MiX99

Solving Large Mixed Model Equations

Release IX/2025

Command Language Interface for MiX99 (CLIM)

Last update: Sept 2025

Luke
NATURAL RESOURCES
INSTITUTE FINLAND

Preface

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

About this manual

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

Disclaimer

The MiX99 software is owned by Natural Resources Institute Finland (Luke). When using this program you agree with the following terms. You are not allowed to distribute, copy, give or transfer MiX99, under the same or any other name. Any decisions based on the information provided by MiX99 are made at your own responsibility and risk. Only limited technical support can be provided, but vital questions on its use can be directed to the authors (mix99@luke.fi). Please report any bugs to the authors. MiX99 can be cited by (MiX99 Development Team, 2025) and (Pitkänen et al., 2022). If you would like to use MiX99, please contact Natural Resources Institute Finland[1].

MiX99 new (NEW) and development (DEV) features

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

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

Authors

Ismo Strandén, Hongding Gao, Martin H. Lidauer, Kaarina Matilainen, Timo Pitkänen, Anna-Kaisa Ylitalo, Napoleon Vargas Jurado, Matti Taskinen

Natural Resources Institute Finland (Luke),
FI-31600 Jokioinen, Finland
http://www.luke.fi/mix99

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

Contents

1 Introduction

MiX99 is a software suite designed for the estimation of breeding values. It includes three types of programs for large models:

- preprocessor: `mix99i`
- solver: `mix99s`, `mix99p`
- approximate reliability calculators: `apax99`, `apax99p`.

An additional program (`exa99`) is available to calculate exact standard errors (SE), prediction error variances (PEV), and accuracies for small models. Additional programs to assist parallel computing using MPI (in `mix99p`, `apax99p`) include `imake99` and `imake4apax`. The main purpose of this manual is to describe the command language interface for MiX99, shortly CLIM, which is used to give instructions in a file to the `mix99i` preprocessor. Some use of `mix99s` and `exa99` is also described in chapters on variance component estimation and reliability computations.

The preprocessor program `mix99i` has two ways to give preprocessor instructions:

- the command language interface for MiX99, called CLIM, and
- the original interface, called directive file.

The directive file answers questions on data and statistical model used. CLIM is the modern interface for the `mix99i` preprocessor and provides many advantages:

- Commands can be given in any order. This removes the requirement for giving the commands in a strict sequence.
- Some commands have default values. Therefore, not all commands need to be given.
- Commands have English language names. This makes the command instruction file somewhat easier to read than a directive file.
- All information about the statistical model is given in the same model area, not divided in several sections as in the directive file.
- Effects in the model are ordered in a way that first fixed effects are specified, followed by the random effects. Within fixed effect, fixed regression effects without nesting are given first.

1.1 Supported statistical models and beta testing features

The MiX99 software supports a wide range of statistical models, and a desired model can be constructed by combining several basic model types, e.g., multi-trait random regression model with weighted observations. The following models are available in CLIM:

- least squares models
- multi-trait linear mixed effect model
- random regression model (with or without covariable table file)
- reduced rank (combining of traits)

- genome information models like SNP-BLUP and G-BLUP
- multiple residual variances
- models with weighted observations

There are some specific models, however, that have **not been implemented** in CLIM:

- random regression with multiple nesting in additive genetics e.g., both maternal and additive genetic effects
- MAS-BLUP (combining of effects)
- non-linear models: threshold and Gompertz models

These models are only available through the directive file interface.

In beta testing:

- order of effects on the model line is free unlike in a directive file.

1.2 Organization of the manual

This manual is organized as follows:

- This chapter gives some introductory remarks on how to use CLIM.
- The second chapter gives some theoretical background, and how it relates to the way models are presented in this manual. In addition, some remarks on computational implementation issues are given which are necessary to understand some of the commands.
- Input files are briefly described in the third chapter.
- The fourth chapter contains important information on how to use the solver and a short description of the solution files.
- The fifth chapter describes in general terms how model is given in CLIM. The section on basic models is a must read as it explains basic concepts of the CLIM interface.
- The sixth and seventh chapters have single trait and multi-trait models, respectively.
- The eigth chapter gives tools to express long and complex models simpler by macros.
- The nineth chapter has genomic data models.
- The tenth chapter describes single-step models.
- The eleventh chapter has some special topics like blending of foreign information and deregression.
- The last chapter has the syntax of all commands.

The manual concentrates on how to give different statistical models to MiX99 using CLIM. However, some options are not covered in detail. Please study them in Chapter 12 which has a summary of all commands. Some of these commands are very important such as `MISSING` and `TMPDIR`.

Table 1.1: Command line options to CLIM, given on the mix99i command line.

Option	Effect
-h,--help	Show options.
-d	CLIM is executed, no preprocessing part in mix99i. File MiX99_DIR.DIR is produced.
-b	Allows use of beta version feature(s), see list above.
-l	Long listing option of mix99i.
--nproc NPROC	Number of MPI processes.
--datafile DATAFILE	Data file name.
--pedfile PEDFILE	Pedigree file name.
--parfile PARFILE	Variance file name.
--checkdata	Enhanced checking of the data file.
--usemacros	Use CLIM macros (DEFINE) and ranges (abcNN:MM).
--keepsol	Keep old solution files.
--keepindir	Do not overwrite MiX99_IN.DIR and .OPT files.
--PAR	PARDISO library used for single-step preconditioner (default).
--CHM	CHOLMOD library used for single-step preconditioner.
-v, --verbose	Show additional information.
-V, --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).

1.3 Invoking CLIM and command line options

The mix99i preprocessor reads CLIM commands from a ***command file*** aka CLIM file, e.g., file named my_model.clm. CLIM interprets the commands in the CLIM file into a directive file MiX99_DIR.DIR which is read by mix99i. Thus, the CLIM interface is an automatic two step process.

CLIM is used by mix99i when a command file is given to it as a command line parameter:

```
mix99i my_model.clm
```

Note that an original directive file is read from the standard input. Thus, executing

```
mix99i < mix99.dir
```

would expect an original directive file to be in mix99.dir.

Some new model features are not yet supported by CLIM, but a similar model may be feasible. By using the command line option '-d' (Table 1.1), i.e.,

```
mix99i -d my_model.clm,
```

a directive file is created, after which the program stops. This directive file can be used as a template for a directive file having the new model feature. When you modify a directive file, please make sure that your intended modifications are correct. For more mix99i command line options, see Table 1.1.

1.4 Simple example

A simple animal model example with one fixed effect (`mean`) and a random additive genetic effect (`individual`) illustrates CLIM:

```
DATAFILE   simple.dat       # Name of the data file
INTEGER    individual mean  # Integer column names in the data file
REAL       y                # Real number column name in the data file

PEDFILE    simple.ped       # Name of the pedigree file
PEDIGREE   individual am    # Additive genetics associated with "individual"
                            # am=animal model
PARFILE    simple.var       # Name of the variance components file
MODEL
  y = mean individual       # Model line
```

Some notes on the example:

- Everything after the '#' character on a line is ignored and is considered as a comment.
- All command information is on one line which can be continued using the '&' continuation symbol.
- Statistical model is given on a line after the `MODEL` command.
- Model effects from integer columns are class effects.
- Effects in the model are ordered: first fixed effect(s), followed by the random effect(s). Within fixed effects, fixed regression effects without nesting are given first.

Command names are not case-sensitive, although in this manual keywords are capitalized. For example, the command `INTEGER` can be written as `integer`. However, all other names are case sensitive, such as `mean` column in the example above. Therefore, the names of the columns in the model lines must be written exactly as they are in the `INTEGER` and `REAL` commands.

2 Theory and notation

Here we introduce some notation and theory for an animal model, which we refer also as pedigree BLUP. For a clear presentation of this and many other models in animal breeding with examples, see Mrode and Thompson (2006). Some examples in this manual are from this book.

2.1 Single trait model

A simple single trait pedigree BLUP (or animal model) has the form

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{b} + \boldsymbol{Z}\boldsymbol{a} + \boldsymbol{e}$$

where

- $\boldsymbol{y}$ is $n \times 1$ vector of observations,
- $\boldsymbol{b}$ is $p \times 1$ vector of fixed effects,
- $\boldsymbol{X}$ is $n \times p$ design matrix to link observations to appropriate fixed effects,

- $\boldsymbol{a}$ is $q \times 1$ vector of random additive genetic effects,
- $\boldsymbol{Z}$ is $n \times q$ design matrix to link observations to appropriate random effects,
- $\boldsymbol{e}$ is $n \times 1$ random residual vector.

Hence, there are q individuals with n observations, and there are p fixed effects.

In a simple mixed effects model, the matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$ are ***incidence matrices***. In other words, these matrices have zeros and ones to indicate which effect corresponds to which observation. However, if the model has regression effects, these matrices have covariates. Thus, we call these matrices ***design matrices*** to indicate wider model possibilities.

It is possible to have both regression and classification variables (categories). Classification variables will be called class effects. For example, herd effect is a typical class effect, an observation belongs only to one herd, and observations with the same herd effect are predicted by the same estimate. Regression effects are not classification effects. For example, a linear function has a coefficient in the design matrix. However, the regression effect can be nested within a classification. In practice, the difference between regression and classification effects is that a class effect number is in the integer number column, but a regression coefficient is in the real number column of the data file.

Common linear mixed effects model assumptions are

$$\begin{array}{rclrcl} \mathrm{E}(\boldsymbol{a}) & = & \boldsymbol{0} & \mathrm{Var}(\boldsymbol{a}) & = & \boldsymbol{K}\sigma_a^2 = \boldsymbol{V}_G \\ \mathrm{E}(\boldsymbol{e}) & = & \boldsymbol{0} & \mathrm{Var}(\boldsymbol{e}) & = & \boldsymbol{I}\sigma_e^2 = \boldsymbol{R} \\ \mathrm{E}(\boldsymbol{y}) & = & \boldsymbol{X}\boldsymbol{b} & \mathrm{Cov}(\boldsymbol{a},\boldsymbol{e}) & = & \boldsymbol{0} \end{array}$$

where $\boldsymbol{K}$ stands for a matrix that describes the relationships between individuals. For instance, the numerator relationship matrix $\boldsymbol{A}$ in a pedigree BLUP model or the genomic relationship matrix $\boldsymbol{G}$ in a GBLUP model.

For the convenience of presentation, it is common to denote the residual covariance matrix by $\boldsymbol{R}$. We denote the genetic covariance matrix by $\boldsymbol{V}_G$. With this notation, the ***mixed model equations*** or MME to solve are

$$\begin{bmatrix} \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{X} & \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{Z} \\ \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{X} & \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{Z} + \boldsymbol{V}_G^{-1} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{b}} \\ \widehat{\boldsymbol{a}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{y} \\ \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{y} \end{bmatrix}$$

where $'$ denotes matrix transpose.

The model can be described by giving its effects. For example, if the above model had a fixed herd effect ($herd$), and an individual genetic effect (a) for the additive genetic effects, then it can be written as

$$y = herd + a + e$$

where e is the residual term. This can be considered as a model for one individual record, although subscripts to indicate this were not used.

2.2 Multiple trait model

Multi-trait model expands the single trait model matrices:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{b} + \boldsymbol{Z}\boldsymbol{a} + \boldsymbol{e}$$

where the matrices and vectors have a trait-wise structure. Thus, for t traits, we can write

$$\boldsymbol{y}' = \begin{bmatrix} \boldsymbol{y}'_1 & \boldsymbol{y}'_2 & \cdots & \boldsymbol{y}'_t \end{bmatrix}$$
$$\boldsymbol{e}' = \begin{bmatrix} \boldsymbol{e}'_1 & \boldsymbol{e}'_2 & \cdots & \boldsymbol{e}'_t \end{bmatrix}$$
$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_T \end{bmatrix}, \quad \boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{X}_t \end{bmatrix}$$
$$\boldsymbol{a} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_t \end{bmatrix}, \quad \boldsymbol{Z} = \begin{bmatrix} \boldsymbol{Z}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Z}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{Z}_t \end{bmatrix}$$

where the vectors and matrices have the same meaning as before and the number in the subscript corresponds to the trait number.

Multiple trait linear mixed effects assumptions are

$$\begin{array}{rclcrcl} \mathrm{E}(\boldsymbol{a}) & = & \boldsymbol{0} & & \mathrm{Var}(\boldsymbol{a}) & = & \boldsymbol{G}_0 \otimes \boldsymbol{K} = \boldsymbol{V}_G \\ \mathrm{E}(\boldsymbol{e}) & = & \boldsymbol{0} & & \mathrm{Var}(\boldsymbol{e}) & = & \boldsymbol{R}_0 \otimes \boldsymbol{I} = \boldsymbol{R} \\ \mathrm{E}(\boldsymbol{y}) & = & \boldsymbol{X}\boldsymbol{b} & & \mathrm{Cov}(\boldsymbol{a}, \boldsymbol{e}) & = & \boldsymbol{0} \end{array}$$

where matrix $\boldsymbol{G}_0$ is a t by t genetic covariance matrix, and $\boldsymbol{R}_0$ is a t by t residual covariance matrix. The mixed model equations are

$$\begin{bmatrix} \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{X} & \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{Z} \\ \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{X} & \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{Z} + \boldsymbol{G}_0^{-1} \otimes \boldsymbol{K}^{-1} \end{bmatrix} \begin{bmatrix} \widehat{\boldsymbol{b}} \\ \widehat{\boldsymbol{a}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}'\boldsymbol{R}^{-1}\boldsymbol{y} \\ \boldsymbol{Z}'\boldsymbol{R}^{-1}\boldsymbol{y} \end{bmatrix}$$

2.3 Solving mixed model equations

The mixed model equations are solved iteratively using the preconditioned conjugate gradient (PCG) method. The following topics need to be considered when using the solver program (see *Technical Reference Guide for MiX99 Solver*):

- iterative method
- preconditioner matrix
- iteration on data
- ordering of equations by blocks

2.3.1 MiX99 solver: PCG iteration

The preconditioned conjugate gradient (***PCG***) algorithm is an iterative method to solve linear models. Unlike direct method, the coefficient matrix is not inverted or decomposed. Instead, every iteration updates the solutions by a defined algorithm.

An ***iterative solving method*** updates solutions until convergence is determined, or the maximum number of iterations is reached. Thus, key values given before iteration starts:

- Convergence criterion.
- Stopping value for the convergence criterion: A small positive value (e.g., 10^{-5}) that determines when convergence has been achieved by some formula.
- Maximum number of iterations: A safeguard to ensure the algorithm terminates even if no convergence has been reached.

In practice, the number of iterations needed by PCG typically does not exceed the number of unknowns. While this has little practical significance for large-scale problems, it can be useful for very small problems, as this limit is used by the solver. For large problems, the maximum number iterations is often set between 5000 to 10000 although convergence is usually achieved much earlier.

Note: The convergence criterion and maximum number of iterations cannot be specified by CLIM. Their values must be provided to the solver program `mix99s` or `mix99p`, not the CLIM preprocessor (see Chapter 4).

2.3.2 Iteration on data

Iteration on data (IOD) means that MiX99 does not make or store the coefficient matrix of the mixed model equations in memory. Every iteration of the PCG method operates on the coefficient matrix by performing matrix times vector products. This involves the model matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$, the pedigree list and/or genomic marker data or relationship matrix, and the variance component information. In practice, IOD means that reliabilities must be calculated by a separate program because the coefficient matrix is never made explicitly.

2.3.3 Ordering of equations by blocks

We described mixed model equations such that the equations are ***ordered*** by effect, which is typical in the literature. This is not always computationally optimal. It is better to order equations of an individual and its close relatives together in the same block when the model has a genetic effect with a pedigree based relationship covariance matrix. Such ordering can lead to data locality which supports efficient computations, minimizes communication between processors in parallel computing, and allows efficient algorithms for reliability computation. Equations can be ordered by common family or contemporary blocks, e.g., by a common environment variable. For more information see Chapter 3.3.3.

CLIM has the optional command `WITHINBLOCKORDER` (Chapter 12.2.43) that can be used to indicate which effects are ***within block equations***. For example, if the herd number is the block code, then it is natural that individual related effects (such as the direct genetic and permanent environment), and herd contemporary effects (such as herd-year-season) are within the block. The use of effects in within blocks affects how the preconditioner is made and instructions to the preconditioner are given for the mixed

model equations. Furthermore, the MPI parallel computing implementation (`mix99p`) depends on good block ordering. When reliabilities are estimated by ApaX (`apax99` or `apax99p`), block information is used to determine the level of approximation. Only effects within blocks are considered in the reliability calculations.

The preprocessor automatically orders equations if the user does not provide specific within-block information. However, this may result in longer solving times for large evaluations, and it may affect approximation of reliabilites. In a multi-trait model, the different trait equations within an effect are always ordered next to each other. This ensures efficient solver performance.

