 Solution files for variance components

REMLlog Contains the estimates of variance components at every **REML** round. The first column in the file specifies the **REML** round and the second column the convergence criterion value of that round. After the second column as many columns follow as there are variance component parameters to be estimated. The order of the lines is as following. The first three lines in the **REMLlog** file describe the order of the parameter columns. The first line has the random effect number and the second and third lines the row-column combination for the particular parameter of a random effect. Hence, the first three lines are identical with the first three columns in the file with the (co)variance components. If **multiple residual variance matrices** are defined, then their variance class number is added to the end of the file. The fourth line contains the initial parameter values used. The following lines contain the estimates of variance components at each **REML** round.

parfile Contains the latest solutions of variance component estimates. The structure of the file is the same as in the parameter file of the **PARFILE** instruction line.

resfile Contains the latest solutions of residual variance component estimates when **multiple residual variance matrices** are defined. The structure of the file is the same as in the multiple residual variance matrices file of the **RESFILE** instruction line.

vceSE This file will be produced when approximated standard errors for REML parameter estimates are calculated. Resembles **parfile**. First three integers indicates the random effect number and row-column combination of that matrix. Fourth column contains the real value and is approximated standard error. If the model has **multiple residual variance matrices**, these are numbered as in **REMLlog** and printed right after the standard errors for other random effects.

vceI This file will be produced when approximated standard errors for REML parameter estimates are calculated. The file has seven columns. Seventh column has the value of the information matrix for each pairwise parameter

combination. The first parameter is indicated by the first three integers as in `vceSE`, and the second parameter is indicated by the next three integers as in `vceSE`.

9.7 Example

Schaeffer has written a technical note on maximum likelihood estimation of variance components (Journal of Dairy Science, 1976). It contains a small example data in Table 1 for milk yields of first lactation daughters of five dairy sires in two herds. The model contains sire groups and herds as fixed effects and uncorrelated sires as random effects. In the following the `CLIM command file` for the `mix99i` pre-processor and the MiX99 `solver option file` for the `mix99s` solver program, as well as the listing in the standard output of the solver are given.

CLIM command file:

```
TITLE data in technical note of Schaeffer (Table 1, J. Dairy Sci., 1976)

DATAFILE data.dat
INTEGER group herd sire
REAL yield

PARFILE data.par

PEDFILE data.ped
PEDIGREE sire sm

MODEL
yield = group herd sire
```

The “`MiX99_DIR.DIR`” file created for `mix99i`:

```
# TITLE data in technical note of Schaeffer (Table 1, J. Dairy Sci., 1976)
data in technical note of Schaeffer (Table 1, J. Dairy Sci., 1976)
# INTEGER group herd sire
group herd sire
# REAL yield
yield
# TRAITS
1
# TRAITGRP
1 -
# datasort: Block_code, Relationship_code
- -
# FIXRAN: Numbers of fixed and random factors in the model
2 1 0 0
# MODEL: Subgr. Trait Weight ... model factors ...
1 1 - 1 2 3
# WITHINBLOCKORDER: Order of effects within blocks
- - 1
# RANDOM
1
# RELATIONSHIPS
1 1
# REGRESS
3 cl cl cl
# COMBINE
n
# PEDIGREE
sm
# DATAFILE data.dat
data.dat
# VAR
3 1 f
```

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

```
# MISSVA
0
# SCALE
n
# PEDFILE data.ped
data.ped
# PARFILE data.par
data.par
# tmpdir
.
# noransol
y
# SOLUNF
n
# precon
b b # Default is block
# parallel: Number of processors used by the solver program
1 # Default: no parallel computing
# COMMONBLOCKS: Number of blocks in common area for parallel computing
0
```

MiX99 solver option file:

```
# RAM: RAM demand: H=high
H
# STOP: Max.iter, Tolerance, Convergence criterion, Enforce
5000 5.0e-5 d f
# RESID: Residuals calculation
N
# VALID: Model validation
N
# VAROPT: Variance options: VCE
E
# STOPE: REMLrounds, nSamples, Conv.value VCE, Conv.value BLUP samples
1000 5 1.0e-9 5.0e-5
# SEED: Type of seed for the random number generator
R
# MIX99PATH:
/share/apps/
# SOLTYP: Solution file options
Y
```

At the following are three excerpts of standard output of MC EM REML evaluation: at the beginning, when the convergence criterion of MC EM REML is reached, and at the end. Comments after # are added.

Parts of standard output file of the solver:

```
.
.
.

-----
MiX99_SOLVE: Start of MC EM REML iteration      Time: 13:11:00.3  18.11.2019
-----

-----
MiX99_SOLVE: Start of Iteration                  Time: 13:11:00.3  18.11.2019
-----

Iteration Statistics
-----
Convergence Indicators
```

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

```

-----
ROUND      CA      CR      CM      CD      MAX.CHA.
-----
Solution vector will be initialized to be zero
  rhs' * rhs =      14014.6999754906
animal rhs' * rhs =      2345.34000116348
-----
  0      1.000      1.000      1.000      0.000      0.000
  1      0.8156E-01      0.4002E-01      0.4602E-01      1.000      1.675
  2      0.4518E-01      0.2589E-01      0.3458E-01      0.2209      -0.6545
  3      0.3199E-01      0.2202E-01      0.2617E-01      0.5391      -1.081
  4      0.8213E-02      0.9357E-02      0.9692E-02      0.8242E-01      -0.3068
  5      0.1470E-02      0.7315E-03      0.1014E-02      0.3065E-01      -0.8918E-01
  6      0.6997E-03      0.4073E-03      0.4696E-03      0.4586E-02      -0.2069E-01
  7      0.4954E-10      0.2027E-10      0.2669E-10      0.1223E-01      -0.3061E-01
  8      0.4274E-12      0.3967E-12      0.4006E-12      0.8750E-10      -0.2393E-09
  9      0.1452E-16      0.1267E-16      0.1258E-16      0.3625E-12      0.6506E-12

CD convergence criterion 0.5E-4 achieved in 9 iterations.
Solutions have converged according to CD criterion.          # Convergence of real data

-----
MiX99_SOLVE: End of Iteration                                Time: 13:11:00.3  18.11.2019
-----

  9      0.1185E-15      0.9644E-16      0.1095E-15      0.4962E-13      0.2709E-13
  9      0.4008E-15      0.5995E-15      0.5419E-15      0.3275E-14      0.1915E-14
  9      0.2205E-15      0.1849E-15      0.1966E-15      0.2568E-13      0.1618E-13
  9      0.2994E-15      0.3145E-15      0.2893E-15      0.7103E-14      -0.5329E-14
  9      0.7586E-16      0.1449E-15      0.1133E-15      0.1982E-15      -0.3053E-15
# Convergences of five simulated data sets
# REML convergence indicator
REML ROUND      1      CONV Cd      0.000
.
.
.

-----
MiX99_SOLVE: Start of Iteration                                Time: 13:11:02.7  18.11.2019
-----

Iteration Statistics
-----

Convergence Indicators
-----
ROUND      CA      CR      CM      CD      MAX.CHA.
-----
Solution vector will be initialized with old solutions      # Solutions from the
                                                             # previous round used
  rhs' * rhs =      2477845.52138173
animal rhs' * rhs =      414663.904911524
-----
  0      0.1222E-02      0.5000E-03      0.8504E-03      0.000      0.000
  1      0.3017E-04      0.1590E-03      0.1578E-03      0.3780E-02      -0.1741E-01
  2      0.5373E-04      0.4800E-04      0.5064E-04      0.4383E-03      -0.1848E-02
  3      0.3687E-04      0.2310E-04      0.2650E-04      0.9638E-04      -0.3164E-03
  4      0.9305E-04      0.5960E-04      0.6905E-04      0.4135E-03      0.1130E-02
  5      0.8204E-04      0.4137E-04      0.5711E-04      0.8518E-03      0.2677E-02
  6      0.3515E-05      0.2416E-05      0.2596E-05      0.6819E-03      0.2034E-02
  7      0.8988E-05      0.3690E-05      0.5059E-05      0.1978E-03      -0.5310E-03
  8      0.1870E-16      0.1799E-16      0.1791E-16      0.3922E-04      -0.1523E-03
  9      0.1120E-17      0.9135E-18      0.1009E-17      0.1369E-16      0.5551E-16

CD convergence criterion 0.5E-4 achieved in 9 iterations.
Solutions have converged according to CD criterion.          # Convergence of real data

-----
MiX99_SOLVE: End of Iteration                                Time: 13:11:02.7  18.11.2019

```

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

```

9      0.3023E-15    0.2992E-15    0.2675E-15    0.1355E-12    0.3701E-13
9      0.4977E-14    0.7166E-14    0.6355E-14    0.7685E-13    0.4430E-13
9      0.8280E-15    0.1071E-14    0.1022E-14    0.1668E-12    -0.3136E-13
9      0.8386E-14    0.1256E-13    0.1209E-13    0.3144E-13    -0.8604E-14
9      0.6958E-15    0.1165E-14    0.8450E-15    0.5171E-12    -0.2358E-12
# Convergences of five simulated data sets
REML ROUND    321    CONV Cd    0.9572E-09    # REML convergence indicator

```

REML convergence criterion 0.1E-8 achieved on round: 321
 - Additional 30 REML rounds will be performed to reduce variation
 in variance component estimates.

