

MiX99

Solving Large Mixed Model Equations

Release XI/2017

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

Last update: Nov 2017
©Copyright 2017

Preface

Development of MiX99 was initiated to allow more sophisticated models in estimation of breeding values for dairy cattle. In the first versions the emphasis was on computational efficiency and the target users were experts on genetic evaluations. Therefore the logic of model definitions were more from an animal breeding perspective. The foremost application of this software is solving of large-scale genetic and genomic evaluations for national dairy cattle evaluations. Nevertheless, we have tried to keep the software as a general tool, where many models can be used. As a result, besides cattle, MiX99 is used in genetic evaluation of other species like pig, horse, sheep, goats, fish, foxes, poultry, and for many types of research work.

Disclaimer

MiX99 software is owned by Biometrical Genetics at 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, neither under the same nor under a different name. Any decisions based on information given 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 (firstname.lastname@luke.fi). Please report any bugs to the authors. MiX99 can be referenced by ([MiX99 Development Team, 2017](#)). If you would like to use MiX99, please contact Biometrical Genetics at Natural Resources Institute Finland¹.

MiX99 development features (DEV)

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

Biometrical Genetics,
Green Technology,
Natural Resources Institute Finland (Luke),
FI-31600 Jokioinen, Finland
firstname.lastname@luke.fi
<http://www.luke.fi/mix99>

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

Contents

1	Introduction	1
2	How to run the mix99i pre-processor	3
2.1	Computing environment	3
2.2	MiX99 pre-processor input files	3
2.3	Running the pre-processor	3
2.4	Command line options	3
3	Editing the input files	5
3.1	Data file	5
3.1.1	Sorted records in the input data file (optional)	5
3.1.2	Multiple input data files	8
3.1.3	Visualizing block-to-block dependencies	8
3.2	Pedigree file	10
3.2.1	Sorting the pedigree file	11
3.3	File with (co)variance components (PARFILE)	11
3.4	Multiple residual (co)variances (RESFILE)	13
3.5	File with a covariable table	13
4	MiX99 instruction file	14
4.1	Instruction lines	14
5	Output files of the MiX99 pre-processor	36
5.1	MiX99.lst file	36
5.2	Log-files	36
5.3	Temporary work files	36
6	Using old solutions	38
7	Examples of MiX99 instruction files	39
7.1	Multiple trait animal model	39
7.2	Multiple-trait model: different models by trait	42
7.3	Random regression animal model	45
7.4	Multiple-trait model with trait groups	48
7.5	Random regression model based on covariance functions	51
7.6	Multiple trait random regression test-day model with covariance functions	54
7.7	Multiple-trait sire model with direct and indirect genetic effects	58
7.8	Non-linear mixed model analysis by Gompertz function	61
7.9	Bivariate model with one categorical trait	64
7.10	Marker-assisted BLUP with one QTL effect	66
7.11	Group Selection	68
8	Acknowledgement	71
9	References	71

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

Appendix A	Convert99: convert data between text and binary forms	73
1.1	Example	73
1.2	Usage	73
Appendix B	Dynamically loaded datafilter plugin	75
2.1	Datafilter plugin examples	75
2.2	Adding columns using plugin “hook” routines	76
2.3	Compiling datafilter plugin	77
2.3.1	Windows	77
2.4	Using datafilter plugin	78
2.5	Reduced size <code>Lambda.data/ySim.data</code> files	78
Index		78

What is new?

New features and options in MiX99 since release XI/2016:

- 1) New parallel solver (`mix99p`) **memory option X** and corresponding **command line option `-x`** have been found to speed up the parallel computations significantly in some cases.
- 2) New preprocessor option to specify the **block code** of the **first common block** instead of the number of the common blocks.
- 3) The ordering of block codes for parallel processing (`mix99p`) is corrected in the case of data-only block codes. This may affect slightly the ordering of the blocks.
- 4) Block communication information program `block_information` can now optionally show the communication due to the data structures and the variance structures separately. Additionally, the communication matrix can be stored as a graph file that can be used to minimize the communication by reordering and partitioning the blocks using external `gmetis` program.
- 5) MiX99 solvers (`mix99s` and `mix99p`) can now be instructed to store intermediate solutions during the iteration by creating a file named `PEEK` or from the command line (options `-peek` and `-peek_step`).
- 6) **ssGTBLUP "Efficient single-step genomic evaluation for a multibreed beef cattle population having many genotyped animals"** has been implemented. When the T matrix is read to memory, parallel BLAS subroutine is used in the multiplication. Thus, even the single processor version can use parallel computing through multiple cores in the computer.
- 7) Single-step in the parallel solver (`mix99p`) no longer requires the genotyped animals to be in the common block area when dense matrix format is used in storing the external matrix needed in single-step. Note, however, that this restriction is still in effect when the external file is in the co-ordinate format.
- 8) New experimental feature has been added to MiX99: ability to dynamically load (DL) user compiled binary object files. This is (first) used as a new user given **datafilter plugin** functionality.

DEV

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 more number of records. In a national evaluation, number of test-day records and number of unknowns in the mixed model equations (MME) are easily more than 100 million.

Computational techniques and algorithms that were found useful for solving animal models may take weeks to obtain solution to large random regression test day model MME. Consequently, 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 [Gompertz function](#). Recent exploitation of genomic data has initiated 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 component for large and complex models. Thus far, a test version of a stochastic EM REML algorithm has been implementation for multiple-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 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 for the number of the following effects: fixed effects with small number of levels, fixed effects with 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 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 multiple 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 to apply different residual (co)variance matrices. This might be useful, if 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 covariance structure must be provided by the user in form of an [inverse correlation matrix](#) (for instance an inverse IBD matrix).
- 14) Models with fixed or random regression variables with large number of regression effects. This is mostly useful for the genomic data where 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.

