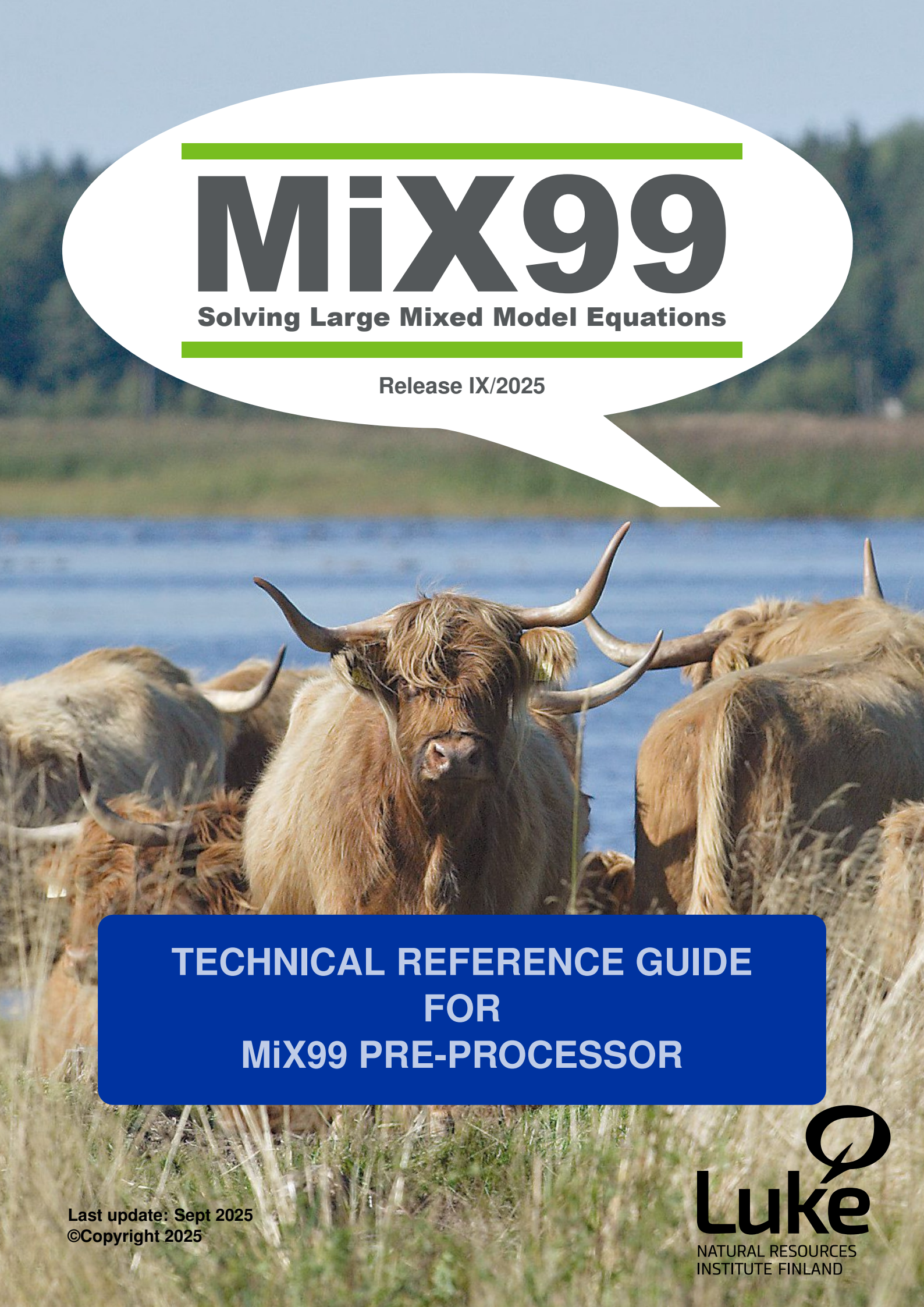
MiX99

Solving Large Mixed Model Equations

Release IX/2025



TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

Last update: Sept 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 Finland[1].

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

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

Contents

What is new?

New features and options in MiX99 since the previous release (2023):

NEW

1) Rewritten CLIM manual where for example single-step model commands are explained better, such as the SSGBLUP and SSGTBLUP commands.
2) mix99i has more output, e.g., the numbers of observations per effect, for checking goodness of data.
3) Maximum length of variable names has been increased from 8 to 80 characters which has lead to increased variable name width in many outputs.
4) Alternative ways to give variance components using the PARFILE command, such as including variance components in the command file and giving variances in a matrix format.
5) Limit to the number of traits lifted.
6) Options in mix99s to allow writing solution files having Mendelian sampling terms (SolMS) and parent averages (SolPA).
7) Fully componentwise ssGTBLUP and ssSNPBLUP models available in mix99p.
8) The preconditioner for the ssSNPBLUP model has been improved, enabling better convergence using a lower value (less than 100) for the second level preconditioner.
9) Variance component estimation for GBLUP and SNPBLUP (REGMATRIX) models.
10) The COVFILE command accepts lower triangle dense matrix format.
11) The REGMATRIX command has option REGINDEX such that the records in the REGFILE do not have to be in the same order as in the data file and only one record is used per index level.
12) REGMATRIX can be defined to be present in some traits only using the REGTRAITS command.
13) Single-trait Reversed reliability approximation (AccurType 20) in apax99 is now parallelized within trait.

DEV

14) Marker-specific (co)variances or weights allowed in the ssSNPBLUP model.

1 Introduction

Dairy cattle breeders around the world have moved to so-called test-day models from plain animal models. Usually, this model upgrade leads to a manifold increase in computations. This is because test-day models have more effects than traditional models, and also a greater number of records. In a national evaluation, the number of test-day records and the number of unknowns in the mixed model equations (MME) are easily more than 100 million.

Computational techniques and algorithms that were found useful for solving animal models may take weeks to obtain a solution to large random regression test day model MME. Consequently, computationally faster solving algorithms had to be developed. Strandén and Lidauer (1999) and Lidauer et al. (1999) advocated the use of preconditioned conjugate gradient (PCG) method. Strandén and Lidauer (2001) showed the usefulness of parallel computing. These techniques have been found to reduce computing time considerably.

Features of the MiX99 software have been developed primarily for the projects we have been involved. New models and options are added to MiX99 as they are needed in practice. In this version, current statistical models used in dairy cattle breeding are mostly covered. MiX99 also supports a threshold model with one categorical trait and several correlated continuous traits, as well as a non-linear model with the Gompertz function. The exploitation of genomic data initiated the development of options useful for genomic selection. Currently, genomic BLUPs can be calculated with simple models. Another new area of development in MiX99 is the estimation of variance components for large and complex models. Thus far, a version of a stochastic EM REML algorithm has been implemented for multi-trait mixed linear effects models.

Statistical Models

The following model options are supported:

1) For each trait, the statistical model can be defined separately. The number of traits is unlimited as long as the traits can be organized in different groups, where within a group there can be a maximum of 31 traits, and as long as residuals of traits from different groups are uncorrelated.

2) There is no limit to the number of the following effects: fixed effects with a small or a large number of levels, regression effects, regression effects nested within fixed effects, random effects, regression effects nested within random effects, random effects with a correlation structure between levels (e.g. QTL effects), and the number of maternal and/or paternal effects.

3) Sire models and animal models are allowed. Grouping of unknown parents by phantom parent groups is possible for both the sire and the animal model.

4) Repeatability models are supported.

5) Reduced rank models in a wide sense are supported. For instance, for multi-trait models, it is possible to define observations on different traits as repeated observations; e.g. milk, protein, and fat yield are repeated observations of the same trait. This feature is useful for certain types of reduced rank random regression models.

6) Combining effects. This option allows combining different fixed effects or combining different random effects within the model. The option can be useful for models with QTL effects or for models with social effects.

7) Models with multiple residual variances are supported. This allows the application of different residual (co)variance matrices. This might be useful, when the residual variance is changing by a time-dependent variable or if the observations are recorded by different recording schemes, i.e., if the accuracy is not the same for all observations. Hence, it allows applying different heritabilities.

8) Models with weighted observations. For each trait, a weighting variable can be provided.

9) LS models and GLS models.

10) Multiplicative models accounting for heterogeneous variance.

11) Model with non-linear Gompertz function as demonstrated by Vuori et al. (2006).

12) Threshold models with one categorical and several linear traits. Unequal design matrices and missing traits are allowed. Thresholds can be estimated or set to be known. (Co)variance components have to be known.

13) Models including QTL effects. In general, MiX99 allows for any random effect, apart from the genetic animal effect, to include a covariance structure between the levels of the random effect. The use must provide the covariance structure in the form of an inverse correlation matrix (for instance an inverse IBD matrix).

14) Models with fixed or random regression variables with a large number of regression effects. This is mostly useful for genomic data where the number of regression effects on a model line can be many thousands.

15) Models with social effects. MiX99 allows modeling a covariance structure between an animal and its contemporary group members.

2 How to run the mix99i pre-processor

2.1 Computing environment

MiX99 is written in standard Fortran 90 and is self-contained. It has been developed in UNIX and Linux environments. The program has been tested to compile under many UNIX and Linux Fortran 90/95 compilers as well as Windows compilers.

2.2 MiX99 pre-processor input files

The following files need to be created before starting MiX99: a data file with the data to be analyzed; a pedigree file with the relationship information; a (co)variance components file with the variance and covariance components for the random effects; an instruction file with the information about the statistical models and run-time parameters, covariable table file (optionally, for models with regression effects). A word of caution in file names. It is advised that no space characters are in file names or in the path. If there is space within the path or file name, the complete file name information with path needs to be given between double quotes (”). For example, ”my directory/file name” in Linux/Unix but ”my directory\file name” in Windows.

2.3 Running the pre-processor

Solving mixed model equations using MiX99 involves the execution of two programs. First, the pre-processing program `mix99i` is executed. Then, a solver program (see *Technical Reference Guide for MiX99 Solver*) is executed.

There are two alternatives how to instruct the `mix99i` pre-processor program. It can be done either by specifying a MiX99 instruction file, which is read from the standard input:

```
mix99i < MiX99_instruction_file
```

or by providing a CLIM command file (see manual *Command Language Interface for MiX99*). In that case, `mix99i` is executed by the command:

```
mix99i CLIM_command_file
```

The CLIM interface is recommended because of its ease of use. CLIM covers a large variety of possible models and options. However, for some special models or options, the instructions need to be given by a MiX99 instruction file.

Some of the MiX99 preprocessor settings can be additionally specified from the command line (see Contents). The option given on the command line overrides the corresponding setting in the MiX99 instruction file or CLIM command file.

MiX99 preprocessor stores its input directives to file `MiX99_IN.DIR` and possible command line options to file `MiX99_IN.OPT`.

After the successful completion of the preprocessor, the file `OK_mix99i` will be created. The file will have the completion time. When this file is missing, the preprocessor was terminated due to some error. When using MiX99 through a script, please check the existence of the `OK_mix99i` file.

2.4 Command line options

Some of the MiX99 preprocessor settings can be additionally specified from the command line. List of available command line options of the MiX99 preprocessor `mix99i`,

can be obtained with

```
mix99i -h
```

This will print the following instructions:

Command line options of mix99i:

```
Usage:

mix99i [-d] [-b] [-l] [--nproc NPROC] [--checkdata] [CLIMFILE]
       [--datafile DATAFILE] [--pedfile PEDFILE] [--parfile PARFILE]
       [-g PATH]
       [--usemacros] [--keepsol] [--keepindir]
       [-h|--help] [-v|--verbose] [-V|--version]
       [--bindir BINDIR] [--datadir DATADIR] [--tmpdir TMPDIR]
where
  -d: Make MiX99_DIR.DIR by CLIM, no preprocessing.
  -b: Beta testing version of CLIM.
  -l: Long listing option in mix99i.
  --nproc NPROC: Number or processes.
  --checkdata: Enhanced checking of data file.
  --datafile DATAFILE: Data     file name.
  --pedfile   PEDFILE: Pedigree file name.
  --parfile   PARFILE: Variance file name.
  -g PATH: Read generated observations from directory PATH.
  --usemacros: Use CLIM macros (DEFINE) and ranges (abcNN:MM).
  --keepsol: Keep old solution files.
  --keepindir: Do not overwrite MiX99_IN.DIR and .OPT files.
  --PAR: Use PARDISO library in inv(A22) diagonal computations.
  --CHM: Use CHOLMOD library in inv(A22) diagonal computations.
  -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
  MIX99_NPROC
Note: Environment variables are used first, then command line options.
```

The option given on the command line overrides the corresponding setting in the MiX99 instruction file or CLIM command file.

MiX99 pre-processor stores the command line options to file `MiX99_IN.OPT`.

2.5 Multi-threaded MiX99 executables

In addition to the normal MiX99 executables, specially compiled ***multi-threaded versions of MiX99 executables*** are also included. Both the “single process” (mix99s) and the parallel solvers (mix99p) have a multithreaded version. Multi-threaded MiX99 executables parallelize some of the calculations, especially large dense matrix computations. These multi-threaded MiX99 executables are located in the `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.

The number of computational threads used by the multi-threaded MiX99 executable is controlled by one or more ***MiX99 binary distribution*** 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, for example, with ten (10) threads:

```
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
```

The multi-threading is especially useful with the MiX99 solvers in the case of genomic information.

Multi-threading is available in mix99p MPI solver. The multiple treads can be taken to use by the above-mentioned environmental variables or by solver command "nt". The effect of the nt command depends on the model 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 thread.

The command line option is like:

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

which uses 4 processes in MPI but 10 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 the multiple threads. 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 Editing the input files

3.1 Data file

The data file contains information on the data to be analyzed together with class and regression variables for the model.

The data file can be a (formatted) ***free format text file*** (columns are separated by at least one space) or in a (unformatted) ***binary format binary file***. Only text files are human-readable but binary files are, on the other hand, faster to read and write. See Appendix A on how to convert between text and binary format.

Each record, i.e., a line in the data file, has been divided into two parts: an integer number part which is followed by a real number part.

1) Integer numbers. The integer data part consists of all class variables in the model. Additionally, depending on the data structure and the model, it may contain sorting variables and an index for the covariable table file. All integers are coded using the default machine integer type. Hence, on 32-bit platforms, the data file can contain integer numbers in the range of ±2.147.483.648. Missing integer numbers must be coded with zeros (0).

2) Real numbers. The real data consists of observations to be analyzed, and again, depending on the model, it may contain covariables or weights. Missing real numbers can be coded with an arbitrary real value, which will be specified in the MiX99 instruction file or CLIM command file.

The data file can contain columns that are not used in a particular run of MiX99. Because MiX99 can read only numerical data, alphabetical character data can be included only after the real numbers in the free format data file.

Information on genetic relationships must be provided in a separate pedigree file. The information in the pedigree and data files are linked together through the class variables for the additive genetic effects. For the animal model, it is the animal code and for the sire model, the sire code. For models with a maternal or a paternal effect or both, a record in the data file must also contain the codes of these classes. The pedigree information of the maternal or paternal effect is given in the pedigree file as well.

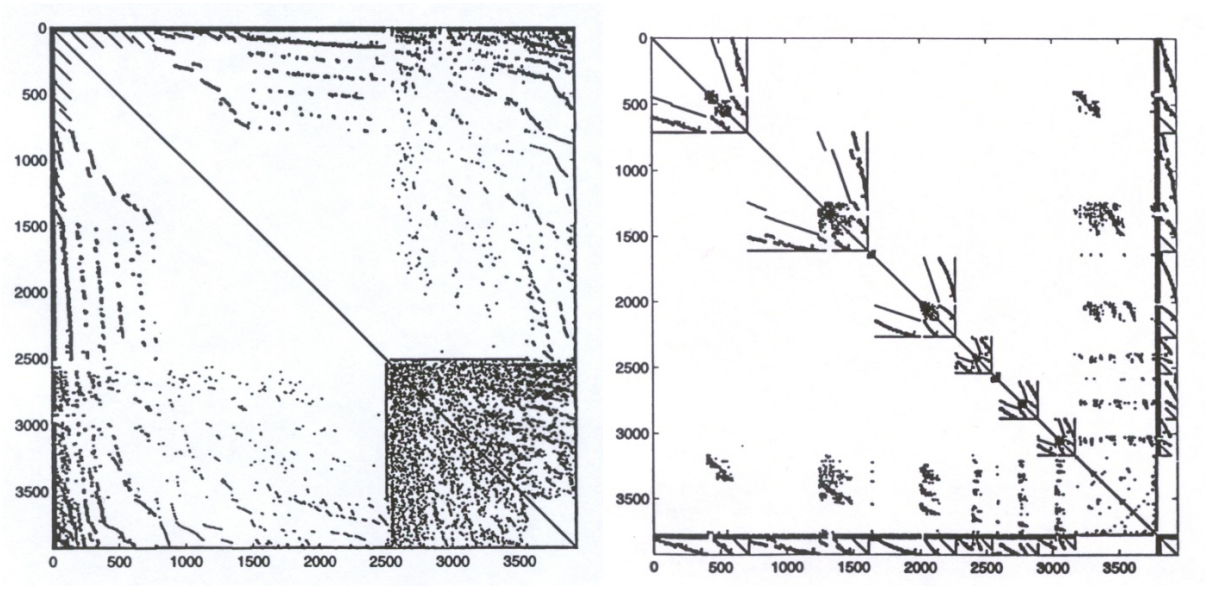
3.1.1 Sorted records in the input data file (optional)

MiX99 allows ordering of equations in ***equation family*** blocks (Lidauer and Strandén, 1999), a concept which is essential for calculation of approximate reliabilities by ApaX and for parallel computing by MiX99 solver and ApaX. To take advantage of this concept, it is important that each equation family block contains as many as possible closely connected equations, and that the number of equations connecting equation family blocks is as low as possible. Equations that connect all (or many) equations can be grouped into one or several common blocks. ***Common blocks*** refer to blocks of equations, which will be made available to all processors when parallel computing is applied. Equations of common blocks must be ordered to appear at the bottom of the MME, i.e., animals of common blocks must have largest block sorting variables. For many animal breeding models this ordering of mixed model equations yields a structured coefficient matrix (Figure 1) that has nearly double-bordered block diagonal form (Duff et al., 1992). MiX99 will solve the mixed model equations even if such a structure cannot be achieved. However, pre-processing time may be longer than usual

and parallel computing may yield no advantage over using one processor. Therefore, it is important to keep this concept in mind when editing the data for an analysis where parallel computing is applied. The concept is equally important for the approximation of reliabilities (see *Technical Reference Guide for MiX99 Solver*).

Records may be sorted by three different variables: *block code*, *relationship code* and *trait group code*. Sorting the records by the *block code* is a prerequisite to create equation families. Sorting the records by the *relationship code* and *trait group code* enhances computation speed for many models.

Unsorted input data: MiX99 will solve your model even the input data is not sorted. However, for large data sets and data sets with repeated observations it may reduce computing speed significantly.



Equations in model effect order | Equations in equation family block order

Figure 1: Non-zeros in the coefficient matrix of the MME when equations are sorted either by model effects or by equation family blocks. The model described TD milk yield data of seven herds including an age effect, a function for stage of lactation effect, a HTD effect and random regressions for non-genetic and genetic animal effects.

Block code (optional)
The concept of equation family blocks requires sorting records by a blocking variable. For example, in dairy cattle, equations for animals in separate herds represent an equation family. Many models contain such effects and are therefore suitable blocking variables to be used to group equations into equation families.

A good blocking variable orders the records such that all (or almost all) records of the same animal and its close relatives (parents and progeny) are in the same block. If the data does not contain such a variable, it might be possible to generate a suitable blocking variable. Again, in dairy cattle, if a model contains a herd-year-season effect (such like a herd-test-day) but not a herd effect, it is advisable to include the herd code into the data and use it as the blocking variable.

The solver program reads the data by blocks with one or several blocks at a time. If there is only one block in a large data file, all iteration files are read into the random

access memory at the same time, which might exhaust computer resources. If the data can be read into the memory then this may be sensible. However, if parallel computing is to be used, then several blocks need to be defined because these blocks are distributed to the processors.

Relationship code (optional)
For data with repeated observations, it is beneficial to sort the records by a classifier for the same relationship information. The same relationship means that the records are from the same animal (same sire, same dam, same maternal or paternal parent). This will allow MiX99 to store relationship information only once for a set of repeated observations, which reduces disk usage and computing time.

Animal model: For an animal model, the animal ID is used as a *relationship code*.

Sire model: For a sire model, one has to be careful to specify the proper *relationship code*. In case of a simple sire model and without any maternal or paternal effects, the sire ID can be used as the *relationship code*. However, **in case the sire model contains a maternal or paternal effect, the sire ID must not be used as the *relationship code***. This is because records may have the same sire, but their maternal or paternal sire may be different, i.e., observations do not have the same relationship information. An example for this is a trait like "calves born dead" that has been modeled under a sire model including the sire as a direct genetic effect and the maternal grand sire of a calf as an indirect genetic effect. In this case, the calf's ID is the suitable relationship code. However, there would be only one observation per calf. Therefore, there is no advantage to sort the data by a relationship code and no need to specify a relationship code in the DATASORT specification of the instruction file (see Example 7.7).

Trait group code (optional)
For certain types of multi-trait models it can be beneficial to arrange the traits in trait groups. Use of trait groups may reduce the size of the data file and computing time substantially. Grouping traits is possible only if some traits are measured at different time or in different environments such that the residual correlation is zero between traits of different trait groups.