2.3.4 Preconditioner

PCG method is a flexible iterative solver. While the core algorithm ***conjugate gradient*** (CG) is the same (with small variations) across implementations, the preconditioning approach varies. Preconditioning aims to improve convergence by transforming the coefficient matrix such that its eigenvalues are near to each other or clustered. This is achieved by approximating the inverse of the coefficient matrix. If the approximation is exact, PCG would converge to the correct solutions in a single iteration. However, in practice, the approximation is never perfect.

A variate of preconditioners are available, each having different memory requirements and time constraints. For large systems of equations, it is impossible to use the most memory-intensive preconditioners implemented. In fact, making the preconditioner can take as much as half of the total preprocessing time.

The `PRECON` command is used to specify the main preconditioner. There is a separate second-level preconditioner available for marker effect. The command to define the main preconditioner is often like:

```
PRECON b b b b  b b
```

where each 'b' letter denotes a block diagonal preconditioner. When no preconditioner is specified, the default preconditioner is 'd' for diagonal for a single-trait model, and 'b' for block diagonal for a multi-trait model.

The order of the letters in the `PRECON` command is important. Preconditioning is defined in the following order:

- Within block effects.
- Across block fixed effects.
- Across block random effects.

The preconditioner definitions for the within block effects follows the order specified by the `WITHINBLOCKORDER` command. If no `WITHINBLOCKORDER` has been given, the last effect in the model is treated as the only within block effect. Preconditioners available for the within block effects are:

d (Diagonal): Uses only the diagonal elements of the MME coefficient matrix.

b (Block diagonal): Applies block diagonal matrices, where the size of each block matrix is equal to the number of equations belonging to an effect level.

Although diagonal and block diagonal preconditioners are functionally the same, when an effect level has a size of one as in a typical single trait model, they are computationally different, which may lead to small differences in the solutions. Furthermore, computations are simpler for a diagonal preconditioner. Use of a block diagonal preconditioner is often computationally most efficient for multi-trait models. However, for complex multi-trait random regression models, they may generate large preconditioning data files with large I/O. In such cases, it may be necessary to use a diagonal preconditioner.

After specifying the within block effects, the next part of the PRECON command defines the preconditioning for the across block fixed effects. Several options are available, listed below in order of increasing computational and memory requirements:

d (Diagonal): Uses only the diagonal elements of the coefficient matrix.

b (Block diagonal): Uses block diagonal matrices of size equal to an effect level.

m (Mixed): A hybrid approach. A block diagonal preconditioner is applied for the levels of the first across block fixed effect, while all remaining across block fixed effects are in one superblock.

m+string (Mixed plus): A more flexible version of the mixed preconditioner, allowing the user to define multiple superblocks using a sequence of integers (see below).

f (Full block): Uses the full matrix corresponding to the across block fixed effects.

In most cases, the block diagonal (b) preconditioner offers a good balance in terms of computational efficiency. However, depending on the model structure, other options may give some advantages.

The mixed preconditioner divides the MME coefficient for the across block fixed effects into two parts:

First part: includes all factors belonging to the first across block fixed effect. In case general regressions (across all data) are specified, also they are included here. Block diagonal preconditioner is used.

Second part: includes all remaining across block fixed effects. One large preconditioner matrix of the size number of equations belonging to the second part will be used.

To optimize performance, the across block effect with the most levels should be defined as the first effect (in the model) when using the "m" option.

The "m+" option allows more control by defining superblocks using a sequence of integers. The number of specified integers must be equal to the number of across block fixed effects. Fixed effects belonging to one superblock must be specified with the same integer number. The first given integer number must be 1 and numbering is in ascending order. Examples, assuming 4 across block fixed effects are specified:

- m 1 2 3 4: same as block diagonal (b) preconditioner.
- m 1 2 3 3: effects one and two use block diagonal preconditioners, and effects three and four are group into one superblock.

- m 1 1 2 2: two superblocks are defined.
- m 1 2 2 2: same as the basic mixed ('m') block.
- m 1 1 1 1: same as the full ('f') block.

The last part of the PRECON command defines the preconditioner for across block random effects. Only two options are available:

d (Diagonal): Uses only the diagonal elements of the coefficient matrix.

b (Block diagonal): Uses block diagonal matrices of size equal to effect level.

If no across block random effects are present in the model, this part can be omitted in the PRECON command.

Note that giving PRECON n disables preconditioning and gives the plain conjugate gradient iteration. This option is useful when computing reliabilities only by apax99 or exa99, as no preconditioning is required in their computation and skipping the computation of preconditioner saves both time and memory in the preprocessor mix99i.

In this manual, all examples use default preconditioner. Thus, although no PRECON command is given, the preconditioner varies by model: single trait models use diagonal preconditioner, multi-trait models use block diagonal preconditioner. The preconditioner used is informed by the preprocessor mix99i and by the solver mix99s. In the preprocessor output, there are lines like:

```
Preconditioner information:
   Across block fixed     : block diagonal
   Across block random    : block diagonal
   Regression matrix      : block diagonal
   1. within block effect : block diagonal
```

In the beginning of the solver output, this same information is given as

```
 Preconditioner information:

 XpX : b  across block fixed
 MpM : b  regression matrix
 WPW : b  within block effects
 WPWa: b  across block random
 d=diagonal, b=block diagonal, f=full block
```

2.4 Second-level preconditioner (sp)

Second-level preconditioner for can be useful when there are several effects that model genetics. For example, model has both marker effects and polygenic effects. The second-level preconditioner is applied to all regress matrix effects (REGMATRIX) simultaneously. It is also applied to all marker effects in the ssSNPBLUP model. The inverse of the given value for the second-level preconditioner is used to multiply all the regress matrix and marker effect values after the first level (regular) preconditioner has been used. mix99s For example, giving value '-sp 100' for the solver mix99s, e.g., mix99s -s -sp 100, can enhance convergence considerably.

3 Input files

Most of the information is read from input files. The input files are:

- data file,
- genotype file(s),
- pedigree file,
- variance components file(s),
- other files.

Individuals in the data and pedigree files must be in the same order. In other words, when data records have been sorted by an individual ID code, then the individuals must be in the same order in the pedigree file as well. The pedigree can be ordered using the RelaX2 program which is a separate utility program for pedigree analysis from Luke (Strandén and Vuori, 2006).

In some models, the genotype file must follow the data file order as well, but not always.

The input files have a certain quite simple format. In the following, the formats of these files are described briefly. For a more complete explanation, see the MiX99 pre-processor manual: *Technical reference guide for MiX99 pre-processor*.

3.1 Data file

The data file contains the observed data to be analyzed. Name of the data file is defined by the `DATAFILE` command, For example, using `data.dat` as the data file is defined by the command `DATAFILE data.dat`. The data file has observations and model effect information such as classification effect numbers and regression covariates. The default format is 'text', which is a ***text format data file*** with columns separated by one or more spaces. A rarely used alternative is the Fortran unformatted ***binary data file***.

Each record, i.e., line in a free format file, has two parts:

- ***Integer numbers*** The integer number data part consists of positive integer numbers for all class variables in the model. In addition, it may contain sorting variables and indices such as an index for heterogeneous residual variance.
- ***Real numbers*** These are observations, covariates, and weights.

The data file may contain columns that are not used in a model. Because the data file can only contain numeric data, alphanumeric data is only allowed in the record after the real number columns in a free format text data file.

All integers are coded using the default machine integer type and must be positive. Typically, integer numbers must be less than 2,147,483,649. ***Missing integers*** must be coded as zero (0). For real (floating-point) numbers, ***missing value*** can be coded by any real number specified using the `MISSING` command. By default, this value is zero.

3.1.1 Example: Multiple trait data

Consider data for a two-trait model. The file has six columns: 4 integer columns, and 2 real columns. Note that the real number columns in this example have integer values. However, for the preprocessor, these are real number columns because observations can have any real value. The file named `example.dat` is:

ID code$_1$	sire$_2$	herd×year$_3$	ones$_4$	trait 1$_1$	trait 2$_2$
4	1	1	1	90	200
6	3	1	1	110	190
8	5	2	1	120	140
9	5	2	1	130	120
10	7	2	1	120	130

The subscripts of the header names are column numbers in the integer and real column areas. No header line is allowed to be present in the data file.

This data file can be described with the following CLIM commands:

```
DATAFILE   example.dat                  # Name of the data file
INTEGER    IDcode sire herdXyear ones   # Integer column names
REAL       trait1 trait2                # Real column names
```

3.2 Genotype file

The genotype file contains the genotypes. There can be multiple genotype files. Consequently, there is no single command for a genotype file. For example, genotypes may be used in a SNP-BLUP model where the genotypes are regression covariates defined by the REGMATRIX command, and the genotypes are in a file defined by the REGFILE command. There can be multiple REGMATRIX commands each with its own REGFILE. When a single-step SNP-BLUP model is used, the genotype file is defined by the SNPFILE command.

Alternative formats are available for genotype file. However, it is always assumed that genotype is a number 0 for homozygous first allele, 1 for heterozygote, and 2 for homozygous second allele. Missing genotype is sometimes allowed but it has to be single digit number such as 3. In general, genotypes are considered as covariates in a regression model. Thus, an additive genetic model is assumed. Please see REGMATRIX and SNPMATRIX commands for the formats.

Each record, i.e., line in a free format text file, of a genotype file has two parts:

- ID code. The ID code is a positive integer number for the ID code of individual owing the genotypes.
- Genotypes. These are genotype values.

The genotype file may contain columns that are not used in a model. Because the genotyped file can only contain numeric data, alphanumeric data is only allowed if FORMAT option for reading in Fortran format is allowed.

3.3 Pedigree file

All pedigree information is contained in the ***pedigree file***. Every individual in the pedigree must have one and only one record in the pedigree file. Each pedigree record has three or four integers, where the fourth integer is optional. Columns of the pedigree file are

1	2	3	4
ID code	sire code	dam code (or maternal grand sire code in case of a sire model)	block code (optional)

The integers must be separated by at least one space. A Pedigree file that is defined to be my.ped using the CLIM command:

```
PEDFILE my.ped
```

When a block code is given, the pedigree and data files must be sorted in the same order by the block code. For more information, see Chapter 3.3.3.

3.3.1 Example: Pedigree file for the data

Let the pedigree file in the file `my.ped` for the multi-trait model data in the example Chapter 3.1.1 be

ID code$_1$	sire$_2$	dam$_3$
1	0	0
2	0	0
3	1	2
4	1	2
5	3	4
6	3	4
7	5	6
8	5	6
9	5	6
10	7	8

3.3.2 Unknown parent groups

Pedigree is often incomplete and has ***missing parents***. Missing parents can be replaced by genetic (unknown parent) groups (UPGs). Similar missing parents are assigned to the same UPG. In the pedigree, an UPG code is used in place of the missing parent. This code must be a negative integer number to distinguish it from an existing pedigree ID code. An UPG code should not have a record in the pedigree file.

For example, the pedigree above (Chapter 3.3.1) had two individuals (1 and 2) that had unknown parents. The missing parents can be assigned to be UPGs (-1 for an unknown sire, -2 for an unknown dam). The pedigree file remains the same for all the others. The changed part of the pedigree file is:

ID code$_1$	sire$_2$	dam$_3$
1	−1	−2
2	−1	−2

3.3.3 Block code

There are some advantages to having a ***block code*** in the data and pedigree files. Block codes are essential for the calculation of reliabilities in ApaX and for the MPI parallel computing implementation in `mix99p`. However, for some large evaluations, the use of the block code with data sorting can improve computational efficiency by improving cache utilization through data locality. Otherwise, the use of a block code may reduce the computation time very little and can often be omitted. The `DATASORT`

command is used to define the block code column in the data file, but the pedigree file has a fixed column structure, assuming the block code is in the fourth column.

The block code in a data file is defined by the BLOCK option in the DATASORT command. For example, DATASORT BLOCK=herd defines integer column herd to have the block code in the data file. The main sort key of the data file is the block code (now herd), within which the individual ID code, referred to as the PEDIGREECODE, is sorted. An example is DATASORT BLOCK=herd PEDIGREECODE=IDcode. Even if an individual has observations in multiple data file blocks, the individual must appear in only one of the blocks in the pedigree file. This special case is considered in a separate section on MPI parallel computing. Note that specifying a BLOCK in the DATASORT command is optional. See below.

An individual's block code is specified in column 4 of the pedigree file. When the pedigree file has the block code column, then every individual must have a block code. In addition, the block code must also be the same in the data file. An individual with records in different data blocks (e.g. in different herds) have to be coded in the pedigree with only one of the block codes where it has observation(s), e.g., the block with most of its observations.

Typically, a block code is an environmental unit such as a herd where an individual's data have been observed. If an individual has no observations, but is a parent of an individual with observation(s) in the data file, then it is natural for the parent with no observations and its offspring to have the same block code. For example, a dairy cow with no observations will be assigned to a block with most of its daughters. This ensures efficient computations in models with a pedigree-based relationship matrix.

When an individual does not belong to any equation family (no observations to give a block code), or it is in many different families through relationship information (e.g. dairy sires have progeny in many herds), an extra block code can be generated. We recommend a separate block code for individuals with close pedigree links to many different equation family blocks. For example, sires in a dairy cattle population can be assigned to a single block code with a number greater than any of the block codes in the data file. Note that a block code can never exceed the maximum integer number of 2147483647. It is advisable to split a large block into several smaller blocks. MiX99 supports as many blocks as possible at the same time, and the block with the most individuals determines some memory requirements, e.g. in making the preconditioner.

3.4 Variance components file

The ***variance components*** file contains (co)variances for all the random effects in the model. The size of these matrices may vary based on the model specification. The random effects are numbered in the order they appear in the RANDOM command. Thus, the random effects should appear in their numbering order in the model line(s), i.e., on the model line(s) from left to right, the random effect one is before random effect two etc. A RANDOM command is not required if the only random effects in the model are the individual additive genetic and the residual effects. The residual effect always assigned the last random effect number, and the additive genetic effect is assigned the second-to-last random effect number.

The variance components file has the values of the (co)variance components. Two file formats are available:

- the traditional sparse matrix format
- a mixed matrix format.

It is also possible to specify that the identity matrix is used for all (co)variance matrices. This can be convenient when setting initial values for variance component estimation.

The name of the variance components file is defined by the PARFILE command. For a traditional sparse matrix file (e.g., ST.var), the command is

```
PARFILE ST.var
```

For a mixed matrix file (e.g., AMmixed.var), the command is NEW

```
PARFILE MIXED AMmixed.var
```

It is possible to specify the variance components in the CLIM file using the command

```
PARFILE CLIM
...
```

Then, the contents of the file are in the CLIM file after the PARFILE command. When all random effects are assumed to have an identity matrix as their (co)variance matrix, then the command is

```
PARFILE IDENTITY
```

When the MIXED, CLIM, or IDENTITY option is used with the PARFILE command, the variance components are written to the Tmpfile.par file in the temporary file directory, using the traditional sparse matrix format.

The traditional sparse matrix format has one line for each non-zero (co)variance. Each line consists of 3 integers followed by a real number representing the (co)variance value. For example, `1 1 1 3.0` indicates that random effect 1 (the first '1') has a variance of 3.0 for the first trait (position '1 1'). The first integer is the random effect number, followed by two numbers for the row and column position, and finally the (co)variance parameter. The row-column position specifies the position of the element in the (co)variance matrix. Only the lower (or upper) triangle of the matrix needs to be specified. If a (co)variance value is not provided, it is assumed to be zero.

The order of the lines in the this type of file does not matter. Identifying the number of the random effect is easy. However, correctly numbering the (co)variances, i.e., the row-column numbers, can be challenging in some models. For example, in a multi-trait model with a random effect that has multiple variances, such as in a random regression model, the numbering is done sequentially by each trait and from left to right on the model line. Therefore, some extra care is needed. The preprocessor output of the (co)variance matrix can be used to verify that the row-column numbers have been entered correctly. This is particularly important when random regression effects are missing in a multi-trait random regression model. For a more detailed explanation, see the examples on multi-trait random regression effects model (Chapter 6.4).

The mixed matrix format file for variance components can include various matrix formats, such as sparse, lower triangle, diagonal, and identity. The first line of a (co)variance matrix in the MIXED matrix format specifies the random effect number and the format type used. For the four formats, the line for the first random effect is NEW

- for the sparse format: `1 SPARSE`
- for the lower triangle format: `1 LOWER`
- for the diagonal format: `1 DIAGONAL`
- for the identity format:`1 IDENTITY`.