.
.
.

```

MiX99_SOLVE: Start of Iteration                                Time: 13:11:02.9  18.11.2019

```

Iteration Statistics

Convergence Indicators

ROUND	CA	CR	CM	CD	MAX.CHA.
-------	----	----	----	----	----------

```

Solution vector will be initialized with old solutions    # Solutions from the
                                                         # previous round used
      rhs' * rhs =      2691132.89311232
animal rhs' * rhs =      450357.241589269

```

0	0.1545E-03	0.6321E-04	0.1075E-03	0.000	0.000
1	0.1501E-05	0.2075E-04	0.2054E-04	0.4940E-03	-0.2280E-02
2	0.7044E-05	0.5680E-05	0.5891E-05	0.5764E-04	-0.2421E-03
3	0.2437E-05	0.2368E-05	0.2522E-05	0.9361E-05	0.2672E-04
4	0.6524E-05	0.3228E-05	0.4053E-05	0.1273E-04	-0.4024E-04
5	0.8385E-05	0.4114E-05	0.5907E-05	0.7934E-04	0.2502E-03
6	0.3123E-06	0.2126E-06	0.2289E-06	0.1667E-03	0.5088E-03
7	0.1162E-05	0.4770E-06	0.6565E-06	0.2334E-04	-0.6286E-04
8	0.1057E-16	0.1040E-16	0.1045E-16	0.7990E-05	-0.2768E-04
9	0.4813E-18	0.2255E-17	0.2079E-17	0.6230E-17	0.2776E-16

CD convergence criterion 0.5E-4 achieved in 9 iterations.
 Solutions have converged according to CD criterion. # Convergence of real data

```

MiX99_SOLVE: End of Iteration                                Time: 13:11:02.9  18.11.2019

```

```

9      0.1635E-15    0.1646E-15    0.1680E-15    0.2608E-12    -0.8683E-13
9      0.1652E-13    0.1800E-13    0.1810E-13    0.6993E-12    -0.3253E-12
9      0.9189E-15    0.1077E-14    0.1097E-14    0.7176E-12    0.2352E-12
9      0.1516E-14    0.1914E-14    0.1454E-14    0.2583E-13    -0.1557E-13
9      0.9651E-15    0.8771E-15    0.9718E-15    0.5607E-13    -0.2165E-13
# Convergences of five simulated data sets
REML ROUND    351    CONV Cd    0.1705E-07 (321+30/30 rounds) # REML convergence indicator

```

```

MiX99_SOLVE: End of MC EM REML iteration                    Time: 13:11:02.9  18.11.2019

```

REML convergence criterion (0.1E-8) was achieved in 321 rounds
 and additional 30 rounds were performed to reduce variation in variance
 component estimates.

First 10 MC EM REML estimates:

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

1	1	1	0.38841920
1	1	1	0.71836584E-01

.			
.			
.			
.			

10 Accounting for heterogeneous variance

The need to account for heterogeneous variances (HV) in several dairy cattle evaluations led to the implementation of this possibility into **MiX99**. The applied method is based on the multiplicative mixed model (MMM) approach given in [Meuwissen et al. \(1996\)](#). The approach requires to solve simultaneously the model for breeding value estimation ([mean model](#)) and a model for dispersion parameters ([variance model](#)). For our needs, to accommodate multiple trait test-day models, we have modified the approach of [Meuwissen et al. \(1996\)](#). The most important modifications are: scaling factors are standardized to keep the across trait covariance structure unchanged; loss in degrees of freedom due to the estimation of fixed effects in the [mean model](#) is accounted (by an approximation) for the estimation of residual variances within strata; variance components for the [mean model](#) are not re-estimated; and the solving algorithm was modified to allow large test-day models. A description of the implemented method is given in [Lidauer et al. \(2008\)](#).

The implementation was targeted towards test-day models for dairy cattle. This required some restrictions on the variety of [variance models](#) that can be fitted. However, the current implementation should allow accounting for [heterogeneous variance](#) in many other models beside test-day models. However, [non-linear models](#) are not supported.

For future versions of MiX99 we target on a larger variety of [variance models](#) and simplified implementation.

10.1 Computation environment

The current implementation works only for the [parallel](#) solver in **MiX99**. Therefore, a MPI (message passing interface) environment is required on the computing platform. The HV method was implemented having shared memory computer architecture in mind. However, meanwhile we have implemented the HV method also on distributed memory platforms. An example of the required scripts for setting up the HV method on a distributed memory platform (Linux-cluster) may be requested from the authors.

10.2 Models for the heterogeneity of variances

In the [multiplicative mixed model](#) approach two models have to be solved simultaneously, the mean model (the model with the breeding values) and the variance model (the model, which describes the heterogeneity of variances). By definition of the [multiplicative mixed model](#) approach, the [mean model](#) must be nested within the [variance model](#). This is because scaling of observations of a particular stratum will change the mean of these observations, which requires that the [mean model](#) is able to account for this change.

10.2.1 Currently supported *variance models* by MiX99

For large models, like random regression test-day model, solving a multiplicative model is a challenging task. Therefore, some restrictions were necessary on the type of [variance models](#) that can be fitted. The possibility for a larger variety of [variance models](#) would lower computation speed, and therefore, is not implemented yet. For an easier understanding, a short explanation of the computations is given:

Considering a [mean model](#) of the form:

$$y_i \lambda_i = X_i b + Z_i a + e_i$$

where, y_i contains all observations of stratum i , which are scaled with the same adjustment factor λ_i ; and a [variance model](#) of the form:

$$s_{ik} = \beta_{1_{ik}} + \beta_{2_{ik}} + \epsilon_{ik}$$

where $\beta_{1_{ik}}$ is a fixed effect and $\beta_{2_{ik}}$ may be a fixed or a random effect.

Solving of the [MMM](#) of those two models will follow the approach given in [Meuwissen et al. \(1996\)](#) apart from a few modifications. The solving scheme starts with the initialization (I step). In the following, step P, E, M, and A will be cycled until the adjustment factors are sufficiently converged. Finally, step P will be performed until solutions to the [mean model](#) are converged. The steps are:

I step: $q = 0, \lambda^{[q]} = 1, \beta^{[q]} = 0$, and $\sigma_{eT}^{2[q]} = \sigma_{eT_{MM}}^2$

P step: $y_{cT_i}^{[q]} = y_{T_i} \lambda_{T_i}^{[q]}$ and iterate [mean model](#)

E step: $\hat{z}_{T_i}^{[q]} = 0.5 \left[y_{cT_i}^{[q]'} \left(y_{cT_i}^{[q]} - \hat{y}_{cT_i}^{[q]} \right) \sigma_{eT}^{-2[q]} - n_{T_i} \right]$ [3]

$$\hat{w}_{T_i}^{[q]} = 0.25 \sigma_{eT}^{-2[q]} y_{cT_i}^{[q]'} \hat{y}_{cT_i}^{[q]} + 0.5 n_{T_i}$$
 [4]

$$s_{T_i}^{[q]} = \left(\hat{z}_{T_i}^{[q]} / \hat{w}_{T_i}^{[q]} \right) + \beta_{1_{T_i}}^{[q]} + \beta_{2_i}^{[q]} - \beta_{T_{BASE}}^{[q]}$$
 [5]

M step: iterate $[S'W^{[q]}S + \Delta\nu] \beta^{[q+1]} = S'W^{[q]}s^{[q]}$ [6]

A step: $\lambda_{T_i}^{[q+1]} = \exp \left[-0.5 \left(\beta_{1_{T_i}}^{[q+1]} + \beta_{2_i}^{[q+1]} - \beta_{T_{BASE}}^{[q+1]} \right) \right]$ [7]

$$\sigma_{eT}^{2[q+1]} = \sigma_{eT}^{2[q]} \exp \left(\beta_{T_{BASE}}^{[q+1]} \right)$$
 [8]

where q is the adjustment cycle; σ_{eT}^2 is the standardization variance for trait T ; $\sigma_{eT_{MM}}^2$ is the residual variance for the trait T used in the [mean model](#); y_{cT_i} includes the adjusted observations for trait T and stratum i ; $\hat{z}_{T_i}$ is an estimate of the heterogeneity of the residual variance for trait T in stratum i , and $\hat{w}_{T_i}$ is the variance of $\hat{z}_{T_i}$, where $\hat{y}_{cT_i}$ is the prediction of y_{cT_i} , and n_{T_i} is the number of observations for trait T in stratum i ; s_{T_i} resembles the observation for the [variance model](#) related to trait T in stratum i ; $\beta_{T_{BASE}}$ is the weighted mean of the $\beta_{1_{T_i}}$ estimates that built the base for trait T ; s contains all s_{T_i} , β contains all β estimates and S is the corresponding design matrix; W is diagonal with all $\hat{w}_{T_i}$ estimates at the diagonal, and if desired, $\Delta\nu$ may present a variance structure (i.e. autoregressive process) for random effect β_2 .

The current version has the following requirements for the definition of the [variance model](#):

- 1) The [variance model](#) must have the same number of traits as the [mean model](#). In case of a multiple trait model, because of computational reasons, traits are analyzed simultaneously even the traits are uncorrelated.
- 2) If traits are grouped by trait groups, traits must be grouped in the same order in both models.
- 3) Two effects must be specified for each trait.

- ⇒ *Fixed across block strata*: The first effect must be a fixed effect. Usually, this effect will describe heterogeneity of variance between strata, which is common across the whole data; for instance, strata like years, seasons, parities, etc.
- ⇒ *Fixed/random within block strata*: The second effect may be a fixed or a random effect. In case of a fixed effect, please see [MODEL](#) instructions, about how to specify LS-models. In case that the second effect is random, optionally, a [first order autoregressive process](#) for strata within same environments (blocks) is supported. For the latter, a program ([mix99hv](#)) is provided, which sets up the autocorrelation structure.

The previous three requirements for the [variance model](#) should be easy to accommodate. However, there is one more restriction on how the strata for the two effects in the [variance model](#) must be defined. These restrictions are because of computational constraints only. For an easier understanding of this restriction, it is worth to know that [mix99p](#) performs all calculations of step E for one data block at the time. For the need of fast computations all required information is stored in vectors, rather than in linked lists. To avoid exhaustion of computer memory only the segment of the strata that relates to one data block is made accessibly during processing of a particular data block. This leads to the following two restrictions:

- 1) The second effect (β_2) must include a block interaction regardless whether it is defined as fixed or as random. The block structure must correspond with the block structure of the [mean model](#). Note: The blocking variable of the [mean model](#) will be included as interaction in the second effect of the [variance model](#) in the form block $\times$ within-block classes (see [10.3.1](#)). Therefore, it is important that the blocking variable in the [mean model](#) is the same for all observations with the same production environment. For instance in dairy cattle, the blocking variable may be herd of production, but it must not be herd of birth, if [heterogeneous variance](#) due to different production environments should be adjusted. Calculations are performed in block-wise manner, where a data block in the [mean model](#) has a corresponding data block in the [variance model](#). This requires, that each stratum of the second effect can only be present in one block.
- 2) Strata of the first effect (β_1) can be associated with data of different blocks. However, because of computational restrictions, within each data block a strata of the first effect (β_1) can only be associated with one strata of the second effect (β_2). Thus, the levels of the first effects must be nested within the levels of the second effect. (This is because within each block only one pointer to the second effect strata is stored for each first effect strata. The restriction could be relaxed if a pointer to the second effect strata would be stored for each observation. However, this would be currently computationally too expensive). Different strata of the first effect can be associated with the same stratum of the second effect. For instance, if the first effect in the [variance model](#) is a *region $\times$ year $\times$ season* effect, then the second effect may be a *herd* effect, a *herd $\times$ year* effect, or a *herd $\times$ year $\times$ season* effect. However, if the first effect is a *year* effect only, then the second effect may not be a *herd $\times$ year $\times$ season* effect.