For instance, considering a data from dairy cattle with repeated observations on milk, protein, and fat yield in the first, second and third lactation, and a model where lactations are modeled as different traits. Thus, the model would include nine different traits. Because observations from the different lactations are measured at different time, each record would contain observations for three traits only and observations for the other traits would be coded as missing. Also some of the environmental effect information would be missing for the other traits. Consequently, each record (data line) in the data file would contain many missing values. By applying the trait group option, each record (data line) can be assigned to a different trait group. Then, continuing this example, there would be three trait groups, one for each lactation, and each trait group would contain three traits: milk, protein, and fat yield. Each record must contain the trait group code and the information related to the traits of the particular trait group, but information about the other traits is redundant (missing by definition of trait group code). Hence, observations of traits from different trait groups can be stored in the same data columns, which reduces sizes of the input and iteration work files. Trait groups are

defined in the TRAITGRP line in the MiX99 instruction file. See also Examples 7.4 and 7.6 how to specify models with the trait groups.

A trait can only be present in one trait group. A trait group may contain a maximum of 31 traits, but lower number is recommended to enhance computations. If the input data is designed to include trait groups, records with the same relationship code must be sorted by the trait group code in ascending order.

3.1.2 Multiple input data files

Some operation systems restrict maximum file size to 2 GB. This may require splitting the input data files to several files when huge amount of data is analyzed. The syntax for specifying multiple input data files is as following: The first file can have any name, the second file should have an identical name with the first one and with "2" added at the end of the filename, for the third file, "3" is added at the end and so on. E.g. `any_name`, `any_name2`, `any_name3`, ...

3.1.3 Visualizing block-to-block dependencies

Performance of the parallel MiX99 solver depends partially on the amount of communication between the parallel computation processes. Parallel communication is mostly due to dependencies between the model effects, for example, pedigree relationships between animals, animals moved from one herd to another, and so on.

MiX99 program `block_information` can be used to analyze the dependencies between the equation family blocks:

```
mix99i < MiX99_instruction_file
block_information
```

Amount of communication needed between the equation blocks is shown in the output of `block_information` program:

```
Communication in parallel computing:
  91.33% using the linked list
   7.98% through the common blocks
   0.69% through the across blocks

   5.78% of equations need communication

Total communicated equations:     21999258
  - linked list              :     20090904
  - common blocks            :      1756440
  - across blocks            :       151914

Total number of equations    :    380436594
  - within blocks            :    380284680
  - across blocks            :       151914
```

In this example 5.78% of equations need communication between the parallel processes.

Additionally, file `BMatrix.dat` is created containing block-to-block communication matrix in sparse “i,j,v” format, number of common blocks, specified by the user in the MiX99 directive file, is stored in file `NcommonB.dat`, and partitioning of the blocks to parallel processes is stored in file `PlastB.dat`. Equation blocks are renumbered

sequentially in the files. Original block codes can be obtained from file `original_blockcode.dat` created by MiX99 preprocessor `mix99i`.

With command line option

```
block_information --separate
```

two communication matrix files are created instead of the default `BMatrix.dat`. File `BM_data.dat` contains the communication due to the data and file `BM_VStruct.dat` the communication due to the variance structures.

Communication matrix can be visualized using command line option

```
block_information -p
```

in which case a PNG image file `BMatrix.png` is created instead of `BMatrix.dat`.

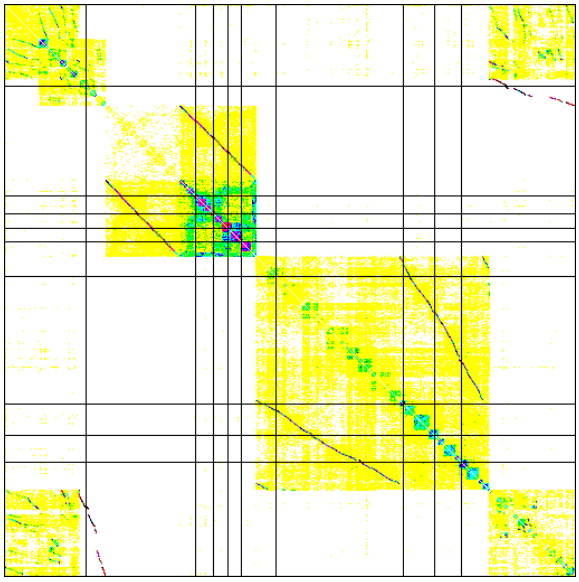


Figure 2: Block-to-block communication matrix of Nordic Holstein test-day model containing 131925 equation blocks. Partitioning of blocks to 10 parallel processes is indicated by vertical and horizontal lines.

Similarly, with command line options

```
block_information -p --separate
```

separate image files `BM_data.png` and `BM_VStruct.png` are created for data and variance structure communications.

PNG image (see Figures 2 and 3) shows the block-to-block communication matrix using eight basic colors in order: white, yellow, green cyan, blue, magenta, red, and black. White means no communication between blocks, yellows contain the smallest communication counts, and blacks are the largest communications so that, in total, each 7 (non-white) color in the image illustrates $\frac{1}{7}$:th of the total communication. Block-to-block communication image shows also the partitioning of the blocks for the parallel processes as a grid of black lines.

By default, communication through the common blocks and between blocks pairs inside the same diagonal pixels are omitted from the image for better color resolution.

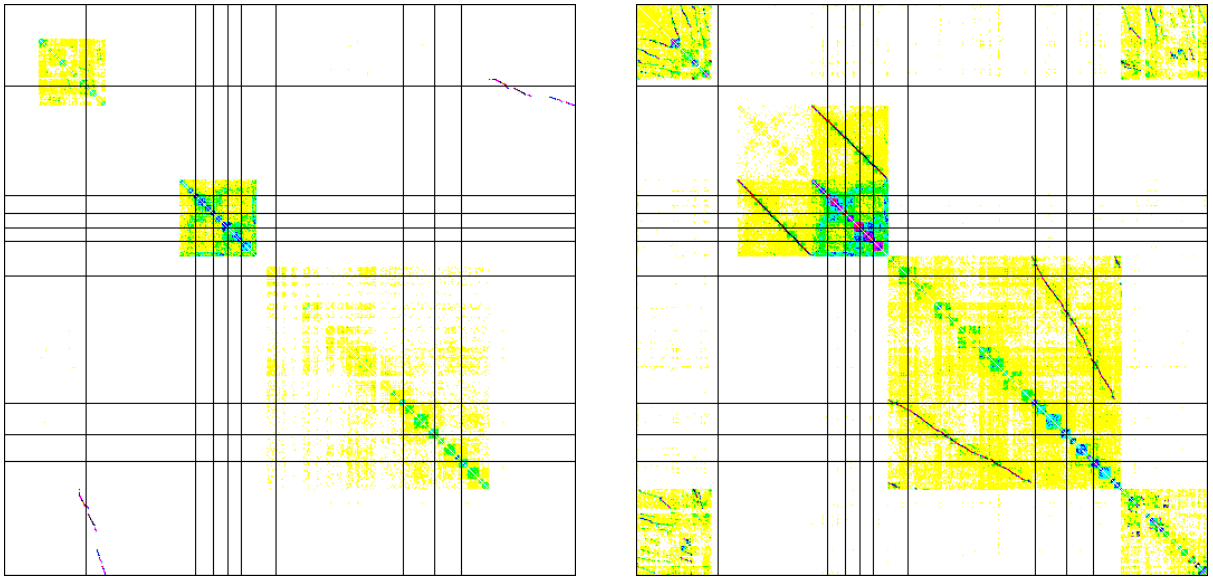


Figure 3: Separate data and variance structure block-to-block communication matrices of Nordic Holstein test-day model containing 131925 equation blocks.

These can be included in the PNG image using options `--common` and `--diag`. The resolution of the PNG image is 512 $\times$ 512 pixels by default and can changed using option `-n <image pixels>` or, alternatively, by specifying resolution for the blocks using option `-r <block pixels>` in which case the pixels of the partition separating lines are added for the overall number of pixels in the image. Line widths of the partition lines can be changed with option `-l <line pixels>`.

During the execution of the `block_information` program, the block-to-block communication matrix is stored in memory using full upper triangle format by default. If the model has very large number of blocks, `block_information` program can take large amount of memory. This can be prevented with option

```
block_information -s
```

in which case the communication matrix is stored as a sparse matrix instead. Memory usage depends, then, on the sparsity structure of the block-to-block communication in the model.

With command line option

```
block_information -g
```

a graph file with file suffix `.graph` is created additionally (i.e. file `BMatrix.graph` and optionally files `BM_data.graph` and `BM_VStruct.graph`). The graph file can be used to minimize the block-to-block communication by reordering and partitioning the blocks using external `gpmetis` program. See utility program `partition_blocks` in the `tools/` subdirectory of the MiX99 binary distribution.

See `block_information -h` (or `--help`) for full usage information of the program.

3.2 Pedigree file

All the pedigree information must be given in the pedigree file. Each animal in the pedigree must have a record with four integers of which the forth integer is optional.

The format is:

1	2	3	4
animal code	sire code	dam / maternal grand sire code	block code (optional)

The integers must be separated by at least one space. After the fourth (third) column the file may contain other information. It may contain a column with within-bull classifications of the daughters, which may be used for the calculation of within-bull daughter yield deviations (DYD) (see Calculation of daughter yield deviations in *Technical Reference Guide for MiX99 Solver*). Further, it can include (right after all integer columns) the inbreeding coefficient of the animal if inbreeding should be considered. Same record can appear only once in the file. Each animal with a record in the data file or animals without a record but coded as maternal or paternal effects must be in the pedigree file. Animals without records in the data file must have a record of their own as well. When a sire or animal model without phantom parent groups is used (specified in the MiX99 instruction file or CLIM command file), missing sires and dams (or maternal grand sires) can be coded with zeros (0) or negative integer numbers.

Phantom parent groups. If missing ancestors are grouped into phantom parent groups, a phantom parent group code is given instead of zero. This group code must be negative (!) to distinguish it from a normal animal code. A phantom parent group code must not have a record of its own in the pedigree file.

Sire model with phantom parent groups. Note, if phantom parent groups are specified for a sire model (for instance in MACE), also the phantom parent group of the maternal grand dams must be provided for each bull. This information is given in the fourth column of the pedigree file. Subsequently, for such a model the optional block code must be given in the column five.

MiX99 allows random phantom parent group effects. This is done by adding for each phantom parent group some value to its diagonal element in the inverse of the relationship matrix (see PEDIGREE in chapter 4).

When benefits of using equation family structure are desired, the block code of the animal has to be given in column 4 (or column 5 for sire models with phantom parent groups) of the pedigree file. Each animal's block code needs to be the same as in the data file. Animals with records in different data blocks (e.g. in different herds) have to be coded with the code of one of the different data blocks where it has observations, e.g., the block with most of its observations. If an animal does not have an observation, but it is a parent to an animal with observations in the data file (e.g. pedigree animal of a particular herd), then it should receive the same block code as its offspring. This is most suitable for a cow without observations. It should be assigned to a block having most of its daughters.

When an animal does not belong to any equation family (no observations to give block code), or it is in many different families through relationship information (e.g. dairy sires have progeny in many herds), a new block code should be given. We recommend a separate block code for animals with links to many different equation family blocks. For instance, sires in a dairy cattle population can be assigned to one block. This

is particularly important for parallel computing. These blocks should be defined as common blocks and largest block code variables have to be given to these blocks. Thus, sorting by the block code variable will ensure that animals of common blocks will appear at the bottom of the MME (see COMMONBLOCKS in chapter 4).

Animals that cannot be included into any equation family can be grouped into one or several own groups, depending on the number of such animals. An equation family should always have a reasonable size. For example, if the pedigree has equation families with 50 to 2000 animals per block and a block with 300,000 animals, it is advisable to split the largest block into several smaller blocks. The solver program reads as many animal blocks at a time as possible, and the largest animal block dictates the memory requirements. Blocking is of greater importance in parallel computing.

3.2.1 Sorting the pedigree file

If equation family structure is desired for the MME (recommended), the pedigree file must be sorted by the block code in the same order as the input data file. The pre-processor program orders the mixed model equations in the same order as the animal family blocks are in the pedigree file. It is essential for the parallel computing that all animal blocks having many links to all other animal families (e.g. sire blocks in dairy cattle) appear at the end of the pedigree file. These blocks can then be treated as common blocks for all processors. The pedigree file can be unsorted if no block code is specified in the MiX99 instruction file. Pedigree file can be prepared using `RelaX2` program. This program allows pruning the pedigree, sorting it according to the data file order and including the block code from the data.

3.3 File with (co)variance components (PARFILE)

A file with variances and co-variances for all the random effects in the model must be prepared. The matrices can be of different size depending on the model specification. The matrices are numbered in the same order as the random effects are given on the RANDOM instruction line(s). The number of the residual (co)variance matrix is always one larger than the largest random effect number on the RANDOM line(s).

The variances and co-variances are specified in a free format. Each line consists of 3 integers followed by a real number (the (co)variance value). The first integer is the random effect number followed by the row and column numbers and by the (co)variance parameter. Only the lower (or upper) triangle of the matrix is given.

Order of the lines in the file is irrelevant. However, it is very important to know correct numbering of the (co)variances. Close attention should be paid to numbering within a random effect when a random effect is modeled with several factors. For instance, multi-trait random regression models or multi-trait maternal/paternal models describe the random effects by more than one factor per trait (e.g., polynomials, or direct plus maternal factor). Ordering of the variance components in the (co)variance matrix of a random effect goes factor-wise and within factor by trait order. This is simple for a single trait model as shown in the following example, but less obvious for a multi-trait model as shown in the second example.

***Examples for single trait model*:**
(Co)Variance components file for the example given by Schaeffer, L. R. and Dekkers, J. C. M. (1994). "Random regressions in animal models for test-day production in dairy cattle". In: *Proc. 5th World Congr. Genet. Appl. Livest. Prod.* Vol. 18, pp. 443–446.

(See Example 7.3). The random additive genetic effect in this single trait model is modeled by three correlated factors. The second random effect in the model is the random residual, which requires one variance parameter.

```
 1   1   1    44.791
 1   1   2    -0.133
 1   1   3     0.351
 1   2   2     0.073
 1   3   2    -0.010
 1   3   3     1.068
 2   1   1   100.000
 |   |   |
 |   |   └── column
 |   └────── row
 └────────── random effect number
```

***Examples for multi-trait model*:**
In the following, numbering of the variance components is illustrated by a hypothetical example for a multi-trait model. All traits have an additive genetic effect but total number of effects is different for each trait. The traits in the model are: birth weight (BW) with a direct and maternal effect, live weight (W), and slaughter weight (SW). For live weight, a third order polynomial is fitted for both additive genetic and non-genetic animal effect. The RANDOM instruction lines are the following:

```
#  Non-genetic animal effect  |          Genetic animal effect
#  int   lin.   quad.   cub.        int.   lin.   quad.   cub.   mater.
   -     -      -       -           2_1    -      -       -      2_7      # BW
   1_1   1_2    1_3     1_4         2_2    2_4    2_5     2_6    -        # W
   -     -      -       -           2_3    -      -       -      -        # SW
```

The subscripts indicate the order of the variances in the variance-covariance matrices. The size of the variance-covariance matrices are 4×4, 7×7, and 3×3, for non-genetic, genetic, and residual effect, respectively.

3.4 Multiple residual (co)variances (RESFILE)

When multiple residual (co)variances are modeled (see DATASORT in chapter 4), a separate residual (co)variance file has to be given. Format of this file is similar to the regular (co)variance file explained above. However, the first number on a line is not the random effect number, but the residual variance class number. Numbering of the multiple residual (co)variances has to start from one (1), up to the total number of residual (co)variances. The number of residual (co)variance matrix in the file is referenced in the data file. Note that a residual (co)variance matrix has to be given in the (co)variance matrix file. These values are used when calculating reliabilities (see Approximate reliabilities using ApaX in *Technical Reference Guide for MiX99 Solver*). However, these residual values are ignored by the solver program.

3.5 File with a covariable table

A covariable table file is needed only if there are regression effects in the model, and the program is instructed to read covariables from a covariable table. Certain models

have several covariables with only limited number of values, e.g., polynomials at age or days in milk. These covariables can be stored to a separate table accessed by an index in the data file in order to make the data file smaller. When the number of covariables indexed by the same variable is large, such as in random regression test day models, savings in storage capacity can be considerable.

The covariable table file has the following design: The first column contains the integer index of the covariable line. The index connects an observation in the data file to the corresponding set of covariables in the covariable table. For example, in test-day models this index is often days in milk. The following columns have the covariables. They must be real numbers and the columns must be separated by one or more spaces. The rows have to be sorted in ascending order by the covariable index column. Within the smallest and largest index line, index lines must not be missing; i.e., number of rows is largest index minus smallest index plus one. The file can also contain covariable columns and rows that are not used in a particular run.

When a non-linear model with ***Gompertz function*** is defined, a covariables file is always needed. The index connecting an observation to a set of covariables is usually time of measurement. The rest of the file content for non-linear models differs from those of the linear random regression models. Because the ***Gompertz function*** has three parameters, first three covariable columns contain only ones. After these, a column with the time variable is set. This time can be original time of measurement, but also a scaled time. Time scaling is needed to improve convergence in case of variance component estimation, because variance components for maturation rate are usually very small. A good choice would be to select time scaling so that estimate for maturation rate is approximately one.

4 MiX99 instruction file

As explained earlier there are two alternatives to instruct `mix99i`. Either by a CLIM command file, which is recommended (see manual for *Command Language Interface for MiX99*), or by the ***MiX99 instruction file***. Setting up the ***MiX99 instruction file*** does not provide the ease and freedom in specifying an analysis as CLIM commands do. However, because the ***MiX99 instruction file*** operates closer to the MiX99 kernel, it always covers all the newest models and options possible with MiX99. Note, in case you instruct `mix99i` by a CLIM command file a ***MiX99 instruction file***, named `MiX99_DIR.DIR` is generated. In this chapter we describe how to set up the ***MiX99 instruction file***.

We recommend the use of a template when setting up a ***MiX99 instruction file*** for the first time. You can find examples of different ***MiX99 instruction files*** in chapter 7 of this technical reference guide. Further, complete examples of MiX99 analyses are included in the MiX99 package. These examples include all necessary information to perform an analysis with MiX99 and give an idea about the variety of models supported by MiX99. Instruction files in the provide examples are named with the suffix **.mix**. However, any name can be given. The ***MiX99 instruction file*** contains all data specifications, the specifications for the applied statistical models, and all parameters that control MiX99. The instruction file is read by `mix99i` from standard input.

4.1 Instruction lines

The MiX99 instruction file consists of instruction lines asked by the program. The instruction lines must be given in the same order presented here. Parameters on an instruction line must be given in a free format, i.e. each parameter must be separated by one or more spaces. An instruction line must not exceed 500 characters. The file can contain as many comment lines and empty lines as wanted. A comment line begins with a **#** character. Within lines, comments can be given after the instruction parameters. For all character instructions, both small and capital letters are allowed.

In the following, a name is given for each instruction. This is followed by a description of all the parameters that can be specified on the particular line(s) of the instruction. For clarity, we use the following terminology from here onwards: ***class variable*** is an explanatory variable in the model that classifies the observations into classes with the same treatment (e.g. age, herd, animal); ***covariable*** is an independent variable in the model that is associated with a regression coefficient (e.g. x, x^2, x^3, e^{-kx}, etc.); ***factor*** can be both, either a class variable or a covariable; ***effect*** is a causal reason affecting the dependent variable (***trait***) and can be modeled by one or several fixed or random ***factors***; an ***effect*** can be modeled as fixed or as random and for each random effect one (co)variance matrix will be provided; the ***numbering of data columns*** starts from one, separately for the integer number columns and for the real number columns of the data file.

TITLE One line with the title of the analysis.

INTEGER A line defining the names of all integer variables in the data file. A name can contain all characters excluding **#** and &. There is no restriction for the length of a name. However, MiX99 uses the first 8 characters of the name only. It is not allowed to give identical names to different integer columns.

The line can be continued by giving the "**&**" character at the end of the line (separated with at least one space from the last word in the line).

REAL A line with the names of all real variables in the data. The same editing rules are applied as for INTEGER.

TRAITS A line with one entry that specifies the number of traits to be analyzed. Optionally, a character **I** or **L** can be specified after the number of traits. This will generate additional output information during pre-processing.

TRAITGRP Two entries to define the trait group information. The first entry is the number of trait groups in the model. The second entry indicates the integer column with the trait group code in the data file. If a grouping of traits is not desired the first entry must be set to 1 and the second entry to dash (-). See also Examples 7.4 and 7.6 on how to specify models with the trait groups.

DATASORT Two or three entries: The first indicates the integer data column of the block code, and the second indicates the integer data column of the relationship code. The third (optional) entry indicates the integer data column with the coding of the residual classes (see multiple residual variances).

Dashes (-) have to be given if the data is not sorted by the block code or by the relationship code. If a dash is specified for the block code, MiX99 will automatically block the pedigree and data by blocks of 30000 records. Then, block code information in the data and pedigree files is no longer needed. However, equations will not be ordered by equation families, which can significantly reduce parallel computing performance as well as the quality of the reliability approximation. If a dash is specified for the relationship code, relationship information will be stored for each record, rather than for a block of records with the same relationship information. This is reasonable when animals do not have repeated observations. In the case of repeated observations, computing time and disk usage will increase.