How to run the mix99i pre-processor

Computing environment

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

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 nor in the path. If there is space within path or file name, the complete file name information with path need to be given between double quotes ("). For example, "my directory/file name" in Linux/Unix but "my directory\file name" in Windows.

Running the pre-processor

Solving mixed model equations using MiX99 involves 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](#).

After successful completion of the preprocessor, 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 existence of the `OK_mix99i` file.

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 `mix99s`, can be obtained with

```
mix99i -h
```


TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

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]
      [-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 of processes.
--datafile DATAFILE: Data file name.
--pedfile PEDFILE: Pedigree file name.
--parfile PARFILE: Variance file name.
--checkdata: Enhanced checking of data file.
-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](#).

Editing the input files

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., 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 $\pm 2.147.483.648$. 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 is allowed to 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.

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 reliabilities and for [parallel computing](#). 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](#). This ordering of mixed model equations yields for many animal breeding problems a structured coefficient matrix (Figure 1) that has nearly doubly-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. As a consequence, 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.

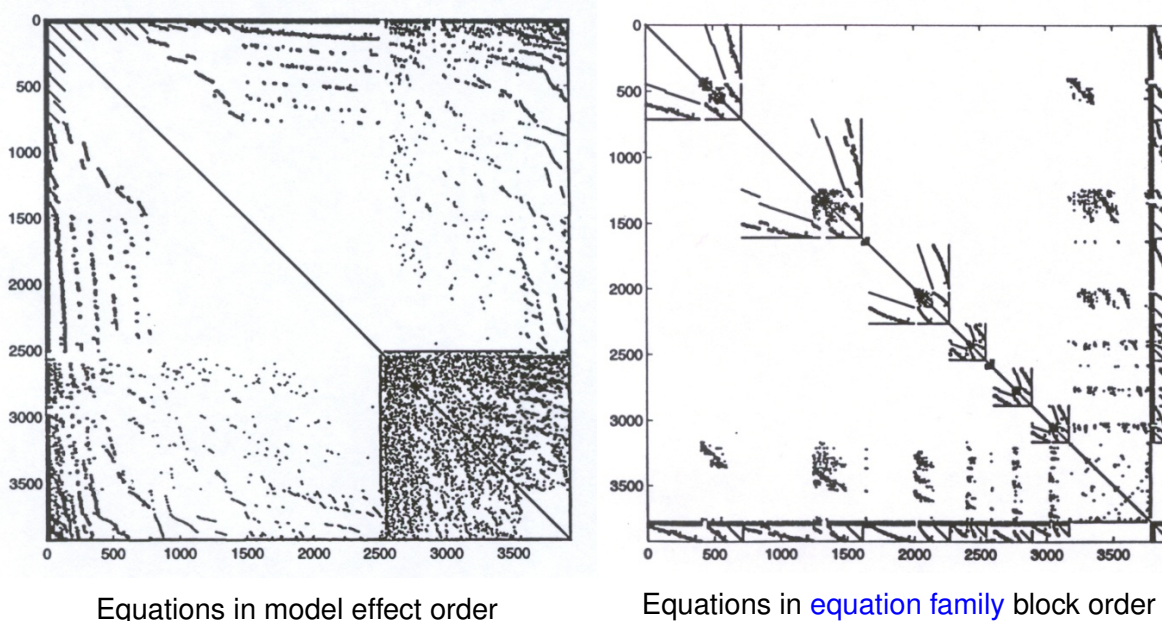


Figure 1: Non-zeros in the coefficient matrix of the MME when equations are sorted either by model effects or by [equation family](#) blocks. The model described TD milk yield data of seven herds including an age effect, a function for stage of lactation effect, a HTD effect and random regressions for 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 multiple-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. As a consequence, each record (data line) in the data file would contain a large number of 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 recommend 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.

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`, ...

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
```

Output of `block_information` program shows the amount of communication needed between the equation blocks:

```
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
```


File `BMatrix.dat` is created by the `block_information` program containing block-to-block communication matrix in sparse “i,j,v” format. Communication matrix can be visualized using Octave script `spy_communication.m` found in the `tools` directory of the MiX99 binary distribution:

```
octave
> spy_communication
```

This will plot an image (see Figure 2) of the block-to-block communication matrix using seven basic colors, each color illustrating $\frac{1}{7}$ th of the total communication. Yellow colored pixels contain the smallest communication counts and blacks are the largest communications.

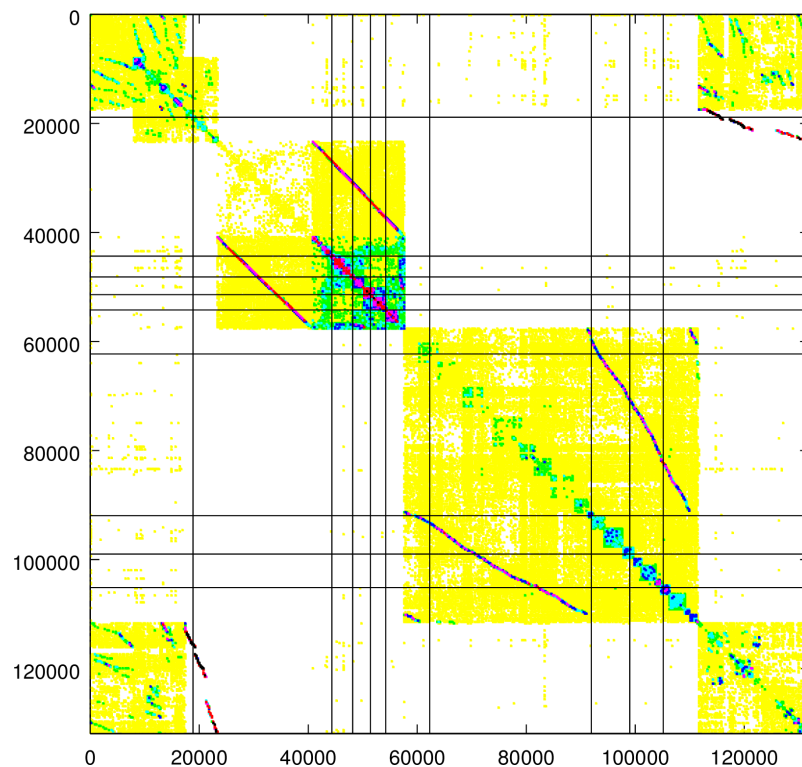


Figure 2: Block-to-block communication matrix of Nordic Holstein test-day model containing 131925 equation blocks. Parallel distribution for 10 processes is indicated by vertical and horizontal lines.

By default, the communication through the `common blocks` is omitted from the image. Number of `common blocks`, specified by the user in the MiX99 directive file, is stored in file `NcommonB.dat` by `block_information` program.

Block-to-block communication image shows also the partitioning of the blocks for the parallel processes. Partitioning is stored in file `PlastB.dat`.

Equation blocks are numbered sequentially in the image. 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 `BM_VStruct.dat` the communication due to the variance structures.

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 `gpmets` program. See utility program `partition_blocks` in the `tools/` subdirectory of the MiX99 binary distribution.

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 fourth 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 taken into account. 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.

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.

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 multiple-trait random regression models or multiple-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 multiple 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 multiple-trait model:

In the following, numbering of the variance components is illustrated by a hypothetical example for a multiple-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.

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 one of the residual (co)variance matrices has to be given in the (co)variance matrix file. However, its values are ignored by the solver program but used by the reliability calculation program (see [Approximate reliabilities using ApaX](#) in *Technical Reference Guide for MiX99 Solver*).

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.

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.

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 lines 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 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 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 for 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 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](#) how to specify models with the trait groups.

DATASORT Two or three entries: The first indicates the [integer data](#) column of the [block code](#), 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 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 multiple-trait models additional information on how to count number of factors is explained in the [MODEL](#) in-

struction. Regression effects which are 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 should be 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 mostly 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 as 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). Number of the factors for the fixed and random effects must be identical with the numbers specified on the **FIXRAN** line.

For **multiple-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 for one trait only. However, there can be only one and the same factor in a "**MODEL** column". In case of multiple-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" and in each column an entry has to be given for all traits. 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 exam-

ple a lactation curve is nested within season classes and is modeled as $season + d(season) + d^2(season) + e^{-0.05d(season)}$, 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 to block the data. 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 “operational” random effect. Specifying the option [ls+b](#) in [PEDIGREE](#) will instruct MiX99 to solve the effect as 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 a [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 across-block effect (see [WITHINBLOCKORDER](#)). If many such effects occur, the effects with 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 multiple-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. They 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 approximation of reliabilities, the [WITHINBLOCKORDER](#) number 1 has to be specified for the genetic animal effect. Otherwise, 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 order within blocks. In many cases, when the problem is small, it is not critical whether

an effect is placed **within blocks** or **across blocks**. 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 quite obvious for some models. For example, a model for milk yield in dairy cattle has the following fixed effects: herd×test-day, age, function on days in milk; and the following random effects: non-genetic animal environment effect, genetic animal effect, and residual. The [block sorting variable](#) should be 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 age effect and 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 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 instruct to 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 have to 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 multiple-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 following: " - 4 - 3 3 3 < 2 2 2 1 ". Regardless 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 has to be provided in the parameter file. The (co)variance matrix has to 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 for 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 code on the **RELATIONSHIPS** instruction line. Further, combining factors within the additive genetic animal effect must be consistent with the **RANDOM** instruction line. A factor which is 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 into the following instructions:

```
#MODEL:  stable sex      litter      animal      mate1      mate2      mate3
#          c1      c2      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 variable number of social effects: 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 the 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 starts 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 corresponds to the order of factors (or factor columns in case of [multiple-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. Pedigree information is used to account inverse of the **numerator relationship matrix** A^{-1} in the mixed model equations assuming zero inbreeding coefficients. Inbreeding coefficients can be included from a given file. Three extra 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 number of id code and inbreeding coefficient value in the inbreeding coefficient file are given. Note that column number of id code must be less than the inbreeding coefficient column. On the following line, name of the inbreeding coefficient 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** (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 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.

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 $G^{-1} - A_{gg}^{-1}$ where G is a genomic relationship matrix and 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 $G^{-1} - A_{gg}^{-1}$ 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". After the lines "fg" and name of the external file, the pedigree identification information is given as explained above.

Example 1 for MiX99 instructions for single step:

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

An alternative is to give the inverse relationship matrices G^{-1} and 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 G^{-1} is given, and then file having A_{gg}^{-1} .