10.3 Input data for the variance model

The program `mix99hv` sets up the required input data files for the [variance model](#). In order to get first heterogeneity observations, a few iterations have to be made on the [mean model](#) prior to the `mix99hv` run.

10.3.1 Input data for `mix99hv`

Three integer columns of the input data file for the [mean model](#) are read by `mix99hv` when setting up the data file for the [variance model](#). These three columns are: the column for the blocking variable, a column with the strata of the first effect of the [variance model](#), and a column with the [within block strata](#) for the second effect of the [variance model](#). The blocking variable must be consistent with the production environment in which the observations were made. For instance, herd of production but not herd of birth. This is important because `mix99hv` uses the blocking variable for coding the second effect of the [variance model](#). For the same reason, the third column must contain a within-block stratification only (without block interaction).

Both columns, that one with the blocking variable and the other one with the within-block stratification, are used to set up information for the first order autoregressive process ([Wade and Quaas, 1993](#)) and to set up the strata for the second effect in the [variance model](#). To accommodate the autocorrelation structure, coding of the within-block classes must be consistent with the distances between classes. When coding starts from n and goes up to k , then $k - n$ is equal to the largest possible distance between classes of the block. Usually, classes are defined by the time when observations were recorded, e.g., year, month, week. Class codes have to be defined that the distance between consecutive classes is one, i.e., $\text{class_code } n+1 - \text{class_code } n = 1$. In case, there is a missing class between consecutive classes, the distance must be 2. It is advisable to number all possible classes consecutively from 1 to k and then give for each class the corresponding code. This way, `mix99hv` will automatically calculate the right distances between classes.

If the second effect in the [variance model](#) is considered as fixed effect or as a normal random effect, coding of the [within block strata](#) can be relaxed. However, in order to save memory, we recommend that within block coding should be consecutive and begin from one. Remember, that the actual class code for the second effect will be formed by the program `mix99hv` from the block sorting variable and the within block class code.

10.3.2 Instruction file for `mix99hv`

Execution of `mix99hv` requires an instruction file, which is read by standard input. Editing rules for this file are the same as for the MiX99 [instruction file](#). The file contains several instruction lines, which have to be in the same order as given in the following. The file can contain as many comment lines as wanted.

DATAFILE The name of the input data file for the [mean model](#). If the data file is in a different directory, the whole path must be specified.

VAR_I One line with one integer and one character. The integer is the number of integer columns in the data file. The character is a code for the type of the file.

- f** Formatted free format (columns are separated by at least one space).
- u** Unformatted format, i.e., a binary file.

TRGRP	One integer value that indicates the integer column with the trait group code in the data file. If trait groups are not needed the number must be set to zero (0).
SORT_B	One integer, which indicates the integer data column of the block sorting variable in the input data file for the mean model . The variables in this column will be used to set up the autocorrelation structure and to form the class codes for the second effect in the variance model (see also 10.3.1).
MODVAR	One line with two integers. The first integer indicates the integer data column with the class variables of the first effect in the variance model (fixed across block strata). The second integer indicates the integer data column with the within-block class variables for the second effect in the variance model (within block strata).
ARFILE	A line with one character. This character defines, whether a file (AR.pedi.reml) with information about the autocorrelation structure is created. This file is only useful for the estimation of variance components for the variance model, using our tailored software, which is not included in MiX99. n No. No file created. y Yes. File created.
STAND	A line with one character. This character defines, whether or not standardization variances (provided in the file HVbase_ResVar, for the calculation of the heterogeneity observations), should be scaled to result standardized adjustment factors, which are on average 1.0 for observations of the defined base strata. For more details, see Contents. The Standardization process has to be done only once for a specific model. The obtained scaled standardization variances can then be used as parameters for future analyses. y Yes. Standardization of multiplicative adjustment factors. n No. No standardization processes. BASEFILE If y is defined, an additional line with the name of the file, that contains the class codes for the base classes must be given (see Contents).
APPROX	A line with one character for each trait. This character defines, whether or not rank approximation, to account for loss in degrees of freedom due to the estimation of fixed effects in the mean model , should be considered for the estimation of within strata residual variances. This option was included for the need to avoid an underestimation of within strata residual variances when size of strata is small. This approximation can only be used if within block fixed effect classes in the mean model match with the strata of the second effect in the variance model . For instance, a fixed <i>herd</i> × <i>year</i> effect in the mean model and <i>herd</i> × <i>year</i> strata for the second effect of the variance model . Otherwise, the loss in degrees of freedom will be underestimated.

- y** Yes. Consider rank approximation.
n No. Do not consider rank approximation.

SETLOSS One line with as many real values as there are traits in the model. This line is only given if **d** is specified at the end of the previous **APPROX** instruction line. Then, for each trait, for which rank approximation is not applied, a preset loss in degrees of freedom may be given. The value must be between 0.0 and 1.0.

MERGE2 One line with as many integers as there are traits in the model. The integers tell whether strata of the second effect in the **variance model** are combined across traits. For the specification of the integers, the same rules apply as for one column in the **MERGE** instruction for **mix99i**. The specified integers must be consistent with the **MERGE** specification in the **variance model** instruction file. If combining is not desired integers from 1 up to n are given, where n is equal to number of traits.

TMPHV The directory where the HV information files **HV.data**, **HV.pedi**, **Lambda.data(i)**, **HV.info(i)**, and optional **ARsiwi.data(i)** will be created. These files are rather large and will be created during the execution of the script **runMMM**. If the HV information files should be in the same directory where the script is executed, a dot (.) is given.

10.3.3 Output files from **mix99hv**

The program **mix99hv** creates a file named **HV.data**, which is the input data file for the **variance model**; a file named **HV.pedi**, which contains information about the autocorrelation structure; the files named **AR.block**, and **Hetlog**, which are read by **mix99i** and **mix99p**; as well as the files **HV.info(i)**, which are needed during updating of the adjustment factors.

HV.data

The file **HV.data** is created by **mix99hv** and is the input data file for the **variance model**. The file is in unformatted (binary) format and contains integer and real data columns. First there are four integer data columns, followed by the real data columns, where the number of real data columns depends on the specified model.

Integer data column	Description
1	renumbered block sorting variable
2	renumbered class variables for the second effect
3	trait group code
4	renumbered class variables for the first effect

There are two groups of real data columns. The first group contains heterogeneity observations and the second group the corresponding weights for the heterogeneity observations. Each group has as many columns as there are number of traits in the largest trait group. E.g., if the first trait group includes two traits and the second trait group three traits, then there will be three columns for the heterogeneity observations and three columns for the weights. The order of the heterogeneity observations and of the weights follows the order of the traits within trait group. If there are fewer traits in a trait group, remaining columns are set to missing. If an observation is missing or if a column is not used for a certain trait group, a code for [missing values](#) will be given. This code is set to **-8192.0** and has to be given in the [instruction file](#) for the [variance model](#).

HV.pedi

The file **HV.pedi** is created by **mix99hv** as well. The file is needed to provide the blocking information for the [variance model](#) and to set up the [autocorrelation structure](#) for second effect in the [variance model](#), if latter is desired. For the [variance model](#), the **HV.pedi** file replaces the normal pedigree file. The file has four integer columns:

Integer column	Description
1	renumbered class variables for the second effect
2	distance to the next classes
3	information about class position within block (1=first class in block; 2=class between; 0=last class in block; 3=first and last class in block)
4	renumbered block sorting variable

In case it is desired to define the second effect as a fixed effect ([LS-models](#)), information on column 2 and 3 will be not used. If the second effect should be a normal random effect, the autocorrelation parameters on the **RHO** instruction line should be set to zero (0.0) in the [instruction file](#) for the [variance model](#). This makes information on 2 and 3 redundant. Alternatively, it is possible to define instead of **ar** (autoregressive model) **am** (animal model) in the instruction file for the random model and set the values in column 2 and 3 to zero.

10.4 Instruction file for the variance model

Setting up the instruction file for the [variance model](#) follows the same rules as given for the [mean model](#). However, a large part of the instructions are already defined by the general setup for the HV-adjustment. These pre-defined instructions are as following:

TITLE

INT-VAR Block BlockxCi Trg Fix

TECHNICAL REFERENCE GUIDE FOR MIX99 SOLVER

REAL-VAR	Name of traits followed by weights for the traits. Names are in same order as given in the mean model .
TRAITS	Number of traits must be the same as in the mean model .
TRGRP	Number of trait groups must be the same as in the mean model . Integer input column for the trait group code is 3.
SORT_R	1 2 The value 1 for the integer column with block code, followed by the value 2 for the integer input column of the second effect in the variance model .
FIXRAN	1 1 Two integers with the value 1, for the number of fixed and random factors.
MODEL	As many model lines as there are traits in the model. <u>Ordering of the traits within trait groups must be the same as in the mean model</u> . For the variance model , all observations are weighted (see Meuwissen et al. (1996)). This requires defining of the real columns with the weights. In HV.data , the real columns with the weights start after the columns with the observations and have the same order as the columns for the observations. Note, if trait groups are defined, for any trait group, the column with the observations for the first trait within trait group is always the first real column. For each trait, the corresponding column with the weights is on position “size of largest trait group plus trait column”. For the first effect in the variance model column 4 has to be defined, for the second effect column 2 (see Example 10.9).
WITHINBLOCKORDER	- 1 A dash (-) is given followed by the integer value 1.
RANDOM	The second effect in the model has to be defined always as random (also for a LS-model ; see defining of LS-models).
RELATIONSHIPS	1 1 Two integers with the value of 1.
REGRESS	Only class variables (<code>c1</code>) are allowed.
COMBINE	(<code>n/y</code>). In case <code>y</code> is given, then last column of the MERGE instruction lines must be consistent with the MERGE2 line in the instruction file for mix99hv .
PEDIGREE	ar for autoregressive model , am for random effect, ls for LS-model .
RHO	In case ar is defined in PEDIGREE , one line with as many autocorrelation values as there are traits in the model must be given. Order of the values must be the same as the order of traits. In case some traits are combined, the order of the autocorrelations must follow the order of the corresponding variances that are applied for the last random effect.
DATAFILE	The HV.data file is placed in the same directory as was defined for the work files in the instruction file for mix99hv : <code>/path for the work</code>

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

`files/HV.data.`

- VAR** An integer value of 4 for the number of integer columns in `HV.data` followed by the number of real columns in `HV.data` and the character **u** for unformatted file. The number of real columns is two times the size of the largest trait group.
- MISSVA** `-8192.0`
The code for missing heterogeneity observations is `-8192.0`.
- SCALE** `n`
A **n** for no scaling.
- PEDFILE** The `HV.pedi` file is placed in the same directory as was defined for the work files in the `instruction file` for `mix99hv`: `/path for the work files/HV.pedi`.
- PARFILE** The name of the file with the variance components for the `variance model`.
- TMPDIR** The directory where the temporary files for the `variance model` are placed. The directory must be a sub-directory of the directory with the temporary files for the `mean model` and must be named `B1`: `/path for the mean model temporary files/B1`.
- RANSOLFILE** `y`
A **y** for the second effect in the model. In case of a `LS-model` a **y** must be given.
- SOLUNF** `n`
A **n** for no unformatted solution files.
- PRECON** `d d`
A **d** for the WpW and **d** for the XpX part, since all traits are uncorrelated.
- PARALLEL** Same number of processes as for the `mean model`.
- COMMONBLOCKS** `0`
An integer value of 0, because there is no common pedigree block in the model.