The third entry is optional. It is the column number in the data file that indicates the number of the residual (co)variance matrix used when multiple residual (co)variances are applied. They have to be numbered in ascending order starting with one (1). The matrices have to be given in a separate file, which is specified in the RESFILE line. The setup of the file is explained in chapter 3.4 describing multiple residual variance-covariance matrices. This option is not implemented for the non-linear Gompertz function models.

FIXRAN Four entries, of which the last two are optional. The first entry indicates the number of all factors for the fixed effects on the MODEL line(s). The second entry indicates the number of all factors for the random effects on the MODEL line(s).

The sum of entry 1 and entry 2 must be equal to the number of factors specified in the MODEL line(s). For multi-trait models, additional information on how to count the number of factors is explained in the MODEL

instruction. Regression effects not nested within a class variable must not be included in the FIXRAN and MODEL specifications. Such regressions are specified on the REGRESS line(s) only.

The third (optional) entry gives the number of random effects, other than genetic animal effect, for which a correlation structure between effect levels is modeled. The correlation structure must be provided in an extra file as an inverse correlation matrix, e.g., an inverse of the IBD matrix. The file name must be specified on the CORRFILE instruction line.

The fourth (optional) entry is needed for genomic data models only and gives the number of across-data regression design matrices in the design matrix file. This is most useful where the number of effects can be many thousands. For this specific case, all columns (for subset see the file names given in REGFILE) in a design matrix file are considered regression effects, and these effects are not expressed on the MODEL line(s) nor on the REGRESS line(s).

MODEL One line for each trait. Each line contains several entries. The first entry indicates the trait group number. Most often, this is one (1). If traits are arranged in groups, MODEL lines have to be ordered by the trait group numbers. Please note, MiX99 does not allow to model for the residual effects co-variance between traits of different trait groups. The second entry indicates the real data column of the trait (dependent variable) to be analyzed. Note, MiX99 numbers the integer data columns of the data file starting from one and the proceeding real data columns of the data file starting from one as well. The third entry indicates a real data column of a weight variable. A dash (-) must be given for this entry if weighting is not applied for a trait. The rest of the MODEL line specifies the integer data columns of all factors in the model. Information on the fixed effects must be given first, then on the random effects. The factors for the genetic effects, i.e., the effects for which the relationship matrix is applied, must be specified last on the MODEL line(s). The number of factors for the fixed and random effects must be identical to the numbers specified on the FIXRAN line.

For ***multi-trait models***, a MODEL line is given for each trait. There can be factors which are the same for several traits but there can be also factors which are specific to one trait only. However, there can be only one and the same factor in a "MODEL column". In the case of multi-trait models, the MODEL instructions are considered in a matrix setup with lines for the traits and columns for the model factors. Thus, each factor occupies one "MODEL column". An entry has to be given for all traits in each column. In the case a factor is not modeled for a trait, a dash (-) must be specified (see Example 7.2).

If it is the aim to nest ***regression effects*** within classes the class effect has to be specified in the MODEL line(s). The covariable (independent variable) will be defined in the REGRESS instruction line(s). For example, a lactation curve is nested within season classes and is modeled as

$season + d(season) + d^2(season) + e^{-005d}(season)$, and the season classification is given in integer data column 6. Then, the integer data column of the season classification has to be included four times on the MODEL line (in this example: 6 6 6 6). If these four numbers are specified in the fixed factor columns of the MODEL line they will be treated as fixed regression effects, whereas specified on the random factor columns of the MODEL line they will be treated as random regression effects. The covariables for the four factors will be defined on the corresponding REGRESS line. No limit is set on the number of covariables.

MiX99 allows solving of ***LS models*** and ***GLS models*** with any number of effects. For such models number of random factors in FIXRAN is set to zero and instructions for RANDOM, RELATIONSHIPS, PEDIGREE, and PEDFILE are skipped. For ***GLS models*** a file with the residual (co)variance matrix needs to be specified in PARFILE.

LS models with data blocking. For solving LS-models with many observations, computations might require ordering the data using a block code. Therefore, a data blocking variable has to be provided in a pseudo pedigree file. For setting up such a model, the last effect in the model needs to be defined as an "operational" random effect. Specifying the option `ls+b` in PEDIGREE will instruct MiX99 to solve the effect as a fixed effect. The pseudo pedigree file must contain for each level of the last fixed effect one row with the following four integer numbers: level_code 0 0 block_code. This option is useful if an LS-model is used for the estimation of heterogeneous variance. (for more information, see the *Technical Reference Guide for MiX99 Solver*).

Non-linear Gompertz function model: A multiplicative Gompertz function model (i.e. $\ln(y) = \ln(a) - b * \exp(-k * t) + e$) as demonstrated in Vuori et al. (2006) is implemented in MiX99. For this, observations should be log-transformed. Model specification for the non-linear model is closely related to the specification of the regression models. Few additional features and restrictions exist.

First, non-linear traits are defined on the model line by adding an extra entry with the character "G" after the first entry of the MODEL line. Second, since there are three parameters in the ***Gompertz function***, factors affecting must be defined thrice (See Example 7.8 and REGRESS section to specify regression effects correctly).

Currently, all non-linear traits introduced in the analysis must have the non-linear ***Gompertz function*** form. Additionally, for all traits, at least one fixed effect needs to be related to all three parameters. This effect is defined first in the MODEL line, and it should be defined as an acrossblock effect (see WITHINBLOCKORDER). If many such effects occur, the effects with the smallest number of levels are recommended to be defined first in the MODEL line.

Models with one categorical trait (threshold model solved by GLMM with probit link function): The categorical trait is defined by adding an

extra entry with the character "**Tn**", where **n** is the number of thresholds, after the first entry on the MODEL line. For example, for a binary trait (records are 1 and 2) the extra entry is in the form **T1**. For the multi-trait models, linear traits are defined first (see Example 7.9). When the threshold model is defined, the following additional instruction lines with two entries must be given:

THR_MHD Define the solving method to be used for the threshold models. Two options exist:

- **EM** Expectation-Maximization algorithm (Gilmour and Thompson, 1998)
- **NR** Newton-Raphson algorithm (Janss and Foulley, 1993; Hoeschele et al., 1995)

By default, thresholds are estimated simultaneously. The second entry **ft** is optional and is needed if fixed threshold values are specified. Then, one additional line with fixed thresholds must follow.

THR_VAL Needed when option **ft** is specified on the previous line. Define the threshold values for the categorical trait. As many real numbers as thresholds are defined in the model line.

WITHINBLOCKORDER One line with as many entries as there are fixed and random factors specified on the MODEL line. The order of the entries corresponds with the order of the factors specified on the MODEL line(s). All effects for which the corresponding equations in the MME should be ordered by the blocking variable must be marked with a positive integer number (see equation families in chapter 3.1.1). These are usually the effects with a large number of levels. All other effects must be marked with a dash (-). The positive integer number must be specified from 1 up to N. The integer number describe the order of the equations within a block. Equations for the effect numbered with 1 will be placed at the beginning of each block and those with the highest WITHINBLOCKORDER number will be placed at the end of each block. For the computation of approximated reliabilities, the WITHINBLOCKORDER number 1 has to be specified for the genetic animal effect. Otherwise, the order of the effects within blocks is arbitrary. If a random effect includes several factors (e.g. random regression function, models with a maternal effect, etc.), then the same WITHINBLOCKORDER number must be given for all the factors. For the non-linear ***Gompertz function***, a dash (-) must be specified for the first effect.

Each block sorting variable in the pedigree file represents one block of the mixed model equations. Therefore, a BLOKORD integer number must be specified for the additive genetic animal effect in the model. For other effects in the model, it is optional whether equations should be ordered within blocks. In many cases, when the data is small, it is not critical whether an effect is in ***within block***s or ***across block***s. If the analyzed

data is large, for parallel computing, or for approximation of reliabilities, the right setup of the WITHINBLOCKORDER option is essential. Block ordering is often obvious for some models. For example, a model for milk yield in dairy cattle has the following fixed effects: herd×test-day, age, a function on the days in milk; and the following random effects: non-genetic animal environment effect, genetic animal effect, and residual. The block sorting variable should be the herd. Then, it is advisable to order equations for the herd×test-day, non-genetic animal environmental, and genetic animal effects within blocks. Equations of the age effect and the stage of lactation effect are connected with equations from all the herds and therefore should be ordered across blocks.

If it is not obvious which effects are best to order within blocks, or if there is no clear animal family structure in the data, one strategy is to group random and fixed effects which have a large number of class levels within blocks. Note that ordering an effect within blocks does not mean that the effect cannot apply across blocks. However, a bad blocking strategy may inhibit efficient use of parallel computing.

Combining model factors within or across traits: MiX99 allows you to combine factors within the model line of a trait and/or across the model lines of different traits. The latter is described in the COMBINE and MERGE instruction and the former is described in the WITHINBLOCKORDER line. Each entry in the WITHINBLOCKORDER line corresponds to a factor "column" in the MODEL line(s). Placing a "<" character between two entries on the WITHINBLOCKORDER line will combine the corresponding factors on the MODEL lines. In that case, levels of both factors will be renumbered together as if they would belong to one effect only. An obvious example would be a QTL effect. Two marker alleles can be modeled as a genotype effect by combining the allele effects. Then, an IBD matrix can be associated with the genotype effect if desired.

Rules for combining factors within trait(s): The combined factors must be ordered either within blocks (must have an integer number) or across blocks (must have a "-" character). Combining is allowed only between random factors or between fixed factors. No limitation exists on the number of factors to be combined. Once combined, the factors cannot be modeled as separate effects for some traits in multi-trait analyses. This is because combining is applied between "MODEL factor columns". Hence, an effect modeled for three traits cannot be combined with an effect modeled for two traits. The following WITHINBLOCKORDER line for a model with 13 model factors instructs to combine two across block factors and five within block factors:

```
- - 7 8 - < - - 6 < 5 < 4 < 3 < 2 1
```

Combining random factors other than the additive genetic animal effects: The size and structure of the (co)variance matrix must be the same for all random effects that should be combined. For instance, if two random effects are modeled with the same number of regression coefficients, they can be combined. Let's have a model with three fixed and three random

effects, where two random effects are modeled by a 2nd order polynomial. The random effects are ordered within blocks with the block order numbers 3, 2, and 1 as follows: " - 4 - 3 3 3 < 2 2 2 1 ". Regardless of whether random effects are combined or not, they should be modeled on the RANDOM line as if no combining was applied. However, only one (co)variance matrix needs to be provided in the parameter file. (Co)variance matrix must be given for the first random factor of the combined random factors. If desired, also the inverse correlation matrix (e.g., inverse IBD matrix, autocorrelation matrix, etc.) can be specified for the first random factor of the combined factors (see CORRFILE).

Combining factors within the additive genetic animal effect: MiX99 allows you to combine factors within the additive genetic animal effect. This is useful for models comprising social effects like in group selection. Instructions for combining different factors are given on the WITHINBLOCKORDER line. Placing "<" characters between the corresponding entries on the WITHINBLOCKORDER line will instruct to combine the corresponding additive factors. Combining additive genetic animal factors is possible only between factors with different relationship information (different animals). Hence, combined factors must have different relationship information codes on the RELATIONSHIPS instruction line. Further, combining factors within the additive genetic animal effect must be consistent with the RANDOM instruction line. A factor combined with another one is specified with a dash "-" on the RANDOM line. For instance, assuming a model with two fixed effects, one random environmental effect, a direct animal effect and three indirect animal effects of pen mates, will need specifications on the WITHINBLOCKORDER, RANDOM and PEDIGREE lines which look as follows:

```
#MODEL:  stable sex litter animal mate1 mate2 mate3
          6     12    8       2     4     5     6
#WITHINBLOCKORDER:
          3     -     2       1     1  <  1  <  1
#RANDOM:
                      1       2     2     -     -
#RELATIONSHIPS: number of add. factors
         4                    1     2     3     4
```

For this model, the variance-covariance matrix for the additive animal effects would be of size 2×2 including variances for the direct animal and the pen mate effects and the covariance between them. Combining factors within the additive animal effect is also possible, if each effect is modeled by a function. Considering the previous example and assuming that each animal effect is modeled with two regression coefficients will lead to the following instructions:

```
#MODEL:  stable sex   litter   animal   mate1   mate2   mate3
#                    c1   c2  c1   c2  c1  c2  c1  c2  c1  c2
            6    12   8    8   2    2   4   4   5   5   6   6
#WITHINBLOCKORDER:
            3    -    2    2   1    1   1   1 < 1   1 < 1   1
#RANDOM:
```

```
                        1    1   2    2   2   2   -   -   -   -
#RELATIONSHIPS: number of add. factors
             8                  1    1   2   2   3   3   4   4
```

Observations with a variable number of social effects: The number of pen or cage mates may vary for observations. MiX99 still does not accept observations with missing effect information. Currently, the only way to circumvent this shortfall is to model a dummy id for missing mates and apply a zero covariable to the factor of the dummy mate. Implementation of such a model can be studied in more detail from Example 7.11 given in this technical reference guide and from the example provided with the software.

RANDOM One line per trait following the same order as the MODEL lines. Each line must contain as many entries as there are random factors defined on the FIXRAN line. If a random effect is not included in the model of a particular trait, a dash (-) must be specified whereas numbering is used to indicate which random factors are correlated. Consequently, correlated factors must be specified with the same number. For instance, if the genetic effect is described by a function with four factors, they all must be numbered identically. The consecutive ordering of the random effects begin from one. The factors of the genetic effect must have the highest number. The numbers correspond to the numbering of the (co)variance component matrices in the file with the (co)variance parameters. (see Examples).

For certain ***reduced rank models*** one might like to combine random factors across traits. In this case, the reduced rank model has to be specified on the RANDOM lines. The reduced rank model has to match with the applied (co)variance matrices and the specifications in the COMBINE section (see also MERGE section and Example 7.6).

If a LS-model with data blocking is specified the last effect in the MODEL line section must be defined as an “operational” random effect for technical reasons, even it will be treated as fixed effect.

RELATIONSHIPS One line with a relationship identification number for each factor of the genetic effect. Order of the specified relationship identification numbers correspond to the order of factors (or factor columns in case of multi-trait models) on the MODEL/RANDOM line(s). The first entry is the number of factors associated with the genetic effect. After this, a relationship identification number is given for each factor. Factors with the same relationship identification number must be grouped together. Numbering must start from one (1). If the same relationship information applies to all factors, only ones (1) are specified. This is the usual case. If there is only one direct genetic animal effect, then the RELATIONSHIPS line contains two ones (1 1). If the genetic animal effect is modeled by five random regression coefficients, then the RELATIONSHIPS line comprises of the integers 5 1 1 1 1 1. If there is a maternal or a paternal effect or both in the model, one needs to specify which factors have the same relationship information. If there is a direct animal effect and a maternal effect, then the RELATIONSHIPS line comprises of the integers 2 1 2.

If the direct animal effect is described by three factors and the maternal effect by two factors, then the RELATIONSHIPS line instruction would be: 5 1 1 1 2 2.

Inbreeding coefficients. Calculation rules for inverse of the numerator relationship matrix $\mathbf{A}^{-1}$ in the mixed model equations assuming zero inbreeding coefficients by default. Inbreeding coefficients can be accounted in the computations by reading an inbreeding coefficients file. Three extra instruction information need to be given. First, on the relationship instruction line, character 'y' need to be given after the integers explained above, e.g., 5 1 1 1 2 2 y. On the following line, column numbers of the ID code and the inbreeding coefficient in the inbreeding coefficients file are given. Note that the column number of the ID code must be less than the inbreeding coefficient column. Finally, name of the inbreeding coefficients file is given.

Example for MiX99 instructions for inbreeding:

```
# PEDIGREE
5 1 1 1 2 2 y
# INBREEDING: columns of ID and inbreeding coefficients
1 3
# INBRFILE: name of inbreeding coefficients file
data.inbr
```

Ignoring the relationship between animals. Specifying only the character "I" or "i" on the RELATIONSHIPS instruction line will instruct MiX99 to ignore the numerator relationship matrix ($\mathbf{A}^{-1}$) for the last random effect in the model. For such a model no pedigree file has to be prepared and the instruction line PEDFILE is skipped.

External inverse relationship matrix. Genomic ***GBLUP*** model can be evaluated by including external inverse relationship matrix. Specifying only the character "f" on the RELATIONSHIPS instruction line will instruct MiX99 to read the inverse relationship matrix (such as inverse genomic relationship matrix $\mathbf{G}^{-1}$) from a file given on the following line. The matrix has to be stored in co-ordinate format where every line has three numbers. The first two numbers are row and column numbers, and the third is element value in the matrix. The row and column numbers refer to animal codes in the data file. Only lower triangle of the matrix should be stored. MiX99 preprocessor checks that the matrix has lower triangle elements only. This check can be ignored by giving character "m" or "M" after "f", i.e., "fm" or "fM". If there are several instances of the same row and column numbers even in symmetric positions, then all values will be summed. Hence, having lines

Matrix file:

Row_1	Column_2	Value_3
712	12	10
712	12	20
12	712	30
⋮	⋮	⋮

is effectively the same as

Matrix file:

Row_1	$Column_2$	$Value_3$
712	12	60
⋮	⋮	⋮

Note that no instruction line PEDFILE should be given, but the relationship identification line information explained above is needed. Note that in earlier versions of MiX99 the relationship identification information was not needed.

There are two approaches to improve performance in use of the $\mathbf{G}^{-1}$ file:

1) The $\mathbf{G}^{-1}$ matrix file can be in unformatted binary form. Often this leads to smaller disk memory usage, faster reading and writing, and more accurate element values in the disk. MiX99 assumes a $\mathbf{G}^{-1}$ matrix file is an unformatted binary file when its suffix is '.bin'.

2) The $\mathbf{G}^{-1}$ can be stored in a more efficient format. The co-ordinate file format for the genomic relationship matrix given above can be inefficient when the number of genotyped animals is large. In the co-ordinate file format, each non-zero element in (lower triangle of) a matrix has a row, and every line in a file has three numbers: row number, column number, value. An alternative matrix format called lower triangle dense format has several advantages over the co-ordinate format:

   (a) the external file takes less disk space,

   (b) computations are faster,

   (c) when parallel computing is used (single-step model!), the genotyped animals need not be in the common block area.

   An example for a 10 by 10 inverse genomic relationship matrix in lower triangle dense format:

```
 10 0
 1 2 3 4 5 6 7 8 9 10
  7.16
-11.32 19.50
 -4.05  6.15  3.70
  2.41 -4.78 -1.24  2.11
  2.01 -3.57 -1.28  0.82 0.86
  2.79 -5.48 -1.09  1.80 1.01  2.17
  5.39 -8.39 -3.31  1.39 1.47  1.76  4.45
 -0.23  0.78  0.41 -0.22 -.29  -.31  -.20 0.34
 -0.30  0.54 -0.22 -0.35 0.01  -.28  -.11 -.17 .29
 -2.27  4.13  0.71 -1.38 -.71 -1.54 -1.42 0.06 .31 1.33
```

Program `hginv` allows making lower triangle dense format matrices by option "-lower" and produces an unformatted binary file suitable for MiX99 when the output file has suffix ".bin".

In MiX99, the lower triangle dense format is indicated by letter "L". Giving 'fL' instead of 'fM' indicates that the file has lower triangle dense format.

In summary: GBLUP is signaled by letter 'f' after which there is space, letter 'M' or letter 'L'. Here are the alternatives:

GBLUP with $\mathbf{G}^{-1}$ in a co-ordinate format file:

```
# RELATIONSHIPS
f  # external inv(G) file
iG_10.dat
```

GBLUP with $\mathbf{G}^{-1}$ in a co-ordinate mixed format file:

```
# RELATIONSHIPS
fM
iG_10.dat
```

GBLUP with $\mathbf{G}^{-1}$ in a lower triangle dense format file:

```
# RELATIONSHIPS
fL
iGL_10.dat
```

Single step. Single-step method has both the external matrix and pedigree information. The pedigree information is given as for standard animal model, i.e., the pedigree identification information (see examples) and the pedigree file PEDFILE has to be given. The additional information due to the single-step method is matrix $\mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$ where $\mathbf{G}$ is a genomic relationship matrix and $\mathbf{A}_{gg}$ is pedigree-based relationship matrix of genotyped animals. The preprocessor program is instructed to take an external file by giving information on the RELATIONSHIPS instruction line. This information can be given in several ways:

A single file having the matrix $\mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$ is indicated by giving "g" or "G" after the "f" character for external inverse relationship matrix, e.g., "fg". MiX99 preprocessor checks that the matrix has lower triangle elements only. No check is done when character "m" or "M" has been given after "fg", i.e., "fgm" or "fgM". For the lower triangle dense format "L" is given instead of "M", i.e., "fgL". After the lines "fg" and name of the external file, the pedigree identification information is given as explained above. Similarly as for the GBLUP model, the $\mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$ matrix file can be in unformatted binary form and is indicated by suffix ".bin".