Example 2 for MiX99 instructions for single step:

```
# 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 A_{gg}^{-1} matrix ($G^{-1} - A_{gg}^{-1}$) 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 variant of the "ssi" option is to instruct the solver to do the required calculations of 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.

Example 3 for MiX99 instructions for single step:

```
# 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
```

Single-step has been implemented in the parallel solver ***mix99p***. However, all genotyped animals must be in the **common blocks**, see **COMMONBLOCKS** when co-ordinate format is used for the external matrix. When a genotyped animal is not in the **common blocks** area, the parallel solver is likely to stop to ListFinds error.

All cases above assume that the external matrix file is in co-ordinate format. Thus, 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. There is an alternative matrix format that is often faster when the matrix is very dense. This is called lower triangle dense format. It is indicated by letter "L". Program **hginv** allows making lower triangle dense format matrices by option "-lower". The use of dense matrix format has several advantages:

- 1) the external file takes less disk space,
- 2) computations are faster,
- 3) when parallel computing is used, the genotyped animals need not be in the common block area.

Example 4 for MiX99 instructions for single step:

```
# inv(G) lower triangle dense format, inv(Agg) calculated by solver
ssIL      # inverse inv(G) in file: lower triangle dense
iGL.dat
PEDIGREE
# Standard pedigree based relationship information
1      1
```

Single-step computations using the ssGTBLUP "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". By giving "TeL", the external T matrix 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 matrix is easiest to make using a dedicated program.

Example 5 for MiX99 instructions for single step by ssGTBLUP:

```
# T matrix in dense rectangular format, inv(Agg) calculated by solver
TeL      # T matrix dense rectangular
TeMatrix.dat
PEDIGREE
# Standard pedigree based relationship information
1      1
```

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.
- c1** 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**

DEV

column **99** is reserved for **c1** (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 **t12**. 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 multiple-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. **t1** for mature weight, **t2** for relative initial weight and **t3** for maturation rate). An additional column for time is needed at the end of each **REGRESS** line. This is not counted 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 multiple trait models it might require to reorder the presentation of 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 multiple-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** If 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, the first column in the **covariable table file** contains the covariable index and is not counted. 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 entry. This is the integer column of the data file, which contains the covariable index. For the non-linear **Gompertz function**, the index is the time of measurement.

PEDIGREE A character code specifying the method used in computing the inverse of the numerator relationship matrix. Inbreeding is not considered when building the 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	1 st 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 One line with 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 value 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 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, if 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 chapter [Calculation of daughter yield deviations](#) in *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 provide as defined in [PEDIGREE](#).

CORRFILE Optional. One line for each random effect of those random effects for which a correlation structure between effect levels is provide by an external file. (see [FIXRAN](#)). Each line contains two 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 name of the file with the correlation structure. The correlation structure between effect levels must be given in form of an inverse correlation matrix. Only the diagonal elements and the lower (or the upper) triangle must be provided. The file consists of three columns. The first two columns are in integer format and contain the ids of the levels. The third column is in real

format and contains the non-zero elements. The ids of the effect levels must be consistent with those given in the data file. Further, only non-zero elements must be included. For the multiple-trait models, **MiX99** will automatically set up the Kronecker product between the provided [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 it is the case in analysis of genomic data (see [FIXRAN](#)). This can be used to make **SNP-BLUP** by considering the regression effects random. Regression design matrix has for each row in the data file regression coefficients. There can be multiple regression design matrices with different properties. Typically, however, one matrix is sufficient. For each [regression design matrix](#), there are two or three lines depending on the nature of the effects. The first and second lines have the following information:

<effect type> <name> <id column> <first regression column> <number of effects> [<centering> <imputing> <impute value>]

<file name of the regression design matrix>

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

<effect type> 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 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 regression column> is the first 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 regression effects/columns> is the number of used (regression) effects in the design matrix.

<centering> is instruction on centering the regression matrix. When letter 'n' is given, then no centering is done. This is the default when no centering nor impute information is given.

<imputing> is instruction for imputing column by the most common genotype. When letter 'n' is given, no imputation is done. Missing value to be imputed is defined by <impute value>. For example, giving 'y 5 ' would impute all values of 5 with most common other value on column. Default is no imputation.

<impute value> is value considered missing in imputation and will be replaced by the most common value in each column.

<file name of the regression design matrix> The file must have an appropriate number of values on each line given above. Lines can be broken down to several lines provided the last line has regression coefficients needed by this model. For example, let the first column be 1, and there are 3 effects. Then, line '1 2 3 4' can be broken to three lines '1' '2' '3 4' but not to two lines '1 2 3' '4'.

If the effects associated with the design matrix are random (given 'R'/'r' or 'H'/'h' for <effect type>), an additional line is given:

<variance component file>

which includes the variance components for the effects. 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 a 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 case of a [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 [Using old solutions](#) 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 to read 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 reliabilities are needed only.

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 may 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. The size of the matrices depends 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 multiple-trait model with three traits where the additive genetic animal effect is modeled by a third order polynomial (four factors per trait) will instruct MiX99 to apply for each animal a preconditioner matrices of size 12×12 .

Diagonal: Only the diagonal elements of the coefficient matrix of the MME 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.
- m** **Mixed blocks**. Block diagonal preconditioner is applied for the first effect in XpX and the remaining part of XpX is applied as full block preconditioner.
- b** **Block diagonal**. Applies block diagonal matrices for preconditioning.
- d** **Diagonal**. Only diagonal elements of the coefficient matrix are used for preconditioning.
- m plus integers** This **alternative** gives the possibility to define the most suitable preconditioner matrix for the XpX equations for a given model. The integers are optional. Numbers have to be separated with a space.

Defining the most suitable preconditioner for the XpX part is model dependent and will require test runs with different preconditioner options. This is of interest when developing models for large routine evaluations where solving time is important. However, experiences have shown that defining a block diagonal preconditioner for XpX part is a good choice for many models. In the following a more detailed description of the preconditioning options is given.

Full block (f): The whole XpX block in the mixed model equations is considered as a preconditioner matrix. In some cases this might yield a large block, for which inversion requires too much memory and computing time. If so, one of the other types might be more suitable.

Mixed blocks (m): The XpX block is separated into two parts. The first part includes all equations related to the regression effects applied across all observations (given they have been modeled; see [REGRESS](#)), as well as all equations that belong to the first specified across block fixed effect on the [MODEL](#) line(s). All factors of the first effect are included. For instance, if the first effect is modeled by a regression function (i.e. for all factors of the regression function the same [integer data](#) column is specified on the [MODEL](#) line(s)), then all equations, which belong to the same effect level will be included and the size of the block precondition matrices will equal to the number of equations per level (see Example [7.5](#)). The second part has all remaining across block fixed effects. The first part of the XpX in the preconditioner matrix is a block diagonal matrix with each level treated as a block, and the second part of the XpX is one large block. If specifying of *Full block* preconditioner will yield a too large matrix, the use of *Mixed blocks* preconditioner can reduce computations considerably given the across block effect with the most equations is specified first on the [MODEL](#) line(s).

Block diagonal: [Same](#) as for the WpW. All equations that belong to the same level form one diagonal block.

Diagonal: Only the [diagonal](#) elements of the XpX of the coefficient matrix of the MME are used for preconditioning. This type of preconditioner is likely to give poor convergence and should be used only if the three other types cannot be used. For the non-linear [Gompertz function](#), the diagonal ([d](#)) preconditioner must be defined.

Mixed block with individual blocking of fixed effects: This option gives the possibility to define the most suitable preconditioner matrix for the XpX equations for a given model. The option is of interest for models with large number of effects and equations in the XpX part. The integer numbers specified after the [m](#)-option define which effects form one single block. If a single block includes more than one effect, the off-diagonal coefficients between the effects are included in the preconditioner matrix as well. This may improve the convergence. In Example [7.5](#), four different fixed effects are specified as across block effects: a regression function nested within season (7), age (5), days carried calf (6), and a year×months interaction (4). This allows to set up different preconditioner matrices. One possibility would be to combine effects 7 and 5 to one block and effects 5 and 6 to a second block, giving the following definition: [m 1 1 2 2](#). Two other alternatives would be to combine the first three effects ([m 1 1 1 2](#)) or the last three effects ([m 1 2 2 2](#)). The latter definition is equal to the default value. Effects which consist of several factors, like the regression function in Example [7.5](#), are considered as one effect and require only one integer value to be specified. For some models, it might be beneficial to reorder

the effects on the **MODEL** line(s) since only consecutive effects can be combined. Numbering of the effects must start from one (1).

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**: work load division method, and size of I/O buffers. The **work load division** is used in parallel computing to divide amount of work to processors evenly. MiX99 has two work load division methods: number of records, and number of equations. Default is work load by number of records, which leads to dividing the data file approximately evenly 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, work load 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).

Output files of the MiX99 pre-processor

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.

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***.

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 round.

- Tmp1.code0** Original and recoded id's of all effect levels. For the animal effect it contains also number of descendants and 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

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

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.

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.

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***.

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 identical 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`.

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-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:
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

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
    
```