From release XII/2014 onwards the variance model can be given also with CLIM syntax. The following illustrates CLIM syntax needed to build variance model.

Variance model CLIM:

```
TITLE Variance Model

DATAFILE BINARY path/to/HV.data
INTEGER BLCK HERDxYR TRGRP MONTH
REAL HET_OBS WGT

MISSING -8192.0
TRAITGROUP TRGRP

DATASORT BLOCK=BLCK PEDIGREECODE=HERDxYR

PEDFILE path/to/HV.pedi
PEDIGREE HERDxYR ar
```

```

PARFILE variance_model.var
AR 0.7 # If pedigree type is ar then autocorrelation values
      # for each trait must be given in AR statement

MODEL # As many model lines as there are traits in the model
      # Ordering of traits must follow mean model!
HET_OBS(1) = MONTH HERDxYR ! WEIGHT=WGHT
      # The first effect of the model is fourth integer column
      # and the second effect is second integer column

PRECON d d
PARALLEL 2 0
WITHINBLOCKORDER HERDxYR
    
```

10.5 Variance components for the variance model

In case the second effect of the [variance model](#) is defined as a random effect, variance components are required. Because variance components depend on the specified model, only a brief explanation about how the variance components for the [variance model](#) are estimated is provided. Generally, the estimation procedure requires first solving the [multiplicative mixed model](#) followed by estimation of the variance components for the [variance model](#). The new variance components are updated in the [multiplicative mixed model](#) and the whole process is repeated until variance components no longer change. **MiX99** authors will provide support for estimating VC for multiplicative mixed models.

At the moment the **MiX99** variance component estimation module does not support models with an auto-regressive correlation structure.

10.5.1 Files with information for the variance component estimation

For the estimation of the variance components, a file with the heterogeneity observations for the [variance model](#) is needed. **MiX99** provides the files named `ARsiwi.data(i)`, which contain the most recent heterogeneity observations plus the corresponding weights. The file is an unformatted file with real columns only. The order of the columns is the same as the order of the real columns in the `HV.data` file. The integer columns of the `HV.data` file and the real columns of the `ARsiwi.data(i)` files may be merged to create an input data file for the variance component estimation. In order to obtain the `ARsiwi.data(i)` files, an `h` must be specified in the `RESID` option line of the **MiX99 solver option file** for the [mean model](#) (`Mean_model.slvM` in the HV directory).

In case, an autocorrelation structure is applied for the second effect in the [variance model](#), **MiX99** may provide two different files with the required information to set up the autocorrelation structure. The first is named `HV.pedi`, which contains the required information as needed for **MiX99**. The second is named `AR.pedi.reml`, and is created if a `y` is specified in the `ARFILE` instruction line of the [instruction file](#) for `mix99hv` (see [10.3.2](#)). The `AR.pedi.reml` file contains the following five integer columns:

Integer column	Description
1	renumbered class variables for the random effect (within block strata)
2	renumbered class variable for the next class of the random effect; this variable is zero (0) in case the class in the first column is the first or the last class of a block.

3	renumbered class variable for the next class of the random effect, in case the class in the first column is the first class of a block; and otherwise zero (0).
4	a column with zeros
5	a column with zeros

10.6 Standardization of the multiplicative adjustment factors

In the method presented by [Meuwissen et al. \(1996\)](#), variance components for the [mean model](#) are re-estimated, because accounting for HV has some effect on the heritability. However, re-estimation of variance components is hardly feasible for large models. Further, even it would be possible to extend the approach of [Meuwissen et al. \(1996\)](#) to multiple trait models, it would be computationally demanding. Therefore, in the current implementation, heterogeneity observations of different traits are considered to be uncorrelated in the [variance model](#). For all these reasons, a standardization procedure was implemented, which should assure that the covariance structure across traits remains unchanged.

The idea of the standardization procedure is to find for each trait a standardization variance $var(s)$, for which the quantity “ $[(y_i' e_i \lambda_i^2) / var(s)] - n_i$ ” becomes zero and the adjustment factors for observations in the base classes are on average 1.0; y_i and e_i are of size n_i and contain observations and residuals of stratum i , respectively; λ_i is the adjustment factor for stratum i . In a perfect world, the standardization variance $var(s)$ would become same as the residual variance of the [mean model](#). The standardization variances have to be provided in a file named `HVbase_ResVar`. The file with converged standardization variances must be provided for any future (routine) runs.

10.6.1 Files for the standardization procedure

File with class codes of the base

The standardization base should include a part of the data for which it is reasonable to assume that standardization factors should be on average 1.0 for all traits. This part of the data must include observation from all traits (if this is not possible see [Contents](#)). Corresponding strata of the first fixed effect in the [variance model](#) need to be provided in the base class file. The file will be renumbered by `mix99hv` and original and renumbered class codes are written to `ID.Base`. The provided file may contain additional columns. The file `ID.Base` is repeatedly read by `mix99p` during the standardization process.

File with the residual variances

A file named `HVbase_ResVar` with as many rows as there are traits in the model. Each row corresponds to the trait in the model and contains the standardization variance. During the standardization process, the file will be updated after each adjustment cycle. The residual variances of the [mean model](#) can be given as starting values for the standardization process. The `HVbase_ResVar` file must be provided in the directory where `mix99p` is executed any time when adjustment for HV is considered.

10.6.2 Standardization process

The standardization process is specified in the [instruction file](#) for `mix99hv`. A `y` must be specified in the `STAND` instruction line, and on the `BASEFILE` line the filename of the

file with the base class codes must be provided. To ensure converged standardization variances, we recommended a minimum of 150 adjustment cycles. Cycle to cycle changes in the standardization variances can be checked from the standard output file. The line marked with “Re:” contains the currently used standardization variances.

The provided standardization method might not work for very complicated models. For instance, if not all traits have observations in the same time period or in some cases where genetic effects are combined over traits. For such situations it is possible to keep the variance ratios between standardization variances of different traits unchanged. Providing the file `HVbase_ResVarLEVEL` will allow this task. The file has the same form as `HVbase_ResVar` but additionally a second column with integer values is included. The first column contains standardization variances and the integers in the second column specify which standardization variances should be changed and which should have a fixed ratio to another standardization variance. If the standardization variance is allowed to change, a trait specific number is given. If the standardization variance should have a fixed ratio with the standardization variance of another trait, then trait number of the other trait is given. For instance, if the standardization variance for trait two should change and the ratio between standardization variance for trait four and trait two should be fixed, then a 2 is specified for trait two and also for trait four.

10.6.3 Multiple residual variance-covariance matrices

Multiple residual variance-covariance matrices (see [Multiple residual \(co\)variances \(RESFILE\)](#) in *Technical reference guide for MiX99 pre-processor*) for the [mean model](#) are automatically considered for the calculation of observations for the [variance model](#). However, for the standardization process one needs to be aware that provided standardization variances in the `HVbase_ResVar` file correspond to the residual variance-covariance matrix of the first residual variance class in the [mean model](#). The base classes need to contain observations that go into the first residual variance class. During the standardization process, standardization variances for all other residual variance classes are automatically changed if the standardization variances for the first residual variance class are changed. The ratios between standardization variances of different residual variance classes are kept constant.

If it is desired that ratios between standardization variances of different residual variance classes are different from those between residual variances of different residual variance classes in the [mean model](#), a file named `HVbase_ResVarCLASS` can be provided. The file has as many rows as there are multiple residual variance classes and as many columns as there are traits in the model. Each row should contain the corresponding residual variances. For classes where higher or lower standardization variances are desired, the variances should be changed accordingly.

10.7 Running a model with heterogeneous variance

The applied [multiplicative mixed model](#) approach, to account for [heterogeneous variance](#), requires a simultaneous solving of two linear models, namely the [mean model](#) and the [variance model](#). Both models are solved by two different MiX99 runs, which interact with each other. To facilitate these interactions between the models a certain directory structure is required when setting up the models.

The `heterogeneous_variance` directory of the MiX99 package includes all required to set up to run a model with [heterogeneous variances](#). For distributed memory

platforms, implementations are more complicated, and we recommend a good understanding of the implementation for shared memory platforms before an implementation is done for a distributed memory platform.

10.7.1 Implementation on shared memory platforms

The [mean model](#) must be set up in an own directory. The [variance model](#) must be set up in a sub-directory of the [mean model](#) directory and must be named [B1](#). The directory with the temporary files for the [mean model](#) must be specified in the [instruction file](#) for the [mean model](#). The temporary files for the [variance model](#) must be in a sub-directory of the [mean model](#) temporary file directory and it must be named [B1](#). Additionally a third directory with [heterogeneous variance](#) information must be specified in the instruction file for [mix99hv](#). The input data files for the [variance model](#) will be placed into this directory. For large model, these three directories may require considerable amount of disk space.

The whole solving process is controlled by the script `runMMM` provided with MiX99 package. The script controls all required **MiX99** runs for solving the [multiplicative mixed model](#). The only required modifications in the script are to set the parameter for the number of processes ([PARALLEL](#)) and the calls to `mpirun`. The program `mix99p` calls the script `solve_HVmodel` during iteration process to start solving of the variance model. Output from this script is written to `CYC.log`

Three different solver option files are needed for multiplicative mixed model. The files in example provided are named as `Mean_model_init.slvI`, which is used to initialize mean model, solver option file `Variance_model.slvV` (in directory [B1](#)) for the variance model, and `Mean_model.slvM` for running the mean model. The convergence criterion for the adjustment procedure is given in the [STOPC](#) instruction line of the stopping criterion file `Mean_model.slvM`. From our experiences, a stopping criterion of $1.0e-7$ for the relative change between adjustment factors of consecutive cycles or at least 80 adjustment cycles was found useful. However, a less conservative criterion may yield already sufficiently converged adjustment factors.