This line is followed by the (co)variance values in the specified format. The `SPARSE` format is identical to the traditional sparse format, except that the random effect number is not needed, as it has already been provided. The `LOWER` dense format includes a lower triangle of the (co)variance matrix. The `DIAGONAL` format requires only the diagonal values, i.e., the variances. The `IDENTITY` format does not include any values after specifying the format type, as the identity matrix is used for the (co)variance matrix. The (co)variance components in a variance component file in the `MIXED` format must be specified in numerical order, from the first to the last random effect.

3.4.1 Example: Variance component file

We illustrate the variance components file for the multi-trait model in Examples Chapters 3.1.1 and 3.3.1. Let the genetic and residual (co)variance matrices be

$$\boldsymbol{G}_0 = \begin{bmatrix} 3.0 & 2.5 \\ 2.5 & 2.5 \end{bmatrix} \text{ and } \boldsymbol{R}_0 = \begin{bmatrix} 7.0 & 2.0 \\ 2.0 & 7.0 \end{bmatrix},$$

respectively. The genetic correlation between the traits is about 91%, and the residual correlation is about 29%. Heritability of the first trait is 30%, and the second trait is about 26%. The traditional (co)variance parameter file has format

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$
1	1	1	3.0
1	1	2	2.5
1	2	2	2.5
2	1	1	7.0
2	1	2	2.0
2	2	2	7.0

NEW

A `MIXED` format variance components file using `SPARSE` and `LOWER` is

SPARSE

```
1 SPARSE
  1 1 3.0
  2 1 2.5
  2 2 2.5
2 SPARSE
  1 1 7.0
  2 1 2.0
  2 2 7.0
```

LOWER

```
1 LOWER
  3.0
  2.5 2.5
2 LOWER
  7.0
  2.0 7.0
```

Assuming the variance matrices were diagonal, the `MIXED` variance components file is

```
1 DIAGONAL
  3.0
  2.5
2 DIAGONAL
  7.0
  7.0
```

A MIXED variance components file can mix the four formats in the same file. For example, a hypothetical variance components file could be:

```
1 IDENTITY
2 DIAGONAL
  2.1
  2.2
3 LOWER
  3.1
  3.0 3.3
4 SPARSE
  1 1 4.1
  2 1 4.0
  2 2 4.2
```

3.4.2 Multiple residual (co)variances

When there are multiple residual (co)variances, an ***additional residual (co)variance file*** must be specified using the RESIDFILE command. The format of this file is the same as the regular (co)variance file described above. However, the first number on each line is not the random effect number, but the number of the residual variance class. The numbering of the residual (co)variance classes has to start from one (1) and go up to the total number of residual (co)variance classes. Each observation has its residual (co)variance class number in the INTEGER column fields of the data file. Note that a residual (co)variance (matrix) has to be also included in the variance components file. Residual (co)variance values in the variance components file are ignored by the solver program (mix99s or mix99p), but are used by the reliability approximation program (apax99 or apax99p). The values in the (co)variance components file are often weighted averages of the values in the multiple residual covariance file and are used in the calculation of heritability in the reliability estimation program.

4 Using the solver

After successfully running the preprocessor, the solver (mix99s/mix99p) can be executed. The solver program assumes that the user gives some instructions about some aspects of the iteration method, the output files to be produced, and possible special computations to be performed (see Chapter 11). The instructions can be given either through standard input or through command line options. The use of both requires the use of the -i option on the command line (see below for more details). By default, the instructions from the standard input are ignored when command line options are given.

In this manual, the command line option approach has been used most of the time for simplicity. This method is only possible when calculating breeding values. The easiest way to run the solver is to give the -s option, which uses default values in solving the breeding values and produces standard output files. So you give mix99s -s. Examples in this manual have been created with this option unless otherwise noted.

Some other common command line options are

- `-n` or `-N` for the number of iterations
- `-ca` or `-Ca` for Ca convergence criterion
- `-cd` or `-Cd` for Cd convergence criterion
- `-cr` or `-Cr` for Cr convergence criterion
- `-cm` or `-Cm` for Cm convergence criterion

For example, giving `mix99s -n 1000 -cr 1e-7` sets the maximum number of iterations to be 1000, and the Cr convergence criterion stopping value to be 10^{-7}. Thus, the iterations are considered converged when the Cr convergence criterion has value less than 10^{-7}. By default, the convergence criterion is Cd with a stopping value 10^{-4}, and the maximum number of iterations is 5000. These defaults may not be sufficient for some models and traits. Therefore, it is important to check that convergence has been reached and that the iterations have not terminated prematurely due to reaching the iteration limit.

The `mix99s` solver computes the values for the four convergence criteria in every iteration, although only one is used to determine convergence. Thus, it is possible to monitor every one of them. Below formulas for the criteria are given. Let the MME be expressed as $\boldsymbol{C}\widehat{\boldsymbol{s}} = \boldsymbol{r}$ where $\boldsymbol{C}$ is the coefficient matrix, $\boldsymbol{r}$ is the right-hand side vector, and $\widehat{\boldsymbol{s}}$ is the vector of unknowns to be solved. Denote by $\widehat{\boldsymbol{a}}$ the solutions of the additive genetic effects which can be associate with relationship matrix $\mathbf{K}$ as in Chapter 2.1. The four convergence criterion are printed to the `Conlog` file by the `mix99s` solver in the following order:

$$
\begin{aligned}
Ca_{(k)} &= \sqrt{\frac{\left(\boldsymbol{r} - \boldsymbol{C}\widehat{\boldsymbol{a}}^{(k)}\right)' \left(\boldsymbol{r} - \boldsymbol{C}\widehat{\boldsymbol{a}}^{(k)}\right)}{\boldsymbol{r}_a' \boldsymbol{r}_a}}, \\
Cr_{(k)} &= \sqrt{\frac{\left(\boldsymbol{r} - \boldsymbol{C}\widehat{\boldsymbol{s}}^{(k)}\right)' \left(\boldsymbol{r} - \boldsymbol{C}\widehat{\boldsymbol{s}}^{(k)}\right)}{\boldsymbol{r}'\boldsymbol{r}}}, \\
Cm_{(k)} &= \sqrt{\frac{\left(\boldsymbol{r} - \boldsymbol{C}\widehat{\boldsymbol{s}}^{(k)}\right)' \boldsymbol{M}^{-1} \left(\boldsymbol{r} - \boldsymbol{C}\widehat{\boldsymbol{s}}^{(k)}\right)}{\boldsymbol{r}'\boldsymbol{M}^{-1}\boldsymbol{r}}}, \\
Cd_{(k)} &= \sqrt{\frac{\left(\widehat{\boldsymbol{s}}^{(k)} - \widehat{\boldsymbol{s}}^{(k-1)}\right)' \left(\widehat{\boldsymbol{s}}^{(k)} - \widehat{\boldsymbol{s}}^{(k-1)}\right)}{\left(\widehat{\boldsymbol{s}}^{(k)}\right)' \left(\widehat{\boldsymbol{s}}^{(k)}\right)}},
\end{aligned}
$$

where

- k is the iteration number.
- Ca is the relative difference between left-hand side and right-hand side (residual) of the part of the MME which includes the equations of the additive genetic effects.
- Cr is the relative residual of all effects of the MME.
- Cm is the preconditioned relative residual of the MME.
- Cd is the relative difference between solutions from consecutive iterations.

Instructions to the solver can also be given in the standard input. This allows a wider range of options and methods than are available in the command line options. Note, however, that giving command line options will by default result in the standard input not being read. It is sometimes more convenient to have the instructions in a file than to type them into the program every time. This can be achieved by reading them from the standard input, e.g., `mix99s < solver_option_file.slv`. Giving

```
mix99s -n 1000 -cr 1e-7 < solver_option_file.slv
```

will not read commands from the file `solver_option_file.slv` but proceed with the command line options only.

The `-i` command line option causes the solver option file (or standard input) to be read first AND ***the command line options to override the corresponding solver option file values***:

```
mix99s -i -n 1000 -cr 1e-7 < solver_option_file.slv
```

An example of a breeding value evaluation instruction file is

```
H                 # RAM:    Demand: H=high, M=medium, L=low, X=extra large
2000 1.0e-7 R F   # STOP:   Max.iter., Conv.value, Conv.criterion, Force
N                 # RESID:  Residuals calculation
N                 # VALID:  Model validation
N                 # VAROPT: Variance options for VCE, PEV, HV
Y                 # SOLTYP: Solution file options
```

The first letter `H` requests the high memory version, which is usually used. The medium and low memory versions are rarely used because even the high memory version uses memory efficiently. Note: `mix99p` has an eXtra large memory requirement approach by letter `X` that requires some more memory than `H` but is much faster.

The most important line is the second line where PCG iteration information is given:

- the number of iterations in the PCG method is limited to 2000 iterations
- convergence value is set to be 10^{-7}
- convergence criterion is set to Cr by letter "`R`"
- the above values are "`F`"orced to be used.

If "`F`" is not given then default values are used. The default values are

- limit to 5000 iterations in the PCG method
- convergence value is 10^{-4}
- convergence criterion is Cd or letter "`D`".

The first three options after the PCG information are not that important for typical breeding value evaluations, and their value are in the example is "n" for no. In the chapter on special topics (Chapter 11) some of these options will be considered. The last "Y" is important. It indicates the type of the solution files. The letter "Y" indicates standard text format solution files. If the last letter is "N", only a binary format solution file named `Solvec` will be created having solutions for all the unknowns.

4.1 Multi-threading

The solver can use multi-threaded computations if the computing environment allows it and the multi-threaded version of the solver is used. MiX99 executables parallelize some of the calculations, especially large dense matrix computations.

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

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

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

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

The multiple threads can be taken to use by the above mentioned environmental variables or by solver command "nt". Effect of the `nt` option depends on the statistical model and memory option (see Chapter 4.6) used. Models that typically benefit from multi-threading are such that use genotypes. For example, CLIM has command `GBLUP`, `COVFILE`, `SSGBLUP REGMATRIX`, or `SNPMATRIX`. When the genomic relationship matrix or marker matrix is in RAM memory, efficient parallel computing can be used.

In `mix99p`:

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

The command line option is like:

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

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

The logic behind the differences is that option “MEL"/"MEB"/"MEM" lead to the Master process to have an additional computational work that will benefit from multi-threading. However, instructing all the other processes to have as many threads may lead to use

of too many threads and inefficient computations. Because this may be often the case, the optimum number of threads in non single-step or "MES" option can be small.

4.2 Terminating iterations cleanly

In some situations, it might be useful that the iteration process is stopped in a controlled fashion before the specified limit of the convergence criterion has been fulfilled. The solver programs can be instructed to stop after the current round of iteration by creating a file named `STOP` in the directory where the solver is executed. Then, the solver will write the most recent solutions to the standard solution files. When MPI parallel computing is applied, the `STOP` file has to be accessible by the master process. In case heterogeneous variance is accounted using the multiplicative mixed model approach described in Meuwissen et al. (1996), a `STOP` file can be used to stop the cycling between the mean model and the variance model. Then, after the current adjustment cycle is finished, the program will continue with iterations on the mean model until the set value for the convergence criterion is reached or the `STOP` file is provided a second time. The solvers will erase the `STOP` file from the directory to avoid trouble in future analysis.

4.3 External PEEK file: intermediate solutions during iteration

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

The `PEEK` file may be either empty or contain one or two integers:

```
[-]PITER PSTEP
```

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

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

```
20 100
```

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

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

4.4 External ITER file: changing parameters during iteration

It may also be useful to be able to modify the parameters of the iterative method during the iteration. This can happen, for example, if it is realized that the original maximum number of iterations is found to be too low or limit for the convergence criterion too tight.

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

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

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

or

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

The fifth parameter is used by threshold-models and deregression. Parameters on the STOP line control the normal PCG iterations of the solvers.

For estimating variance components, optional second line similar to STOPE line of the solver option file can be specified:

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

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

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

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

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

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

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

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

```
*********************  M i X 9 9 s   M e s s a g e  **********************
 Message:                                    Time: 13:30:10.7  02.07.2019
```

```
   Updating iteration parameters from ITER file:
   - Solver STOP line in ITER file:
  >> 6000     1.0e-6      M                     F
     - Max. number of iterations changed from 5000 to 6000.
     - Stopping criterion value changed from .5e-6 to .1E-7.
     - Stopping criterion changed from D to M.
   - Renaming ITER file to ITER.OLD.
*******************************************************************************
```

4.5 The solver option file details

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

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

4.6 Solver option lines

The `mix99s` solver option file consists of one or more option lines. These lines must appear in the exact order specified below. Including all option lines is not mandatory. However, if an option line is omitted, all successive lines must also be left out. In such a case, the solver will use default values for the missing options. Comment lines are allowed in the same manner as in CLIM: any text after the `#` character on a line is ignored. Option values can be specified using either uppercase or lowercase letters.

RAM A line with at least one character,

```
# RAM demand:
   H
```

which defines the use of random access memory. See previous chapter.

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

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

nt n Number of CPUs used in multi-threading is set to be n.

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

sp v Second-level preconditioner of value v.

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

IOP/IM/CHM/PAR in single-step method, product of $\mathbf{A}_{gg}^{-1}$ times vector can be performed using one of four altenative approaches. Approach IOP uses iteration on pedigree, IM uses iteration in memory, but

CHM uses CHOLMOD and PAR uses MKL PARDISO library. Use of memory from lowest to most: IOP, IM, CHM/PAR, where memory need by IOP and IM is quite close but CHM/PAR use much more RAM. Computing time from highest to lowest: IOP, IM, CHM/PAR, where there is substantial difference between all approaches. The CHM and PAR options can use multi-threading.

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

MEM is like option MEL but slower and uses slightly less memory, when $\mathbf{T}$ matrix is used. Same as MEL when $\mathbf{G}^{-1}$.

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

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

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

RDS instructs to use a small block size when calculating contributions from regression design matrices. Keeps SNP marker matrix byte-packed in memory. Currently implemented in `mix99s` only.

RDM similar to RDS but uses a medium block size. Currently in `mix99s` only.

RDL similar to RDM but uses a large block size. Currently in `mix99s` only.

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

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

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

ω value for the ω multiplier of matrix $\mathbf{A}_{gg}^{-1}$.

Defaults: **h** for high memory, **YES** for check release information, **PAR** for

MKL PARDISO library, **MEL** for having $\mathbf{G}^{-1}$ or $\mathbf{T}$ in memory, block size 1 for regression design matrices (except if all are SNP matrices: 1/5/10 for Low/Medium/High memory options), and 1 for value of ω. Option **MEA** is default for the fully component-wise ssGTBLUP and ssSNPBLUP models that use 1 byte marker matrix.

The following will change to use CHM, MEB, and 10 CPU threads:

```
   H CHM MEB 1000 nt 10
```

STOP One line with four entries:

```
# maxiter, tolerance,  criterion (A/R/M/D), [enforce (F)]:
  5000     5.0e-5       D                    F
```

The solver will consider the `STOP` option line only if an enforcing character **f** is specified for the fourth entry. Otherwise, the line is ignored and default values will be used.

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

- **a** Ca. Relative difference (residual) between right-hand and left-hand side of the MME considering all equations of the additive genetic effects only.
- **r** Cr. Relative difference between right-hand and left-hand side of the MME considering all equations.
- **m** Cm. Relative difference between preconditioned right-hand and left-hand sides of the MME considering all equations.
- **d** Cd. Relative difference between solutions of the last two iteration rounds. Note that if Cd is chosen as the convergence criterion, it should be less than the stopping value in two consecutive iterations.

Default values: **5000 1.0e-5 d**

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

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

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

y **Yes**. Residuals will be written into the file(s) `eHat.data(i)`. When using `mix99s`, (i) will be zero (0). In case of MPI parallel processing, each process writes an own `eHat.data(i)` file with the process number (i) at the end of the filename. The order of the residuals corresponds with the order of observations in the input data file. The `eHat.data(i)` files have as many columns with residuals as the maximum number of traits in the largest trait group. This is equal to the mxntra parameter shown in the `Parlog` file which is produced by `mix99i`. The residual columns are ordered in the same sequence as the traits in the trait groups. For missing observations, the corresponding value in the residual files are set to the ***missing value -8192.0***.
In case of MPI parallel processing, the order of the residual files correspond to the order of observations in the input data file beginning with file zero (0) up to number of processes minus one.

n **No**. No residuals are written.

h This option is only available in `mix99p` and will create the file(s) `ARsiwi.data(i)`, which contain information about the heterogeneity of variance in the residuals. These files are needed only when accounting for heterogeneous variance.