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, multiple 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
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

-      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 precondition. 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%	
.	.	.	.	

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

. . .
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
. . .      residual
. . .
3 6 7      residual covariance between MT and FF
3 7 7      residual variance for FF

```

CLIM command file:

```

TITLE    Pig evaluation, multiple 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

```

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

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.
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```
# REGRESS: number  beta1  beta2  herd-test-day  gamma0  gamma1  gamma2
              6      1      2              c1      c1      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
```


TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

MODEL

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

Multiple-trait model with trait groups

Traits:	TD-milk 1 st lac.(M1), TD-protein 1 st lac.(P1), TD-milk 2 nd lac.(M2), TD-protein 2 nd lac.(P2)
Model:	Within each lactation, each animal has repeated observations, i.e. the model has features of multiple-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
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
<u>34</u>	<u>10</u>	<u>7</u>	0	<u>5</u>	0	<u>.967</u>	<u>.042</u>	-16.	-16.	<u>12.1</u>	<u>3.40</u>	-16.	-16.
<u>34</u>	<u>10</u>	<u>7</u>	0	<u>6</u>	0	<u>.562</u>	<u>.084</u>	-16.	-16.	<u>8.7</u>	<u>3.52</u>	-16.	-16.
<u>34</u>	<u>10</u>	0	<u>17</u>	0	<u>10</u>	-16.	-16.	<u>.661</u>	<u>.035</u>	-16.	-16.	<u>28.2</u>	<u>3.37</u>
<u>34</u>	<u>10</u>	0	<u>17</u>	0	<u>10</u>	-16.	-16.	<u>.430</u>	<u>.087</u>	-16.	-16.	<u>32.7</u>	-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
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```
# 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	

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

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
.  .  .          .
2 12 12          variance for P2,G2
1  1  1          residual variance for M1
1  1  2  residual covariance between M1 and P1
1  1  3  residual covariance between M1 and M2
1  1  4  residual covariance between M1 and P2
1  2  2          residual variance for P1    residual
1  2  3  residual covariance between P1 and M2
1  2  4  residual covariance between P1 and P2
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)

```

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
# REGRESS: number:
           16      herd b0 b1 b2 b3 b4 age dcc ym htm z0 z1 z2 g0 g1 g2
           c1  c1 t2 t3 t4 t5 c1 c1 c1 c1 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)