Single step GBLUP with $\mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$ in a co-ordinate format file:

```
# Single step inv(G)-inv(Agg)
fg      # external file (lower triangle) and relationship matrix
iH.dat
# Standard pedigree-based relationship information
1   1
```

Single step GBLUP with $\mathbf{G}^{-1}-\mathbf{A}_{gg}^{-1}$ in lower triangle dense format file:

```
# Single step inv(G)-inv(Agg)
fgL     # external file (lower triangle) and relationship matrix
iHL.dat
# Standard pedigree-based relationship information
1   1
```

An alternative is to give the inverse relationship matrices $\mathbf{G}^{-1}$ and $\mathbf{A}_{gg}^{-1}$ in separate files. Then, letters "ssi" need to be given instead of "fg" on the RELATIONSHIPS instruction line. After this line, file having $\mathbf{G}^{-1}$ is given, and then file having $\mathbf{A}_{gg}^{-1}$.

Single step GBLUP with separate $\mathbf{G}^{-1}$ and $\mathbf{A}_{gg}^{-1}$ files:

```
# Single step: separate inv(G) and inv(Agg)
ssi     # single-step with inverse files
iG.dat
iAgg.dat
# Standard pedigree-based relationship information
1   1
```

The minus sign needed in the calculations before the $\mathbf{A}_{gg}^{-1}$ matrix ($\mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$) is performed by the solver program.

Note that for "ssi", MiX99 preprocessor assumes that the matrix has lower triangle elements in a file with co-ordinate format, and no checks on this are made. Do not give "ss" only because this will lead to slow incorrect single-step if inverse matrices are given.

A more efficient variant of the "ssi" option is to instruct the solver to do the required calculations of $\mathbf{A}_{gg}^{-1}$ without giving any file having this matrix. A reserved word PEDIGREE is given instead of a file. The solver has alternatives (see the solver manual, *Technical Reference Guide for MiX99 Solver*) for making these calculations.

Single step GBLUP with $\mathbf{G}^{-1}$ in file and $\mathbf{A}_{gg}^{-1}$ by solver computations:

```
# inv(G) file given but inv(Agg) calculated by the solver
ssi     # single-step with inverse inv(G) in file
iG.dat
PEDIGREE
# Standard pedigree-based relationship information
1   1
```

Lower triangle dense format is used with "L" option, i.e., "ssiL".

Single step GBLUP with prepared $\mathbf{G}^{-1}$ and $\mathbf{A}_{gg}^{-1}$ by solver:

```
# inv(G) file given but inv(Agg) calculated by the solver
ssiL    # single-step with inverse inv(G) in file
iGL.dat
PEDIGREE
# Standard pedigree-based relationship information
1   1
```

When the "PEDIGREE" approach is used for single-step, the preprocessor does not compute explicitly the $\mathbf{A}_{gg}^{-1}$ matrix. Thus, it is not readily available

to the preconditioner. By default, the diagonal of the $\mathbf{A}_{gg}^{-1}$ matrix is computed by a Monte Carlo approach. When the same genotyped individuals are used in many evaluations this step is done is several times. In order avoid, this, a separate program called `calc_diag_iA22` is available that calculates the diagonal elements of $\mathbf{A}_{gg}^{-1}$. Then, diagonal values of $-\mathbf{A}_{gg}^{-1}$ can be provided to the preprocessor by giving letter 'p' as the tenth letter on the command line when ssGBLUP or ssGTeBLUP is used. Note that ssGTABLUP needs the diagonal of the $(\frac{1}{w}-1)\mathbf{A}_{gg}^{-1}$ matrix where w is the residual polygenic proportion. For some more information, see the `IHPRECON` command in the *Command Language Interface for MiX99* manual.

Additional preconditioner information in file indicated by 'p':

```
# inv(G) file given but inv(Agg) calculated by the solver
ssi      p # single-step with inverse inv(G) in file
iG.dat
PEDIGREE
minus_diA22.dat  # minus diagonal of inv(Agg)
# Standard pedigree-based relationship information
1   1
```

The external file can be in lower triangle dense format like was the case in GBLUP. This is often faster and takes less disk space when the matrix is very dense. Like as for GBLUP, this format is indicated by letter "L".

Lower triangle dense binary file and additional precond. information:

```
# inv(G) lower triangle dense format, inv(Agg) calculated by solver
ssiL     p # single-step with inverse inv(G) in file
iGL.bin
PEDIGREE
minus_diA22.dat  # minus diagonal of inv(Agg)
# Standard pedigree-based relationship information
1   1
```

Single-step computations using the ssGTeBLUP "Efficient single-step genomic evaluation for a multibreed beef cattle population having many genotyped animals" formulation can be done by giving letters "TeL" instead of "ssiL". This assumes that the genomic relationship matrix has form $\mathbf{G}_\epsilon = \mathbf{Z}_c\mathbf{D}\mathbf{Z}_c' + \frac{1}{\epsilon}\mathbf{I}$ where ϵ is a small number, e.g., $\epsilon = 0.01$, $\mathbf{Z}_c$ is centered marker matrix, $\mathbf{D}$ is a scaling matrix. Use of Woodbury matrix identity allows expressing the $\mathbf{G}_\epsilon^{-1}$ conveniently in form $\mathbf{G}_\epsilon^{-1} = \mathbf{I}\frac{1}{\epsilon} - \mathbf{T}_e\mathbf{T}_e'$. By giving "TeL", the external $\mathbf{T}_e$ matrix is assumed to be a dense rectangular matrix. Note that there are some additional information provided on the first line of the T matrix file. Thus, the T matrix is easiest to make using a dedicated program. Similarly as for the regular ssGBLUP with the PEDIGREE option, diagonal of minus the $\mathbf{A}_{gg}^{-1}$ matrix is good to use for the preconditioner update.

ssGTeBLUP with preconditioner correction due to $-\mathbf{A}_{gg}^{-1}$:

```
# T matrix in dense rectangular format, inv(Agg) calculated by solver
TeL      p # T matrix dense rectangular
TeMatrix.dat
PEDIGREE
```

```
minus_diA22.dat
# Standard pedigree-based relationship information
1   1
```

An alternative called ssGTABLUP allows including so-called residual polygenic part efficiently. In practice, the inverse genomic relationship matrix has now form $\mathbf{G}_w^{-1} = \frac{1}{w}\mathbf{A}_{gg}^{-1} - \mathbf{T}_A\mathbf{T}_A'$, where w is the residual polygenic proportion. In MiX99, there are two approaches. A similar to the ssGTeBLUP approach given above is use "TAL" instead of "TeL". The other approach uses component computations and requires two files instead of one ($\mathbf{Z}_c$ and $\mathbf{C}^{-1}$) and have command "TAZc" and is available only in mix99s. Both of these will be described below. CLIM has commands for these approaches as well (TAFILE for the first, and `ICFILE` and `ZCFILE` for the second).

Note that when the preconditioner correction is made due to the diagonal of $\mathbf{A}_{gg}^{-1}$, a full correction requires giving in the file having diagonal values of $(\frac{1}{w} - 1)\mathbf{A}_{gg}^{-1}$. In the example below the $\mathbf{T}_A$ matrix has been precomputed and stored in a (Fortran unformatted) binary file.

ssGTABLUP with preconditioner correction due to $(\frac{1}{w} - 1)\mathbf{A}_{gg}^{-1}$***:***

```
# T matrix in dense rectangular format, inv(Agg) calculated by solver
TAL      p # T matrix dense rectangular
TAMatrix.bin
PEDIGREE
diA22_TA.dat
# Standard pedigree-based relationship information
1   1
```

Single-step has been implemented in the parallel solver ***mix99p***. However, when the co-ordinate format in ssGBLUP is used with "MES" option, all genotyped animals need to be in the common blocks, see COMMONBLOCKS. When co-ordinate format has been used, the parallel solver tries to find the non-common block individuals and write them in file `MIX99_NON_COMMONBLOCK.TXT`. If the program does not succeed in finding such individuals (but there is one), the parallel solver is likely to stop to ListFinds error. It is recommended to use the lower triangle dense format to avoid these kind of problems. The ssGTBeBLUP and ssGTABLUP do not suffer from these problems as no co-ordinate format can be used with these approaches.

Single-step GTABLUP has been implemented using the "TAL" approach as described above. In the "TAL" approach, the T matrix for the ssGTABLUP model is computed and used. An alternative is to compute the component matrices used in ssGTABLUP. This is computationally less demanding for the preprocessor. However, this leads to some increased computing time in the solver but the overall computing time is often lower.

In the new component-wise ssGTABLUP, it is assumed that the genomic relationship matrix has form $\mathbf{G}_w = \mathbf{Z}_c\mathbf{B}\mathbf{Z}_c' + w\mathbf{A}_{gg}$ where

1) w is weight for the residual polygenic proportion,

2) $\mathbf{B} = \mathbf{I}\frac{1-w}{k}$ is weight matrix for the markers,

3) k is a scaling factor,

4) $\mathbf{Z}_c$ is centered marker matrix,

5) $\mathbf{A}_{gg}$ is pedigree-based relationship matrix for the genotyped.

The component matrices needed in MiX99 are

1) $\mathbf{Z}_c$ and

2) $\mathbf{C}^{-1} = (\frac{1}{w}\mathbf{Z}_c'\mathbf{A}_{gg}^{-1}\mathbf{Z}_c + \mathbf{B}^{-1})^{-1}$

An advantage of the component-wise ssGTABLUP over regular ssGTABLUP implementation is that marker effect solutions are computed as a by-product to a file called SolSNP. Consequently, it is possible to compute estimates for the newly genotyped individuals when their genotypes are available using the marker effect and parent GEBV solutions. A separate program called `predict_GEBV` allows making these computations.

Assume that the $\mathbf{Z}_c$ matrix is in file Zc.bin, and the $\mathbf{C}^{-1}$ in file iC.bin, i.e., both have been stored as a (Fortran unformatted) binary file.

The new ssGTABLUP with preconditioner correction for $\mathbf{A}_{gg}^{-1}$:

```
# T matrix in dense rectangular format, inv(Agg) calculated by solver
TAZc     p # T matrix dense rectangular
Zc.bin
iC.bin
PEDIGREE
diA22_TA.dat
# Standard pedigree-based relationship information
1   1
```

Note that here the same preconditioner correction file was used as for the standard ssGTABLUP approach. This is not optimal but for some data and model, this has given as good convergence as having the diagonal of $\mathbf{G}_w^{-1} - \mathbf{A}_{gg}^{-1}$.

The approach with "TAZc" described above reads the centered marker matrix $\mathbf{Z}_c$ to RAM. This matrix can be large because it is stored in double precision. An alternative is to use a lower RAM version which reads the original marker matrix to RAM in integer 1 byte, but the additional information is needed for centering to be done by the solver. Thus, instead of one Zc file, there is a need for two files: the marker matrix and the centering information for each marker. The "iC" file is still needed. We call this approach the fully component-wise ssGTABLUP.

An example of the approach:

The fully component-wise ssGTABLUP using marker matrix:

```
TAZ1
markers.dat
2 50241 f 2
(i10,26x,50240i1)
AF.dat
iC.bin
```

```
PEDIGREE
```

Here, after the "TAZ1" command, the file name for having the markers is given. Next, there is information on which columns have the accepted markers, now from columns 2 to 50241, i.e., 50240 markers. ID code is always assumed to be in the first column. Then follows a character 'f' indicating that a reading format is used, which is given on the following line in Fortran format style. Instead of 'f', currently, the only other option is 'm' which means space-separated markers and no need to give a format line. The last line is '2' to indicate that centering uses allele frequencies in the file following the format line, i.e., now in AF.dat. The allele frequency file has 2 columns: marker number (i) and allele frequency (p_i). Note that the marker number i starts from one onwards, where one means the first marker, i.e., column 2 in the marker file. The markers are centerd by adding to them $-2p_i$.

The "TAZ1" option is given above to use ssGTABLUP. It can be considered a special case of ssSNPBLUP in "A single-step genomic model with direct estimation of marker effects". The ssGTABLUP model is reached by absorbing the marker effects in ssSNPBLUP model into the genotyped individual breeding values. The ssSNPBLUP model can be taken to use by giving further information on the scaling and residual polygenic proportion w, and removing the iC.bin file. An example of the ssSNPBLUP model:

The component-wise ssSNPBLUP model:

```
TAZ1
markers.dat
2 50241 f 2 2 2 0.2
(i10,26x,50240i1)
AF.dat
PEDIGREE
```

In addition to the number for the component-wise ssGTABLUP given above, the second line has numbers '2 2 0.2'. The first '2' here indicates that the scaling factor is $k = 2\sum_i p_i(1-p_i)$ where p_i is the allele frequency of marker i from the allele frequency file AF.dat. The second additional '2' indicates that residual polygenic proportion is given. The last number is the residual polygenic proportion w. For information on the other scaling factors, see the numbers of SCALE in the SNPMATRIX command in *Command Language Interface for MiX99*.

The above models having TAZ1 will use one byte for each SNP marker. The SNP marker matrix is read to memory by the solver program. A more memory efficient version packs several SNP markers to a byte. However, this can increase computing time in comparison to TAZ1/TAZc. The command is TAZp instead of TAZ1:

The fully component-wise ssSNPBLUP model:

```
TAZp
markers.dat
2 50241 f 2 2 2 0.2
(i10,26x,50240i1)
```

```
AF.dat
PEDIGREE
```

The ssGTeBLUP approach is avaiable as the regular component-wise (TeZc) and the fully component-wise (TeZ1) as well as packed version (TeZp). Also, the ssSNPBLUP approach can be done with the GTe approach. Then, instead of '2' as the second last number should be '1'. For example,

The fully component-wise ssSNPBLUP model with GTe:

```
TeZ1
markers.dat
2 50241 f 2 2 1 0.01
(i10,26x,50240i1)
AF.dat
PEDIGREE
```

where it is assumed that 0.01 is added to the diagonal of $\mathbf{ZZ}'$. Packed storage form is available with TeZp.

REGRESS One line per trait in the same order as the MODEL lines. The first entry gives the number of specifications following on this line. This number must be the same for all REGRESS lines. The number is the sum of regression effects applied across all observations plus the number of factors specified on the MODEL line. After the first entry the entries for across-data regression effects (optional) are given, followed by the entries corresponding to the factors specified in the MODEL line(s):

1) Entries for across-data regressions: A covariable specification for each regression effect applied across all observations. These entries are optional. However, number of entries must be the same for each trait. Therefore, if an across data regression effect is not modeled for a trait a dash (-) has to be specified.

2) Entries for the factors specified in the MODEL line(s). For each factor in the MODEL line(s) a covariable specification must be given regardless whether the factor is a class or a regression effect. The order of the specifications must correspond to the order of the factors in the MODEL line(s).

There are five types of covariable specifications:

- Whenever an effect is missing for a particular trait a dash (-) is given. Dashes must be on the same positions as on the MODEL line.

cl The model factor is declared as a class variable.

Real data column number of a covariable The corresponding factor in the model is considered to be a regression effect. The real data column **99** is reserved for cl (internal covariable of 1.0) and must not be specified.

t*n* where *n* is the covariable column number in of a covariable table file. For instance, **t**12. In this case the covariable is read from a covariable

table file rather than from the data file. The specified column (e.g. *12*) corresponds to the covariable column in the file containing the covariable table. MiX99 numbers the covariable columns beginning with one. Numbering does not include the index variable which is given in the first column (see also “File with covariables”). There is no restriction on the number of columns in the table.

`zr` Model factor is nullified. A covariable of 0.0 is applied for this factor of the model. This may be useful for model testing and validation purposes as well as in certain multi-trait models with combined effects.

If a non-linear ***Gompertz function*** will be employed, **t** must be specified for each effect, i.e., all factors must be considered as covariables read from the separate file. Real input columns of the covariables in the data file indicate the parameters of the ***Gompertz function***, to which the effects are related to (e.g. **t**1 for mature weight, **t**2 for relative initial weight and **t**3 for maturation rate). An additional column for time is needed at the end of each REGRESS line. This is not counted among the number of the regression effects specified first on the line. For the contents of the covariable file, see chapter 3.5.

COMBINE ***Combining model factors across traits***: MiX99 allows you to combine factors across the model lines of different traits. A line with one character indicating whether effects should be combined across traits.

y Yes.

n No.

When **y** is given, the following additional instruction lines must be specified.

MERGE One line for each trait with the same order as on the REGRESS instruction line(s), but without the leading integer number.

Instructions are used to merge factors across traits, i.e., one and the same factor can be specified for two or more traits. For instance somebody may consider a model where milk yields of 1st, 2nd, and 3rd lactation are modeled as three different traits but the same herd effect is modeled for 2nd and 3rd lactation observations.

For simplicity, first the lines are initialized with the trait number. Traits are numbered in ascending order. Thus, the line for the first trait is filled with ones (1), the line for the second trait with twos (2), the line for the third trait with threes (3), and so on. If a factor is missing on the REGRESS line(s), it must be marked to be missing (-) here as well. After the initialization step any factors which should be merged across traits will be given the same trait number. The specified trait number is strictly the lowest trait number of the factors which are merged. For some multi-trait models, this might require reordering the traits on the MODEL, RANDOM and REGRESS lines.

An example:

```
# COMBINE
  y
# MERGE
# factors   1   2   3   4   5
            1   1   1   1   1
            1   2   2   2   1
            3   3   3   3   1
            3   4   4   4   1
```

The interpretation of the given specification is as following: For the first factor, equations of the first and second traits are merged and equations of the third and fourth traits are merged as well. The second, third and fourth factors are treated as in a usual multi-trait model. For the fifth factor, equations of all four traits are merged. If the last factor is the genetic effect, its variance-covariance matrix is of size 1×1, i.e., a scalar. If the second last effect is a random effect, its variance-covariance matrix is of size 4×4. The residual variance-covariance matrix is 4×4 for all five factors.

CVRFIL, CVRNUM and **CVRIND** f covariables are read from a covariable table, rather than from the data file (covariable columns have been specified with a preceding "**t**" on the REGRESS line), then the following three additional instruction lines need to be specified (see Examples 7.5 and 7.6). This option is obligatory for the non-linear ***Gompertz function***.

CVRFIL Name of the covariable table file.

CVRNUM Number of covariable columns in the covariable table file. Note that the first column in the covariable table file contains the covariable index and is not counted when referring to columns in the covariable table file. For the non-linear ***Gompertz function***, number of columns needed is at least 4 (i.e., number of the parameters in the non-linear Gompertz function plus one for the time dependent).

CVRIND A line with one integer number. The integer number is the column number in the data file having the covariable index of data record. For the non-linear ***Gompertz function***, the index is the time of measurement.

PEDIGREE A character code specifying the method used in calculations having the inverse of the numerator relationship matrix.

Code	Method	Available information
sm	Sire model	Sires and maternal grandsires.
sm+p [value]	Sire model	Sires and maternal grandsires, phantom parent groups for missing parents.
am	Animal model	Sires and dams.
am+p [value]	Animal model	Sires and dams, phantom parent groups for missing parents.
ar	1st order autoregressive process	Block information, distances between classes within block.
ls	LS-model	-
ls+b	LS-model	Data blocking information from a pseudo pedigree file.

In case [**value**] is given phantom parent group effect will be treated as random (optional). The provided value will be added to the diagonal elements of the phantom parent groups of the inverse of the numerator relationship matrix. For instance, a value of 0.3333 would be equal to one additional phantom parent group offspring. A value of 1.0 is used for MACE models.

If `ar` is defined, a 1st order autoregressive process is applied on the last random effect instead of the numerator relationship matrix. This may be desired for the *variance model* when accounting for heterogeneous variance. The required information is given in the pedigree file (see the paragraph: "Accounting for heterogeneous variance").

When `ar` is employed, the following instruction line must be included.

RHO One line with as many autocorrelation parameters as there are traits in the model. Order of the values must be the same as the order of the traits. If the last random effect is combined across traits, then the order follows the order of the applied variances for the last random effect.

For the PEDIGREE option `ls+b`, a pseudo pedigree file has to be specified instead. This pseudo pedigree file has to contain for each level of the last effect in the model a line with four integers of the form "level code, 0, 0, block code".

DATAFILE Name of the data file. If the data file is in a different directory the whole path must be specified. If multiple input data files are used, only the name of the first data file is given.

VAR Line having three or four entries. The first entry is the number of integer columns in the data file, the second entry is the number of real columns in the data, and the third entry is a code for the type of the file.

f Formatted free format (columns are separated by at least one space) text file.

u Unformatted, i.e., a binary file. See Appendix A on how to convert between text and binary format.

The fourth entry is optional and can be:

a which is useful when testing models with heterogeneous variance adjustment. If **a** is given, observations will be scaled with the adjustment factors of a latter analysis. The analysis must contain all data of the previous run and not any new data.

g will instruct MiX99 to replace the real observation with simulated observations (see VALID option line in *Technical Reference Guide for MiX99 Solver*.

LAMPATH If **a** or **g** is specified in VAR, an additional line has to be given. The line contains the path for the directory where the files with the latter lambda values (option **a**) or the file with the simulated observations (option **g**) are stored.

MISSVA One real value. The value defines the code for missing real values in the data file.

SCALE A character that defines whether or not data is scaled. If scaling is desired, all observations will be scaled to units of residual standard deviation before the analysis. Scaling often improves numerical performance. Before any output is generated, all solutions are scaled back to the original units.

y Scaling.

n No scaling.