Default value: **n**

VALID A line with one entry,

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

which instructs the solver to calculate for each observation a corresponding, here specified, sub-quantity of the applied model line, or to instruct the solver to simulate observations based on the specified model, or to deregress breeding values. By default, the calculated quantities for options (P, S, Y, D, I) are written to binary files after the iteration process has ended. A text file is written when the option letter is appended with letter t, e.g. pt. DYDs are always written to a text file. Missing value is -8192.0. Alternatives are:

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

p **Predictions**. For each observation, the predicted value ($\widehat{y}$) is written to the file(s) `yHat.data(i)`.

s **Selected Model Factors' Sum**. For each observation the sum of selected model factors is written to the file(s) `sHat.data(i)`. The selected factors must be specified on a following line.

y **Yield Deviations (YD)**. For each observation the corresponding YD will be written to the file(s) named `YD.data(i)`. The factors included into the YD must be specified on a following line.

i **Individual Daughter Deviations (IDD)**. For each observation the corresponding IDD will be written into the file(s) named `IDD.data(i)`. The factors included into the IDD must be specified on a following

line.

d **Daughter Yield Deviations (DYD)**. The solver will calculate for each observation the corresponding IDD and will use it for the calculation of DYDs based on the approach of Mrode and Swanson (2004). For this option the calculated DYD will be written to a formatted filenamed `Soldyd`. The factors included into the DYD must be specified on a following line.

g **Generate Observations**. This option is available in `mix99s` only. The `mix99s` solver will not solve the model, but instead will generate for each observation in the data a simulated observation ($\widetilde{y}$). Therefore, for all effects in the model true solutions will be simulated based on the provided variance components. Fixed effect solutions will be set to zero. The true solutions are written to the standard solution files. The generated observations will be written into the file named `ySim.data0`. This file can be used in a future MiX99 run to replace real observation by simulated observations. See the VAR instruction line in the *Technical reference guide for MiX99 pre-processor* for reading and using of the generated observations instead of the real observations.
When specifying **g** a SEED option line must be included after the VAROPT option line. The SEED option line has one entry, which must be one of those given in the SEED option description below (see VAROPT).

r **Deregression** (See Chapter 11.2).

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

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

FACTOR One line with as many integers as there are factor (effect) columns (missing included) defined in `MODEL` line. In the `MiX99_DIR.DIR` file, the number is the same as the first integer value on the REGRESS instruction line (see *Technical reference guide for MiX99 pre-processor*). The order of the integer values on the FACTOR line corresponds to the order of the factors specified on the `MODEL` as well as on the REGRESS instruction line. Each integer specifies whether or not the corresponding model factor is included into the calculation of the desired quantity.

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

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

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

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

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

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

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

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

VAROPT A line with at least one entry that specifies different options related to the estimation of variance components, the estimation of prediction error variances by Monte Carlo, or the adjustment for heterogeneous variance.

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

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

e <f *n*> **Estimation of Variance Components**. The option **e** will instruct `mix99s` to estimate variance components by a stochastic Monte Carlo Expectation Maximization REML (MC EM REML) algorithm. A special case is the **ei** option for estimation of variance components of a MACE model (see Chapter 11.3.8).

An additional (optional) instruction can be given after the **e** (or **ei**) character. This instruction has two entries; the character **f** and an integer number *n*. The optional instructions are needed in case certain variance component parameters are meant to be fixed.

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

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

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

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

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

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

Default value: **d**

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

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

c **Cycle between models for solving the multiplicative mixed model**. The option is needed for the heterogeneous variance adjustment and will instruct `mix99p` to discontinue in certain intervals the iteration

process and make system calls for solving the variance model by a second, simultaneous ***MiX99*** analysis. The process will continue until both models have converged.

ADJUST In case **s** or **c** is defined on the VAROPT option line, an additional line needs to be specified with as many integers as there are factor columns (missing included) defined in `MODEL` line. In the `MiX99_DIR.DIR` file, the number is the same as the first integer value on the REGRESS instruction line (see *Technical reference guide for MiX99 pre-processor*). The order of the integer values on the ADJUST line corresponds to the order of the factors specified on the REGRESS instruction line. Each integer specifies whether or not the corresponding factor of the model is included into the adjustment of heterogeneous variance.

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

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

$$\boldsymbol{y}_i = \boldsymbol{X}_i\widehat{\boldsymbol{b}} + (\boldsymbol{Z}_i\widehat{\boldsymbol{p}} + \boldsymbol{Z}_i\widehat{\boldsymbol{a}} + \widehat{\boldsymbol{e}}_i)\,\gamma_i,$$

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

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

$$Cd_{(k)} = \frac{\left(\boldsymbol{l}^{(k)} - \boldsymbol{l}^{(k-1)}\right)'\left(\boldsymbol{l}^{(k)} - \boldsymbol{l}^{(k-1)}\right)}{\left(\boldsymbol{l}^{(k)}\right)'\left(\boldsymbol{l}^{(k)}\right)}$$

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

Default values: **1000 1.0e-7**

SOLTYP A line with one character:

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

which specifies the way solutions are handled. Solving a standard linear model expects **y**.

4.7 Files for model validation purposes

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

eHat.data(i) File(s) with residuals.

yHat.data(i) File(s) with predicted observations.

sHat.data(i) File(s) with a sum of selected model factors.

YD.data(i) File(s) with yield deviations.

IDD.data(i) File(s) with individual daughter deviations.

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

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

All values on a row are stored as single precision real values and simple Fortran programs can be written to transfer the files to text files. However, the ***MiX99*** packages provides MiXtools programs, which allows simple analyses of the information (means and SD by classification), or merging of the files with the input data file. See MiXtoolmerge.f90 and MiXtoolms.f90 in the MiXtools directory of the package. Alternatively, the files may be requested to be in text format by adding 't' after the option name when requesting it.

4.8 Solution files

The solver will write ***solution files*** depending on the model. Different types of solutions are written to different files:

- `Solani`: Solutions for additive genetic effects. (`Sol_mn` in case of an LS-model.)
- `Solfix`: Solutions for all across blocks fixed effects.
- `Solf`*nn*: Solutions for the n^{th} within block fixed effect. For instance, `Solf02` is the solution file for the second within block fixed effect.
- `Solr`*nn*: Solutions for the n^{th} random effect in the model. For example, `Solr03` is the solution file for the random effects with the random effect number 3.
- `Solreg`: Solutions for the regression effects of the first regression effect group (applied across all observations).
- `Solreg_mat`: Solutions for the regression effects of `REGMATRIX`.
- `SolDGV`*nn*: Genomic breeding value solutions for the n^{th} `REGMATRIX` in the model. For example, `SolDGV02` has the genomic breeding values for the second `REGMATRIX` command in the CLIM file.
- `SolSNP`: Solutions for all SNP marker effects. Available in component-wise single-step models.
- `SolPA`: Solutions for parent averages of the additive genetic effects.
- `SolMS`: Solutions for Mendelian sampling deviations of the additive genetic effects.

The structure of the text solution files depends on the model, but the general form of a particular solution file is the same. However, the number of columns in a `Solani` file depends on the number of traits. An explanation of the columns of the solution files is given at the end of the printout of the solver program.

Below are descriptions of the two most common solution file formats. In general, the `Solani` file has the following columns:

1) individual ID code
2) number of offspring
3) number of observations
4) solution for trait 1
5) solution for trait 2
6) ...

In this manual, column titles are shown for `Solani`, although they are not present in the file. When there are random regression effects, solutions are in the numbering order of the random regression effects. The numbering order is explained in Chapter 7.3.

The `SolPA` and `SolMS` files share the same format as the `Solani` file. However, instead of containing solutions for the additive genetic effects, they have parent averages and Mendelian sampling deviations derived from those solutions.

In general, the `Solfix` file has the following columns :

1) factor number
2) trait number
3) level code
4) number of observations
5) solution
6) name of factor (integer number column)
7) name of trait.

The `Solfnn` file has the solutions for the within-block fixed effect number *nn*. The file has the following columns :

1) level code
2) number of observations
3) solution for trait 1
4) solution for trait 2
5) ...

The `Solrnn` file has the solutions for random effect *nn*. The file has the following columns :

1) level code
2) number of observations
3) solution for trait 1
4) solution for trait 2
5) ...

The `Solreg` file has the solutions for regression effects. The file has the following columns :

1) trait number
2) regression number within trait
3) solution
4) trait name
5) covariable name

The `Solreg_mat` file has the solutions for regression effects defined by the `REGMATRIX` commands. The file has the following columns :

1) trait number
2) `REGMATRIX` number within trait
3) effect number within `REGMATRIX`
4) solution
5) `REGMATRIX` name

The `SolDGVnn` file has the genomic breeding value solutions for `REGMATRIX` number *nn*. The file has the following columns :

1) individual ID code number
2) number of observations
3) solution for trait 1
4) solution for trait 2
5) ...

The SolSNP file has the following columns:

1) SNP marker number
2) solution for trait 1
3) solution for trait 2
4) ...

4.9 Using old solutions for the solver

Solutions from a previous run can be reused as initial values by the solver. The solver automatically writes all solutions in binary format to a file named Solvec. When the solver is restarted in a directory containing a Solvec file, it uses the solutions in that file as initial values for the iterative process. When new data is added or variance components are updated to an evaluation, where the model effects in the mixed model equations remain the same, it may still be possible to use the solutions in the existing Solvec as initial values. To achieve this, the preprocessor must be invoked using the --keepsol option. Without this option, the Solvec file will be ignored and deleted. The Solvec file approach for initial values is commonly used when additional iterations are needed to achieve convergence.

A more general approach is to create a Solunf file, especially when new data introduces new effect classes (e.g., new years or individuals). This file is generated when the MODEL command has the the RESTARTSOL option. To use the Solunf file as a source of initial values, it must be renamed to be Solold for the preprocessor to recognize it. Unlike the Solvec file, the Solunf file has information to accommodate increase in the number of levels in the model effects. For example, new data may have new individuals with new levels of fixed effects.

5 Describing model

5.1 Naming of model components

The data file is expected to have two types of values: integer and real number columns. Column names are specified by the INTEGER and REAL commands. The names can contain any alphanumeric characters, i.e., letters and numbers. Many other characters such as the underscore (_) are also allowed. However, there are some reserved characters that are not allowed in names: =, (,), @, |, !, <, & and #. These characters have special meanings. For example, # starts a comment, and & marks a line continuation. Others are model component separators, and will be discussed later in this manual.

A statistical model is used to describe observations. Statistical model has effects. Data column names can be used as effect names in the model. If a data column name is an integer number column name, then it is assumed to be an effect with classes or factors. If a data column name is a real number column name, then it is assumed to be a regression effect. CLIM expects all effect names on a model line to be unique such that an effect name can appear only once on a model line. When many model effects refer to the same data column, component names can be used. See the repeatability model example (Chapters 6.2 and 6.2.1). Length of names have been restricted to 80 characters.

CLIM models have a predefined structure. In general, the model can have four types of

effects, and the effects have to be given on a model line by this order. The the order and types of effects are:

- Fixed regression effects
- Fixed classification effects
- Random effects other than additive genetics
- Random additive genetic effects

We will clarify these effect types in this manual elsewhere. In the following examples of this Chapter, we will consider models having fixed effects only. However, note that the distinction of a random effect to be an additive genetic effect depends on the covariance structure used, not so much on the effect being an additive genetic effect. In MiX99 , an additive genetic effect is associated with a pedigree-based relationship matrix giving a pedigree BLUP or a single-step GBLUP.

5.2 Model basics by linear model examples

We illustrate some key consepts of giving a model using linear models without random effects other than the residual. We use data in file named `LS.dat`:

ones$_1$	year1$_2$	year2$_3$	x$_1$	x2$_2$	y$_3$	y2$_4$
1	1999	2001	−1.0	1.00	14	1
1	1999	2002	−0.8	0.64	12	12
1	1999	2001	−0.6	0.36	10	3
1	1999	2002	−0.4	0.16	8	15
1	1999	2001	−0.2	0.04	9	7
1	1999	2002	0.0	0.00	10	19
1	2000	2001	0.2	0.04	21	11
1	2000	2002	0.4	0.16	23	23
1	2000	2001	0.6	0.36	26	15
1	2000	2002	0.8	0.64	30	27
1	2000	2001	1.0	1.00	34	19

where columns `year1` and `year2` are observation years, columns `y` and `y2` have observations of two traits. Columns `x` and `x2` are linear and quadratic functions, respectively, used as covariates. The first column `ones` can be used to include the general mean to a model.

This data file can be described to CLIM using commands:

```
DATAFILE ls.dat

INTEGER    ones year1 year2
REAL       x x2 y y2
MISSING    -99
```

Note the need to use the `MISSING` command to define missing `REAL` column variable to be -99 because a valid covariate of an observation has value zero. Default value for missing is zero.

5.2.1 Example: single trait regression with nesting

Consider a single trait model

$$y = year1 + x * s_l + x^2 * s_q + e$$

where

- year1 is year effect
- x is linear regression covariate
- s_l is unknown linear regression
- s_q is unknown quadratic regression
- e is the random residual.

The model terms cannot be given in the order presented because MiX99 expects that regression terms precede class effects. Thus, the model needs to be given as `y = x x2 year1`.

```
DATAFILE ls.dat

INTEGER   ones year1 year2
REAL      x x2 y y2
MISSING -99

MODEL
  y = x x2 year1
```

Assuming both regressions are also nested within year, the CLIM code is

```
DATAFILE ls.dat

INTEGER   ones year1 year2
REAL      x x2 y y2
MISSING -99

MODEL
  y = x x2 year1 curve(x x2| year1)
```

The term `curve` is not a reserved keyword but can be chosen to describe the nested components in the model. Naming certain model components can be a convenient way to organize and clarify model effects. For example, in this model we can group the non-nested regression effects under a common name to improve readability. In addition, the `year1` effect can be included in the nesting component:

```
  y = regression(x x2) curve(1 x x2| year1)
```

The estimates of this and the previous model line will be the same but the latter model line may be easier to read.

For this model with nested regression, three solution files are generated:

- `Solreg` has estimates for the linear and quadratic terms.
- `Solfix` has estimates for the nested regression terms, except the last one.
- `Solf01` has estimates of the last nested regression term.

The reason for splitting the nested regression estimates into two files is that MiX99 assigns a special status to the last effect. The last effect is treated as a within block effect, when `WITHINBLOCKORDER` command is not used to give other instructions. In a fixed effect model, all nested effects are uncorrelated leading to the last effect being a separate effect.

The `Solreg` file is

```
Trt Reg-No  Solution    Trait  Covariable
  1    1   3.3527       y    x
  1    2   5.2455       y    x2
```

The `Solfix` file having the `curve(1 x| year1)` model part is

```
Fact. Trt      Level    N-Obs  Solution              Factor Trait
  1    1       1999         6   9.8929                 year1 y
  2    1       1999         6   4.2723               x(year1) y
  1    1       2000         5   19.400                 year1 y
  2    1       2000         5   2.4330               x(year1) y
```

The `Solf01` file having `curve(x2|year1)` model part is

```
       1999         6   6.8080
       2000         5   3.6830
```

5.2.2 Example: two-trait regression with nesting

Consider extending the previous Chapter model to include a second trait to get a two-trait model. Assume that residuals of the two traits are uncorrelated. In this model, both traits share the same covariate values. Furthermore, assume that `year1` is the nesting variable for the curves of both traits. By default, each trait gets its own set of solution for all model effects.

CLIM code for a two-trait model is

```
DATAFILE ls.dat

INTEGER   ones year1 year2
REAL      x x2 y y2
MISSING -99

MODEL
   y  = regression(x x2) curve(1 x x2| year1)
   y2 = regression(x x2) curve(1 x x2| year1)
```

This model gives again three solution files.

The `Solreg` file is

```
Trt Reg-No  Solution    Trait  Covariable
  1    1   6.5937       y    x
  1    2   5.2455       y    x2
  2    1   12.809       y2    x
  2    2  -7.9241       y2    x2
```

The `Solfix` file having the `curve(1 x| year1)` and `curve(1 x| year1)` model parts is

```
Fact. Trt      Level    N-Obs  Solution              Factor Trait
  1    1       1999         6   9.8929                 year1 y
  1    2       1999         6   16.250                 year1 y2
  2    1       1999         6   1.0312               x(year1) y
  2    2       1999         6   3.6376               x(year1) y2
  1    1       2000         5   19.400                 year1 y
  1    2       2000         5   3.0000                 year1 y2
  2    1       2000         5 -0.80804               x(year1) y
```

```
   2    2       2000          5    40.048                x(year1) y2
```

The `Solf01` file having `curve(x2|year1)` and `curve(x2|year1)` model parts is

```
       1999          6    6.8080        11.942
       2000          5    3.6830       -27.790
```

The first trait solutions are in column 3 and the second trait solutions are in column 4.

5.2.3 Example: different model effects by trait

We make the following two changes to the model of the previous Chapter:

- Assume a 50% residual correlation between the traits.
- Add a second nested regression for the second trait, nested by `year2`.

The first change requires defining a residual variance structure using the `PARFILE` command. In this case, we assume a residual variance of 1 and a correation of 0.5. The second change requires adding an additional nested regression term for the second trait. Because the first trait does not have this regression, it must be explictly defined as missing for that trait.