```

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 1 st lac. (M1), 2 nd lac. (M2), and 3 rd 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
⋮	⋮	⋮	⋮

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

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 multiple-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  6  7  5  4  2  2  2  2  2  2  2  2  2  2
#   2   1   -   1  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  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
#   2   2   2   2   2   2   2   2   2   2   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
# CVRFIL:
#       covarmilk.tab
# CVRNUM:
#       34
# CVRIND:
#       9
# PEDIGREE:
#       am+p
# DATAFILE:
#       /koel/testi/data/multidat.s
# VAR:
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

10 1 f
# MISSVA:
0.0
# SCALE:
y
# PEDFILE:
/koel/testi/data/new.ped.s
# PARFILE:
llacmpf.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	

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

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 1 2      residual covariance between M1 and M2
4 1 3      residual covariance between M1 and M3      residual
4 2 2      residual variance for M2
4 2 3      residual covariance between M2 and M3
4 3 3      residual variance for M3

```

CLIM command file:

```

TITLE      Milk 1., 2., and 3. Lactation, MT-RTDM

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    llacmpf.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

```

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 paternal grandsire, cow's sire's maternal grandsire
direct (D):	calf's paternal grandsire, calf's sire'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:
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

      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:
      2      cow's sire, calf's sire
      1      2
# REGRESS:      number:      cmth sex noc cow hy, cow's sire, calf's sire
      7      cl 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 cbd-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

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
    
```

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.)
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

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 with 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
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
2	2	2	10.0
2	3	1	0.0

non-gen env.

sire

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

2	3	2	0.0	
2	3	3	1.0	
3	1	1	1.0	residual

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
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```

# 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	

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
3	1	1	0.2850
4	1	1	0.7000

QTL effect
polygenic effect
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
      -
```


TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```
# 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.
# MODEL: 1 3 1 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 c1 c1 c1 c1
# COMBINE:
# n
# PEDIGREE:
# am
# DATAFILE:
# MAS.dat
# VAR:
# 6 1 f
# MISSVA:
# 0.0
# SCALE:
# n
# PEDFILE:
# MAS.ped
# CORRFIL: 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.
# The matrix is associated with the random effect 1 (=haplotype1).
# 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
```

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 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
```

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

```
# 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

Acknowledgement

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

References

- Duff, Iain S., Erisman, Albert M., and Reid, John K. (1992). *Direct methods for sparse matrices*. Oxford, UK: Clarendon Press (cit. on p. 5).
- 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. 18).
- 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. 18).
- Janss, L. L. G. and Foulley, J. L. (1993). "Bivariate analysis for one continuous and one threshold dichotomous trait with unequal design matrices and an application to birth weight and calving difficulty". In: *Livest. Prod. Sci.* 33.3–4, pp. 183–198. DOI: [10.1016/0301-6226\(93\)90001-X](https://doi.org/10.1016/0301-6226(93)90001-X) (cit. on p. 18).
- 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. 5).
- Lidauer, M., Strandén, I., Mäntysaari, E. A., Pösö, J., and Kettunen, A. (1999). "Solving large test-day models by iteration on data and preconditioned conjugate gradient". In: *J. Dairy Sci.* 82.12, pp. 2788–2796. DOI: [10.3168/jds.S0022-0302\(99\)75536-0](https://doi.org/10.3168/jds.S0022-0302(99)75536-0) (cit. on p. 1).
- Lidauer, M., Matilainen, K., Mäntysaari, E. A., Pitkänen, T., Taskinen, M., and Strandén, I. (2017). *Technical Reference Guide for MiX99 Solver*. Release XI/2017. Natural Resources Institute Finland (Luke) (cit. on pp. 3, 6, 10, 13, 17, 24, 29, 32).
- 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: [doi:10.2527/jas2017.1912](https://doi.org/10.2527/jas2017.1912) (cit. on pp. v, 25).
- MiX99 Development Team (2017). *MiX99: A software package for solving large mixed model equations*. Release XI/2017. Natural Resources Institute Finland (Luke). Jokioinen, Finland. URL: <http://www.luke.fi/mix99> (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. 12, 45).
- Strandén, I. (2017). *Command Language Interface for MiX99*. Release XI/2017. Natural Resources Institute Finland (Luke) (cit. on pp. 3, 14).
- Strandén, I. and Lidauer, M. (1999). "Solving large mixed linear models using preconditioned conjugate gradient iteration". In: *J. Dairy Sci.* 82.12, pp. 2779–2787. DOI: [10.3168/jds.S0022-0302\(99\)75535-9](https://doi.org/10.3168/jds.S0022-0302(99)75535-9) (cit. on 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](https://doi.org/10.3168/jds.S0022-0302(01)74477-3) (cit. on p. 1).
- Vuori, K., Strandén, I., Sevón-Aimonen, M.-L., and Mäntysaari, E. A. (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](https://doi.org/10.1186/1297-9686-38-4-343) (cit. on pp. 2, 17).

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**.