PEDFILE Name of the pedigree file plus one optional integer. If the pedigree file is in a different directory, the whole path to the file must be specified. If the optional integer is given, `mix99i` will prepare information for the calculation of daughter yield deviations (DYD). DYDs will be calculated for each sire with daughters in production when a zero "0" entry is specified. Otherwise, the integer indicates the column in the pedigree file that contains a within-bull classification variable of daughters (e.g. production year). In this case, DYDs will be calculated for each class within a sire. For more information, see the chapter Calculation of daughter yield deviations in the *Technical Reference Guide for MiX99 Solver*. For LS and GLS models, the PEDFILE line will be ignored. For LS models with blocking structure, a pseudo pedigree file must be provided as defined in PEDIGREE.

CORRFILE Optional. One line for those random effects that an inverse correlation structure between effect levels is provided in an external file (see FIXRAN). Each line contains three entries. The first entry is the random effect number that is specified on the RANDOM instruction line(s) for the random effect in question. The second entry is the format number of the inverse correlation matrix in the file (third entry). The format number is: 1= co-ordinate format, or 2=lower triangle dense format. The third entry is the name of the file having the correlation structure. The correlation structure between effect levels must be given in the form of an inverse correlation matrix. Only the diagonal and the lower (or the upper) triangle elements must be provided. In the co-ordinate format file, every line has three

columns. The first two columns are integer numbers and contain the id numbers of the levels. The third column is a non-zero real value in the inverse correlation matrix. The id numbers of the effect levels must be consistent with those given in the data file. Further, only non-zero elements must be included. For the multi-trait models, **MiX99** will automatically set up the Kronecker product between the inverse correlation matrix and the inverse variance-covariance matrix of the random effect.

REGFILE Optional. This is given only when there are regression design matrices as is the case in the analysis of genomic data (see FIXRAN). This can be used to make ***SNP-BLUP*** by considering the regression effects random. See *Command Language Interface for MiX99* for more details.

For each row in the data file, there must be a row of regression coefficients in the REGFILE. There can be multiple regression design matrices with different properties. Typically, however, one matrix is sufficient.

For each regression design matrix, there are two to five lines. The first line has mandatory and optional parameters:

```
<mandatory parameters> [<opt1> [<opt2> [<opt3> ...]]]
```

where the mandatory parameters are:

```
<type> <name> <id column> <first column> <number of effects>
```

and the optional parameters are:

```
[<center> [<impute> <missing> [<format> [<precond> [<scale>]]]]]
```

The information between square brackets are optional and can be left out.

The parameters are:

<type> of the effect can be 'F'/'f' for fixed effects, 'R'/'r' for random effects with the same variance for all effects in the design matrix, or 'H'/'h' for heterogeneous random effects with all effects having a different variance.

<name> is the name of the regression design matrix.

<id column> is the column reserved for the identity in the regression matrix. Value can be zero to indicate that no such column is present.

<first column> is the first regression column in the file considered as the design matrix column. All columns before this are skipped. Despite this, they must be numerical columns.

<number of effects/columns> is the number of (regression) effects in the design matrix.

** ** is instruction on ***centering*** the regression matrix:

- **n** No centering is done. This is the default.
- **c** Centering around averages of the matrix columns.
- **<real value>** Centering around given constant real value.

f Separate centering for each matrix column. Centering values are stored in a file whose name is given in the second line (see below).

<impute> is the command for ***imputing*** missing values. When the letter 'n' is given, no imputation is done. The letter 'a' replaces the missing values with average values on each column. The default is no imputation. The missing value to be imputed is defined by **<**missing**>**. For example, giving ' a 5 ' would impute all values of 5 with averages.

<missing> is the value considered missing and will be replaced, or imputed, by the average value on each column.

<format> is file format of the REGFILE:

n(ormal) Either 'n' or 'normal': columns of real valued regression coefficient are separated by spaces.

m(arkers) Integer (0, 1, 2, and optional **<**missing**>**) coded SNP marker values with separating spaces.

s(queezed) SNP marker values without separating spaces.

pb Binary PLINK .bed file format in individual-major mode (PLINK2 export format `ind-major-bed`). Reads ids and determines the number of SNPs from corresponding .bim and .fam files that must be available. **<**Id column**>** must be either 1 (ids are verified) or 0 (not verified). If any missing values, value 3 must be specified for **<**missing**>** (above). The file name must be given without the extension (.bed).

Optional minus sign prevents byte-packing of SNP matrices.

<precond> is the preconditioner type of the REGFILE. Options include 'n' for none, 'd' for diagonal, and 'b' for block diagonal. Default is 'd'.

<scale> is instruction on ***scaling*** the regression matrix:

n No scaling is done. This is the default.

<real value> Scaling with given constant real value.

2pq Scaling with $\frac{1}{\sqrt{2\sum_i p_i q_i}}$ where $p_i = \frac{\mu_i}{2}$ are half of the mean coefficients (markers) (μ_i) and $q_i = 1 - p_i$.

m Scaling with $\frac{1}{\sqrt{m}}$ where m is the number of coefficients (markers).

m2 Scaling with $\frac{1}{\sqrt{\frac{m}{2}}}$.

f Separate scaling for each matrix column. Scaling values are stored in a file whose name is given in the second or third line (see below).

After the first line, one to four file names, each on their own line:

<file name of the centering> Optionally given, if is 'f'. The file must contain <number of effects> values for centering of each column.

<file name of the scaling> Optionally given, if <scale> is 'f'. The file must contain <number of effects> values for scaling of each column.

<file name of the regression design matrix> The file must have an appropriate number of values on each line as specified in "number of effects/columns" above.

<file name of the variance component(s)> Optionally given, if the effects associated with the design matrix are random (given 'R'/'r' or 'H'/'h' for <effect type>). Includes the variance components for the effects. The format of this file is the following:

```
<effect number> <first trait> <second trait> <(co)variance>
```

where

<effect number> is the column number of the regression effect. Numbering starts from one from the first regression column in the file of the regression matrix,

<first trait> and ***<second trait>*** are the trait numbers related with the (co)variance component. In single trait models, both <first trait> and <second trait> equal one.

An example of a regression design matrix that is random. So, this is an SNP-BLUP model with common marker variance in file day4.par. Marker information is in columns 1 to 10, and no id column. No centering, nor imputation.

```
# Regression matrix information
RANDOM SNP  0  1 10  n  n 0
# REGFILE    xvec_day4.inp
xvec_day4.inp
# REGPARFILE day4.par
day4.par
```

day4.par file with (co)variance components:

1	2	3	1	
1	1	1	0.1	1st design matrix variance

PARFILE Name of the file with the (co)variance components for the model. If the file is in a different directory the whole path must be specified. This line will be ignored in the case of an LS-model.

RESFILE Name of the file with the residual (co)variance matrices. This information is optional and is required only for models where different residual classes are specified (see DATASORT). If the file is in a different directory the whole path must be specified.

TMPDIR The directory where the temporary work files will be created. These files will be created during the execution of `mix99i`. Depending on the model

and the amount of data analyzed, the files can be large. Specifying a dot (.) will locate the temporary files in the same directory where the program is executed.

RANSOLFILE One line with entries equal with the number of random effects specified in the model. The order of the entries is the same as the numbering of the random effects. Each entry defines whether or not a formatted solution file is created for the corresponding random effect.

y Yes.

n No.

This option is included to avoid unnecessary large solution files. For the fixed effects, formatted solution files are always created (see Formatted solution files in *Technical Reference Guide for MiX99 Solver*).

SOLUNF An entry specifying whether or not an unformatted solution file "***Solunf***" should be created. The unformatted solution file can be used as input file for future evaluations. Then, the solution vector will be initialized with the old solutions stored in this unformatted solution file (see chapter Contents and chapter Unformatted solution files in *Technical Reference Guide for MiX99 Solver*).

y Yes.

n No.

Additionally, ***mix99s*** creates a binary file called "***Solvec***", which contains a copy of the solution vector. If ***mix99s*** is requested to restart, the program will read "`Solvec`" and continue with these solutions.

For non-linear ***Gompertz function models***, restarting of the analysis is needed because of the iterative algorithm. Then, both options are possible: if variance component estimation is done similarly, **y** must be defined because the new variance components are not updated without calling `mix99i` first; when the variance components remain the same from iteration to iteration, restarting ***mix99s*** under the option **n** is possible as well.

PRECON One line with two sets of characters defining which type of preconditioner is used. The first set of characters defines the type of preconditioner applied to the equations belonging to within block effects (WpW). Usually, this part includes most of the equations. The second set of characters defines the type of preconditioner applied to the equations belonging to the across block fixed effects (XpX). Before specifying the preconditioner options we recommend reading the chapter Effect of preconditioning on convergence in *Technical Reference Guide for MiX99 Solver*.

Specifying **n** as the first entry on the PRECON line will instruct MiX99 to skip preconditioning. This can save much computing time and is useful if only reliabilities need to be calculated.

For each within block effect (WpW), one character must be given. The

order of the characters must be the same as the defined order of within blocks effects on the WITHINBLOCKORDER line. Two alternative preconditioners can be defined:

b Block diagonal. Applies block diagonal matrices for preconditioning, where the size of matrices are equal to the number of equations belonging to an effect level. For complex models, it may generate large preconditioning data files with significant increase of I/O operations.

d Diagonal. Only the diagonal elements of the coefficient matrix are used for preconditioning.

If DYDs will be calculated, a block diagonal preconditioner matrix (**b**) must be defined for the genetic animal effect. For the non-linear ***Gompertz function***, a diagonal preconditioner matrix (**d**) must be defined.

Block diagonal: The block diagonal preconditioner consists of as many matrices as there are effect levels. Size of these matrices depend on the number of equations which belong to one level, i.e., this is usually equal to the number of traits, or it is the number of correlated factors describing the effect. For example, assuming a multi-trait model with three traits, where the additive genetic animal effect of a trait is modelled by four regression coefficients, will instruct MiX99 to apply for each animal a preconditioner matrices of size 12×12.

Diagonal: Only the diagonal elements of the coefficient matrix are used for preconditioning.

For across block fixed effects (XpX), there are five alternative preconditioners:

f Full block. The whole XpX matrix is used for preconditioning.

b Block diagonal. Applies block diagonal matrices for preconditioning.

d Diagonal. Only diagonal elements of the coefficient matrix are used for preconditioning.

m Mixed. A block diagonal preconditioner is applied for the levels of the first fixed effect specified in XpX and one superblock is defined for all remaining equations in XpX.

m plus optionally a certain series of integers This option allows to set up a very specific preconditioner matrix for the XpX part as explained in detail below.

Based on our experiences, for most models a block diagonal preconditioner is a good choice for the XpX part. Nevertheless, for some models the other options may give some advantages.

Full block: The whole XpX block of the MME is considered as a preconditioner matrix. In some cases, inversion of that matrix will require too much memory and computing time.

Block diagonal: Same as for the WpW. All equations that belong to the same effect level form one diagonal block.

Diagonal: Only the diagonal elements of the XpX part of the coefficient matrix are used for preconditioning. This type of preconditioner is likely to yield poorer convergence for multi-trait models. For the non-linear ***Gompertz function***, the diagonal (**d**) preconditioner must be defined.

Mixed: The XpX part of the coefficient matrix is separated into two parts. The first part includes all factors that belong to the first across block fixed effect. In case general regressions (across all data) are specified, also these regressions are included into the first part. The second part includes all factors of all remaining across block fixed effects. For the first part block diagonal matrices will be used for preconditioning. For the second part, one large preconditioner matrix, of size number of equations belonging to the second part, will be used for preconditioning. Hence, the across fixed effect with most levels should be defined as first effect when applying option Mixed.

Mixed plus optionally a certain series of integers: For some models the option "Mixed" may result too large preconditioner matrices. Specifying a series of integers behind the "**m**" character will allow to define preconditioning superblocks for the XpX part. The number of specified integers must be equal to the number of across block fixed effects. Fixed effects that are belonging to one superblock must be specified with the same integer number. The first given integer number must be 1 and numbering is in ascending order. For instance, if 4 across block fixed effects are specified in the model, then the specification i) `m 1 2 3 4`, ii) `m 1 2 3 3`, iii) `m 1 1 2 2`, iv) `m 1 2 2 2`, and v) `m 1 1 1 1` have the following meaning: i) same as block diagonal preconditioner, ii) for effect one and two block diagonal preconditioners are defined and for effect three and four one superblock is defined, iii) two superblocks are defined, iv) same as Mixed, v) same as Full block.

PARALLEL A line with one entry specifying the number of processors used for solving the MME. In case the serial solver program ***mix99s*** is used, number one (1) needs to be given. Models with non-linear or categorical traits are not yet possible to analyze with parallel computing. There are two optional arguments after the number of processors: workload division method, and size of I/O buffers. The workload division is used in parallel computing to divide amount of work to processors evenly. MiX99 has two workload division methods: number of records, and number of equations. Default is workload by number of records, which leads to dividing the data file approximately evenly to to all processors. Work load division by number of equations can be requested by giving letter e or E. Then, each processor will have about the same number of equations. Total size of I/O buffers can be given in megabytes. Example: Giving `4 w 100` will lead to using 4 processors, workload division (i.e. number of records), and I/O buffer size of 100 megabytes. Note that the preprocessor does not necessarily allocate all given buffer amount, e.g., 100MB given in the example above.

The amount is divided to different I/O buffers such as data, pedigree, and preconditioner I/O buffers. For the data and pedigree files, memory optimization is always done so that buffer size will not exceed too much the needed size. However, for buffer of the preconditioner file, the buffer size will not be memory optimized. Consider the 100MB example above. Assume that the preconditioner is estimated to take 20 percent of the 100MB but the data is small and only 1MB is needed by the other I/O buffers, then only 21MB will be used instead of the requested 100MB.

COMMONBLOCKS An integer indicating number of pedigree blocks which should be included to the common equations part (or common blocks) in the parallel computing set up of the MME. (See equation family structure in 3.1.1.)

```
# COMMONBLOCKS: <number of common area blocks>
          17
```

Alternatively, common blocks can be defined by specifying the block code of the first common area block instead and giving letters "FIRST" (or "f", "F") after it.

```
# COMMONBLOCKS: <block code of the first common block> FIRST
          9000   FIRST
```

The information is not used, if the single processor program ***mix99s*** is executed. However, some number has to be given (e.g., 0).

5 Output files of the MiX99 pre-processor

5.1 MiX99.lst file

The MiX99 pre-processor will print information on instructions, model, data, etc. to the standard output. Most important information about the analysis is given in the `MiX99.lst` file. This includes information about the input files, applied model, applied variance components, a description of the pedigree and data, the number of effect levels, as well as the structure of the mixed models equations.

5.2 Copy of input directives and command line options

MiX99 pre-processor stores its input directives, i.e. content of the MiX99 instruction file or CLIM command file (via `MiX99_DIR.DIR` file) to file `MiX99_IN.DIR`.

Similarly, the command line options used when executing the pre-processor are stored to file `MiX99_IN.OPT`.

These two files (`MiX99_IN.DIR` and `MiX99_IN.OPT`) can later be used to specify the original pre-processor input directives, for example, when calculating approximate reliabilities using ApaX or estimating variance components.

5.3 Log-files

More technical information is written to log-files. Some of this information will be utilized by the programs scheduled after the execution of `mix99i`.

`Modlog` Contains model and data parameters. This file will be read by ***mix99s***, ***mix99p***, ***apax99***, ***apax99p***, and ***exa99***.

`Parlog` Contains program dimension parameters. The information is used when adjustment for heterogeneous variance is applied.

`Memlog` Contains information about the amount of random access memory used during the execution of the pre-processor and solver programs. In practice, the amount actually is often larger due to memory overhead and depends on the computer.

`Hetlog` Contains information about the structure of the data for the *variance model*. This file is created by ***mix99hv*** when adjusting for heterogeneous variance and it is used by ***mix99p***.

5.4 Temporary work files

Temporary work files are created by the pre-processor program `mix99i`. They are in a binary format and will be used by the programs scheduled after the execution of `mix99i`. The temporary files 4, 5, 6, and 10 will be read by the solver programs every iteration. Some other work files are created depending on the model.

`Tmp1.code0` Original and recoded id codes of all effect levels. For the animal effect, it has also the number of descendants and the number of observations. Created in the data recording phase. Used while writing out the solutions or the reliabilities.

`Tmp4.pedi0` Pedigree information of the animals. Created after the data recording phase. Used in the iteration process and by the programs for calculation of reliabilities.

Tmp5.clas0 Pre-processed data with class information. Used in the iteration process and by the programs for calculation of reliabilities.

Tmp6.diab0 LDL'-decomposition of the preconditioner matrix. Used in the iteration process

Tmp9.strc0 Information about the structure of the pre-processed data. The file is created at the end of the data pre-processing phase and is read before the start of the iteration process. The information controls the I/O tasks during the iteration process.

Tm10.trco0 Pre-processed data with observations and covariables. Used in the iteration process.

Tm20.inbr0 Inbreeding coefficients.

Tm21.regr0 Regress coefficient matrix values.

Tmp.cov0 Matrix element values for $\mathbf{G}^{-1}/\mathbf{T}$.

Tmp.ijcov0 Matrix equation numbers for $\mathbf{G}^{-1}/\mathbf{T}$.

TmpiG.cov0 Matrix element values for the metafounder matrix.

TmpiG.ijcov0 Matrix equation numbers for the metafounder matrix.

TmpiC.cov0 Matrix element values for the $\mathbf{C}^{-1}$ matrix in the new ssGTABLUP.

Tmp.para Information needed for parallel processing.

In case of parallel processing (using programs ***mix99p*** or ***apax99p***), each process has iteration files of its own. The names of the files end with the corresponding process number.

6 Using old solutions

The solver programs ***mix99s*** and ***mix99p*** can create a solution file in binary format named `Solunf`. This file contains all solutions to the MME. Solutions in this file can be used as initial values for future evaluations when more data has been accumulated. When carrying out the future evaluation the `Solunf` file from the previous evaluation run must be renamed to "`Solold`". The pre-processor `mix99i` checks for the existence of the "`Solold`" file in the execution directory. In case such a file exists mix99i will initialize the solution vector with the solutions from the previous evaluation run. Solutions of new effect levels will be preset to zero. The initialized solution vector will be written to a file named "`Solvec`", which will be read by the solving program ***mix99s*** or ***mix99p***.

7 Examples of MiX99 instruction files

This chapter gives examples on how MiX99 instruction files are set up for various models supported by MiX99. Some of the examples are identically with the test examples provided in the MiX99 package. The examples are chosen to cover a wide range of possible models and analysis and may be helpful when setting up more complicated model. For some of the following examples, also the corresponding CLIM command file is provided. Note a CLIM command file can be transferred to a MiX99 instruction file by executing the command '`mix99i -d instr.clm`' where `instr.clm` is the CLIM file. This option will instruct `mix99i` to produce a MiX99 instruction file named `MiX99_DIR.DIR`.