CLIM code becomes

```
DATAFILE ls.dat

INTEGER   ones year1 year2
REAL      x x2 y y2
MISSING -99

PARFILE CLIM
1 LOWER
  1
  .5 1

MODEL
  y  = regression(x x2) curve1(1 x x2| year1) curve2(- - -)
  y2 = regression(x x2) curve1(1 x x2| year1) curve2(1 x x2| year2)
```

This model gives again three solution files.

The `Solreg` file is

```
 Trt Reg-No  Solution     Trait  Covariable
   1    1    5.9545        y    x
   1    2    15.003        y    x2
   2    1    4.1136        y2    x
   2    2    8.1043        y2    x2
```

The `Solfix` file has the `curve1(1 x x2| year1)`, `curve1(1 x x2| year1)`, and `curve2(1 x| year2)` model part solutions:

```
 Fact. Trt       Level     N-Obs  Solution              Factor Trait
   1    1        1999          6   9.8929                 year1 y
   1    2        1999          6   11.328                 year1 y2
   2    1        1999          6   1.6705               x(year1) y
   2    2        1999          6   6.2784               x(year1) y2
   3    1        1999          6  -2.9490              x2(year1) y
   3    2        1999          6  -5.2005              x2(year1) y2
```

```
  1   1       2000          5   19.400                  year1 y
  1   2       2000          5   11.319                  year1 y2
  2   1       2000          5 -0.16883               x(year1) y
  2   2       2000          5   5.1949               x(year1) y2
  3   1       2000          5  -6.0740              x2(year1) y
  3   2       2000          5  -9.1532              x2(year1) y2
  4   2       2001          6  -2.4818                  year2 y2
  5   2       2001          6   1.1802               x(year2) y2
  4   2       2002          5   7.8259                  year2 y2
  5   2       2002          5   1.0596               x(year2) y2
```

The `Solf01` file has the `curve2(x2|year2)` model part solutions:

```
      2001          6  0.81246
      2002          5  0.58657E-01
```

6 Single trait mixed effect models

Single trait mixed models have only one model line and one residual variance. However, there are exceptions to these rules through the use of trait groups and heterogeneous residual variances. In the following we will present different kinds of mixed effect models where the additive genetic effects have a pedigree-based relationship matrix. Many of the same principles apply to multiple trait models (Chapter 7) and genomic data models (Chapter 9).

6.1 Pedigree BLUP

Consider a simple individual animal model

$$trait1 = herd \times year + a + e$$

where

- herd $\times$ year is fixed the $herd$ times $year$ interaction effect
- a is the random additive genetic effect
- e is the random residual.

CLIM does not support multiplication operations between class effects or covariates in the model line. For example, the $herd \times year$ interaction has to be coded in an integer column of the data file as a class effect. Below this interaction is in column **herdXyear**. However, class effect interactions allow within class effect regressions such as random regression models.

Complete CLIM instruction file is (named `amodel.clm`)

```
DATAFILE  example.dat
INTEGER   IDcode sire herdXyear ones
REAL      trait1 trait2

PEDFILE   my.ped
PEDIGREE  IDcode am

DATASORT PEDIGREECODE=IDcode

PARFILE  ST.var
```

```
MODEL
  trait1 = herdXyear IDcode
```

The data file example.dat and the pedigree file my.ped are the same as given earlier (Chapters 3.1.1 and 3.3.1). The variance components file (ST.var) is for the first trait (trait1):

Random effect$_1$	Row$_2$	Column$_3$	Variance$_1$
1	1	1	3.0
2	1	1	7.0

In order to solve the model, two steps are needed. First, the preprocessor is executed: mix99i amodel.clm. Second, the solver is executed: mix99s -s. The solver will produce solution files Solfix having fixed effects, and Solani having the breeding values. The Solfix file is

```
 Fact.  Trt       Level     N-Obs   Solution              Factor Trait
   1    1           1          2    99.538              herdXyear trait1
   1    1           2          3    122.69              herdXyear trait1
```

The Solani file is (column names have been added)

```
     IDcode N-Desc  N-Obs Solution
          1      2      0  0.18516E-13
          2      2      0  0.18516E-13
          3      2      0  0.92308
          4      2      1 -0.92308
          5      3      0  0.42454E-13
          6      3      1   1.8462
          7      1      0  0.65421
          8      1      1  0.64447E-01
          9      0      1   2.0506
         10      0      1 -0.17840
```

Solutions may be slightly different due to computing precision when the example is tested on another computer. For example, the solutions close to zero are likely to be different (breeding values for individuals 1, 2, and 5).

6.1.1 Example: Unknown parent groups in a pedigree

Unknown parent groups (Chapter 3.3.2) are as easy to give in CLIM. The groups can be given as coded unknown parents in the pedigree file where a group is indicated by giving negative number as a parent code instead of zero for an unknown parent. Unknown parent groups are used, when the PEDIGREE command has '+p'. For example, the previous CLIM instruction file needs only one change:

```
PEDIGREE  IDcode am+p
```

in order to have fixed UPGs. Note that no space is allowed between the characters in am+p.

Unknown parent groups are defined to be random by including a genetic (co)variance matrix multiplier after am+p. For example,

```
PEDIGREE  IDcode am+p 0.5
```

The multiplier, here 0.5, is in the inverse scale. Thus, the (co)variance matrix for the random UPGs is now 2.0 times the additive genetic (co)variance matrix.

Notes:

- if the pedigree has negative parent numbers, and the model instruction file does not have '+p' then all negative parent numbers are considered to indicate an unknown parent and are effectively the same as zero.
- if '+p' is given but parent number of an individual is zero (0) (instead of a negative number), this parent assigned to group code -99999999.

6.1.2 Example: Inbreeding and the pedigree relationship matrix

The pedigree-based additive relationship matrix does not account for non-zero inbreeding coefficients by default. However, it is possible to use pre-calculated inbreeding coefficients in the additive relationship matrix. Because the processor does not calculate inbreeding coefficients, a separate program such as RelaX2 (Strandén and Vuori, 2006) has to be used.

Consider the example in Chapter 6.1 but now account the inbreeding coefficients in the relationship matrix computations. Inbreeding coefficients calculated using RelaX2 are (file `my.inbr`):

```
 1  1 0.00000
 2  2 0.00000
 3  3 0.00000
 4  4 0.00000
 5  5 0.25000
 6  6 0.25000
 7  7 0.37500
 8  8 0.37500
 9  9 0.37500
10 10 0.50000
```

In this file, the first column has the ID codes, the second has renumbered ID codes, and the last column has the inbreeding coefficients. In order for CLIM to use this file, two additional lines are needed:

```
INBRFILE    my.inbr
INBREEDING PEDIGREECODE=1 FINBR=3
```

According to the `INBRFILE` command, the inbreeding coefficients are in the `my.inbr` file. The `INBREEDING` command specifies that the ID codes are in the first column (`PEDIGREECODE=1`), and the inbreeding coefficients are in the third column (`FINBR=3`) of the `INBRFILE` `my.inbr`.

The use of inbreeding coefficients in the relationship matrix changes the solutions slightly. Fixed effect solutions in the `Solfix` file are

```
 Fact.  Trt       Level     N-Obs   Solution              Factor Trait
   1    1           1           2    99.538           herdXyear trait1
   1    1           2           3    122.61           locationXyear trait1
```

Breeding value estimates in the `Solani` file are

```
     IDcode N-Desc   N-Obs Solution
          1      2       0 -0.79627E-16
```

```
         2       2       0 -0.79627E-16
         3       2       0  0.92308
         4       2       1 -0.92308
         5       3       0  0.11618E-13
         6       3       1   1.8462
         7       1       0  0.70457
         8       1       1  0.24590
         9       0       1   1.8188
        10       0       1  0.11106
```

6.1.3 Example: Weighting residuals in a model

Weight can be used to indicate that an observation is either a mean from many records or that observations have varying accuracy. A weight of an observation divides the common residual variance for that observation. Thus, weights can be used to achive heterogeneous residual variances.

In the data file, weight is a real number. In CLIM, the `WEIGHT` option in the model line specifies the weight used for a trait. Model options for a trait follow the "`!`" sign. For example, when the data file column 'weight' contains weights, the option to use them for a trait is "`! WEIGHT=weight`" at the end of the model line.

Consider the previous example (Chapter 6.1.2) again but with weights. However, first a least squares model was used to correct for the environment and weights were computed. The new data file (`example_w.dat`) is:

ID code$_1$	sire$_2$	herd×year$_3$	ones$_4$	trait 1$_1$	trait 2$_2$	yc$_3$	weight$_4$
4	1	1	1	90	200	−10.0	1.00
6	3	1	1	110	190	−10.0	1.00
8	5	2	1	120	140	−3.3	0.95
9	5	2	1	130	120	6.7	0.95
10	7	2	1	120	130	−3.3	0.95

where columns `yc` and `weight` have the corrected observations and weights, respectively. Because precorrected observations are used, the only fixed effect is the general mean as defined by column `ones`.

The CLIM code is

```
DATAFILE    example_w.dat

INTEGER     IDcode sire herdXyear ones
REAL        trait1 trait2 yc weight
MISSING     -999

PEDFILE     my.ped
PEDIGREE    IDcode am

INBRFILE    my.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE     ST.var

MODEL
  yc = ones IDcode ! WEIGHT=weight
```

In a single trait model, the use of weights is equivalent to using observation specific residual variance which is computed as the residual variance divided by the weight.

Fixed effect solutions (`Solfix`) are

```
 Fact. Trt       Level     N-Obs  Solution              Factor Trait
   1   1           1           5 -0.58673                 ones yc
```

Breeding value estimates in the `Solani` file are

```
          1       2       0  0.63451E-16
          2       2       0  0.63451E-16
          3       2       0  0.91096
          4       2       1 -0.91096
          5       3       0 -0.25403E-01
          6       3       1   1.8473
          7       1       0  0.69541
          8       1       1  0.24391
          9       0       1   1.7555
         10       0       1  0.11040
```

6.2 Repeatability model

Repeatability model has usually two effects with the same incidence matrix but different covariance structures. In each observation of an individual, the permanent environment describes the non-genetic part, and the additive genetic effect has the heritable part. Thus, in the model line, the same integer number column in the data file is referred to by two different effects: permanent environment and direct genetics. However, because the same name cannot appear twice on the model line, component names must be used. Component names are user defined (renamed) names of one or more components in the model line. Basically, any classification effect can be renamed. For example, there is a column `IDcode` but we want to rename this effect to be `individual`. This is achieved by giving `individual(IDcode)` on the model line.

Consider a repeatability model

$$y = herd \times year + p + a + e$$

where $herd \times year$ is fixed $herd$ times $year$ interaction effect, p is random permanent environment effect, a is the additive genetic effect, and e is the random residual. Both $\boldsymbol{p}$ and $\boldsymbol{a}$ have the same design matrix relating observations to individuals. However, they have different covariance structures. The usual repeatability model assumptions are

$$\begin{array}{rclcrcl} \mathrm{E}(\boldsymbol{p}) & = & \boldsymbol{0} & & \mathrm{Var}(\boldsymbol{p}) & = & \boldsymbol{I}\sigma_p^2 \\ \mathrm{E}(\boldsymbol{a}) & = & \boldsymbol{0} & & \mathrm{Var}(\boldsymbol{a}) & = & \boldsymbol{A}\sigma_a^2 \\ \mathrm{E}(\boldsymbol{e}) & = & \boldsymbol{0} & & \mathrm{Var}(\boldsymbol{e}) & = & \boldsymbol{I}\sigma_e^2 \end{array}$$

where σ_p^2 is the permanent environment variance, σ_a^2 is the additive genetic variance, and σ_e^2 is the residual variance.

The model has two random effects which refer to the same class name, named `IDcode`, that is present in the data file. The following model statement is unacceptable (note that only commands relevant to the model line are given):

```
INTEGER   IDcode sire herdXyear ones
REAL      trait12

PEDIGREE  IDcode am
MODEL
```

```
  trait12 = herdXyear IDcode IDcode
```

because IDcode appears twice as an effect.

An alternative would be to have two identical columns with ID code numbers in both of them. Thus, our model line would be

```
INTEGER   IDcode pe_IDcode sire herdXyear ones
REAL      trait12

PEDIGREE  IDcode am # IDcode is for additive genetic
RANDOM    pe_IDcode # for permanent environment

MODEL
  trait12 = herdXyear pe_IDcode IDcode
```

This model is acceptable. However, now the data file is larger, and it is necessary to have two columns having the same content. Instead, a component name can be used to name the model effects.

The preferred way is to use component names when referring to the same integer column name. In the following, the name 'G' is given to the additive genetic effect, and 'PE' for the permanent environment. These names try to be descriptive and are not reserved names in CLIM. Alternatives are (only the relevant command lines are given):

1:

```
PEDIGREE   G am #  G for additive genetics
RANDOM     PE   # PE for permanent environment
MODEL
  trait12 = herdXyear PE(IDcode) G(IDcode)
```

2:

```
PEDIGREE   IDcode am # ID code for individual additive genetics
RANDOM     PE        # PE     for permanent environment
MODEL
  trait12 = herdXyear PE(IDcode) IDcode
```

3:

```
PEDIGREE  G am   # G for additive genetics
RANDOM    IDcode # permanent environment
MODEL
  trait12 = herdXyear IDcode G(IDcode)
```

Note that the effect name in the PEDIGREE command is always an additive genetic random effect and need not be present in the RANDOM command. Furthermore, it is assumed that the random effect number in the PARFILE for an additive genetic effect is one less than for the residual. These alternatives will produce the same instructions for mix99i. Chapter 6.4 will show that component names can be given to fixed effects as well.

6.2.1 Example: Repeatability model in detail

Let's use the two-trait data presented for a two-trait model (Chapter 3.1.1) and modify it for a single trait repeatability model. The model is

$$tr_{12} = herd \times year + p + a + e$$

where $herd \times year$ is the fixed herd-year effect, p is the random permanent environment effect, a is the random additive genetic effect, and e is the random residual.

Variance components are: permanent environment $\sigma_p^2 = 2.0$, genetic $\sigma_a^2 = 3.0$, and residual $\sigma_e^2 = 5.0$. The parameter file `RM.var` is

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	Comment
1	1	1	2.0	permanent environment
2	1	1	3.0	additive genetic
3	1	1	5.0	residual variance

The data file having two traits is modified for the repeatability model. The observations of both traits are giving its own record line and the same column for herd×year and observation contains data from both of the traits. The new data file:

ID code$_1$	sire$_2$	herd×year$_3$	ones$_4$	trait12$_1$
4	1	11	1	90
4	1	21	1	200
6	3	11	1	110
6	3	21	1	190
8	5	12	1	120
8	5	22	1	140
9	5	12	1	130
9	5	22	1	120
10	7	12	1	120
10	7	22	1	130

```
DATAFILE   example_repeat.dat
INTEGER    IDcode sire herdXyear ones
REAL       trait12

DATASORT PEDIGREECODE=IDcode

PEDFILE    my.ped
PEDIGREE   G am   #  G for additive genetics
RANDOM     PE     # PE for permanent environment

PARFILE    rep.var

MODEL
  trait12 = herdXyear PE(IDcode) G(IDcode)
```

The fixed effect solutions in the `Solfix` file are

```
Fact. Trt       Level     N-Obs   Solution         Factor Trait
 1    1          11            2   99.833         herdXyea trait12
 1    1          12            3   123.01         herdXyea trait12
 1    1          21            2   194.83         herdXyea trait12
 1    1          22            3   129.68         herdXyea trait12
```

Permanent environment solutions in the `Solr01` file are

```
   IDcode      N-Obs Solution
        4          2 -.88889
        6          2 0.88889
        8          2  1.1867
        9          2 -.55846
```

```
      10          2 -.62828
```

The breeding values estimates in the `Solani` file are

```
   IDcode N-Desc  N-Obs Solution
        1      2      0 -.58526E-06
        2      2      0 -.58526E-06
        3      2      0 0.33333
        4      2      2 -.33333
        5      3      0 -.81159E-06
        6      3      2 0.66667
        7      1      0 0.97735E-01
        8      1      2 0.98778
        9      0      2 -.85516E-01
       10      0      2 0.71556E-01
```

6.3 Sire model

In a sire model, records are associated with the sire of the individual having a record. In a typical sire model, sire has all its daughter's observations. Consequently, observation has only half of the genes to which the observation has been associated. Furthermore, the pedigree based relationship has often only sires.

In the following, we use the data introduced for animal model (Chapter 6.1) to illustrate a sire model.

6.3.1 Example: Simple sire model

Consider a simple sire model:

$$trait1 = herd \times year + s + e$$

where $herd \times year$ is the fixed herd-year effect, s is the random sire effect, and e is the random residual.