10.8 Workflow and needed files for running multiplicative mixed model

The following table shows all necessary files and directories needed to set up and run multiplicative mixed effects model. The filenames are the same as in HV example shipped with **MiX99**. Files that have to be named with a certain name are marked with a star (*).

Directory and file setup for multiplicative mixed model	
Directories within mean model directory	
*B1	Directory for variance model
tmpMiX	Directory for temporary files (Mean model)
*tmpMiX/B1	Directory for temporary files (Variance model). Should be a subfolder of mean model temporary files
tmpHV	Directory for HV information files (see TMPHV)
Files	

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

<code>runMMM</code>	Script that does all steps
*B1/solve_variance_model	Script that runs variance model. Created by <code>runMMM</code> script.
* <code>solve_HVmodel</code>	Script called from <code>mix99p</code> to run variance model. An example script is shipped with MiX99 . Script runs B1/solve_variance_model
Files for Mean model	
mean_model.[clm mix]	Definition for the mean model
mean_model.dat	Data
mean_model.ped	Pedigree
mean_model.var	Variance components for mean model
mean_model.resvar	Optional: Variance components for residuals if more than one residual class
mean_model_init.slv	Solver option file to initialize the mean model
mean_model.slvM	Solver option file for the mean model
Files for variance model	
hvdata.dir	Instruction file for mix99hv (see section 10.3.2)
B1/variance_model.[clm mix]	Model definition for the variance model (see section 10.4)
B1/variance_model.slvV	Solver option file to variance model
B1/variance_model.var	Variance components for the variance model
*B1/HVbase_ResVar	Standardization variances for traits
*B1/HVbase_ResVarCLASS	Optional: Ratios of standardization variances between different residual classes
Additional files for standardization of adjustment factors	
HVbaseclasses	Standardization base (see section 10.6.1)
*HVbase_ResVarLEVEL	Optional: Control standardization variance estimation for traits
*Name can not be changed	

Solving multiplicative mixed effect model requires several steps. To help users we provide an example script `runMMM` located in examples/heterogeneous_variance directory that can be used as a starting point when solving larger models with HV correction. Workflow of running MMM is presented in the following table. Input files are the same as in previous table.

Workflow for running MMM	
In mean model directory	
1 <code>mix99i</code> mean_model.clim	Initialize mean model
2 <code>imake99</code>	Additional pre-processing program mean model

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

3	<code>mix99p < mean_model_init.slv</code>	Run couple of iterations of mean model to get initial values for variance model
4	<code>mix99hv < hvdata.dir</code>	Create data for variance model
In B1 directory		
5	<code>mix99i variance_model.clim</code>	Initialize variance model
6	<code>imake99</code>	Additional pre-processing program for variance model
7	<code>mix99p < variance_model.slvV</code>	Solve variance model
In mean model directory		
8	<code>mix99p < mean_model.slvV</code>	Solve MMM. Mean model solver calls script <code>solve_HVmodel</code> . That runs step 7 in each cycle and updates lambda values.

In addition to files created by standard MiX99 solver the HV solving process creates files to exchange information between mean and variance model and provide information to user. The following table lists the most important files. Filenames containing standard output from MiX99 software follows the script `runMMM` shipped with MiX99 package.

Files created by MiX99hv

<code>tmpHV/HV.data</code>	Data for variance model
<code>tmpHV/HV.pedi</code>	Pedigree file for variance model

Optional files created by mean model solver

<code>ARsiwi.data(i)</code>	Heterogeneity observations and weights for variance model. Used when estimating VC for variance model. See section (10.5.1)
<code>AR.pedi.reml</code>	Information about AR correlation structure

Log files created by `runMMM`

<code>Mi.log</code>	Mean model initialization (<code>mix99i</code>)
<code>imake99.log</code>	Mean model <code>imake99</code>
<code>Ma.log</code>	Log for first iterations on mean model (<code>mix99p</code>)
<code>HVd.log</code>	Log from variance model data creation (<code>mix99hv</code>)
<code>B1/Vi.log</code>	Variance model initialization (<code>mix99i</code>)
<code>B1/imake99.log</code>	Variance model <code>imake99</code>
<code>B1/Vs.log</code>	Variance model solving. All cycles. (<code>mix99p</code>)
<code>Ms.log</code>	Log for solving multiple mixed model

Log files created by MiX99

<code>CYC.log</code>	Output from <code>solve_HVmodel</code> . (see section 10.10.3)
<code>Lambda.log</code>	Lambda values. (see section 10.10.4)

10.9 Example

Accounting for [heterogeneous variance](#) in a small test-day model. This example is also shipped with MiX99 package and can be run with script [runMMM](#). The script carefully verifies that all the steps are performed without errors.

[Mean model](#) is a random regression model. [Variance model](#) includes fixed month strata and autocorrelation structure is applied between consecutive years within herd.

Instruction file for mean model (mean_model.clm):

```
TITLE Heterogeneous variance adjustment by Meuwissen approach: a demonstration

DATAFILE mean_model.dat
INTEGER Month Animal Herd HerdYear Year
REAL Covar_1 Covar_2 Milk
DATASORT BLOCK=Herd PEDIGREECODE=Animal

PEDFILE mean_model.ped
PEDIGREE G am

PARFILE mean_model.var

TMPDIR tmpMiX
MISSING 0.0

MODEL
Milk = Covar_1 Covar_2 Month HerdYear G( 1 Covar_1 Covar_2 | Animal )

WITHINBLOCKORDER G HerdYear

PRECON d d b
PARALLEL 2 1
```

Instruction file for mix99hv hvdata.dir :

```
# Instruction file for mix99hv
# This is needed to allow mix99hv to create a data and pedigree file
# for the variance model
#-----
#
# Name of the data file
mean_model.dat
# Number of integer columns; type of file (f/u)
5 f
# Column with the trait group code
0
# Column with the block variable for the auto regress. proc.
# (HERD)
3
# Heterogeneity Model:
# Fixed across block strata; random within block strata
# (Month) (Year within Herd)
1 5
# Create pedigree file for REML-analysis? (y/n)
n
# Standardisation of multiplicative adjustment factors? (y/n)
n
# Name of file with the class codes of the base
# ID.Base.Classes
# Approximation of rank across traits (y/n)
y
# Random HY effect: Combining of second and third lactation in the variance model
1
# Directory for the temporary files
./tmpHV
```

Instruction file for the variance model (variance_model.clm):

```
TITLE Variance Model

DATAFILE BINARY ../tmpHV/HV.data
INTEGER BLCK HERDxYR TRGRP MONTH
REAL HET_OBS WGHT
MISSING -8192.0
TRAITGROUP TRGRP

DATASORT BLOCK=BLCK PEDIGREECODE=HERDxYR

PEDFILE ../tmpHV/HV.pedi
PEDIGREE HERDxYR ar

PARFILE variance_model.var
AR 0.7

MODEL
HET_OBS(1) = MONTH HERDxYR ! WEIGHT=WGHT

PRECON d d
PARALLEL 2 0
WITHINBLOCKORDER HERDxYR
TMPDIR ../tmpMiX/B1
```

File with standardization variances HVbase_ResVar:

1.089320

MiX99 solver option file mean_model_init.slvI:

```
# RAM      RAM demand: H=high
           H
# STOP      Max.iter, Tolerance (CR), Convergence criterion, Force
           20      1.0e-7      R      F
# RESID      Residuals calculation
           N
# VALID      Model validation
           N
# VAROPT      Variance options: HV (Start)
           S
# SOLTYP      Solution file options: No solution files
           N
```

MiX99 solver option file variance_model.slvV:

```
# RAM      RAM demand: H=high
           H
# STOP      Max.iter, Tolerance (CR), Convergence criterion, Force
           25      1.0e-7      R      F
# RESID      Residuals calculation
           N
# VALID      Model validation
           N
# VAROPT      Variance options
           N
# SOLTYP      Solution file options: Half-Chebyshev-accelerated solutions
           H
```

MiX99 solver option file mean_model.slvM:

```
# RAM      RAM demand: H=high
           H
# STOP      Max.iter, Tolerance (CR), Convergence criterion, Force
           25      1.0e-7      R      F
```

```
# RESID   Residuals calculation
N
# VALID   Model validation
N
# VAROPT   Variance options: HV (Continue)
C
# STOPC   Max.HVcycles, Conv.value for lambda effects (Cd^2)
80      1.0e-7
# SOLTYP  Solution file options
Y
```

File with variance components for the variance model *variance_model.var*:

1	1	1	0.078485
2	1	1	0.590860

10.10 Output files

There are several sources to follow the progress of the [heterogeneous variance adjustment](#) process: information written to [standard output](#), [HVD.log](#), [CYC.log](#) and [Lambda.log](#).

10.10.1 Mi.log and Ms.log

Standard output from [mix99i](#) and [mix99p](#) are written to the files [I.log](#) and [S.log](#) for the *mean model*, respectively, and for the [variance model](#) it is written to [Vi.log](#), [Vs.log](#), respectively. Each cycle, [mix99p](#) output from the [variance model](#) will be appended to [Vs.log](#).

[Standard output](#) from the main script [runMMM](#) contains general information about the progress of the HV adjustment and is explained in more detail. The output informs about the current stage of calculation: INITIALIZATION, START ADJUSTMENT CYCLES, CYCLE END, FINAL ITERATION ON MEAN MODEL.

During each cycle the following information is provided (for a simple single trait model):

```
.
Solve Variance-Model
  Last HV cycle:      2
  Lambda criterion CD = 4.59369973910834D-006
8.85770073643941D-004

                                # convergence of the adjustment factors
                                # for lambda values of first and second
                                # effect in the variance model
      rhs' M^-1 * rhs = 318.854161366479
-----
      8   0.6804E-02   0.000   0.000   0.000
      9   0.3163E-02   0.000   0.000   0.000
     10   0.3698E-02   0.000   0.000   0.000
     11   0.3698E-02   0.000   0.000   0.000

                                # iteration on mean model
                                # Standardization variance
Re:      56.1916
Residual class: 1
NStrata      22.0000      # number of strata for each trait
zHatMean     0.3248      # average absolute value of z(z is minimized)
wHatMean     2.4492      # average weight associate with z's
.
```

10.10.2 HVd.log

This output provides information about the data and model structure for the [variance model](#).

For instance:

```
Number of Blocks in the Data:      57949
Largest Class in Blocks found:      21
Largest Class in Fixed Strata:      675
Shift variable:                    100
Largest Random Stratum Code:      5794921
```

For routine evaluations it might give a large amount of warnings of the following form:

```
Block without si and wi values: renum.block code=      3771
Current number of blocks:          3713
```

This means that the block (herd) has not enough information to calculate residual variances for the strata in the block. For instance if the strata are herd test-days and there is only one cow in a herd.