7.1 Multiple trait animal model

Traits:		milk, protein
Model:		
	Fixed effects:	herd, year×season (ys)
	Random effects:	animal
Pedigree:		animal model, phantom parents

Data file:

$Animal_1$	$Herd_2$	$Year\text{-}Season_3$	Age_4	$Milk_1$	$Protein_2$
5	102	3	17	5123.5	180.4
6	102	3	13	7597.0	243.8
7	103	4	25	6410.3	-888.0
8	103	3	20	-888.0	210.7
⋮	⋮	⋮	⋮	⋮	⋮

Pedigree file:

$Animal_1$	$Sire_2$	Dam_3	$Herd_4$
1	-10	-20	100
2	-10	-20	100
3	1	2	100
4	-15	-20	100
5	1	2	102
6	4	-25	102
7	3	-25	103
8	3	7	103
⋮	⋮	⋮	⋮

MiX99 instruction file:

```
# TITLE:
          Multiple Trait, Milk and Protein
# INTEGER:
          Animal Herd Year-Season Age
# REAL:
          Milk Protein
# TRAITS:
          2
# TRAITGRP:  trait_group, column_with_group_code
```

```
                    1                    -
# DATASORT: block_code, relationship_code, (no multiple residual)
                    2     1
# FIXRAN: number of fixed and random factors in the model
                    2                    1
# MODEL:  trait_group  trait  weight  herd  ys  animal
                    1     1     -     2     3     1
                    1     2     -     2     3     1
# WITHINBLOCKORDER: order of effects within block
                                        2     -     1  # Herd is the only fixed
# RANDOM:         animal                               # effect within blocks
                    1
                    1
# RELATIONSHIPS:        number: animal
                              1      1
# REGRESS:        number:  herd   ys   animal             # all effects are
                    3      cl     cl     cl               # classifications, no
                    3      cl     cl     cl               # regression effects
# COMBINE:
             n
# PEDIGREE:  animal model with phantom parent groups for missing parents
             am+p
# DATAFILE:
             example1.dat
# VAR:       integer  real  formatted
                4     2     f
# MISSVA:    code for missing real values
             -888.0
# SCALE:
             y
# PEDFILE:
             example1.ped
# PARFILE:
             variance_comp.ex1
# TMPDIR:
             .
#RANSOLFILE: animal
            y
# SOLUNF: no unformatted solution file
            n
# PRECON: block diagonal preconditioner for all WpW, full block for XpX equations
            b  b     f
# PARALLEL:  number of processors used by the solver program
            1
# COMMONBLOCKS: number of blocks in common             # only for parallel proc.
            0
```

File with (co)variance components:

1	2	3	1	
1	1	1	2.1708e+05	
1	1	2	4.2379e+03	animal
1	2	2	1.2928e+02	
2	1	1	4.9496e+05	
2	1	2	1.4427e+04	residual
2	2	2	4.3169e+02	

CLIM command file:

```
TITLE    Multiple Trait, Milk and Protein

INTEGER Animal Herd YS Age          # Integer columns in the data
REAL    Milk Protein                # Real columns in the data
DATAFILE example1.dat               # Data file

PEDFILE  example1.ped               # Pedigree file
```

```
PEDIGREE Animal am+p                  # Pedigree is associated with Animal effect
PARFILE  variance_comp.ex1            # Var.comp. file

DATASORT BLOCK=Herd PEDIGREECODE=Animal
MISSING  -888.0
PRECON   b b f
WITHINBLOCKORDER Animal Herd

MODEL
  Milk    = Herd YS Animal  # Trait 1
  Protein = Herd YS Animal  # Trait 2
```

7.2 Multiple-trait model: different models by trait

Traits:		growth (G), feed efficiency (FE), meat quality (MQ), fat% (F%), meat% (M%), motility (MT, scored from 1 to 5), front feet (FF, binary trait, 1 = sick, 2 = healthy)
Model:		
	Fixed effects:	age, sex, station-year-season (sys)
	Random effects:	litter, animal
Pedigree:		animal model and phantom parent groups

Data file:

Class variables						Traits						
$Sort_1$	Ani_2	Age_3	Sex_4	$Lit._5$	SYS_6	G_1	FE_2	MQ_3	$F\%_4$	$M\%_5$	MT_6	FF_7
1	3781	5	2	5906	9901	1013	2.40	34.8	13.6	65.3	4	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Pedigree file:

$Animal_1$	$Sire_2$	Dam_3	$Sort_4$
3521	−1085	−1085	0
2941	−1085	−1085	0
1552	3521	2941	0
1699	−2080	−2080	0
⋮	⋮	⋮	⋮

MiX99 instruction file:

```
# TITLE:
          Pig evaluation, multi-trait animal model
# INTEGER:
          Sort Ani Age Sex Litter SYS
# REAL:
          G FE MQ F% M% MT FF
# TRAITS:
          7
# TRAITGRP:  trait_groups, column with the trait_group_code
          1     -
# DATASORT: block_code, relationship_code (animal, only 1 res.var.)
          1     2
# FIXRAN: number of fixed and random factors in the model
          3     2
# MODEL:  trait_group, trait, weight,  age,  sex,  sys,  litter,  animal
                  1     1     -         3     4     6     -     2
                  1     2     -         3     4     6     -     2
                  1     3     -         3     4     6     -     2
                  1     4     -         3     4     6     -     2
                  1     5     -         3     4     6     -     2
                  1     6     -         -     -     6     5     2
                  1     7     -         -     -     6     5     2
# WITHINBLOCKORDER: order of effects within blocks
                                        -     -     -     1     2
# RANDOM: litter, animal
             -     2
             -     2
             -     2
```

```
             -     2
             -     2
             1     2
             1     2
# RELATIONSHIPS: number: animal
             1     1
# REGRESS: number,    age,    sex,   sys,   litter,   animal
             5        cl      cl      cl       -      cl
             5        cl      cl      cl       -      cl
             5        cl      cl      cl       -      cl
             5        cl      cl      cl       -      cl
             5        cl      cl      cl       -      cl
             5         -       -      cl      cl      cl
             5         -       -      cl      cl      cl
# COMBINE:
          n
# PEDIGREE:
          am+p
# DATAFILE:
          kanta.dat
# VAR:    formatted data file
          6     7     f
# MISSVA:
          0.0
# SCALE:
          y
# PEDFILE:
          kanta.ped
# PARFILE:
          kanta.var
# TMPDIR:
          .
#RANSOLFILE: litter, animal
          n     y
# SOLUNF:
          n
# PRECON: block diagonal preconditioner for all effects
          b b b # (XpX is too large that the full block precond. could be used.)
# PARALLEL:  number of processors used by the solver program
          1
# COMMONBLOCKS: number of common blocks (parallel computing)
          0
```

File with (co)variances for random effects:

1	2	3	1	
1	1	1	variance for MT	
1	1	2	covariance between MT and FF	litter
1	2	2	variance for FF	
2	1	1	variance for G	
2	1	2	covariance between G and FE	
2	1	3	covariance between G and MQ	
2	1	4	covariance between G and F%	
2	1	5	covariance between G and M%	
2	1	6	covariance between G and MT	
2	1	7	covariance between G and FF	animal
2	2	2	variance for FE	
2	2	3	covariance between FE and MQ	
2	2	4	covariance between FE and F%	
0	0	0	.	
0	0	0	.	
2	6	7	covariance between MT and FF	
2	7	7	variance for FF	

```
3  1  1          residual variance for G
3  1  2   residual covariance between G and FE
3  1  3   residual covariance between G and MQ
0  0  0                    .                    residual
0  0  0                    .
3  6  7  residual covariance between MT and FF
3  7  7         residual variance for FF
```

CLIM command file:

```
TITLE   Pig evaluation, multi-trait animal model

INTEGER Sort Ani Age Sex Litter SYS # Integer columns in data
REAL    G FE MQ Fp Mp MT FF         # Real columns in data
DATAFILE kanta.dat                  # Data file
DATASORT BLOCK=Sort PEDIGREECODE=Ani

PEDFILE  kanta.ped      # Pedigree file
PEDIGREE Ani am+p       # Pedigree is for Animal

PARFILE  kanta.var      # Var.comp. file

RANDOM   Litter Ani     # Ani not necessary, it is in PEDIGREE
NORANSOL Litter

MODEL
  G  = Age Sex SYS        Ani
  FE = Age Sex SYS        Ani
  MQ = Age Sex SYS        Ani
  Fp = Age Sex SYS        Ani
  Mp = Age Sex SYS        Ani
  MT =         SYS Litter Ani
  FF =         SYS Litter Ani

WITHINBLOCKORDER Litter Ani
```

7.3 Random regression animal model

Example given by Schaeffer, L. R. and Dekkers, J. C. M. (1994). "Random regressions in animal models for test-day production in dairy cattle". In: *Proc. 5th World Congr. Genet. Appl. Livest. Prod.* Vol. 18, pp. 443–446.

Data file:

Herd_Test_Day$_1$	Animal$_2$	Covariable 1$_1$	Covariable 2$_2$	Milk kg$_3$
1	1	73.0	1.4298500	26.0
1	2	34.0	2.1939499	29.0
1	3	8.0	3.6408701	37.0
2	1	123.0	0.9081270	23.0
⋮	⋮	⋮	⋮	⋮

Pedigree file:

Animal$_1$	Sire$_2$	Dam$_3$	Block_sort._var.$_4$
1	9	7	0
2	10	8	0
3	9	2	0
4	10	8	0
5	11	7	0
6	11	1	0

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   example3.ped
PEDIGREE  G am

PARFILE   variance_comp.ex3

PRECON    d d f
WITHINBLOCKORDER G HTD

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

MiX99 instruction file:

```
# Trait:                          milk
# Model:
   # Fixed regressions:           beta1 (B1), beta2 (B2)
   # Fixed effect:                herd-test-day
   # Random regression effects:   gamma0 (G0), gamma1 (G1), gamma2 (G2); G0, G1,
   #                              and G2 describe the animal effect
# Pedigree:                       animal model
#
# TITLE:
          RANDOM REGRESSION, L.Schaeffer & J.Dekkers (1994)
# INTEGER:
          HTD Animal
# REAL:
          Covar_1 Covar_2 Milk
# TRAITS:
          1
```

```
# TRAITGRP:
          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
                                  # data column (2) for animal class is repeated
                                  # three times for three model factors
# WITHINBLOCKORDER: order of effects within blocks
                                       2         1         1         1
# RANDOM: gamma0  gamma1  gamma2
            1       1       1     # as the animal factors are correlated, they
                                  # all belong to the same random effect
# RELATIONSHIPS: number: gamma0  gamma1  gamma2
            3       1       1       1             # each animal factor uses the
                                                  # same pedigree.
# REGRESS: number  beta1  beta2  herd-test-day  gamma0  gamma1  gamma2
            6       1       2          cl         cl        1        2
                          # there are two regression effects (B1 & B2) and four
                          # class variables in the model. For two class variables
                          # regressions are nested within class (G1 & G2).
                          # numerical values for the covariables B1 & B2 and G1 &
                          # G2 are in the 1st and 2nd real columns of the data.
# COMBINE:
          n
# PEDIGREE:
          am
# DATAFILE:
          example3.dat
# VAR:
          2  3  f
# MISSVA:
          0.0
# SCALE:
          n
# PEDFILE:
          example3.ped
# PARFILE:
          variance_comp.ex3
# TMPDIR:
          .
#RANSOLFILE: animal effect
          y
# SOLUNF:
          n
# PRECON: diagonal preconditioner for WpW and full block for XpX
          d  d     f
# PARALLEL:  number of processors used by the solver program
          1
# COMMONBLOCKS:
          0
```

File with (co)variances for random effects:

1	2	3	1	
1	1	1	44.791	
1	1	2	−0.133	
1	1	3	0.351	animal
1	2	2	0.073	
1	2	3	−0.010	
1	3	3	1.068	
2	1	1	100.000	residual

CLIM command file:

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

DATAFILE  example3.dat          # Data file
INTEGER   HTD Animal            # Integer column names
REAL      Covar_1  Covar_2   &  # Covariables
          Milk                  # Milk yield

PEDFILE   example3.ped          # Pedigree file
PEDIGREE  G am                  # Genetics associated with the animal code

PARFILE   variance_comp.ex3     # Variance component file
PRECON    d d f
WITHINBLOCKORDER G HTD

DATASORT BLOCK=HTD  PEDIGREECODE=Animal

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

7.4 Multiple-trait model with trait groups

Traits:		TD-milk 1st lac.(M1), TD-protein 1st lac.(P1), TD-milk 2nd lac.(M2), TD-protein 2nd lac.(P2)
Model:	Within each lactation, each animal has repeated observations, i.e. the model has features of multi-trait and repeatability models.	
	Fixed effects:	herd, age
	Fixed regressions (season×lactation curve interaction):	beta0 (B0), beta1 (B1), beta2 (B2)
	Random effects:	non-genetic animal environment (AE)
	Random regressions (genetic animal effect):	gamma0 (G0), gamma1 (G1), gamma2 (G2)
Pedigree:		animal model

If we do not use trait groups, the data file will have the following structure. Missing integers are coded with "0" and missing real values are coded with "-16.". Information utilized is underlined.

Class variables						Covariables				Traits			
Hrd_1	Ani_2	$Ag1_3$	$Ag2_4$	$Se1_5$	$Se2_6$	$C11_1$	$C12_2$	$C21_3$	$C22_4$	$M1_5$	$P1_6$	$M2_7$	$P2_8$
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
34	10	7	0	5	0	.967	.042	-16.	-16.	12.1	3.40	-16.	-16.
34	10	7	0	6	0	.562	.084	-16.	-16.	8.7	3.52	-16.	-16.
34	10	0	17	0	10	-16.	-16.	.661	.035	-16.	-16.	28.2	3.37
34	10	0	17	0	10	-16.	-16.	.430	.087	-16.	-16.	32.7	-16.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Data file: Building two trait groups (Trg = trait group code), one for the first lactation and one for the second lactation allows the following data file structure:

Hrd_1	Ani_2	Trg_3	Age_4	Sea_5	$Cv1_1$	$Cv2_2$	M_3	P_4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
34	10	1	7	5	0.967	0.042	12.1	3.40
34	10	1	7	6	0.562	0.084	8.7	3.52
34	10	2	17	10	0.661	0.035	28.2	3.37
34	10	2	17	10	0.430	0.087	32.7	-16.
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

MiX99 instruction file:

```
# TITLE:
          Multiple-trait model with trait groups
# INTEGER:
          Hrd Ani Trg Age Sea
# REAL:
          Cv1 Cv2 Mlk Prt
# TRAITS:
          4
# TRAITGRP:  trait_groups, column with the trait_group_code
```

```
             2     3
# DATASORT: block_code, relationship_code (animal, single res. var)
             1     2
# FIXRAN: number of fixed and random factors in the model
             5     4
# MODEL:  trait_group,  trait,  weight,  herd, B0, B1, B2,  age,   AE, G0, G1, G2
                   1     3      -          1    5   5   5     4     2   2   2   2
                   1     4      -          1    5   5   5     4     2   2   2   2
                   2     3      -          1    5   5   5     4     2   2   2   2
                   2     4      -          1    5   5   5     4     2   2   2   2
# WITHINBLOCKORDER: order of the effects within blocks
                                           3    -   -   -     -     1   2   2   2
# RANDOM:           AE,   G0,   G1,   G2
                   1     2     2     2
                   1     2     2     2
                   1     2     2     2
                   1     2     2     2
# RELATIONSHIPS: number:         G0,   G1,   G2
                   3              1     1     1
# REGRESS: number:  herd, B0,  B1,  B2,  age,    AE,  G0,  G1,  G2
             9        cl   cl   1    2     cl     cl   cl   1    2
             9        cl   cl   1    2     cl     cl   cl   1    2
             9        cl   cl   1    2     cl     cl   cl   1    2
             9        cl   cl   1    2     cl     cl   cl   1    2
# COMBINE:
          n
# PEDIGREE:
          am
# DATAFILE:
          /home/martin/data/example4.dat.bin
# VAR:   unformatted data file
          5  4  u
# MISSVA:
          -16.
# SCALE:
          y
# PEDFILE:
          /home/martin/data/example4.ped
# PARFILE:
          variance_comp.ex4
# TMPDIR:
          /disk3/tmp
#RANSOLFILE: AE, animal
           n  y
# SOLUNF:
           n
# PRECON:
           d  b  d     f
# PARALLEL:  number of processors used by the solver program
           4
# COMMONBLOCKS: number of common blocks (parallel computing)
           2
```

File with (co)variances for random effects:

1	2	3	1	
1	1	1	variance for M1	
1	1	2	covariance between M1 and P1	
1	1	3	covariance between M1 and M2	
1	1	4	covariance between M1 and P2	
1	2	2	variance for P1	animal env.
1	2	3	covariance between P1 and M2	
1	2	4	covariance between P1 and P2	
1	3	3	variance for M2	
1	3	4	covariance between M2 and P2	

```
1   4   4            variance for P2
2   1   1          variance for M1-G0
2   1   2   covariance between M1,G0 and P1,G0
2   1   3   covariance between M1,G0 and M2,G0
2   1   4   covariance between M1,G0 and P2,G0
2   1   5   covariance between M1,G0 and M1,G1
2   1   6   covariance between M1,G0 and P1,G1
2   1   7   covariance between M1,G0 and M2,G1
2   1   8   covariance between M1,G0 and P2,G1
2   1   9   covariance between M1,G0 and M1,G2      animal
2   1  10   covariance between M1,G0 and P1,G2
2   1  11   covariance between M1,G0 and M2,G2
2   1  12   covariance between M1,G0 and P2,G2
2   2   2          variance for P1,G0
2   2   3   covariance between P1,G0 and M2,G0
0   0   0                   .
2  12  12          variance for P2,G2
1   1   1        residual variance for M1
1   1   2 residual covariance between M1 and P1
1   2   2        residual variance for P1          residual
1   3   3        residual variance for M2
1   3   4 residual covariance between M2 and P2
1   4   4        residual variance for P2
```

CLIM command file:

```
TITLE     Multiple-trait model with trait groups

DATAFILE  BINARY /home/martin/data/example4.dat.bin
INTEGER   Hrd Ani Trg Age Sea
REAL      Cv1 Cv2 Mlk Prt
MISSING   -16.

PEDFILE   /home/martin/data/example4.ped
PEDIGREE  G am

TMPDIR    /disk3/tmp
PARFILE   variance_comp.ex4
PRECON    d b d f
WITHINBLOCKORDER Ani G Hrd

PARALLEL  4 2

DATASORT     BLOCK=Hrd  PEDIGREECODE=Ani
TRAITGROUP   Trg
RANDOM       Ani
NORANSOL     Ani

MODEL SCALE
 Mlk(1) = Hrd fix_curve(1 Cv1 Cv2| Sea) Age Ani G(1 Cv1 Cv2| Ani)
 Prt(1) = Hrd fix_curve(1 Cv1 Cv2| Sea) Age Ani G(1 Cv1 Cv2| Ani)
 Mlk(2) = Hrd fix_curve(1 Cv1 Cv2| Sea) Age Ani G(1 Cv1 Cv2| Ani)
 Prt(2) = Hrd fix_curve(1 Cv1 Cv2| Sea) Age Ani G(1 Cv1 Cv2| Ani)
```

7.5 Random regression model based on covariance functions

Random regression test-day model based on covariance functions for the 1st lactation milk yield. Covariables are read from a covariable table file, rather than from the data file.

Traits:		milk
Model:		
	Fixed effects:	herd, season×lactation curve interaction, age, days carried calf (dcc), and year-month (ym). The season×lactation curve interaction is modeled by 5 regression coefficients (b0,. . . ,b4).
	Random effects:	herd×test-month (htm), non-genetic_animal_env. (covariance function with three polynomials; z0, z1, z2), additive_genetic_animal (covariance function with three polynomials; g0, g1, g2)
Pedigree:		animal model, phantom parent groups

Data file:

herd$_1$	animal$_2$	htm$_3$	ym$_4$	age$_5$	dcc$_6$	season$_7$	dim$_8$	milk$_1$
1001	50070	18810	8810	17	1	3	36	23.9
1001	50070	18811	8811	17	1	3	64	22.4
1001	50070	18812	8812	17	1	3	91	24.1
1001	50070	18901	8901	17	1	3	128	27.1
1001	50070	18902	8902	17	1	3	161	24.1
1001	50070	18903	8903	17	1	3	183	23.4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Pedigree file:

animal$_1$	sire$_2$	dam$_3$	herd$_4$
50054	34788	50068	1001
50055	90670	50067	1001
50056	28400	−172	1001
50057	30099	−173	1001
50058	29598	−175	1001
50059	33338	50057	1001
⋮	⋮	⋮	⋮

File with covariable table:

Index	Covariable columns				
DIM$_1$	1	2	3	4	5
4	−0.7071	−0.3189	−0.4110	0.1045	0.4303
5	−0.7071	−0.3230	−0.3755	0.0894	0.4293
⋮	⋮	⋮	⋮	⋮	⋮

MiX99 instruction file:

```
# TITLE:
          Test-day data: milk, 1st lac., third order polynomial
# INTEGER:
          herd animal htm ym age dcc season dim
# REAL:
          milk
# TRAITS:
```

```
          1
# TRAITGRP:
          1     -
# DATASORT:
          1     2
# FIXRAN:
          9     7
# MODEL: TRAITGRP trait wgt: herd b0 b1 b2 b3 b4 age dcc ym htm z0 z1 z2 g0 g1 g2
                  1     1  -      1  7  7  7  7  7   5   6  4   3  2  2  2  2  2  2
# WITHINBLOCKORDER:
                               4  -  -  -  -  -   -   -   -  3  1  1  1  2  2  2
# RANDOM:                                                    htm z0 z1 z2 g0 g1 g2
                                                             1  2  2  2  3  3  3
# RELATIONSHIPS: number:
                        3                                                  1  1  1
# REGRESS: number:           herd b0 b1 b2 b3 b4 age dcc ym htm z0 z1 z2 g0 g1 g2
           16                cl   cl t2 t3 t4 t5  cl  cl cl  cl t1 t2 t3 t1 t2 t3
# COMBINE:
          n
# -------------------------
# Use covariable table
# -------------------------
       # CVRFIL: name of the file with covariable table
                  covarmilk.tab
       # CVRNUM: number of the covariable columns in the file
                  5
       # CVRIND: integer column in the data, which contains the covariable index
                  8
# PEDIGREE:
          am+p
# DATAFILE:
          /koel/testi/data/mTab.dat
# VAR:
          8  1  f
# MISSVA:
          0.0
# SCALE:
          y
# PEDFILE:
          /koel/testi/data/blk.ped
# PARFILE:
          legendrepol.par
# TMPDIR:
          .
#RANSOLFILE: solution files for the random effects: htm n-ga animal
          y     n     y
# SOLUNF: unformatted solution file
          y
# PRECON: diagonal preconditioner for all WpW effects except for the animal
#         effect, and mixed block preconditioner for the XpX
#         m: first effect (season x lactation curve; 5 equations / effect level:
#         b0, b1, b2, b3, b4) and second effect (age) form one diagonal block,
#         remaining equations of dcc and ym effect form another diagonal block.
          d  d  b  d     m 1 1 2 2
# PARALLEL:
          1
# COMMONBLOCKS:
          0
```

File with (co)variances for random effects:

```
1 2 3                1
1 1 1        variance for htm          herd-test-month
2 1 1        variance for z0
2 1 2 covariance between z0 and z1
2 1 3 covariance between z0 and z2      non-genetic
```

```
2  2  2        variance for z1          animal environment
2  2  3  covariance between z1 and z2
2  3  3        variance for z2
3  1  1        variance for g0
3  1  2  covariance between g0 and g1
3  1  3  covariance between g0 and g2    additive genetic
3  2  2        variance for g1             animal
3  2  3  covariance between g1 and g2
3  3  3        variance for g2
4  1  1       residual variance            residual
```

CLIM command file:

```
TITLE Test-day data: milk, 1st lac., third order polynomial

DATAFILE  /koel/testi/data/mTab.dat
INTEGER   herd animal htm ym age dcc season dim
REAL      milk
DATASORT  BLOCK=herd PEDIGREECODE=animal

PEDIGREE  G am+p
PEDFILE  /koel/testi/data/blk.ped

PARFILE    legendrepol.par
TABLEFILE  covarmilk.tab
TABLEINDEX dim

PRECON   d d b d m 1 1 2 2

RANDOM   htm PE G
NORANSOL PE

WITHINBLOCKORDER PE G htm herd

MODEL SCALE
  milk = herd curve(1 t2 t3 t4 t5| season) age dcc ym &
         htm PE(t1 t2 t3| animal) G(t1 t2 t3| animal)
```

7.6 Multiple trait random regression test-day model with covariance functions

Multiple trait random regression test-day model where random animal effects are modeled by covariance functions. The model includes three traits: milk yield of first (M1), second (M2), and third (M3) lactations. The non-genetic effects and the genetic effects are modeled by covariance functions. Thus, the covariance functions for non-genetic animal effects and genetic animal effects will be modeled across traits; therefore, the COMBINE option will be applied.