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

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 5123.50000000000000
6 102 3 7597.00000000000000
7 103 4 6410.30000000000000
8 103 3 -888.00000000000000
```


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.

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.

DEV

- 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.

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
<code>init_datafilter(nfi,nfr)</code>	opening data file
<code>rewind_datafilter()</code>	rewinding data file
<code>close_datafilter()</code>	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 column 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
```

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
```

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 `DLL_EXPORT` declaration.

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
*****
```

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: [79](#)
- also referred: [79](#)
-  see example on page

across block, [19](#), [32](#)
across data regression, [25](#)
additional output information, [15](#)
am
 [46](#), [50](#)
am, [28](#)
 [46](#), [49](#), [67](#), [69](#)
am+p
 [41](#), [44](#), [53](#), [57](#)
am+p, [28](#)
 [40](#), [43](#), [52](#), [55](#)
ar, [28](#), [28](#)
autoregressive process, [28](#), [28](#)

BINARY
 [50](#)
binary file, [5](#), [28](#), [32](#), [73–75](#)
binary format, [5](#), [5](#), [28](#), [36](#), [38](#), [73](#)
how to convert, [73](#)
BLOCK
 [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)
block code, [v](#), [6](#), [10](#), [15](#), [35](#)
block sorting variable, [5](#), [6](#), [18](#), [19](#)
block_information, [v](#), [8](#)
BM_data.dat, [10](#)
BM_data.graph, [10](#)
BM_VStruct.dat, [10](#)
BM_VStruct.graph, [10](#)
BMatrix.dat, [9](#), [10](#)
BMatrix.graph, [10](#)

categorical trait, see threshold model,
[1](#), [2](#), [18](#), [18](#), [35](#), [64](#)
cl, [25](#), [26](#)
 [40](#), [43](#), [46](#), [49](#), [52](#), [55](#), [59](#), [65](#),
[67](#), [69](#)
class variable, [14](#)
CLIM command file, [3](#), [4](#), [5](#), [10](#), [14](#), [39](#)

COMBINE, [19](#), [21](#), [26](#), [54](#)
Combining factors
model factors within traits, [19](#)
random factors other than additive
genetic animal effects, [19](#)
within additive genetic animal
effect, [20](#)
command line options, [3](#)
common blocks, [5](#), [9](#), [11](#), [24](#), [35](#)
COMMONBLOCKS, [11](#), [24](#), [35](#)
convert99, [73](#), [73](#)
correlation structure between effect
levels, [29](#)
CORRFILE, [16](#), [20](#), [29](#)
covariable, [14](#)
covariable table file, [3](#), [5](#), [13](#), [26](#), [27](#), [51](#)
(co)variance components file, [3](#), [11](#)
CVRFIL, [27](#)
CVRIND, [27](#)
CVRNUM, [27](#)

data file, [3](#), [5](#), [73](#), [75](#), [76](#), [78](#)
DATAFILE, [28](#)
DATAFILE
 [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)
datafilter, [v](#), [75](#)
plugin, [75](#)
DATASORT, [7](#), [13](#), [15](#), [31](#)
DATASORT
 [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)
daughter yield deviation, [10](#), [29](#), [29](#)
design matrix, [30](#)
design matrix file, [16](#), [31](#)
Development features (DEV), [ii](#), [v](#), [25](#),
[75](#)
DYD, daughter yield deviation, [10](#), [29](#),
[29](#), [33](#)

- EM, [18](#)
- equation family, [5](#), [6](#), [8](#), [10](#), [11](#), [15](#), [18](#), [35](#)
- examples, [39–70](#)
- Expectation-Maximization algorithm, [18](#)
- external inverse relationship matrix, [22](#)
- first common block, [v](#), [35](#)
- fixed effect, [14](#)
- FIXRAN, [15](#), [16](#), [17](#), [21](#), [29](#), [30](#)
- free format, [5](#), [5](#), [11](#), [14](#), [28](#), [73](#)
- GBLUP, [22](#)
- GLS models, [2](#), [17](#), [29](#)
- Gompertz function, [1](#), [2](#), [13](#), [17](#), [17](#), [18](#), [26](#), [27](#), [33](#), [34](#)
- Gompertz function model, [1](#), [2](#), [13](#), [15](#), [17](#), [32](#)
- heterogeneous variance, [2](#), [17](#), [28](#)
- Hetlog, [36](#)
- hginv, [24](#)
- ignoring relationship between animals, [22](#)
- inbreeding coefficients, [22](#)
- instruction file, [3–5](#), [8](#), [10](#), [11](#), [14](#), [14](#), [39](#)
- INTEGER, [14](#), [15](#)
- INTEGER
 - ☞ [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)
- integer data, [5](#), [15–17](#), [34](#), [73](#), [75](#)
- inverse correlation matrix, [2](#), [16](#), [20](#), [29](#), [30](#)
- LAMPATH, [29](#)
- ListFinds error, [24](#)
- log files, [36](#)
- ls, [28](#)
- LS models, [2](#), [17](#), [17](#), [28](#), [29](#), [31](#)
- LS models with data blocking, [17](#), [21](#), [28](#), [29](#)
- ls+b, [17](#), [28](#), [28](#)
- Memlog, [36](#)
- MERGE, [19](#), [21](#), [26](#)
- MISSING
 - ☞ [41](#), [50](#)
- missing integer values, [5](#), [48](#)
- missing real values, [5](#), [29](#), [48](#)
- MISSVA, [29](#)
- MiX99.lst, [36](#)
- MiX99_DIR.DIR, [14](#), [39](#)
- mix99i, [3](#), [3](#), [9](#), [14](#), [29](#), [31](#), [32](#), [36](#), [38](#), [39](#), [75](#), [78](#)
 - mix99i_datafilter, [78](#)
- mix99s, [3](#)
- MODEL, [15](#), [16](#), [16–19](#), [21](#), [25](#), [27](#), [34](#), [35](#), [68](#)
- MODEL
 - ☞ [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)
- Modlog, [36](#)
- multiple input data files, [8](#)
- multiple residual variances, [2](#), [13](#), [15](#)
- multiple-trait models, [16](#), [21](#)
- NcommonB.dat, [9](#)
- Newton-Raphson algorithm, [18](#)
- NORANSOL
 - ☞ [44](#), [50](#), [53](#)
- NR, [18](#)
- number of processors, [35](#), [35](#)
- numerator relationship matrix, [22](#), [27](#), [28](#)
- Observations with variable number of social effects, [21](#)
- OK_mix99i, [3](#)
- old solutions, [32](#), [38](#)
- original_blockcode.dat, [9](#)
- output files, [36](#)
- PARALLEL, [35](#)
- PARALLEL
 - ☞ [50](#), [57](#)
- parallel computing, [5](#), [6](#), [8](#), [11](#), [15](#), [19](#), [35](#), [35](#)
 - work load division, [35](#)
- PARFILE, [11](#), [17](#), [31](#)
- PARFILE
 - ☞ [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)
- Parlog, [36](#)
- PEDFILE, [17](#), [22](#), [23](#), [29](#), [29](#)
- PEDFILE
 - ☞ [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)
- PEDIGREE, [10](#), [17](#), [20](#), [24](#), [27](#), [28](#), [29](#)
- PEDIGREE
 - ☞ [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)