For such a block a second warning will be written:

```
Warning!!! Block without HV-obs. in AR classes.
Block=      3713  ignored.
```

For such a case the adjustment factors will be set to 1.0. If the number of such block is rather large one might consider a different [variance model](#).

10.10.3 CYC.log

Every cycle `mix99p` calls the script `solve_HVmodel`. Output from this script as well as a sample of adjustment factors it written to `CYC.log`.

During each cycle the following information is provided (for a simple single trait model):

```
.
Tue Apr 20 11:11:23 EEST 2004
Variance_model-iterations: 12 Convergence crit (Cr): 0.3716E-07
Lambda convergence (Cd):5.25599751543677D-006 0.00000000000000D+000
Beta convergence (Cr): 2.21850223850421D-002
Acceleration = 16.0800328372335 2 Used = 1.25000000000000
Lambda values for first 30 fixed strata:
-----
Code,      Trait 1, 2, 3, ...
  1  0.9130
  2  0.9416
  3  0.9674
  4  1.0524
  5  1.1232

Lambda values for random strata:
-----
Block, AR_class, Traits: 1, 2, 3, ...

  1  1  1.0152
  1  2  1.0140
  2  1  1.0173
  2  2  1.0142
```

3	1	0.9769
3	2	0.9849
4	1	0.9911
4	2	0.9874

The sample of adjustment factors allows visual inspection (by plotting the adjustment factor of one class) of convergence.

10.10.4 Lambda.log

This file is created every cycle by the master process of `mix99p`. On distributed memory platforms the file will be written to the directory of the master node. `Lambda.log` contains the lambda values of all strata of the first effect and a sample of lambda values of strata of the second effect in the [variance model](#). It also provides information about the standardization of variances. From the `Lambda.log` file one can check whether the lambda values are within a reasonable range. Normally between 0.6 and 1.9. Some values might be smaller or larger. In case a large amount of values is 0.37 ($\exp(-1)$) or 2.72 ($\exp(1)$) something might go wrong. The latter values are the maximum allowed deviations of adjustment factors from 1.0.

11 Acknowledgement

The Genomics and Breeding Group at Natural Resources Institute Finland is kindly acknowledged for the valuable suggestions, comments and for being the beta tester of new MiX99 versions.

12 References

- Gilmour, A. R. and Thompson, Robin (1998). "Reformulated generalised linear (mixed) model aids multiple trait genetic evaluation with polychotomous calving ease". In: *Proc. 6th World Congr. Genet. Appl. Livest. Prod.* Vol. 20, pp. 613–616 (cit. on p. 51).
- Harris, B. and Johnson, D. (1998). "Approximate Reliability of Genetic Evaluations Under an Animal Model". In: *J. Dairy Sci.* 81.10, pp. 2723–2728. DOI: [10.3168/jds.S0022-0302\(98\)75829-1](https://doi.org/10.3168/jds.S0022-0302(98)75829-1) (cit. on p. 43).
- Hesterberg, T. (2005). "Staggered Aitken Acceleraton for EM". In: *Proc. of the American Stat. Assoc.* Minneapolis, Minnesota, USA, 2101–2110. URL: <http://home.comcast.net/~timhesterberg/articles/JSM05-accelerateEM.pdf> (cit. on p. 16).
- Hoeschele, I., Tier, B., and Graser, H. U. (1995). "Multiple-trait genetic evaluation for one polychotomous trait and several continuous traits with missing data and unequal models". In: *J. Anim. Sci.* 73.6, pp. 1609–1627. URL: <http://www.journalofanimalscience.org/content/73/6/1609> (cit. on p. 51).
- Jamrozik, J., Schaeffer, L. R., and Jansen, G. B. (2000). "Approximate accuracies of prediction from random regression models". In: *Livest. Prod. Sci.* 66.1, pp. 85–92. DOI: [10.1016/S0301-6226\(00\)00158-5](https://doi.org/10.1016/S0301-6226(00)00158-5) (cit. on p. 30).
- Janss, L. L. G. and Foulley, J. L. (1993). "Bivariate analysis for one continuous and one threshold dichotomous trait with unequal design matrices and an application to birth weight and calving difficulty". In: *Livest. Prod. Sci.* 33.3–4, pp. 183–198. DOI: [10.1016/0301-6226\(93\)90001-X](https://doi.org/10.1016/0301-6226(93)90001-X) (cit. on p. 51).
- Lidauer, M., Emmerling, R., and Mäntysaari, E. A. (2008). "Multiplicative random regression model for heterogeneous variance adjustment in genetic evaluation for milk yield in Simmental". In: *J. Anim. Breed. Genet.* 125.3, pp. 147–159. DOI: [10.1111/j.1439-0388.2008.00728.x](https://doi.org/10.1111/j.1439-0388.2008.00728.x) (cit. on p. 65).
- Lidauer, M., Matilainen, K., Mäntysaari, E. A., Pitkänen, T. J., Taskinen, M., and Strandén, I. (2023). *Technical reference guide for MiX99 pre-processor*. Release X/2023. Natural Resources Institute Finland (Luke) (cit. on pp. 1, 3–5, 12, 15, 21, 28, 29, 32, 48, 52, 53, 55, 57, 76).
- Lidauer, M. and Strandén, I. (1998). "Experience on using parallel computing to solve large test-day models". In: *Proc. 49th Ann. Meeting EAAP*. 4. Warsaw, Poland, p. 46 (cit. on p. 1).
- Lidauer, M. and Strandén, I. (1999). "Fast and flexible program for genetic evaluation in dairy cattle". In: *INTERBULL Bulletin*. 20. Tuusula, Finland, pp. 20–25. URL: <https://journal.interbull.org/index.php/ib/article/view/468/466> (cit. on pp. 1, 21).
- Lidauer, M., Strandén, I., Mäntysaari, E. A., Pösö, J., and Kettunen, A. (1999). "Solving large test-day models by iteration on data and preconditioned conjugate gradient". In: *J. Dairy Sci.* 82.12, pp. 2788–2796. DOI: [10.3168/jds.S0022-0302\(99\)75536-0](https://doi.org/10.3168/jds.S0022-0302(99)75536-0) (cit. on pp. 1, 2).
- Matilainen, K., Mäntysaari, E. A., Lidauer, M., Strandén, I., and Thompson, R (2012). "Employing a Monte Carlo algorithm in expectation maximization restricted maximum

- likelihood estimation of the linear mixed model". In: *J. Anim. Breed. Genet.* 129.6, pp. 457–468. DOI: [10.1111/j.1439-0388.2012.01000.x](https://doi.org/10.1111/j.1439-0388.2012.01000.x) (cit. on p. 54).
- Matilainen, K., Mäntysaari, E. A., Lidauer, M., Strandén, I., and Thompson, R. (2013). "Employing a Monte Carlo algorithm in Newton-type methods for restricted maximum likelihood estimation of genetic parameters". In: *PLoS ONE* 8.12. e80821. DOI: [10.1371/journal.pone.0080821](https://doi.org/10.1371/journal.pone.0080821) (cit. on p. 57).
- Meuwissen, T. H. E., De Jong, G., and Engel, B. (1996). "Joint estimation of breeding values and heterogeneous variances of large data files". In: *J. Dairy Sci.* 79.2, pp. 310–316. DOI: [10.3168/jds.S0022-0302\(96\)76365-8](https://doi.org/10.3168/jds.S0022-0302(96)76365-8) (cit. on pp. 15, 65, 66, 72, 75).
- MiX99 Development Team (2025). *MiX99: A software package for solving large mixed model equations*. Release IX/2025. Natural Resources Institute Finland (Luke). Jokioinen, Finland. URL: <http://www.luke.fi/mix99> (cit. on p. ii).
- Misztal, I., Tsuruta, S., Aguilar, I., Legarra, A., VanRaden, P. M., and Lawlor, T. J. (2013). "Methods to approximate reliabilities in single-step genomic evaluation". In: *J. Dairy Sci.* 96.0, pp. 647–654. DOI: [10.3168/jds.2012-5656](https://doi.org/10.3168/jds.2012-5656) (cit. on p. 31).
- Misztal, I. and Wiggans, G. R. (1988). "Approximation of prediction error variance in large-scale animal models". In: *J. Dairy Sci.* 71, Supplement 2.0, pp. 27–32. DOI: [10.1016/S0022-0302\(88\)79976-2](https://doi.org/10.1016/S0022-0302(88)79976-2) (cit. on p. 30).
- Mrode, R. A. and Swanson, G. J. T. (2004). "Calculating cow and daughter yield deviations and partitioning of genetic evaluations under a random regression model". In: *Livest. Prod. Sci.* 86.1–3, pp. 253–260. DOI: [10.1016/j.livprodsci.2003.09.001](https://doi.org/10.1016/j.livprodsci.2003.09.001) (cit. on pp. 12, 47).
- Pitkänen, T. J. et al. (2022). "From data to genomic breeding values with the MiX99 software suite". In: *Proc. 12th World Congr. Genet. Appl. Livest. Prod.* Wageningen Academic Publishers. Rotterdam, The Netherlands, pp. 1534–1537 (cit. on p. ii).
- Shewchuk, J. R. (1994). *An introduction to the conjugate gradient method without the agonizing pain*. URL: <http://www.eletrica.ufpr.br/artuzi/te804/arquivos/cg.pdf> (cit. on p. 2).
- Strandén, I. (1999). "Parallel benefits in test-day evaluations". In: *INTERBULL Bulletin*. 20. Tuusula, Finland, pp. 26–32. URL: <https://journal.interbull.org/index.php/ib/article/view/469/467> (cit. on p. 1).
- Strandén, I. (2023). *Command Language Interface for MiX99*. Release X/2023. Natural Resources Institute Finland (Luke) (cit. on pp. 1, 5, 12).
- Strandén, I. and Lidauer, M. (1999). "Solving large mixed linear models using preconditioned conjugate gradient iteration". In: *J. Dairy Sci.* 82.12, pp. 2779–2787. DOI: [10.3168/jds.S0022-0302\(99\)75535-9](https://doi.org/10.3168/jds.S0022-0302(99)75535-9) (cit. on pp. 1, 3).
- Strandén, I., Lidauer, M., Mäntysaari, E. A., and Pösö, J. (2000). "Calculation of Interbull weighting factors for the Finnish test day model". In: *INTERBULL Bulletin*. 26. Verden, Germany, pp. 78–79. URL: <https://journal.interbull.org/index.php/ib/article/view/366/366> (cit. on p. 30).
- Tier, B. and Meyer, K. (2004). "Approximating prediction error covariances among additive genetic effects within animals in multiple-trait and random regression models". In: *J. Anim. Breed. Genet.* 121.2, pp. 77–89. DOI: [10.1111/j.1439-0388.2003.00444.x](https://doi.org/10.1111/j.1439-0388.2003.00444.x) (cit. on pp. 30, 43).
- Tyrisevä, A.-M. et al. (2011). "Principal component approach in variance component estimation for international sire evaluation". In: *Genet. Sel. Evol.* 43.21. DOI: [10.1186/1297-9686-43-21](https://doi.org/10.1186/1297-9686-43-21) (cit. on p. 57).