Traits:		Milk 1st lac. (M1), 2nd lac. (M2), and 3rd lac. (M3)
Model:		
	Fixed effects:	The model for the fixed effects is the same as in the Example 7.5.
	Random effects:	The same as in the Example 7.5, but both the non-genetic animal environment effect and the additive genetic animal effect, are modeled by 5 regression coefficients.
Pedigree:		animal model, phantom parent groups

Data file:

herd$_1$	animal$_2$	trgr$_3$	htm$_4$	ym$_5$	age$_6$	dcc$_7$	season$_8$	dim$_9$	month$_{10}$	milk$_1$
1001	50070	1	18810	8810	9	1	3	36	10	23.9
1001	50070	1	18811	8811	9	1	3	64	11	22.4
1001	50070	1	18812	8812	9	1	3	91	12	24.1
1001	50070	1	18901	8901	9	1	3	128	13	27.1
1001	50070	1	18902	8902	9	1	3	161	14	25.1
1001	50070	1	18903	8903	9	1	3	213	15	23.4
1001	50070	1	18905	8905	9	3	3	270	17	17.1
1001	50070	1	18906	8906	9	3	3	301	18	18.9
1001	50070	1	18907	8907	9	4	3	332	19	15.7
1001	50070	2	18911	8911	17	1	1	25	23	27.1
1001	50070	2	18912	8912	17	1	1	52	24	32.3
1001	50070	2	19001	9001	17	1	1	87	25	35.2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Pedigree file:

animal$_1$	sire$_2$	dam$_3$	herd$_4$
50054	34788	50068	1001
50055	90670	50067	1001
50056	28400	−172	1001
50057	30099	−173	1001
50058	29598	−175	1001
50059	33338	50057	1001
⋮	⋮	⋮	⋮

File with covariable table:

Index	Covariable columns									
dim_1	1	2	3	4	5	6	7	8	...	34
4	−0.0810	−0.3189	−0.4110	0.1045	0.4303	−0.0132	−0.4889	−0.2613	...	−0.4341
5	−0.0773	−0.3230	−0.3755	0.0894	0.4293	−0.0152	−0.4755	−0.2443	...	−0.4278
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
365	−0.9614	0.9244	0.0001	0.03211	0.8762	−0.5621	0.9852	−0.1232	...	0.5999

MiX99 instruction file:

```
# Estimation of breeding values for the 1st, 2nd, and 3rd lactation milk yield in
# dairy cattle using a multi-trait random regression test day model based on
# the covariance function.
#
# TITLE:
          Milk 1., 2., and 3. Lactation, MT-RRTDM
# INTEGER:
          herd animal trgr htm ym age dcc season dim month
# REAL:
          milk
# TRAITS:
          3
# TRAITGRP:
          3     3
# DATASORT:
          1     2
# FIXRAN:
          9     11
# MODEL:
#  trgr trt wgt: herd s0 s1 s2 s3 s4 age dcc ym htm n1 n2 n3 n4 n5 a1 a2 a3 a4 a5
    1    1   -      1  8  8  8  8  8   6   7  5   4  2  2  2  2  2  2  2  2  2  2
    2    1   -      1  8  8  8  8  8   6   7  5   4  2  2  2  2  2  2  2  2  2  2
    3    1   -      1  8  8  8  8  8   6   7  5   4  2  2  2  2  2  2  2  2  2  2
# ORDER:  within blocks effects
                    4  -  -  -  -  -   -   -  -   3  1  1  1  1  1  2  2  2  2  2
# RANDOM:                                          htm n1 n2 n3 n4 n5 a1 a2 a3 a4 a5
                                                     1  2  2  2  2  2  3  3  3  3  3
                                                     1  -  -  -  -  -  -  -  -  -  -
                                                     1  -  -  -  -  -  -  -  -  -  -
# RELATIONSHIPS: number:                                                 a1 a2 a3 a4 a5
        5                                                                 1  1  1  1  1
# REGRESS:
# n:herd,s0, s1, s2, s3, s4,age,dcc,ym,htm,n1, n2, n3, n4, n5, a1, a2, a3, a4, a5
 20   cl cl t31 t32 t33 t34  cl  cl cl cl  t1  t2  t3  t4  t5 t16 t17 t18 t19 t20
 20   cl cl t31 t32 t33 t34  cl  cl cl cl  t6  t7  t8  t9 t10 t21 t22 t23 t24 t25
 20   cl cl t31 t32 t33 t34  cl  cl cl cl t11 t12 t13 t14 t15 t26 t27 t28 t29 t30
# COMBINE:
          y
  # MERGE: combine traits for non-genetic and genetic animal effect
  # herd,s0, s1, s2, s3, s4,age,dcc,ym,htm,p1, p2, p3, p4, p5, a1, a2, a3, a4, a5
      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   2  1   1   1   1   1   1   1   1   1   1
      3   3   3   3   3   3   3   3  3   3  1   1   1   1   1   1   1   1   1   1
# CVRFIL:
          covarmilk.tab
# CVRNUM:
          34
# CVRIND:
          9
# PEDIGREE:
          am+p
# DATAFILE:
          /koel/testi/data/multidat.s
# VAR:
          10  1  f
# MISSVA:
```

```
          0.0
# SCALE:
          y
# PEDFILE:
          /koel/testi/data/new.ped.s
# PARFILE:
          1lacmpf.par
# TMPDIR:
          /usr/tmp
#RANSOLFILE: htm  non-g  animal
            y    y       y
# SOLUNF:
          n
# PRECON: diagonal preconditioner for the WpW, block diagonal preconditioner for
# the XpX
          d  d  d  d     b
# PARALLEL:
          6
# COMMONBLOCKS:
          1
```

File with (co)variances for random effects:

1	2	3	1	
1	1	1	variance for M1	
1	1	2	covariance between M1 and M2	
1	1	3	covariance between M1 and M3	herd-test-month
1	2	2	variance for M2	
1	2	3	covariance between M2 and M3	
1	3	3	variance for M3	
2	1	1	variance for n1	
2	1	2	covariance between n1 and n2	
2	1	3	covariance between n1 and n3	
2	1	4	covariance between n1 and n4	
2	1	5	covariance between n1 and n5	
2	2	2	variance for n2	
2	2	3	covariance between n2 and n3	non genetic
2	2	4	covariance between n2 and n4	animal environment
2	2	5	covariance between n2 and n5	
2	3	3	variance for n3	
2	3	4	covariance between n3 and n4	
2	3	5	covariance between n3 and n5	
2	4	4	variance for n4	
2	4	5	covariance between n4 and n5	
2	5	5	variance for n5	
3	1	1	variance for a1	
3	1	2	covariance between a1 and a2	
3	1	3	covariance between a1 and a3	
3	1	4	covariance between a1 and a4	
3	1	5	covariance between a1 and a5	
3	2	2	variance for a2	
3	2	3	covariance between a2 and a3	additive genetic
3	2	4	covariance between a2 and a4	animal
3	2	5	covariance between a2 and a5	
3	3	3	variance for a3	
3	3	4	covariance between a3 and a4	
3	3	5	covariance between a3 and a5	
3	4	4	variance for a4	

```
3  4  5  covariance between a4 and a5
3  5  5        variance for a5
4  1  1    residual variance for M1
4  2  2    residual variance for M2
4  3  3    residual variance for M3
```

CLIM command file:

```
TITLE    Milk 1., 2., and 3. Lactation, MT-RRTDM

DATAFILE    /koel/testi/data/multidat.s
INTEGER     herd animal trgr htm ym age dcc season dim month
REAL        milk
DATASORT    BLOCK=herd PEDIGREECODE=animal
TRAITGROUP  trgr

TABLEFILE   covarmilk.tab
TABLEINDEX  dim

PEDFILE   /koel/testi/data/new.ped.s
PEDIGREE  G am+p

PARFILE   1lacmpf.par
TMPDIR    /usr/tmp

PRECON    d d d d        b
PARALLEL  6 1

RANDOM    htm PE
WITHINBLOCKORDER PE G htm herd

MODEL SCALE
  milk(1) = herd fix(  1 t31 t32 t33 t34|season) age dcc ym &
            htm   PE( t1  t2  t3  t4  t5|animal)@1          &
                  G(t16 t17 t18 t19 t20|animal)@FST
  milk(2) = herd fix(  1 t31 t32 t33 t34|season) age dcc ym &
            htm   PE( t6  t7  t8  t9 t10|animal)@1          &
                  G(t21 t22 t23 t24 t25|animal)@FST
  milk(3) = herd fix(  1 t31 t32 t33 t34|season) age dcc ym &
            htm   PE(t11 t12 t13 t14 t15|animal)@1          &
                  G(t26 t27 t28 t29 t30|animal)@FST
```

7.7 Multiple-trait sire model with direct and indirect genetic effects

Traits:		calves born dead (cbd), calving difficulty (cdiff)
Model:		
	Fixed effects:	calving×month (cmth), number_of_calvings (noc), sex
	Random effects:	herd×year (hy), cow, cow's sire (cosi), calf's sire (casi)
Pedigree:	Sire model;	
	indirect (I):	cow's sire, cow's maternal grandsire
	direct (D):	calf's sire, calf's maternal grandsire

Data file:

noc_1	sex_2	$cmth_3$	cow_4	hy_5	$calf_6$	$casi_7$	$cosi_8$	cbd_1	$cdiff_2$
1	1	8	1237	17093	12371	8920	6953	1	3
2	1	10	1237	17094	12372	9278	6953	1	1
3	2	11	1237	17095	12373	9347	6953	1	1
1	2	5	4329	17095	43291	8920	6511	2	2
1	1	3	6482	17096	64821	7743	7131	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Pedigree file:

$bull_1$	$sire_2$	mgs_3	hy_4
1239	0	0	0
2498	0	0	0
4371	1239	0	0
3981	2498	0	0
5432	0	0	0
6953	5432	4371	0
8920	3981	1239	0
⋮	⋮	⋮	⋮

MiX99 instruction file:

```
# TITLE:
          Sire model: Calves born dead, Calving difficulties
# INTEGER:
          noc sex cmth cow hy calf casi cosi
# REAL:
          cbd cdiff
# TRAITS:
          2
# TRAITGRP:
          1     -
# DATASORT: Block_code, relationship_code
          5     -
# FIXRAN:
          3     4
# MODEL: TRAITGRP trt wgt: cmth sex noc cow hy, cow's sire(=I), calf's sire (=D)
            1     1    -   1    2   3   4   5   8              7    # cbd
            1     2    -   1    2   3   -   5   8              7    # cdiff
# ORDER: within blocks effects
```

```
                              -    -   -   1   3   2                2
# RANDOM:                                  cow hy, cow's sire,      calf's sire
                                           1   2   3                3
                                           -   2   3                3
# RELATIONSHIPS: number:                                cow's sire,      calf's sire
                    2                                    1                2
# REGRESS:        number:   cmth sex noc cow hy, cow's sire,      calf's sire
                    7       cl   cl  cl  cl  cl  cl               cl
                    7       cl   cl  cl   -  cl  cl               cl
# COMBINE:
          n
# PEDIGREE:
          sm
# DATAFILE:
          keino.dat
# VAR:
          8  2  f
# MISSVA:
          0.0
# SCALE:
          y
# PEDFILE:
          keino.ped
# PARFILE:
          par.ay
# TMPDIR:
          .
#RANSOLFILE: cow, hy, sire
           y   y   y
# SOLUNF:
          n
# PRECON: block diagonal preconditioner for WpW and full block for XpX
          b  b  b     f
# PARALLEL:
          1
# COMMONBLOCKS:
          0
```

File with (co)variances for random effects:

1	2	3	1	
1	1	1	variance for cbd	cow
2	1	1	variance for cbd	
2	1	2	covariance between cbd and cdiff	herd-year
2	2	2	variance for cdiff	
3	1	1	variance for cbd-I	
3	1	2	covariance between cbd-I and cdiff-I	
3	1	3	covariance between cbd-I and cbd-D	
3	1	4	covariance between cbd-I and cdiff-D	
3	2	2	variance for cdiff-I	genetic
3	2	3	covariance between cdiff-I and cbd-D	
3	2	4	covariance between cdiff-I and cdiff-D	
3	3	3	variance for cbd-D	
3	3	4	covariance between cbc-D and cdiff-D	
3	4	4	variance cdiff-D	
4	1	1	residual variance for cbd	
4	1	2	residual covariance between cbd and cdiff	residual
4	2	2	residual variance for cdiff	

CLIM command file:

```
TITLE    Sire model: Calves born dead, Calving difficulties

DATAFILE keino.dat
INTEGER  noc sex cmth cow hy calf casi cosi
REAL     cbd cdiff
DATASORT BLOCK=hy

PEDFILE  keino.ped
PEDIGREE G sm

PARFILE  par.ay

PRECON   b b b       f

RANDOM   cow hy

MODEL SCALE
  cbd   = noc sex cmth cow hy G(cosi casi)
  cdiff = noc sex cmth     hy G(cosi casi)

WITHINBLOCKORDER cow G hy
```

7.8 Non-linear mixed model analysis by Gompertz function

This model can be specified by MiX99 instructions only.

Traits:		live weight
Model:		
	Fixed effects:	sex
	Random effects:	non-genetic animal effect, genetic sire effect
Pedigree:		sire model

Data file:

$block_1$	$sire_2$	sex_3	$animal_4$	$time_5$	y_1	$ln(y)_2$
1	11	1	211	50	23.4835	3.156296
1	11	1	211	57	28.4946	3.349713
1	11	1	211	64	34.1581	3.530999
1	11	1	211	71	40.2846	3.695969
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Pedigree file:

$bull_1$	$sire_2$	mgs_3	$block_4$
1	0	0	1
11	1	0	1
12	1	0	1
13	1	0	1
14	1	0	1
⋮	⋮	⋮	⋮

File with covariable table:

Index	Covariable columns				
$time_1$	1	2	3	4	5
50	1	1	1	50	0.850
51	1	1	1	51	0.867
52	1	1	1	52	0.884
⋮	⋮	⋮	⋮	⋮	⋮

MiX99 instruction file:

```
# TITLE:
          Growth curve data
# INTEGER:
          block sire sex animal time
# REAL:
          y ln(y)
# TRAITS:
          1
# TRAITGRP: trait_groups, trait_group_code
          1     -
# DATASORT: block_code, relationship_code (no multiple resid. var.)
          1     4
# FIXRAN: number of fixed and random factors in the model
          3     6
# MODEL: trgr  type  trait  weight:   sex   animal   sire
           1     G     2     -       3 3 3   4 4 4   2 2 2
```

```
# WITHINBLOCKORDER: order of effects within block
                                    - - -   2 2 2   1 1 1
# RANDOM: animal, sire
          1 1 1   2 2 2
# RELATIONSHIPS: number: sire
          3       1 1 1
# REGRESS: number:   sex         animal        sire
           9      t1 t2 t3     t1 t2 t3     t1 t2 t3     t5
# COMBINE:
          n
# -------------------------
# Use covariable table
# -------------------------
          # CVRFIL: name of the file with covariable table
                    tmp.cov
          # CVRNUM: number of covariable columns in the file
                    5
          # CVRIND: integer column in the data file containing the covariable index
                    5
# PEDIGREE: sire model
          sm
# DATAFILE:
          tmp.dat
# VAR:    integer  real  formatted
          5  2  f
# MISSVA: code for missing real values
          0.0
# SCALE:
          n
# PEDFILE:
          NEW.PED
# PARFILE:
          PARIN
# TMPDIR:
          .
#RANSOLFILE: non-gen env., sire
          y     y
# SOLUNF: unformatted solution file
          y
# PRECON: diagonal preconditioner for all WpW, and for the XpX equations
          d  d     d
# PARALLEL:  number of processors used by the solver program
          1
# COMMONBLOCKS: number of blocks in common
          0
```

File with (co)variance components:

1	2	3	1	
1	1	1	100.0	
1	2	1	0.0	non-gen env.
1	2	2	10.0	
1	3	1	0.0	
1	3	2	0.0	
1	3	3	1.0	
2	1	1	100.0	
2	2	1	0.0	sire
2	2	2	10.0	
2	3	1	0.0	
2	3	2	0.0	
2	3	3	1.0	
3	1	1	1.0	residual

7.9 Bivariate model with one categorical trait

The analysis includes two traits, where one is a continuous trait (somatic cell count) and one is a categorical trait (clinical mastitis). The categorical trait is modeled by a threshold model. So far, this model can be specified by MiX99 instructions only.

Traits:		somatic cell count (scc), clinical mastitis (cm)
Model:		
	Fixed effects:	age, year×month (YM)
	Random effects:	herd×year with a 3-years period (HY3), sire
Pedigree:		sire model

Data file:

$block_1$	$sire_2$	age_3	$HY3_4$	YM_5	scc_1	cm_2
0	4905090	6	1000013	966	368	1
0	3718628	4	1000013	975	585	1
0	3725954	6	1000013	971	421	1
0	4844676	6	1000015	1001	681	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Pedigree file:

$bull_1$	$sire_2$	Mgs_3	$block_4$
3313937	4904314	4904331	0
3336938	4903585	4902160	0
3363715	4904314	4904934	0
3363717	4904314	4901592	0
⋮	⋮	⋮	⋮

MiX99 instruction file:

```
# TITLE:
          Threshold analysis (scc and cm)
# INTEGER:
          block sire age HY3 YM
# REAL:
          scc cm
# TRAITS:
          2
# TRAITGRP:  trait_groups, column_with_group_code
          1      -
# DATASORT: block_code, relationship_code (no multiple res. var.)
          1      2
# FIXRAN: number of fixed and random factors in the model
          2      2
# MODEL:  trait_group  type  trait  weight:   age    YM    HY3    sire
                  1      L      1      -      3      5      4      2
                  1     T1      2      -      3      5      4      2
   # THR_METHOD
          nr    ft
   # THRESHOLDS
          0.0
# WITHINBLOCKORDER: order of effects within block
                                             -      1      2      3
# RANDOM: HY3, sire
          1      2
```

```
          1     2
# RELATIONSHIPS: number: sire
            1     1
# REGRESS: number:  age    YM     HY     sire                  # all effects are
            4       cl     cl     cl     cl                    # classifications, no
            4       cl     cl     cl     cl                    # regression effects
# COMBINE:
          n
# PEDIGREE: sire model
          sm
# DATAFILE:
          data.dat
# VAR:    integer  real  formatted
          10  5  f
# MISSVA: code for missing real values
          -9.0
# SCALE:
          n
# PEDFILE:
          data.ped
# PARFIL:
          THRESHcmscc.para
# TMPDIR:
          .
#RANSOLFILE: HY3, sire
          y     y
# SOLUNF: no unformatted solution file
          n
# PRECON: block diagonal preconditioner for all WpW, full block for the XpX
# equations
          b  b  b     f
# PARALLEL:  number of processors used by the solver program
          1
# COMMONBLOCKS: number of blocks in common
          0
```

File with (co)variance components:

1	2	3	1	
1	1	1	747.60	
1	1	2	−10.8340	herd-year
1	2	2	2.464	
2	1	1	251.98	
2	1	2	4.31636	sire
2	2	2	0.1857	
3	1	1	7941.3	
3	1	2	33.582	residual
3	2	2	1	

7.10 Marker-assisted BLUP with one QTL effect

Marker-assisted BLUP with one QTL effect (Herman Mulder, Wageningen University). The QTL is modeled by two haplotype effects, which are combined within the model. An IBD matrix is provided for the QTL effect. So far, this model can be specified by MiX99 instructions only.