The data file is `example.dat`, the same as used in Chapter 3.1.1. In this sire model, all sires are assumed to be unrelated. Thus, the pedigree file (`SM.ped`) is

individual$_1$	sire$_2$	maternal grand sire$_3$
1	0	0
3	0	0
5	0	0
7	0	0

The variance components file needs to be changed. Sire genetics make up only a quarter of the total additive genetic variance. Thus, the variance components file (`SM.var`) is

Random effect$_1$	Row$_2$	Column$_3$	Variance$_1$
1	1	1	0.75
2	1	1	9.25

The CLIM code for the sire model is

```
DATAFILE   example.dat

INTEGER    IDcode sire herdXyear ones
REAL       trait1 trait2
```

```
PEDFILE    SM.ped
PEDIGREE   G sm

PARFILE    SM.var

MODEL
  trait1 = herdXyear G(sire)
```

The fixed effect solutions (Solfix) are

```
Fact. Trt       Level     N-Obs   Solution          Factor Trait
 1    1           1          2   100.00          herdXyea trait1
 1    1           2          3   123.25          herdXyea trait1
```

The breeding values estimates (Solani) are

```
     Bull N-Desc  N-Obs Solution
        1      0      1 -.75000
        3      0      1 0.75000
        5      0      2 0.24390
        7      0      1 -.24390
```

6.3.2 Example: Maternal grand sire model

The previous simple sire model example can also be analyzed by a sire model where a sire maternal grand sire relationship matrix is used. This allows taking into account relationships between sires. The command file does not change, but the pedigree file is different.

The pedigree file (smgms.ped) is

individual$_1$	sire$_2$	maternal grand sire$_3$
1	0	0
3	1	0
5	3	1
7	5	3

As before, the solver will produce the Solfix file having the fixed effect solutions

```
Fact. Trt       Level     N-Obs   Solution          Factor Trait
 1    1           1          2   99.976          herdXyea trait1
 1    1           2          3   123.19          herdXyea trait1
```

The Solani file has breeding values for the sires is

```
     Bull N-Desc  N-Obs Solution
        1      2      1 -.35736
        3      2      1 0.40621
        5      1      2 0.20334
        7      0      1 0.24076E-01
```

6.4 Fixed and random regressions with nesting

Fixed and random regression models can have regression effects nested within levels of a class variable. Typical random regression model in dairy cattle is test-day model where the lactation curve is fitted for test-day observations. Test-day models often have fixed regression curves nested within a class variable such as lactation number.

The class variable is an integer number column name in the data file. Nested regression models usually use component names (Chapter 5.1) in order to nest several regression

covariables within the same class variable. In the following, we extend the use of the component name as a class effect. Earlier we renamed a class effect such as the additive genetic effect `G(IDcode)`. The same notation will be used for nested regression effects. When the model has different kinds of regression effects, the order of the effects on the model line becomes important. We will cover this at the end of the next Chapter.

6.4.1 Nested regression effects

A regression effect can be nested within a class variable. This is similar to the component name concept introduced for the repeatability model. However, we can go even further and combine several effects into a component with the same nesting class variable. For example, assume a fixed cubic polynomial curve nested within a season. The model could be

```
y = fixed_curve(1 linear quadratic cubic | season) IDcode
```

where `season` and `IDcode` are integer number column names in the data file, and `linear`, `quadratic`, and `cubic` are real number column names in the data file. The number `1` above means intercept term, i.e., season effect, in this example. The '`fixed_curve`' is a component name with the common nesting class `season` applied to all its regression effects. Here `fixed_curve` defines a fixed regression effect. However, when the component name appears in the `RANDOM` command, then a random regression effect is assumed instead.

In an alternative equivalent model, the intercept in the `fixed_curve` has been moved to be a separate class effect. Therefore, the model line can also be written as:

```
y = season fixed_curve(linear quadratic cubic | season) IDcode
```

This moving of the `season` effect to a separate effect works for fixed effects. However, for random effects, this may not be feasible because component names are used to distinguish correlated random regression effects. See the examples below for random regression models.

Consider a more general model with a non-genetic random effect (`PE`) and an additive genetic effect (`G`). A regression without nesting can be given:

```
y = linear quadratic cubic season PE(IDcode) G(IDcode)
```

This and the previous models has some important things to consider:

Model structure: The model has four types of effects:

- Fixed regression effects: `linear`, `quadratic`, `cubic`
- Fixed classification effects: `(`season), fixed_curve(1 linear quadratic cubic | season)
- Random effects: `PE`
- Random additive genetic effect: `G`

The order of effects on the model lines must follow these types. Within each type, the order of effects is free.

Genetic effects: The additive genetic (here `G`) is linked to the pedigree by the `PEDIGREE` command, and must always be the last effect on the model line.

Following these rules, fixed effects with regression and nested regression can be given:

```
y = linear quadratic cubic &
    season fixed_curve(linear quadratic cubic | season) &
    PE(IDcode) &
    G(IDcode)
```

Note the four types of effects following each other as blocks in correct order.

6.4.2 Example: Random regression test-day model

We consider a single trait random regression model presented in Schaeffer and Dekkers (1994). This model is known also as random regression test-day model, where several observations are modeled throughout lactation using a linear regression function.

Cows have repeated observations of milk production. The data file includes milk yield and the time since calving, also known as days in milk (DIM). The model is

$$milk = DIM \cdot b_1 + log(305/DIM) \cdot b_2 + HTD + f(a, DIM) + e$$

where

$milk$	is milk yield observation,
DIM	is the days in milk linear regression covariable,
b_1	is fixed regression effect of DIM,
$log(305/DIM)$	is the logarithm of DIM regression covariable,
b_2	is fixed regression effect of the logarithm of DIM,
HTD	is the fixed herd test-day effect,
$f(a, DIM)$	is the random additive genetic regression function, and
e	is the random residual.

The random regression function f for individual i has form

$$f(\boldsymbol{a}, DIM) = a_{i,1} + DIM \cdot a_{i,2} + log(305/DIM) \cdot a_{i,3}$$

Thus, for each individual, three random regression coefficients are estimated which can be used to estimate breeding values.

Variance components are: residual variance $\sigma_e^2 = 100$, and random regression effect covariance matrix

$$\boldsymbol{G}_0 = \begin{bmatrix} 44.791 & -0.133 & 0.351 \\ -0.133 & 0.073 & -0.010 \\ 0.351 & -0.010 & 1.068 \end{bmatrix}$$

The parameter file `RRM.var` is

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	Comment
1	1	1	44.791	additive genetic: intercept
1	2	1	-0.133	intercept, DIM linear
1	3	1	0.351	intercept, ln(DIM/305)
1	2	2	0.073	DIM, DIM
1	2	3	-0.010	DIM, ln(DIM/305)
1	3	3	1.068	ln(DIM/305), ln(DIM/305)
2	1	1	100.000	residual variance

The parameter file can be given in the MIXED format as

```
1 LOWER
  44.791
  -0.1333 0.073
   0.351 -0.010 1.068
2 LOWER
  100.0
```

The pedigree and data files for the random regression model example:

pedigree file RRM.ped				data file RRM.dat					
individual$_1$	sire$_2$	dam$_3$	block$_4$	HTD$_1$	individual$_2$	block$_3$	DIM$_1$	ln(305/DIM)$_2$	milk$_3$
1	9	7	1	1	1	1	73.0	1.4298500	26.0
2	10	8	1	2	1	1	123.0	0.9081270	23.0
3	9	2	2	3	1	1	178.0	0.5385280	21.0
4	10	8	3	1	2	1	34.0	2.1939499	29.0
5	11	7	3	2	2	1	84.0	1.2894900	18.0
6	11	1	4	3	2	1	139.0	0.7858380	8.0
7	0	0	8	4	2	1	184.0	0.5053760	1.0
8	0	0	8	1	3	2	8.0	3.6408701	37.0
9	0	0	8	2	3	2	58.0	1.6598700	25.0
10	0	0	8	3	3	2	113.0	0.9929240	19.0
11	0	0	8	4	3	2	158.0	0.6577170	15.0
				5	3	2	218.0	0.3358170	11.0
				6	3	2	268.0	0.1293250	7.0
				2	4	3	5.0	4.1108699	44.0
				3	4	3	60.0	1.6259700	29.0
				4	4	3	105.0	1.0663500	22.0
				5	4	3	165.0	0.6143660	14.0
				6	4	3	215.0	0.3496740	8.0
				4	5	3	14.0	3.0812500	35.0
				5	5	3	74.0	1.4162500	23.0
				6	5	3	124.0	0.9000300	17.0
				5	6	4	31.0	2.2863200	28.0
				6	6	4	81.0	1.3258600	22.0

The CLIM code for the random regression model is

```
DATAFILE   RRM.dat
INTEGER    HTD IDcode block
REAL       DIM  ln305DIM  &   # Covariables
           milk               # Milk yield observation

PEDFILE    RRM.ped
PEDIGREE   G am

PARFILE    RRM.var

MODEL
  milk = Lact_curve(DIM ln305DIM) HTD G(1 DIM ln305DIM| IDcode)
```

The component name 'G' is used for the last effect on the model line with a pedigree structure. Thus, it has the additive genetic random regression effects. Note that the component name Lact_curve is informative for the user only, because it is used for fixed regression effects and there is no nesting. The name Lact_curve will remind

the user that these regression effects model the fixed lactation curve.

The fixed effect solutions (Solfix) are

```
Fact. Trt      Level    N-Obs  Solution         Factor Trait
  1   1          1          3  19.950              HTD milk
  1   1          2          4  20.373              HTD milk
  1   1          3          4  20.610              HTD milk
  1   1          4          4  19.728              HTD milk
  1   1          5          4  18.605              HTD milk
  1   1          6          4  17.852              HTD milk
```

The Lact_curve regression effect solutions (Solreg) are

```
Trt Reg-No  Solution    Trait  Covariable
  1    1 -.49839E-01   milk       DIM
  1    2  5.2910       milk       ln305DIM
```

The random regression effect estimates (Solani) for each individual are

```
   ID code N-Desc  N-Obs Intercept     DIM         ln305DIM
         1      1      3 -.44256     0.36869E-01 -.36961E-01
         2      1      4 0.26977     -.66036E-01 0.32266E-01
         3      0      6 -.72875     0.68317E-02 -.47899E-01
         4      0      5  1.1019     -.53652E-02 0.76755E-01
         5      0      3 -.16240     0.69360E-02 -.14924E-01
         6      0      2 -.48256     0.16641E-01 -.37788E-01
         7      2      0 -.98533E-01 0.13337E-01 -.10336E-01
         8      2      0 0.45724     -.23800E-01 0.36344E-01
         9      2      0 -.62847     0.35030E-01 -.47914E-01
        10      2      0 0.45724     -.23800E-01 0.36344E-01
        11      2      0 -.18720     -.76675E-03 -.14550E-01
```

6.4.3 Covariable table in a regression model

Regression models in dairy cattle breeding evaluations are typically so called test-day models, where repeated observations of a cow are modeled throughout lactation. The regression covariables are functions of days in milk (DIM) which take only specific values, such as integers from 1 to 350. An index in the data file can be used to indicate a row in the covariable table file, and its columns are specified in the model line. The use of a covariable file helps to reduce the size of the data file, as covariable files are often small due to the limited number of possible values for the index, like DIM. For example, if the data file contains DIM values ranging from 1 to 350 days, the covariable table file will have values in only 350 rows.

Assume the same fixed effect regression curve as given above. A covariable table file can be specified by the command TABLEFILE. The data file must include an integer index column, such as DIM. In our example, the file will have five columns: DIM, intercept (constant ones), linear (equals to DIM), quadratic (DIM^2), and cubic (DIM^3). The covariable table values are referenced by the letter *t* and a column number: t*n*. The model line can now be written as:

```
  y = fixed_curve(t1 t2 t3 t4 | season) IDcode
```

where $t1$, $t2$, $t3$, $t4$ refer to columns two, three, four, and five, respectively, in the covariable table file. Column one in the covariable table file contains the table index and is not considered an accessible column by the model. The input column in the data file is indicated by the command TABLEINDEX. See the example in Chapter 6.4.4.

Only one covariable table file and index is supported. The covariable table indices in the data file must be consecutive positive numbers greater than zero. For example, the covariable table file can have rows for indices 1, 2, 3, and 4, but having rows only for 1, 2, and 4 is unacceptable. The data file does not have to reference all indices in the covariable table and the indices do not need to start from one.

6.4.4 Example: Covariable table in test-day model

Consider again the single trait random regression test-day model example by Schaeffer and Dekkers (1994) (Chapter 6.4.2). We use the covariable table approach to reduce the number of columns in the data file. The idea is to use an index in the data file to specify which regression covariates are used.

In dairy cattle test-day models, a natural covariable table index is days in milk (DIM). However, for this example, an artificial index was used instead. This allowed the covariable table to contain only the necessary rows and remain small.

An artificial index was created to be consecutively numbered starting from one to the number of used test-days. This gave only 23 distinct covariate lines, and covariable indices numbered from 1 to 23. The covariable table file has this index in the first column, and its covariates on the same line.

The covariable table file (`RRM_table.cov`) is

index$_1$	DIM$_1$	log(305/DIM)$_2$
1	5	4.1108699
2	8	3.6408701
3	14	3.0812500
4	31	2.2863200
5	34	2.1939499
6	58	1.6598700
7	60	1.6259700
8	73	1.4298500
9	74	1.4162500
10	81	1.3258600
11	84	1.2894900
12	105	1.0663500
13	113	0.9929240
14	123	0.9081270
15	124	0.9000300
16	139	0.7858380
17	158	0.6577170
18	165	0.6143660
19	178	0.5385280
20	184	0.5053760
21	215	0.3496740
22	218	0.3358170
23	268	0.1293250

Use of a covariable table file leads to changes in data file where there is no longer the need to have the covariate values but instead the covariable index. The new data file (`RRM_table.dat`) is

HTD$_1$	individual$_2$	block$_3$	index$_4$	milk$_1$
1	1	1	8	26.0

```
  2       1       1     14   23.0
  3       1       1     19   21.0
  1       2       1      5   29.0
  2       2       1     11   18.0
  3       2       1     16    8.0
  4       2       1     20    1.0
  1       3       2      2   37.0
  2       3       2      6   25.0
  3       3       2     13   19.0
  4       3       2     17   15.0
  5       3       2     22   11.0
  6       3       2     23    7.0
  2       4       3      1   44.0
  3       4       3      7   29.0
  4       4       3     12   22.0
  5       4       3     18   14.0
  6       4       3     21    8.0
  4       5       3      3   35.0
  5       5       3      9   23.0
  6       5       3     15   17.0
  5       6       4      4   28.0
  6       6       4     10   22.0
```

The CLIM instruction file is changed as well. The covariable table file has to be defined using commands `TABLEFILE` and `TABLEINDEX`. In addition, the covariates are referenced in the model using `t` and the covariate table column number.

The CLIM file is

```
DATAFILE   RRM_table.dat
INTEGER    HTD IDcode block index
REAL       milk

TABLEFILE  RRM_table.cov
TABLEINDEX index

PEDFILE    RRM.ped
PEDIGREE   G am

PARFILE    RRM.var

MODEL
  milk = Lact_curve(t1 t2) HTD G(1 t1 t2| IDcode)
```

The solution files will be the same. However, there is a small difference in the `Solreg` file. The file is now

```
 Trt Reg-No  Solution     Trait  Covariable
   1    1 -0.49839E-01    milk   T1
   1    2   5.2910        milk   T2
```

Thus, instead of the covariable names `DIM` and `ln305DIM`, there are the table covariable column names `T1` and `T2`.

CLIM range expansion Giving several successive covariable table columns can be shortened using CLIM range expansion (command line option `--usemacros` is

needed):

```
MODEL
  milk = Lact_curve(t1:2) HTD G(1 t1:2| IDcode)
```

A range expansion is a macro that has the form

```
  <identifier><N>:<M>
```

which replaces this syntax by a space separated list of

```
  <identifier><N>  <identifier><N+1>  ... <identifier><M-1>
<identifier><M>
```

The identifier can be a column name in the data file or a 't' character indicating column in a table file. Note that the range expansion macro will lead to expanding all expressions in a CLIM file that follow the above described form. Thus, it is recommended not to use the ':' character in a column or component name when no range expansion is wanted.

6.5 Example: Heterogeneous residual variances in a test-day model

Consider the random regression model example by Schaeffer and Dekkers (1994) (Chapter 6.4.4). However, let's now assume that the residual variance differs across blocks. There are four blocks, with residual variances of 100, 110, 105, and 90 in blocks 1, 2, 3, and 4, respectively.

Important commands in CLIM for the use of heterogeneous residual variance are `RESIDFILE` and `RESIDUAL`. Command `RESIDFILE` has the name of the residual variance file. Command `RESIDUAL` indicates the integer number column having the residual variance number in the data file.