TECHNICAL REFERENCE GUIDE FOR MiX99 SOLVER

- VanRaden, P.M. and Wiggans, G.R. (1991). "Derivation, Calculation, and Use of National Animal Model Information". In: *J. Dairy Sci.* 74.8, pp. 2737–2746. DOI: [10.3168/jds.S0022-0302\(91\)78453-1](https://doi.org/10.3168/jds.S0022-0302(91)78453-1) (cit. on p. 47).
- Wade, K. M. and Quaas, R. L. (1993). "Solutions to a system of equations involving a first-order autoregressive process". In: *J. Dairy Sci.* 76.10, pp. 3026–3032. DOI: [10.3168/jds.S0022-0302\(93\)77642-0](https://doi.org/10.3168/jds.S0022-0302(93)77642-0) (cit. on p. 68).
- Vuori, K., Strandén, I., Lidauer, M., and Mäntysaari, E. A. (Aug. 2006a). "MiX99 – Effective solver for large and complex linear mixed models". In: *Proc. 8th World Congr. Genet. Appl. Livest. Prod.* Belo Horizonte, MiG, Brazil, pp. 27–33 (cit. on p. 52).
- Vuori, K., Strandén, I., Sevón-Aimonen, M.-L., and Mäntysaari, E. A. (June 2006b). "Estimation of non-linear growth models by linearization: a simulation study using a Gompertz function". In: *Genet. Sel. Evol.* 38, pp. 343–358. DOI: [10.1186/1297-9686-38-4-343](https://doi.org/10.1186/1297-9686-38-4-343) (cit. on p. 52).

Index

Index entry styles:

- normal index entry
- CLIM input commands
- file names
- MiX99 input commands
- shell commands

Page numbers:

- primary definition: [88](#)
- also referred: [88](#)
-  see example on page

- 99999, [53](#)
-8192.0, [11](#), [11](#), [29](#), [71](#), [73](#)
ACCani, [42](#)
AccurType (ApaX), [33](#), [43–45](#)
across block strata, [67](#), [69](#)
ADJUST, [15](#), [15](#)
Aitken acceleration, [16](#)
am
 [48](#), [80](#)
am, [71](#), [72](#)
 [39](#), [49](#)
apax99, [1](#), [30](#), [30](#), [31](#), [33](#), [34](#), [37](#), [40](#),
[43](#)
apax99p, [1](#), [30](#), [30–33](#), [37](#), [43](#)
APAX_VAR_COMP.TXT, [34](#)
APPROX (mix99hv), [69](#), [70](#)
AR
 [74](#), [81](#)
ar
 [73](#), [81](#)
ar, [71](#), [72](#), [72](#)
AR.block, [70](#)
AR.pedi.reml, [69](#), [74](#), [79](#)
ARFILE (mix99hv), [69](#), [74](#)
ARsiwi.data(i), [11](#), [70](#), [74](#), [79](#)
autoregressive model, [71](#), [72](#)
B1, [77](#), [77](#)
for temporary files, [73](#), [77](#)
BASEFILE (mix99hv), [69](#), [75](#)
BINARY
 [73](#), [81](#)
BLOCK
 [48](#), [73](#), [80](#), [81](#)
BR2.dat, [37](#)
BR2.f90, [30](#), [37](#)
CA, [10](#), [20](#), [20](#), [22](#)
Ca, [10](#)
ca, [20](#)
categorical variable, [51](#)
CD, [20](#), [20](#), [22](#), [27](#)
Cd, [10](#), [10](#)
cd, [20](#), [27](#)
CG, conjugate gradient, [2](#)
CHM, [8](#), [9](#)
cl
 [38](#), [49](#), [60](#)
CM, [20](#)
Cm, [10](#)
cm, [20](#), [20](#)
COMBINE (HV var), [72](#)
command line options, [7](#), [8](#), [17](#), [23](#)
COMMONBLOCKS (HV var), [73](#)
conjugate gradient method, [2](#)
convergence criterion, [10](#), [19](#)
Cd, [10](#)
Cm, [10](#)
Cr, [10](#)
convergence indicator, [14](#), [15](#), [19](#),
[19–22](#), [27](#), [56](#)
ca, [20](#)
cd, [20](#)
cm, [20](#)
cr, [20](#)
largest change, [20](#), [27](#)
convergence information, [27](#)
covariable file, [34](#), [52](#)
covariable table, [34](#), [52](#)
covariances of REML estimates, [58](#)
CovarInfo (ApaX), [34](#)
CR, [20](#), [20](#), [22](#)
Cr, [10](#)
cr, [20](#)
CYC.log, [77](#), [82](#), [83](#)
DATAFILE

- ☞ 48, 60, 73, 80, 81
- DATAFILE (HV var), 72
- DATAFILE (mix99hv), 68
- DATASORT
 - ☞ 48, 73, 80, 81
- daughter yield deviation, 12, 13, 28, 47, 48
- Deregression, 12, 23
- Development features (DEV), ii
- DYD, daughter yield deviation, 12, 13, 47, 47, 48
- EDC, effective daughter contribution, 37, 37
- effective daughter contribution, 37
- effective record contributions, 43
- eHat.data(i), 11, 29
- EM REML, 54
- EM, Expectation Maximization
 - for estimation of variance components, 54
 - for Threshold-model, 51
- ERC, 37, 43
- estimation of variance components, 13, 23, 54
 - changing parameters during estimation, 23, 54
 - continuing estimation, 13, 54, 57
 - restarting estimation, 54, 58
 - stopping evaluation, 54
- exa99, 1, 40, 40–42
- FACTOR, 12, 12, 48
- first order autoregressive process, 67, 68, 69, 71
- fixed variance components, 13, 57
- FIXRAN (HV var), 72
- generating observations, 12
- genomic BLUP, 30
- Gompertz-model, 16, 51, 52, 54
- H2calc (ApaX), 35
- Half-Chebyshev, 16
- heterogeneous variance, 1, 11, 22, 24, 32, 65, 65, 67, 76, 77, 80
- heterogeneous variance adjustment, 7, 13, 14–16, 20, 82
- heterogeneous_variance, 76
- Hetlog, 70
- HV, heterogeneous variance, 65
 - HV.data, 70, 70, 72–74, 79
 - HV.info(i), 70, 70
 - HV.pedi, 70, 71, 71, 73, 74, 79
 - HVbase_ResVar, 69, 75, 75, 76, 78, 81
 - HVbase_ResVarCLASS, 76, 78
 - HVbase_ResVarLEVEL, 76, 78
 - HVd.log, 83
- I.log, 82
- ID.Base, 75, 75
- IDD, 11, 13, 47
- IDD.data(i), 11, 29
- IM, 8
- imake4apax, 32
- imake99, 4, 4, 6, 78, 79
- Index.bin, 4
- individual daughter deviation, 11, 13, 29, 47
- information matrix, 58
- instruction file
 - ApaX, 33, 38, 43, 45
 - CLIM command file, 5, 51, 54, 55, 60
 - Exa, 40
 - HV var, 71, 71
 - mix99hv, 68, 72–75, 77
 - preprocessor directive file, 4, 5, 7, 38, 51, 52, 54, 55, 68
 - solver option file, 5, 7, 7, 18, 23, 24, 54, 57, 58, 60, 74
- INT-VAR (HV var), 71
- INTEGER
 - ☞ 48, 60, 73, 80, 81
- IOD, iteration on data, 2
- IOP, 8
- ITER, 6, 23, 23, 24, 54
- ITER.LOCK, 24, 24
- ITER.OLD, 24
- iteration on data, 1, 1, 2, 4
- iterative method, 2, 23
 - changing parameters during iteration, 6, 23
 - convergence indicator, 19
 - determining convergence, 19
 - intermediate results, 6, 22
 - interrupting, 6, 22