pedigree file, [3](#), [5](#), [10](#), [15](#), [18](#), [22](#), [28](#),
[29](#), [35](#), [68](#)

PEDIGREECODE

☞ [41](#), [44](#), [46](#), [50](#), [53](#), [57](#)

Phantom parent groups, [10](#), [10](#), [28](#)

PlastB.dat, [9](#)

PRECON, [32](#), [32](#)

PRECON

☞ [41](#), [46](#), [50](#), [53](#), [57](#), [60](#)

preconditioning, [32–35](#)

block diagonal, [33](#), [33](#), [34](#)

diagonal, [33](#), [33](#), [34](#)

full block, [33](#), [34](#)

mixed blocks, [33](#), [34](#)

with individual blocking of fixed
effects, [33](#), [34](#)

QTL effect, [1](#), [2](#), [2](#), [19](#), [66](#)

RANDOM, [11](#), [12](#), [17](#), [20](#), [21](#), [21](#), [27](#),
[29](#)

RANDOM

☞ [44](#), [50](#), [53](#), [57](#), [60](#)

random effect, [14](#)

random phantom parent group effects,
[10](#)

RANSOLFILE, [32](#)

REAL, [15](#)

REAL

☞ [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)

real data, [5](#), [16](#), [25](#), [73](#), [75](#)

reduced rank models, [21](#)

REGFILE, [16](#), [30](#)

REGRESS, [16](#), [17](#), [25](#), [25–27](#), [34](#)

regression design matrix, [16](#), [30](#), [30](#)

regression effects, [16](#)

relationship code, [7](#), [15](#)

RELATIONSHIPS, [17](#), [20](#), [21](#), [21–23](#)

RelaX2, [11](#)

RESFILE, [13](#), [15](#), [31](#)

residual (co)variance matrix, [31](#)

restart, [32](#)

RHO, [28](#)

Rules for combining factors within
trait(s), [19](#)

SCALE, [29](#)

SCALE

☞ [50](#), [53](#), [57](#), [60](#)

scaling observations, [29](#)

simulated observations, [29](#), [29](#)

single-step, [23](#)

Sire model with phantom parent
groups, [10](#)

sm

☞ [60](#)

sm, [28](#)

☞ [59](#), [62](#), [65](#)

sm+p, [28](#)

SNP-BLUP, [30](#)

social effect, [2](#), [2](#), [20](#), [21](#), [68](#)

Solold, [38](#), [38](#)

SOLUNF, [32](#)

Solunf, [32](#), [38](#)

Solvec, [32](#), [32](#), [38](#)

spy_communication.m, [9](#)

ssGTBLUP, v, [25](#), [25](#)

syntax change
GBLUP, [23](#)

TABLEFILE

☞ [53](#), [57](#)

TABLEINDEX

☞ [53](#), [57](#)

temporary work files, [31](#), [36](#)

text file, [5](#), [28](#), [73](#), [74](#)

THR_MHD, [18](#)

THR_VAL, [18](#)

threshold model, [1](#), [2](#), [18](#), [18](#), [64](#)

TITLE, [14](#)

TITLE

☞ [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)

Tm10.trco0, [37](#)

Tmp.para, [37](#)

Tmp1.code0, [36](#)

Tmp4.pedi0, [36](#)

Tmp5.clas0, [36](#)

Tmp6.diab0, [36](#)

Tmp9.strc0, [36](#)

TMPDIR, [31](#)

TMPDIR

☞ [50](#), [57](#)

total I/O buffer size, [35](#)

trait, [14](#)

trait group code, [7](#), [15](#), [16](#)

TRAITGROUP

☞ [50](#), [57](#)

TECHNICAL REFERENCE GUIDE FOR MiX99 PRE-PROCESSOR

TRAITGRP, [8](#), [15](#)

TRAITS, [15](#)

VAR, [28](#), [29](#)

weight, [2](#), [5](#), [16](#)

within block, [19](#), [32](#)

WITHINBLOCKORDER, [17](#), [18](#),
[18–20](#), [33](#)

WITHINBLOCKORDER

 [41](#), [44](#), [46](#), [50](#), [53](#), [57](#), [60](#)

work load division, [35](#), [35](#)

zr, [26](#)