Our example data stays the same. However, a heterogeneous residual variance file (`RRM_res.var`) is needed:

```
1  1  1  100.0
2  1  1  110.0
3  1  1  105.0
4  1  1   90.0
```

In a residual (co)variance file, the first column specifies the residual (co)variance block number. This number is referred by the integer number in the data file column defined by the `RESIDUAL` command. The second and third columns indicate the matrix position. In our example, the position is always (1,1) as we have a single trait model. The last column contains the value of the residual variance.

The CLIM instructions for the analysis are

```
DATAFILE   RRM_table.dat
INTEGER    HTD IDcode block index
REAL       milk

TABLEFILE  RRM_table.cov
TABLEINDEX index

PEDFILE    RRM.ped
PEDIGREE   G am
```

```
PARFILE     RRM.var      # regular variance file
RESIDFILE   RRM_res.var  # the residual variances
RESIDUAL    block        # index for residual variance

MODEL
  milk = Lact_curve(t1 t2) HTD G(1 t1 t2| IDcode)
```

Fixed effect solutions (`Solfix`) are

```
 Fact. Trt       Level     N-Obs   Solution               Factor Trait
   1    1           1         3    19.950                    HTD milk
   1    1           2         4    20.374                    HTD milk
   1    1           3         4    20.610                    HTD milk
   1    1           4         4    19.728                    HTD milk
   1    1           5         4    18.605                    HTD milk
   1    1           6         4    17.852                    HTD milk
```

Fixed regression effect solutions (`Solreg`) are

```
 Trt Reg-No  Solution     Trait  Covariable
   1    1 -0.49839E-01   milk   DIM
   1    2   5.2910       milk   ln305DIM
```

Random regression effect solutions (`Solani`) are

```
    ID code N-Desc  N-Obs Intercept     DIM         ln305DIM
          1      1      3 -0.44276      0.36871E-01 -0.36886E-01
          2      1      4  0.26983     -0.66036E-01  0.32231E-01
          3      0      6 -0.72869      0.68315E-02 -0.47906E-01
          4      0      5   1.1020     -0.53657E-02  0.76718E-01
          5      0      3 -0.16257      0.69346E-02 -0.14852E-01
          6      0      2 -0.48278      0.16646E-01 -0.37681E-01
          7      2      0 -0.98720E-01  0.13336E-01 -0.10283E-01
          8      2      0  0.45727     -0.23800E-01  0.36316E-01
          9      2      0 -0.62850      0.35026E-01 -0.47885E-01
         10      2      0  0.45727     -0.23800E-01  0.36316E-01
         11      2      0 -0.18730     -0.76072E-03 -0.14474E-01
```

6.6 Maternal genetic model

The random regression effect models allow quite flexible model descriptions. However, random regression effects have the same covariance structure, e.g., numerator relationship matrix. Random maternal and paternal effects with correlated genetic effects have different ID codes to refer to a covariance structure. Thus, nesting within a component is now by a different class variable. So, we have multiple correlated factors within the genetic effect.

A random effect may have multiple class effects. For example, the genetic component has both a maternal and a direct genetic effect. A component name is again needed.

A simple model with a fixed herd effect, random maternal and direct individual genetic effects is

```
PEDIGREE  G am
MODEL
          y = herd G(dam individual)
```

Although `G(dam individual)` looks similar to the random regression models, there is a notable difference. Here `dam` and `individual` are different class effects, not regression effects by the same class variable. This model specification requires a 2 by 2 genetic covariance matrix for the maternal and individual genetic effects.

Note that the maternal genetic model is different from the model

```
PEDIGREE   G am
RANDOM     dam
MODEL
           y = herd dam G(individual)
```

In this context, `dam` is a common dam environment effect on all of its progeny. This `dam` effect does not have a covariance structure by a relationship matrix, instead the common dam effects are assumed uncorrelated.

6.6.1 Example: Animal model for a maternal trait

Consider model

$$tr_1 = herd \times year + p_m + a_m + a_g + e$$

where $herd \times year$ is fixed herd-year effect, p_m is random common dam permanent environment effect, a_m is random additive maternal genetic effect, and a_g is random additive individual genetic effect, and e is random residual.

The variance components are: maternal permanent environment variance $\sigma^2_p = 1$, residual variance $\sigma^2_e = 7$, and genetic covariance matrix

$$\boldsymbol{G}_0 = \begin{bmatrix} 2.0 & 1.0 \\ 1.0 & 3.0 \end{bmatrix}$$

The parameter file `mat.var` is

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	Comment
1	1	1	1.0	maternal permanent env.
2	1	1	2.0	maternal genetic
2	1	2	1.0	cov(maternal, individual)
2	2	2	3.0	individual genetic
3	1	1	7.0	residual variance

The `MIXED` type parameter file (`mat_mix.var`) can be

```
1 LOWER
  1.0
2 LOWER
  2.0
  1.0 3.0
3 LOWER
  7.0
```

Thus, `PARFILE MIXED mat_mix.var`.

We use the previously introduced data (Chapter 3). The pedigree file (Chapter 3.3.1) can be kept the same. For the purposes of this example, we modify the data (Chapter 3.1.1) to have the `dam` column instead of the `sire` column (`example_mat.dat`):

individual$_1$	dam$_2$	herd×year$_3$	ones$_4$	trait 1$_1$	trait 2$_2$
4	2	1	1	90	200
6	4	1	1	110	190
8	6	2	1	120	140
9	6	2	1	130	120
10	8	2	1	120	130

The CLIM code is

```
DATAFILE   example_mat.dat

INTEGER    individual dam herdXyear ones
REAL       trait1 trait2

PEDFILE    my.ped
PEDIGREE   G  am
RANDOM     PE

PARFILE    mat.var

MODEL
  trait1 = herdXyear PE(dam) G(dam individual)
```

The herd-year solutions (`Solfix`) are

```
 Fact. Trt       Level     N-Obs   Solution          Factor Trait
   1   1           1           2   99.338          herdXyear trait1
   1   1           2           3   121.71          herdXyear trait1
```

The maternal non-genetic or permanent environment (PE) solutions (`Solr01`) are

```
    Dam code     N-Obs Solution
           2         1 -1.0095
           4         1  1.0095
           6         2 0.24832
           8         1 -.24832
```

The maternal (dam) and individual direct (individual) genetic effect solutions (`Solani`) are

```
     ID code   N-Desc N-Obs Maternal    Individual
           1       2      0   1.0095       0.50476
           2       2      1  -1.0095      -0.50476
           3       2      0  0.25237       0.75714
           4       2      1  0.75714      -0.25238
           5       3      0  0.38060       0.19030
           6       3      2   1.1337        1.8287
           7       1      0  0.69506       0.82328
           8       1      1  0.22383      0.30388E-01
           9       0      0   1.1042        2.0507
          10       0      0  0.33529      0.54356E-01
```

Note that the content of genetic effect solutions in the Solani file depends on the given order in the model line. In this example, we gave `G(dam individual)` with the maternal genetic effect first, and the direct genetic effect second. Changing the order of the effects changes the order in the solution file as well, and assumes a different order of variances in the parameter file (`mat.var`).

7 Multi-trait models

Each trait in a multiple trait model is defined by a separate model line, with traits numbered sequentially so that the trait number corresponds to the model line number. For example, the first model line represents trait number 1, the second represents trait number 2, and so on. This numbering system is crucial when creating a covariance matrix in the `PARFILE`.

Numbering variance components can sometimes be challenging, especially when certain traits lack specific effects (such as random regression or maternal effects) that are present in other traits. These missing effects are indicated by a minus ($-$) sign in the model lines.

In the following sections, we will focus on individual animal models, although the principles apply similarly to sire models.

7.1 Basic multi-trait model

A simple multi-trait model has several traits that are equal in the sense of having the same effects. For example, both traits have herd-year effects and additive genetic effects. These effects have different solutions by trait. However, the important fact is that both traits refer to the same classification column in the data file.

7.1.1 Example: Simple multi-trait model

Consider the multi-trait model data presented in Chapter 3.1.1. First, consider a simple model where both traits have the same effects:

$$tr_1 = herd \times year_1 + a_1 + e_1$$
$$tr_2 = herd \times year_2 + a_2 + e_2$$

where subscripts 1 and 2 refer to traits 1 and 2. The variance components file (name `mt.var`) was already presented in Chapter 3.4.1. CLIM instruction file is:

```
DATAFILE   example.dat

INTEGER    IDcode sire herdXyear ones
REAL       trait1 trait2

PEDFILE    my.ped
PEDIGREE   IDcode am
DATASORT   PEDIGREECODE=IDcode

PARFILE    mt.var

MODEL
  trait1 = herdXyear IDcode
  trait2 = herdXyear IDcode
```

The fixed effect solutions (`Solfix`) are

```
 Fact. Trt      Level    N-Obs  Solution              Factor Trait
   1   1          1          2   99.760             herdXyear trait1
   1   2          1          2   194.87             herdXyear trait2
   1   1          2          3   122.93             herdXyear trait1
   1   2          2          3   129.73             herdXyear trait2
```

The breeding value estimates (Solani) are

```
    ID code N-Desc  N-Obs    trait1        trait2
          1      2      0 -0.13665E-05 -0.68995E-05
          2      2      0 -0.13665E-05 -0.68995E-05
          3      2      0  0.47969      0.26920
          4      2      1 -0.47969     -0.26922
          5      3      0 -0.35700E-05 -0.64338E-06
          6      3      1  0.95938      0.53842
          7      1      0  0.23058      0.78158E-01
          8      1      1  0.77386      0.86993
          9      0      1  0.43462     -0.14044
         10      0      1  0.40004E-02  0.91922E-01
```

7.2 Including trait-specific effects

MiX99 supports multi-trait models with different effects. However, when using CLIM, there are some important things to consider. By default, CLIM model line works similarly to the MiX99 directive file, meaning model effects are column-restricted. In a model line, each effect has a column which is either present (indicated by a model name) or missing (indicated by a minus sign) for each trait. Each column (or a set of adjacent columns) is reserved to one effect only. Therefore, it is crucial to specify effects in a consistent and correct order. The beta testing version of CLIM removes this restriction, allowing model effects to be specified without the minus sign indicator for missing effects. However, when using the beta testing version of CLIM (argument -b), there may be cases where models are interpreted incorrectly. It is essential to check that the model generated by CLIM is correct in the MiX99_DIR.DIR file.

7.2.1 Example: Multi-trait model with missing effects

Consider the multi-trait model data as in Chapter 7.1.1 but use model

$$tr_1 = herd \times year + a_1 + e_1$$
$$tr_2 = \mu + a_2 + e_2$$

where the traits refer to different fixed effect columns. We use the same variance components file as before. CLIM instruction file is the same except for the model lines:

```
MODEL
  trait1 = -    herdXyear IDcode
  trait2 = ones -         IDcode
```

The fixed effect solutions (Solfix) are

```
 Fact. Trt      Level     N-Obs   Solution              Factor Trait
   1   2          1          5    160.28                   ones trait2
   2   1          1          2    88.759                herdXyear trait1
   2   1          2          3    137.73                herdXyear trait1
```

Breeding value estimates in the Solani file are

```
    ID code N-Desc  N-Obs    trait1        trait2
          1      2      0  0.14293E-05  0.60920E-06
          2      2      0  0.14293E-05  0.60920E-06
          3      2      0 -2.2842      -2.5112
          4      2      1  2.2842       2.5112
          5      3      0 -5.8969      -5.8969
          6      3      1  1.3284       0.87448
```

```
          7       1       0   -4.2747       -4.4635
          8       1       1   -7.6804       -7.6189
          9       0       1   -6.6913       -7.2448
         10       0       1   -9.9586       -9.9459
```

7.2.2 Example: Different effects by trait using CLIM beta features

The order of effects is unimportant in the CLIM beta version. The model lines in the example above in Chapter 7.2.1 can be given differently using the CLIM beta version (e.g., giving `mix99i -b model.clm`). A natural way of giving would be

```
MODEL
  trait1 =      herdXyear IDcode
  trait2 = ones           IDcode
```

The additional space between the effect names is not important for CLIM, it is just to make the model easier to read. Another acceptable model would be

```
MODEL
  trait1 =      IDcode herdXyear
  trait2 = ones IDcode
```

which does not look easier interpret as `IDcode` seems misordered.

The breeding value estimates in the `Solani` solution file would be the same as before. However, solutions in the `Solfix` file are printed in different order:

```
 Fact. Trt       Level     N-Obs   Solution              Factor Trait
   1   1           1         2     88.759              herdXyear trait1
   1   1           2         3     137.73              herdXyear trait1
   2   2           1         5     160.28                   ones trait2
```

The reason for different ordering is the `-b` option. The `-b` option leads to ordering by column number in the data file. Column `herdXyear` is before the `ones` in the data file. The difference can also be seen in the `MiX99_DIR.DIR` file.

7.3 Multi-trait random regression model

Multiple trait random regression models are an extension of single trait models. Multiple traits gives several model lines which requires careful numbering of the variances of the random regression effecst. The numbering proceeds column-wise, starting with the first regression effect of the first trait, then the first regression effect of the second trait, continuing across all traits. Next, the second regression effect of the first trait, followed by the second regression effect of the second trait, and so on. As a result, in multi-trait random regression models, the variances for the regression effects of the same trait are typically not consecutively numbered.

Consider the quadratic random regression function of an individual for two traits:

$$\begin{bmatrix} f(\boldsymbol{a}_1, x_1) \\ f(\boldsymbol{a}_2, x_2) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \boldsymbol{a}_3 \end{bmatrix}$$

where subscripts 1 and 2 refer to trait number, x_1 and x_2 are covariates, and $\boldsymbol{a}$ has random regression coefficients to be estimated. The functions can be written also

$$f(\boldsymbol{a}, x_1) = a_{1,1} + x_1 \cdot a_{1,2} + x_1^2 \cdot a_{1,3}$$
$$f(\boldsymbol{a}, x_2) = a_{2,1} + x_2 \cdot a_{2,2} + x_2^2 \cdot a_{2,3}$$

where the first subscript in $\boldsymbol{a}$ is trait number, and the second is random regression effect number. The random regression effects are numbered comlumn-wise

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} = \begin{bmatrix} (1) & (3) & (5) \\ (2) & (4) & (6) \end{bmatrix}$$

where the numbers in parenthesis mean effect numbers.

The random regression effect numbers are used to identify variance components in the `PARFILE`. They also determine the order of estimates in the solution files such as `Solani`.

7.3.1 Example: Multi-trait test-day model

Consider the single trait random regression test-day model data presented in Chapter 6.4.2. We expand the data by adding a column of observations for a second trait. The observation column of the single trait is copied to this new column.