Data file:

animal$_1$	sire$_2$	dam$_3$	mean$_4$	haplotype1$_5$	haplotype2$_6$	phenotype$_1$
1	0	0	1	1	2	0.380925
2	0	0	1	3	1	0.375538
3	0	0	1	4	1	2.618100
4	0	0	1	5	6	0.336157
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Pedigree file:

animal$_1$	sire$_2$	dam$_3$
1	0	0
2	0	0
3	0	0
4	0	0
⋮	⋮	⋮

File with (co)variance components:

1	2	3	1	
1	1	1	0.0075	QTL effect
3	1	1	0.2850	polygenic effect
4	1	1	0.7000	residual effect

File with Inverse IBD matrix:

haplotype id$_1$	haplotype id$_2$	nonzero element$_1$
1	1	1.50000
2	2	5.69483
3	2	−0.92179
3	3	5.69483
4	1	0.50000
4	2	−0.92179

MiX99 instruction file:

```
# TITLE:
          Marker-assisted BLUP for a single trait with IBD-matrix (H. Mulder)
# INTEGER:
          animal sire dam mean haploty1 haploty2
# REAL:
          phenotype
# TRAITS:
          1
# TRAITGRP:
          1 -
# DATASORT: Block_code, Relationship_code
          -              -
# FIXRAN: Fixed factors, Random factors, External correlation matrix
                                        # One random effect (haploty1) has an
                                        # correlation matrix between levels.
                                        # This matrix will be read from a
                                        # file provided by the user. For this
```

```
                                        # example it is an inverse IBD matrix.
          1              3               1
# MODEL:  Traitgrp., Trait, Weight; mean haplotype1 haplotype2 animal
          1          1      -       4    5          6          1
# WITHINBLOCKORDER: Order of effects within blocks
          # The character "<" instructs to combine haplotype1 and haplotype2,
          # i.e. levels of haplotype2 effect are considered as levels of
          # haplotype1 effect.
                                  -    -     <    -          1
# RANDOM: haplotype1 haplotype2 animal
          1          2          3
# RELATIONSHIPS: Number: animal
          1       1
# REGRESS: Number: mean haplotype1 haplotype2 animal
          4       cl   cl         cl         cl
# COMBINE:
          n
# PEDIGREE:
          am
# DATAFILE:
          MAS.dat
# VAR:
          6 1 f
# MISSVA:
          0.0
# SCALE:
          n
# PEDFILE:
          MAS.ped
# CORRFILE: Random effect number, Filename
          # The file contains an inverse IBD matrix for the QTL effect.
          # The size of the matrix is equal to the total number of different
          # haplotypes present in the animals in the complete pedigree.
          # The file contains the diagonal and the lower triangle non-zeros of
          # of the inverse IBD matrix in the co-ordinate format.
          # The matrix is associated with the random effect 1 (=haplotype1).
          1 1 MAS.ibd
# PARFILE:
          MAS.var
# TMPDIR:
          .
#RANSOLFILE: animal haplotype1 haplotype2
          y      y          y
# SOLUNF:
          n
# PRECON: WpW, XpX
          b    d
# PARALLEL:  Number of processors used by the solver program
          1
# COMMONBLOCKS  Number of blocks in common area when using parallel processing
          0
```

7.11 Group Selection

Example for net daily gain in pigs, where the statistical model considers also the social effect of the animal's pen mates. The model and the sample data as well as the variance components were provided by Piter Bijma, Wageningen University.

Trait:
Net daily gain in pigs.

Effects in the model:
Fixed: number of pen mates, line, and sex
Random: litter, group, animal (direct effect), mate2, mate3, . . . , mate12, error term
Some effects were left out due to demonstration purposes.

Variable number of pen mates:
It is usual for such a model that not all animals have equal number of pen mates. Because MiX99 does not allow missing class information yet (in this case missing id's for mates) the following strategy can be adopted:

- the model includes as many mate effects as there are in the largest group.
- for pens with less animals dummy mates are specified. All dummy mats can have the same id. The dummy mate id has to be also in the pedigree file with sire and dam set to zero.
- in case a dummy-id is coded for a mate effect, a zero covariable must be associated with this "dummy" mate effect.

The zero covariables can be provided in the data file or in a covariable table. In case a covariable table is used, a strict ordering of the missing mates in the data file is required. For each observation, the order of the animal effect id's in the provided data columns (specified in the MODEL line) has to be: in the first animal id column (column 1 in this example) the id of the animal with the direct effect is given (on which the observation was made), followed by the id's of mates (starting with column 5 in this example). The remaining columns (up to column 15 in this example) are filled with the dummy id for missing mates, in case of missing mats. Further, each record must contain the number of animals in a pen, i.e., animal plus number of mates. The number is used as a reference index for the covariable table. This way a covariable of 1.0 will be associated to all mates and a zero covariable to all "dummy" mates.

File with covariable table:

size	animal	mate1	mate2	mate3	mate4	mate5	mate6	mate7	mate8	mate9	mate10	mate11
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0
11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0
12	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

MiX99 instruction file:

```
# Example for group selection provided by Piter Bijma, University Wageningen
#
# Model:
#  net_daily_gain = number_of_pen_mates + line + sex + litter_id + group +
#                   + animal + mate2 + mate3 + ... + mate12 + e
#
#  random effects: litter_id, group, animal(direct & mate)
#
#  Variances/covariances ASReml:
#   group                     2011  2011  0.425608      1370.73    13.38   0 U
#   litterid                  2056  2056  0.731717E-01   235.660    5.59   0 U
#   residual                 16434 13959  1.00000       3220.64    33.54   0 P
#   animal (direct)       UnStruct     1  0.472707      1522.42     9.72   0 U
#   animal (cov dir mate) UnStruct     1  0.173330E-01    55.8234   2.07   0 U
#   animal (mate)         UnStruct     2  0.157223E-01    50.6359   5.60   0 U
#
#-------------------------------------------------------------------------------
# title
  Group selection for net daily gain in pigs (model & data by Piter Bijma)
# INTEGER
#   1       2       3       4       5     6     7     8     9    10    11
  animal Npenmate group compartm mate2 mate3 mate4 mate5 mate6 mate7 mate8   &
#  12    13     14     15       16     17  18
  mate9 mate10 mate11 mate12 litterid sex line
# REAL
#    1
  netdgain
# traits
  1   L
# trait-subgroups, input column
  1                -
# input column of block code and animal code; residual class
  -  2
# number of fixed and random effect factor columns in the model lines
  3  14
# MODEL:
# trgr trait wgt:    line  litter
#               :Nmates   sex   group ani m2 m3 m4 m5 m6 m7  m8  m9  m10 m11 m12
  1      1    -    2  18  17  16  3    1  5  6  7  8  9  10  11  12  13  14  15
# WITHINBLOCKORDER: order of effects within block
                   -   -   -   3  2    1  1 <1 <1 <1 <1 < 1 < 1 < 1 < 1 < 1 < 1
# RANDOM:                litter group ani m2 m3 m4 m5 m6 m7  m8  m9  m10 m11 m12
                            1     2    3   3  -  -  -  -  -   -   -   -   -   -
# PEDIG:  N:                          ani m2 m3 m4 m5 m6 m7  m8  m9  m10 m11 m12
         12                           1   2  3  4  5  6  7   8   9   10  11  12
# trgr trait wgt:    line  litter
# REGRESS:
#     N: Nmates line sex litter group ani m2 m3 m4 m5 m6 m7  m8  m9  m10 m11 m12
     17    cl    cl   cl   cl    cl    t1 t2 t3 t4 t5 t6 t7  t8  t9  t10 t11 t12
# combining of the effects
  n
# covariable file name:
  GS.cov
# number of the covariable columns
  12
# integer data column with the covariable index (= number of animals in pen)
  2
# method used for relationship
  am
# input file
  GS.dat
# int-col. real-col. form;
   18         1       f
# code for missing real values
  -99.
# scaling (y/n)
```

```
   n
# pedigree file
  GS.ped
# parameter file
  GS.par
# directory for the temporary files
  .
#solution files for the random effects: litterid group animal_effects
                                          y      y        y
# binary solution file
  n
# type of preconditioner WpW, XpX
# WpW   1      2      3;      XpX
#     animal group litterid
        d      d      d        b
# PARALLEL
  1
# COMMONBLOCKS last pedigree block in common
  1
```

File with (co)variance components:

1	2	3	1	
1	1	1	235.66	Litter
2	1	1	1370.73	Group
3	1	1	1522.42	animal (direct)
3	2	1	55.8234	cov. (animal, mate)
3	2	2	50.6359	mate
4	1	1	3220.64	residual effect

8 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.

9 References

Duff, Iain S., Erisman, Albert M., and Reid, John K. (1992). *Direct methods for sparse matrices*. Oxford, UK: Clarendon Press (cit. on p. 6).

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. 20).

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. 20).

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 (cit. on p. 20).

Lidauer, M., Matilainen, K., Mäntysaari, E. A., Pitkänen, T. J., Taskinen, M., and Strandén, I. (2023). *Technical Reference Guide for MiX99 Solver*. Release X/2023. Natural Resources Institute Finland (Luke) (cit. on pp. 3, 7, 12, 14, 19, 27, 36, 40).

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 p. 6).

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 (cit. on p. 1).

Liu, Z., Goddard, M. E., Reinhardt, F., and Reents, R. (2014). "A single-step genomic model with direct estimation of marker effects". In: *J. Dairy Sci.* 97.9, pp. 5833–5850. DOI: 10.3168/jds.2014-7924 (cit. on p. 31).

Mäntysaari, E. A., Evans, R. D., and Strandén, I. (2017). "Efficient single-step genomic evaluation for a multibreed beef cattle population having many genotyped animals". In: *J. Anim. Sci.* 95.11, pp. 4728–4737. DOI: 10.2527/jas2017.1912 (cit. on p. 28).

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).

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).

Schaeffer, L. R. and Dekkers, J. C. M. (1994). "Random regressions in animal models for test-day production in dairy cattle". In: *Proc. 5th World Congr. Genet. Appl. Livest. Prod.* Vol. 18, pp. 443–446 (cit. on pp. 13, 53).

Strandén, I. (2023). *Command Language Interface for MiX99*. Release X/2023. Natural Resources Institute Finland (Luke) (cit. on pp. 3, 16, 28, 31, 37).

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 (cit. on p. 1).

Strandén, I. and Lidauer, M. (2001). ”Parallel computing applied to breeding value estimation in dairy cattle”. In: *J. Dairy Sci.* 84.1, pp. 276–285. DOI: 10.3168/jds.S0022-0302(01)74477-3 (cit. on p. 1).

Vuori, K., Strandén, I., Sevón-Aimonen, M.-L., and Mäntysaari, E. A. (June 2006). ”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 (cit. on pp. 2, 19).

Appendix A Convert99: convert data between text and binary forms

Some of the MiX99 input and output files are of binary format. For example, MiX99 input data file can be either a (formatted) free format text file or an unformatted (binary format) binary file.

While text files are human-readable and can be easily modified with a text editor, binary files, on the other hand, are harder to read and modify.

If binary file has a repetitive structure of similar records, it can be converted to a (formatted) free format text file using MiX99 program `convert99`. Similarly, if each row of a text file contains equal number of space separated columns with similar information, the file can be converted to binary format using `convert99`.

1.1 Example

MiX99 program `convert99` is operated with command line parameters. By default it assumes that a text file is going to be converted into binary file so that in each line there are some integer valued columns followed by some real valued columns, each column separated with one or more spaces.

Example text file (`file.txt`) with 3 integer and 2 real columns:

```
5  102  3  5123.5   180.4
6  102  3  7597.0   243.8
7  103  4  6410.3  -888.0
8  103  3  -888.0   210.7
```

To convert the text file (`file.txt`) into binary file using `convert99` program, number of integer (`3`) and real columns (`2`) must be indicated with the command line parameters:

```
convert99 3 2 < file.txt > file.bin
```

where, by default, standard input and output streams are used.

To convert the binary file back to text file, command line option `-b` must be added:

```
convert99 -b 3 2 < file.bin > file2.txt
```

Content of the file `file2.txt` is after the conversion:

```
5 102 3 5123.500 180.4000
6 102 3 7597.000 243.8000
7 103 4 6410.300 -888.0000
8 103 3 -888.0000 210.7000
```

1.2 Usage

Above binary file `file.bin` is written with ***32-bit integer*** and ***single precision floating point*** values, format that is assumed for the unformatted MiX99 input data file.

Other types can be specified with optional first command line parameter “TYPES”:

```
convert99 [-b] [--checkdata] [TYPES] COUNT1 [COUNT2 ...] [INFILE [OUTFILE]]
```

where

-b Convert from BINARY to TXT. Default: TXT to BINARY.

--checkdata Enhanced checking of the input file.

TYPES Characters describing groups of number values. Default: "SF".

- **S** Small integer.
- **I** Default integer. Typically the same as S.
- **L** Large integer.
- **F** Single precision float.
- **D** Double precision float.

COUNT1 Count of the first group of numbers.

COUNT2 Count of the second group of numbers.

... Other group counts.

INFILE Input file name. Default: uses standard input.

OUTFILE Output file name. Default: uses standard output.

Note that if the input text file has more columns on a line or binary file has more binary values on a record than is specified with the command line COUNTS, the rest of the line/record is skipped.

The conversion example above from text to binary is the same as:

```
convert99 SF 3 2 file.txt file.bin
```

The same input file (file.txt) could be used to convert into different types of integers and floats:

```
convert99 ILD 1 2 2 file.txt > file2.bin
```

where one of the three integers is converted into default and two into large integers, and both reals into double precision floats. This can be converted back to text and additionally skipping the last real value with:

```
convert99 -b ILD 1 2 1 < file2.bin
```

in which case the output is

```
5 102 3 5123.500000000000
6 102 3 7597.000000000000
7 103 4 6410.300000000000
8 103 3 -888.0000000000000
```

Appendix B Dynamically loaded datafilter plugin

Previously, some of the MiX99 users edited and modified `mix99i.f90` source code to limit or filter the data file lines that the MiX99 preprocessor (`mix99i`) accepted for further processing. For this, most of the MiX99 source code repository needed to be available and compiled separately for each different acceptance rule.

MiX99 user can now implement his/hers own data file filter rule by creating a simple Fortran subroutine and compiling just a single source code file into a binary object file (`.so` or `.dll`). MiX99 can then be instructed to ***dynamically load*** that specific object file and use the user given data file filter subroutine to limit, filter, or even change the data file content.

For each data file line (or record if binary data file) MiX99 preprocessor calls subroutine named `accept_datarecord()` to determine whether the data file line is included (accepted) or omitted from the calculations.

2.1 Datafilter plugin examples

Example Fortran source file `accept_all.f90` that accepts all data file lines:

```
module Kinds
  implicit none
  integer, parameter :: machine_integer = 0
  integer, parameter :: int_defau = kind(machine_integer)
  integer, parameter :: int_small = selected_int_kind(9)
  integer(kind=int_defau), parameter :: real_low = selected_real_kind(p=6,r=30)
end module Kinds

function accept_datarecord(i_data_records, n_integer_cols, integer_cols, &
                 n_real_cols, real_cols) bind(C, name="accept_datarecord")
  use Kinds
  implicit none
  integer(kind=int_defau), intent(in)    :: i_data_records, n_integer_cols
  integer(kind=int_small), intent(inout) :: integer_cols(*)
  integer(kind=int_defau), intent(in),    optional :: n_real_cols
  real(kind=real_low),     intent(inout), optional :: real_cols(*)
  logical :: accept_datarecord

  ! All records are accepted:
  accept_datarecord = .TRUE.

end function accept_datarecord
```

Each data line/record can be accepted depending on the line/record content:

- Subroutine arguments:
  - `i_data_records` Line number (record number for binary file) of the data file.
  - `n_integer_cols` Number of integer columns in the data file.
  - `integer_cols` Array of the integer column on this line/record.
  - `n_real_cols` Number of real columns in the data file.
  - `real_cols` Array of the real column this line/record.
- Note that the preprocessor reads only the integer columns when reading the data file the first time!
- Column values can also be modified here.

- Return value: logical
  - Return `.TRUE.` if this data line/record is accepted.
- Module `Kinds` is included in the source code in order to be make sure that the variable types match the MiX99 types.

The data filter could filter the data line, for example, according to the line number (`accept_first3.f90`):

```
    ! Accepted first 3 lines:
    accept_datarecord = (i_data_records <= 3)
```

according to line content (`accept_cows.f90`):

```
    ! All records having value 2 at the second integer column:
    accept_datarecord = (integer_cols(2) == 2)   ! Bull = 1, cow = 2.
```

or change the line content (`accept_modify.f90`):

```
    ! Values of the first real value column are multiplied by 2:
    if (present(real_cols)) then
      real_cols(1) = 2.0 * real_cols(1)
    endif
```

These examples are available in the `examples/datafilter` directory of the MiX99 distribution.

2.2 Adding columns using plugin "hook" routines

The data filter plugin can also add columns to the lines read from the data file. This can be done by including user defined plugin "hook" routines in the plugin source file.

If found, the following plugin routines are called when reading the data file:

Subroutine name	Called when
`init_datafilter(nfi,nfr)`	opening data file
`rewind_datafilter()`	rewinding data file
`close_datafilter()`	closing data file

For example, if the data file contains one integer column less than is declared in the model file, the missing column can be added in the `accept_datarecord` routine. For this, in order to prevent MiX99 preprocessor from reading one too many integer columns from the data file, the number of integer columns needs to be temporarily corrected, i.e., decreased by one in `init_datafilter` routine:

```
subroutine init_datafilter(n_file_int_cols, n_file_real_cols) &
           bind(C, name="init_datafilter")
  use Kinds
  implicit none
  integer(kind=int_defau), intent(inout) :: n_file_int_cols, n_file_real_cols

  n_file_int_cols = n_file_int_cols - 1

end subroutine init_datafilter
```

and `accept_datarecord` routine can then fill in the missing column. For example:

```
  integer_cols(3) = integer_cols(1) * integer_cols(2)
  accept_datarecord = .TRUE.
```

The two other “hook” routines can be used to rewind and close the user defined data filter processing:

```
subroutine rewind_datafilter() bind(C, name="rewind_datafilter")
end subroutine rewind_datafilter

subroutine close_datafilter() bind(C, name="close_datafilter")
end subroutine close_datafilter
```

2.3 Compiling datafilter plugin

The data filter subroutine needs to be compiled into an object file. This can be done with the help of `compile_dll99` script:

```
    compile_dll99 accept_all
```

where `accept_all` is the name of the Fortran source file (`accept_all.f90`) without the file extension (`.f90`).

This will create a shared object file `accept_all.so` (or `accept_all.dll` in Windows) that can be dynamically loaded on demand.

Intel Fortran compiler (`ifort`) is used by default but can be changed in Linux environment to GNU Fortran compiler using `-gnu` option:

```
    compile_dll99 -gnu accept_first3
```

Data filter plugin can also be compiled using debugging settings with option `-debug`:

```
    compile_dll99 -debug accept_cows
```

2.3.1 Windows

Currently, only Intel Fortran compiler (`ifort`) is supported in Windows. Also, the plugin hook routines need to be exported:

```
subroutine init_datafilter(n_file_int_cols, n_file_real_cols) &
           bind(C, name="init_datafilter")
#ifdef FORWINDOWS
  !DEC$ ATTRIBUTES DLLEXPORT :: init_datafilter
#endif
  use Kinds
  implicit none
  integer(kind=int_defau), intent(inout) :: n_file_int_cols, n_file_real_cols
end subroutine init_datafilter

subroutine rewind_datafilter() bind(C, name="rewind_datafilter")
#ifdef FORWINDOWS
  !DEC$ ATTRIBUTES DLLEXPORT :: rewind_datafilter
#endif
end subroutine rewind_datafilter

subroutine close_datafilter() bind(C, name="close_datafilter")
#ifdef FORWINDOWS
  !DEC$ ATTRIBUTES DLLEXPORT :: close_datafilter
#endif
end subroutine close_datafilter
```

Routine `accept_datarecord` is exported by default in `compile_dll99.bat` so it does not need the `DLLEXPORT` declaration.

2.4 Using datafilter plugin

Currently, special version of the MiX99 preprocessor, `mix99i_datafilter` needs be used for the data filter plugin. This is because of an extra loop in the preprocessor which can slow down the operation if no data filter plugin is used. In the future, the data filter feature may be included in the (normal) preprocessor (`mix99i`).

The data filter plugin name needs to be given to MiX99 preprocessor using option `--datafilter`:

```
    mix99i_datafilter --datafilter ./accept_all
```

Note that the absolute path needs to be given (here `./`) for the object file but the file extension (`.so` or `.dll`) can be omitted.

MiX99 preprocessor will dynamically load (DL) the shared object file (`accept_all.so` or `accept_all.dll`) and searches for the subroutine `accept_datarecord` from it. If found, the routine is called for each line/record of the data file to accept or omit the line/record from the rest of the operations.

```
************************  M i X 9 9 i   N o t i c e  ************************
  Notice:                                          Time: 15:36:52.8  04.08.2017
    Data file filter plugin loaded and activated: ./accept_all
********************************************************************************
```

2.5 Reduced size `Lambda.data`/`ySim.data` files

MiX99 preprocessor reads certain files in sync with the data file in the case of heterogeneous variance adjustment (`Lambda.data(i)`) and with generated observations (`ySim.data(i)`).

By default, these files are assumed to be of the full size, i.e. containing the same number of lines as the full data file, regardless of the data filter (`accept_datarecord` routine) accepting or rejecting the lines.

If the files are, instead, of the reduced size, i.e. containing the same number of lines as the `accept_datarecord` routine accepts, this needs to be specified on the command line using `--reduced_data` option:

```
    mix99i_datafilter --datafilter ./accept_odd --reduced_data
```

```
************************  M i X 9 9 i   N o t i c e  ************************
  Notice:                                          Time: 13:13:20.1  07.08.2017
    Data file filter plugin loaded and activated: ./accept_odd
      - Lambda.data and ySim.data files are assumed to be of reduced size.
********************************************************************************
```

Index

Index entry styles:
- normal index entry
- CLIM input commands
- file names
- MiX99 input commands
- shell commands

Page numbers:
- primary definition: **86**
- also referred: 86
- ☞ see example on page