- maximum number of iterations, [10](#), [23](#)
- nested iterations, [51](#)
- restarting, [28](#)
- stopping criterion, [19](#), [23](#)
- JFilter (ApaX), [35](#), [36](#), [44](#)
- `Lambda.data(i)`, [70](#)
- `Lambda.log`, [82](#), [84](#), [84](#)
- `ls`, [72](#)
- LS-model, [28](#), [67](#), [71–73](#)
- MACE model, [57](#)
- MAS BLUP, [30](#)
- maximum number of iterations, [10](#)
- MaxNonZ (ApaX), [34](#), [44](#)
- MC, Monte Carlo, [54](#)
- MC EM REML, [13](#), [14](#), [24](#), [54](#), [56](#), [57](#)
- MEA, [9](#)
- mean model, [22](#), [65](#), [65–69](#), [71–77](#), [80](#)
- `Mean_model.slvM`, [74](#), [77](#), [77](#)
- `Mean_model_init.slvI`, [77](#)
- MEB, [5](#), [8](#), [9](#)
- MEL, [5](#), [8](#), [8](#), [9](#)
- MEM, [5](#), [8](#)
- MERGE2 (mix99hv), [70](#), [72](#)
- MES, [5](#), [8](#)
- MISSING
 -  [73](#), [80](#), [81](#)
- missing covariable, [53](#)
- missing observation, [53](#)
 - heterogeneity observation, [73](#)
- missing trait, [51](#), [52](#)
- missing value, [11](#), [11](#), [29](#), [71](#)
- MISSVA (HV var), [73](#)
- MiX99 binary distribution, [4](#)
- `MiX99_DIR.DIR`, [55](#), [55](#), [60](#)
- `MiX99_IN.DIR`, [55](#)
- `MiX99_IN.OPT`, [55](#)
- `mix99hv`, [14](#), [67](#), [68](#), [68](#), [70–75](#), [77](#), [79](#)
- `mix99i`, [1](#), [4](#), [5](#), [6](#), [11](#), [14](#), [28](#), [29](#), [31](#), [32](#), [34](#), [35](#), [40](#), [45](#), [53–55](#), [60](#), [70](#), [78](#), [79](#), [82](#)
- `mix99p`, [1](#), [4](#), [4–8](#), [10](#), [11](#), [14–16](#), [18](#), [19](#), [25](#), [28](#), [29](#), [31](#), [67](#), [70](#), [75](#), [77–79](#), [82–84](#)
- MIX99PATH, [14](#), [56](#)
- `mix99s`, [1](#), [4](#), [4–7](#), [9–14](#), [17–19](#), [25](#), [28](#), [29](#), [31](#), [53–57](#), [60](#)
- mixed model equations, [2](#)
- `MiXtoolmerge.f90`, [29](#)
- `MiXtoolms.f90`, [29](#)
- `MiXtools`, [29](#), [29](#)
- MME, [1](#), [2](#), [2–4](#), [10](#), [19–21](#), [29–31](#), [34](#), [51](#)
- MMM, multiplicative mixed model, [65](#), [66](#)
- MODEL
 -  [48](#), [60](#), [74](#), [80](#), [81](#)
- MODEL (HV var), [72](#)
- model validation, [29](#), [47](#)
- MODVAR (mix99hv), [69](#)
- Monte Carlo, [54](#)
- `mp`, [4](#), [43](#)
- multi-threaded versions of MiX99
 - solvers, [4](#), [8](#)
- multi-threading, [4](#), [8](#)
- multiple residual variance matrices, [32](#), [57–59](#), [76](#)
- multiplicative mixed model, [14](#), [15](#), [20](#), [65](#), [65](#), [74](#), [76](#), [77](#)
- mxntra, [11](#), [29](#)
- New features (NEW), [ii](#), [43](#)
- Newton-Raphson algorithm, [51](#)
- noc, [18](#)
- nocov, [8](#)
- non-linear models, [51](#), [65](#)
- noQ, [8](#)
- nt, [5](#)
- number of data samples, [13](#), [56](#), [57](#)
- NumBVs (ApaX), [35](#), [35](#), [44](#)
- OK-file, [26](#)
- `OK_mix99p`, [6](#), [26](#)
- `OK_mix99s`, [6](#), [26](#)
- OriginalDir (ApaX), [35](#), [43](#)
- `OutFile`, [37](#)
- PAR, [8](#)
- PARALLEL
 -  [74](#), [80](#), [81](#)
- PARALLEL (HV var), [73](#), [77](#)
- parallel computing, [1](#), [3](#), [4](#), [4](#), [6](#), [8](#), [11](#), [22](#), [23](#), [29](#), [30](#), [32–34](#), [65](#)
- `apax99p`, [30](#)

- `mix99p`, [4](#)
- multi-threading, [4](#), [8](#), [9](#)
- PARFILE
 - ☞ [48](#), [60](#), [73](#), [80](#), [81](#)
- `parfile`, [55](#), [58](#), [59](#), [59](#)
- PARFILE (HV var), [73](#)
- PCG, preconditioned conjugate
 - gradient, [2](#), [2](#), [4](#), [10](#), [20](#), [23](#), [27](#), [51](#), [54](#)
- PEDFILE
 - ☞ [48](#), [60](#), [73](#), [80](#), [81](#)
- PEDFILE (HV var), [73](#)
- PEDIGREE
 - ☞ [48](#), [60](#), [73](#), [80](#), [81](#)
- PEDIGREE (HV var), [72](#), [72](#)
- PEDIGREECODE
 - ☞ [48](#), [73](#), [80](#), [81](#)
- `PEEK`, [6](#), [22](#), [22](#), [23](#)
- `PEV.bin`, [33](#), [33](#), [34](#)
- `PEVani`, [37](#), [37](#), [45](#)
- `PEVrnn`, [42](#)
- PRECON
 - ☞ [48](#), [74](#), [80](#), [81](#)
- PRECON (HV var), [73](#)
- preconditioned conjugate gradient
 - method, [1](#), [2](#)
- preconditioner matrix, [2](#), [2](#), [14](#), [21](#), [22](#)
- preconditioning, [2](#), [10](#), [19–22](#), [32](#), [53](#), [55](#)
 - block diagonal, [21](#), [21](#), [47](#)
 - diagonal, [21](#), [21](#)
 - full block, [21](#)
 - updating, [14](#)
- predicted observations, [11](#), [29](#)
- RAM, [7](#)
- RANDOM (HV var), [72](#)
- random_seed, [14](#)
- RANSOLFILE (HV var), [73](#)
- RDB, [9](#), [9](#)
- RDL, [9](#), [9](#)
- RDM, [9](#), [9](#)
- RDS, [9](#), [9](#)
- RDU, [9](#)
- RDX, [9](#)
- REAL
 - ☞ [48](#), [60](#), [73](#), [80](#), [81](#)
- REAL-VAR (HV var), [72](#)
- REGRESS (HV var), [72](#)
- RELATIONSHIPS (HV var), [72](#)
- reliabilities, [30](#), [31](#), [36](#)
 - approximate, [1](#), [30](#)
 - exact, [1](#), [40](#)
 - maternal genetic, [33](#)
 - output files, [37](#)
 - reversed, approximation, [43](#)
 - multi-trait, [45](#)
 - single-trait, [43](#)
- REML, [13](#), [14](#), [54](#), [55–57](#), [59](#)
- REMLlog, [54](#), [55](#), [58](#), [59](#), [59](#)
- resfile, [55](#), [59](#)
- RESID, [10](#), [29](#), [74](#)
- residuals, [7](#), [10](#), [11](#), [13](#), [29](#)
- restart iteration, [28](#)
- reversed reliability approximation, [33](#), [34](#), [43](#)
 - multi-trait, [44](#), [45](#)
 - single-trait, [43](#), [45](#)
- RHO (HV var), [72](#)
- runMMM, [70](#), [77](#), [78–80](#), [82](#)
- `S.log`, [82](#)
- SCALE (HV var), [73](#)
- Second level preconditioner, [19](#)
- SEED, [12](#), [14](#), [56](#)
- SEfnn, [42](#)
- SEfix, [42](#)
- SEreg, [42](#)
- SETLOSS (mix99hv), [70](#)
- `sHat.data(i)`, [11](#), [29](#)
- simulating observations, [12](#)
- `SireFile`, [37](#)
- `SiWi.data(i)`, [14](#)
- sm
 - ☞ [60](#)
- sm
 - ☞ [60](#)
- `Sol_mn`, [28](#)
- `Solani`, [28](#), [37](#), [48](#)
- `SolaniB`, [16](#)
- `Solcommon`, [28](#)
- `Soldyd`, [12](#), [28](#), [48](#), [50](#)
- `Solfnn`, [28](#)
- `Solfix`, [28](#), [52](#)
- `SolfixB`, [16](#)
- `SolMS`, [28](#)

[Solold](#), [29](#)
[SolPA](#), [28](#)
[Solpriv\(i\)](#), [28](#)
[Solrnn](#), [28](#)
[Solreg](#), [28](#)
[SolSNP](#), [28](#)
[SOLTYP](#), [15](#), [53](#)
[Solunf](#), [29](#)
[SOLUNF](#) (HV var), [73](#)
[solve_HVmodel](#), [77–79](#), [83](#)
[Solvec](#), [28](#), [28](#), [29](#)
[SORT_B](#) (mix99hv), [69](#)
[SORT_R](#) (HV var), [36](#), [72](#)
[sp](#), [8](#)
 sparse regression matrix, [18](#)
[Sparse_Matrix.DMPz](#), [34](#)
[srm](#), [18](#)
[ssGTABLUP](#), [28](#)
[STAND](#) (mix99hv), [69](#), [75](#)
 standard errors
 for fixed effects, [42](#)
 for variance components, [57](#)
 standard output, [25](#), [48](#), [82](#)
[StartDIM](#) (ApaX), [34](#)
[STOP](#), [10](#), [10](#), [14](#), [23](#), [51](#)
[STOP](#), [6](#), [22](#), [22](#), [27](#), [54](#)
[STOPC](#), [15](#), [24](#), [77](#)
[STOPE](#), [13](#), [23](#), [24](#), [54](#), [56](#)
 stopping criterion, [19](#), [27](#)
 Ca, [10](#)
 stopping iteration, [22](#), [27](#)
 sum of selected model factors, [11](#), [29](#)
 threshold-model, [10](#), [23](#), [51](#), [51](#), [54](#)
[TITLE](#)
 ☞ [48](#), [60](#), [73](#), [80](#), [81](#)
[TITLE](#) (HV var), [71](#)
[Tm10.trco\(i\)](#), [4](#)
[Tm10.trco0](#), [4](#)
[Tmp4.pedi\(i\)](#), [4](#)
[Tmp4.pedi0](#), [4](#)

[Tmp5.clas\(i\)](#), [4](#)
[Tmp5.clas0](#), [4](#)
[Tmp6.diab\(i\)](#), [4](#)
[Tmp6.diab0](#), [4](#)
[TMPDIR](#)
 ☞ [80](#), [81](#)
[TMPDIR](#) (HV var), [73](#)
[TMPHV](#) (mix99hv), [70](#), [77](#)
[TRAITGROUP](#)
 ☞ [73](#), [81](#)
[TRAITS](#) (HV var), [72](#)
[TRGRP](#) (HV var), [72](#)
[TRGRP](#) (mix99hv), [69](#)

[VALID](#), [11](#), [15](#), [29](#), [47](#), [48](#)
[VAR](#) (HV var), [73](#)
[VAR_I](#) (mix99hv), [68](#)
 variance model, [14–16](#), [22](#), [65](#), [65–77](#),
 [80](#), [82–84](#)
[Variance_model.slvV](#), [77](#)
[VAROPT](#), [12](#), [13](#), [13](#), [15](#), [16](#), [20](#), [54](#), [56](#),
 [57](#)
[vceI](#), [58](#), [59](#)
[vceSE](#), [58](#), [59](#), [60](#)
[Vi.log](#), [82](#)
[Vs.log](#), [82](#), [82](#)

[WEIGHT](#)
 ☞ [74](#), [81](#)
[Weights](#) (ApaX), [35](#), [35](#)
 within block strata, [67](#), [68](#), [69](#), [74](#)
[WITHINBLOCKORDER](#)
 ☞ [48](#), [74](#), [80](#), [81](#)
[WITHINBLOCKORDER](#) (HV var), [72](#)
 work files, [2](#), [4](#), [4](#)

[YD](#), yield deviation, [11](#), [47](#)
[YD.data\(i\)](#), [11](#), [29](#)
[yHat.data\(i\)](#), [11](#), [29](#)
 yield deviation, [7](#), [11](#), [13](#), [29](#), [47](#)
[ySim.data0](#), [12](#)