The two traits will have the same random regression function. Assume that within a trait the genetic covariance matrix stays the same, and between the traits, the correlation is 95 %. Remember that the numbering of regression effects is column-wise. Thus, the genetic covariance matrix is

$$\boldsymbol{G}_0 = \begin{bmatrix} 44.791 & 42.55145 & -0.133 & -0.12635 & 0.351 & 0.33345 \\ 42.55145 & 44.791 & -0.12635 & -0.133 & 0.33345 & 0.351 \\ -0.133 & -0.12635 & 0.073 & 0.06935 & -0.010 & -0.0095 \\ -0.12635 & -0.133 & 0.06935 & 0.073 & -0.0095 & -0.010 \\ 0.351 & 0.33345 & -0.010 & -0.0095 & 1.068 & 1.0146 \\ 0.33345 & 0.351 & -0.0095 & -0.010 & 1.0146 & 1.068 \end{bmatrix}$$

and let the residual covariance matrix be

$$\boldsymbol{R}_0 = \begin{bmatrix} 100.0 & 50.0 \\ 50.0 & 100.0 \end{bmatrix}$$

Then, the variance parameter file (`RRM_mt.var`) is

```
1  1  1    44.791
1  2  2    44.791
1  1  2    42.55145
1  3  1    -0.133
1  4  2    -0.133
1  4  1    -0.12635
1  3  2    -0.12635
1  3  3     0.073
1  4  4     0.073
1  3  4     0.06935
1  3  5    -0.010
1  4  6    -0.010
1  3  6    -0.00950
1  4  5    -0.00950
1  5  1     0.351
1  6  2     0.351
1  5  2     0.33345
1  6  1     0.33345
1  5  5     1.068
```

```
 1  6  6     1.068
 1  5  6     1.01460
 2  1  1   100.0
 2  2  1    50.0
 2  2  2   100.0
```

This parameter file can be given using the MIXED format as

```
1 LOWER
  44.791
  42.55145  44.791
  -0.133    -0.12635  0.073
  -0.12635  -0.133    0.06935 0.073
   0.351     0.33345 -0.010  -0.0095 1.068
   0.33345   0.351   -0.0095 -0.010  1.0146 1.068
2 LOWER
  100
  50 100
```

The CLIM code is

```
DATAFILE   RRM_mt.dat
INTEGER    HTD IDcode block
REAL       DIM  ln305DIM milk_1 milk_2

PEDFILE    ../data/RRM.ped
PEDIGREE   G am

PARFILE    RRM_mt.var

MODEL
  milk_1 = Lact_curve(DIM ln305DIM) HTD G(1 DIM ln305DIM| IDcode)
  milk_2 = Lact_curve(DIM ln305DIM) HTD G(1 DIM ln305DIM| IDcode)
```

Fixed effect solutions (Solfix) are

```
Fact. Trt       Level     N-Obs   Solution              Factor Trait
  1   1           1           3    20.151                   HTD milk_1
  1   2           1           3    20.151                   HTD milk_2
  1   1           2           4    20.488                   HTD milk_1
  1   2           2           4    20.488                   HTD milk_2
  1   1           3           4    20.718                   HTD milk_1
  1   2           3           4    20.718                   HTD milk_2
  1   1           4           4    19.891                   HTD milk_1
  1   2           4           4    19.891                   HTD milk_2
  1   1           5           4    18.753                   HTD milk_1
  1   2           5           4    18.753                   HTD milk_2
  1   1           6           4    17.992                   HTD milk_1
  1   2           6           4    17.992                   HTD milk_2
```

Fixed regression effects (Solreg) are

```
Trt Reg-No  Solution    Trait  Covariable
  1    1 -0.50424E-01   milk_1   DIM
  1    2   5.2366       milk_1   ln305DIM
  2    1 -0.50424E-01   milk_2   DIM
  2    2   5.2366       milk_2   ln305DIM
```

Random regression breeding values (Solani) are

```
ID        (1)      (2)       DIM(1)       DIM(2)     ln305DIM(1)
 1 1 3 -0.54418 -0.54418  0.37792E-01  0.37792E-01 -0.43454E-01 ...
 2 1 4  0.34400  0.34401 -0.67204E-01 -0.67204E-01  0.37635E-01 ...
 3 0 6 -0.85094 -0.85095  0.75663E-02  0.75663E-02 -0.54885E-01 ...
 4 0 5   1.2933  1.2933  -0.65258E-02 -0.65258E-02  0.88939E-01 ...
 5 0 3 -0.18603 -0.18603  0.70228E-02  0.70228E-02 -0.16765E-01 ...
 6 0 2 -0.57906 -0.57906  0.17727E-01  0.17727E-01 -0.44573E-01 ...
 7 2 0 -0.12304 -0.12304  0.13492E-01  0.13492E-01 -0.12031E-01 ...
 8 2 0  0.54577  0.54577 -0.24577E-01 -0.24577E-01  0.42191E-01 ...
 9 2 0 -0.75278 -0.75278  0.36106E-01  0.36106E-01 -0.55565E-01 ...
10 2 0  0.54577  0.54577 -0.24577E-01 -0.24577E-01  0.42191E-01 ...
11 2 0 -0.21570 -0.21570 -0.44588E-03 -0.44588E-03 -0.16793E-01 ...
```

The last column has been omitted here due to page width restrictions. It is identical to the second-to-last column. Numbers in parentheses indicate the trait number of the effect.

7.4 Combining of trait estimates

Combining trait estimates means forcing the effects of different traits to be the same. By default, it is assumed that effects in different traits will differ and get separate estimates. Combining trait estimates, or shortly combining of traits, allows for estimating the same solutions for effects across different traits. These can be fixed or random effects, although in practice, combining of traits is mostly used in the reduced rank random regression models. However, the concept can be illustrated by a multi-trait model that is equivalent to a repeatability model.

The combining of traits is indicated by the '`@`' sign in the model lines. After the '`@`' sign, a combining group name is provided. This name can be any valid name that has not already been used. Combining of traits can be specified for any effect that has a component name. Therefore, it is not possible to use integer number column names directly when combining traits. This issue can be resolved by assigning these effects a component name. For example, `G(IDcode)@fst`. The same applies to (fixed and random) regression effects. In particular, several effects can be included using the component name. For example, `curve(x logx sqrtx)@first` or `G(1 DIM DIM2|IDcode)@common`.

7.4.1 Example: Repeatability model by multi-trait model

The repeatability model is a multi-trait model where the genetic correlation between traits is one, residual variances are equal, and residual covariance values are equal. Residual correlations are equal to $\sigma_p^2/\left(\sigma_p^2+\sigma_e^2\right)$ where σ_p^2 is permanent environment variance and σ_e^2 is residual variance. We consider the repeatability model example presented in Chapter 6.2.1.

An equivalent two-trait model is

$$tr_1 = herd \times year + a + e_1$$
$$tr_2 = herd \times year + a + e_2$$

where the random additive genetic effect *a* can be considered common to both traits. Genetic variance is as before $\sigma_a^2 = 3$. Residual covariance matrix is

$$\boldsymbol{R}_0 = \begin{bmatrix} 7.0 & 2.0 \\ 2.0 & 7.0 \end{bmatrix}$$

Note that the residual variances are equal to the sum of the permanent environment and the residual variances in the repeatability model, and the covariance equals repeatability variance.

The variance components file (`mt_repeat.var`) is

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	Comment
1	1	1	3.0	individual genetic
2	1	1	7.0	residual variance
2	1	2	2.0	permanent environment
2	2	2	7.0	residual variance

The repeatability data file changed to multi-trait format (`mt_repeat.dat`):

individual	sire	herd×year$_1$	herd×year$_2$	ones	trait 1	trait 2
4	1	11	21	1	90	200
6	3	11	21	1	110	190
8	5	12	22	1	120	140
9	5	12	22	1	130	120
10	7	12	22	1	120	130

The CLIM code is

```
DATAFILE   mt_repeat.dat

INTEGER    IDcode sire hy_1 hy_2 ones
REAL       trait1 trait2

PEDFILE    my.ped
PEDIGREE   G am

PARFILE    mt_repeat.var

MODEL
  trait1 = hy_1 -    G(IDcode)@1
  trait2 = -    hy_2 G(IDcode)@1
```

Note that the additive genetic effect (`IDcode`) must have a name (`G` in the example) and assigned a combining group name or indicator ('1' in the example). Additionally, a minus sign (`-`) is required to indicate the use of separate integer columns for the herd-year effects. Note that the group name or indicator need not be a number but can have many letters. Thus, the following model lines would be correct as well:

```
  trait1 = hy_1 -    G(IDcode)@fst
  trait2 = -    hy_2 G(IDcode)@fst
```

Estimated herd-year solutions (`Solfix`) are

```
 Fact.  Trt       Level     N-Obs   Solution              Factor Trait
   1    1          11          2    99.833                   hy_1 trait1
   1    1          12          3    123.01                   hy_1 trait1
   2    2          21          2    194.83                   hy_2 trait2
   2    2          22          3    129.68                   hy_2 trait2
```

Estimated breeding values (`Solani`) are

```
   ID code N-Desc  N-Obs    trait12
         1      2      0 -0.18506E-14
         2      2      0 -0.18506E-14
         3      2      0  0.33333
         4      2      1 -0.33333
         5      3      0 -0.19257E-14
         6      3      1  0.66667
         7      1      0  0.97731E-01
         8      1      1  0.98778
         9      0      1 -0.85515E-01
        10      0      1  0.71553E-01
```

The solutions are the same as in the repeatability model in Chapter 6.2.1. However, no solutions for the permanent environment effects are calculated because no permanent environment effect was in this two-trait model.

7.4.2 Example: Reduced rank random regression model

Rank reduction can be used to make covariance matrices of highly correlated random regression coefficients more independent. Consequently, the size of the covariance matrix is reduced. In a reduced rank model, coefficients from two or more traits multiply the same solutions. Convergence of the iterative solver becomes faster for at least two reasons: equations become less correlated, and there are fewer unknowns to solve.

7.4.3 Example: Reduced rank two-trait model

Consider the two-trait random regression example presented in Chapter 7.3.1. We now assume that the genetic effects of the first and second traits have been combined, while the data remains unchanged. For this example, we use the same genetic covariance matrix as in the single trait example in Chapter 6.4.2:

$$\boldsymbol{G}_0 = \begin{bmatrix} 44.791 & -0.133 & 0.351 \\ -0.133 & 0.073 & -0.010 \\ 0.351 & -0.010 & 1.068 \end{bmatrix}$$

In this exmaple, the reduced rank (co)variance matrix has been derived from the original genetic (co)variance matrix. In practice, such a reduced rank multi-trait random regression (co)variance matrix simplifies the model by reducing the number of variance components (here from the full 6 by 6 matrix), typically based on criteria such as keeping the amount of (genetic) variance explained above some limit. This reduction uses eigendecomposition of the (co)variance matrix, from which the eigenvectors of the most important eigenvalues are selected. Because only a subset of eigenvectors are used, the values in the new (co)variance matrix may differ those in the original (co)variance matrix.

The residual covariance matrix will be the same as for the multi-trait model:

$$\boldsymbol{R}_0 = \begin{bmatrix} 100.0 & 50.0 \\ 50.0 & 100.0 \end{bmatrix}$$

The variance parameter file (`RRM_mt_RR.var`) is

```
1  1  1    44.791
1  2  1    -0.133
1  2  2     0.073
1  2  3    -0.010
```

```
  1   3   1      0.351
  1   3   3      1.068
  2   1   1    100.0
  2   2   1     50.0
  2   2   2    100.0
```

This can be given using the MIXED format as

```
1 LOWER
  44.791
  -0.133   0.073
   0.351 -0.010 1.068
2 LOWER
  100.0
   50.0 100.0
```

The CLIM code is

```
DATAFILE   RRM_mt.dat
INTEGER    HTD IDcode block
REAL       DIM ln305DIM milk_1 milk_2

PEDFILE     RRM.ped
PEDIGREE    G am

PARFILE     RRM_mt_RR.var

MODEL
  milk_1 = Lact_curve(DIM ln305DIM) HTD G(1 DIM ln305DIM| IDcode)@fst
  milk_2 = Lact_curve(DIM ln305DIM) HTD G(1 DIM ln305DIM| IDcode)@fst
```

Fixed effect solutions (Solfix) are

```
 Fact. Trt       Level     N-Obs   Solution                Factor Trait
   1   1           1           3    20.171                     HTD milk_1
   1   2           1           3    20.171                     HTD milk_2
   1   1           2           4    20.499                     HTD milk_1
   1   2           2           4    20.499                     HTD milk_2
   1   1           3           4    20.728                     HTD milk_1
   1   2           3           4    20.728                     HTD milk_2
   1   1           4           4    19.908                     HTD milk_1
   1   2           4           4    19.908                     HTD milk_2
   1   1           5           4    18.769                     HTD milk_1
   1   2           5           4    18.769                     HTD milk_2
   1   1           6           4    18.007                     HTD milk_1
   1   2           6           4    18.007                     HTD milk_2
```

Fixed regression effects (Solreg) are

```
 Trt Reg-No  Solution     Trait  Covariable
   1    1 -0.50486E-01   milk_1   DIM
   1    2   5.2312       milk_1   ln305DIM
   2    1 -0.50486E-01   milk_2   DIM
   2    2   5.2312       milk_2   ln305DIM
```

Random regression breeding values (Solani) are

```
    ID code N-Desc  N-Obs Intercept     DIM         ln305DIM
          1      1      3 -0.55454      0.37887E-01 -0.44107E-01
          2      1      4  0.35144     -0.67308E-01  0.38179E-01
```

```
          3      0      6 -0.86297      0.76386E-02 -0.55550E-01
          4      0      5   1.3125     -0.66443E-02  0.90125E-01
          5      0      3 -0.18819      0.70254E-02 -0.16933E-01
          6      0      2 -0.58904      0.17835E-01 -0.45274E-01
          7      2      0 -0.12543      0.13507E-01 -0.12184E-01
          8      2      0  0.55465     -0.24651E-01  0.42767E-01
          9      2      0 -0.76524      0.36213E-01 -0.56331E-01
         10      2      0  0.55465     -0.24651E-01  0.42767E-01
         11      2      0 -0.21860     -0.41763E-03 -0.17029E-01
```

Note that the number of estimates in the Solani file is the same as in the single trait example but estimates are different because more data is used.

7.4.4 Example: old Finnish test-day model

This example is not a complete presentation of the first Finnish test-day model. No data is given, and only the first lactation is considered. This illustrates the potential for solving complex test-day models currently used to solve dairy cattle breeding values.

This was the Finnish test-day model for first lactation milk, protein, and fat yield. The full model had two additional traits: the second lactation, and the third with later lactations. In this subset model, both the permanent environment and the additive genetic effects are modeled by a curve with six coefficients. However, covariance matrices of these effects have size six because all traits are combined into one.

```
DATAFILE    Ter.dat
INTEGER     block IDcode HTM YM SEASON AGE DCC DIM
REAL        milk protein fat

PEDFILE     miniTDM.pedi
PARFILE     TDMpara.in
PEDIGREE    G am+p
RANDOM      HTM PE

TABLEFILE   finTDMpara.cov
TABLEINDEX DIM

MODEL
  milk     = Curve(t1 t2 t3 t4 t96| SEASON) AGE DCC YM HTM  &
               PE(t5  t6  t7  t8  t9  t10| IDcode)@1st     &
                G(t59 t60 t61 t62 t63 t64| IDcode)@FST

  protein = Curve(t1 t2 t3 t95 t97| SEASON) AGE DCC YM HTM &
               PE(t11 t12 t13 t14 t15 t16| IDcode)@1st     &
                G(t65 t66 t67 t68 t69 t70| IDcode)@FST

  fat      = Curve(t1 t2 t3 t95 t98| SEASON) AGE DCC YM HTM &
               PE(t17 t18 t19 t20 t21 t22| IDcode)@1st     &
                G(t71 t72 t73 t74 t75 t76| IDcode)@FST
```

7.5 Multiple trait maternal effects model

Consider first a two-trait maternal effects model. In addition, some fixed effects are different by trait. Note that spaces between the effect names are only to help read the model lines.

```
DATAFILE   Beef.dat
INTEGER    BREED HERD id dam sex twin damXage &
```

```
          birthXmth HYbirth HY200d HY365d
REAL      BirthWeight 200dWeight 365dWeight age200d age365d

PEDFILE   Beef.ped
PEDIGREE  G am+p

PARFILE   Beef.var
RANDOM    PE G

MODEL
 BirthWeight =     -   twin sex     -    HYbirth -      PE(dam) G(dam id)
 200dWeight  = age200d twin sex birthXmth  -     HY200d PE(dam) G(dam id)
```

There are several aspects to consider:

Model structure: The model has four types of effects:

- Fixed regression effect: `age200d`
- Fixed classification effects: `twin`, `sex`, `birthXmth`, `HYbirth`, and `HY200d`
- Random effect: `PE`
- Random additive genetic effect: `G`

The order of effects on the model lines must follow these types. Within each type, the order of effects is free. However, each effect must consistently be on the same column on every model line. If an effect is missing in a particular trait line, it should be indicated with the minus sign (`-`).

Genetic effects: Both maternal and direct genetic effects are included in the same named class effect, here defined `G`. This effect (`G`) is linked to the pedigree by the `PEDIGREE` command, which refers to a pedigree-based relationship matrix (`am`).

(Co)variances: The (co)variances for the maternal and genetic effects are numbered column-wise from left to right, following the same convention as in the multi-trait random regression model. Thus, the `PARFILE Beef.var` is

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	Comment
1	1	1	1.0	non-genetic maternal variance of BirthWeight
1	2	1	0.1	non-genetic cov(BirthWeight, 200dWeight)
1	2	2	0.5	non-genetic maternal variance of 200dWeight
2	1	1	2.0	maternal genetic variance of BirthWeight
2	2	1	0.1	maternal cov(BirthWeight, 200dWeight)
2	2	2	0.5	maternal genetic variance of 200dWeight
2	3	1	1.0	cov(maternal, direct) of BirthWeight
2	3	2	0.1	cov(maternal 200dWeight, direct BirthWeight)
2	3	3	3.0	direct genetic variance of BirthWeight
2	4	1	0.0	cov(maternal BirthWeight, direct 200dWeight)
2	4	2	0.3	cov(maternal, direct) of 200dWeight
2	4	3	1.0	direct genetic cov(BirthWeight, 200dWeight)
2	4	4	3.0	direct genetic variance of 200dWeight
3	1	1	7.0	residual variance of BirthWeight
3	2	1	1.0	residual cov(BirthWeight, 200dWeight)

3	2	2	15.0	residual variance of 200dWeight

- Using the `MIXED` format, i.e., `PARFILE MIXED BeefMix.var`, the file can be

```
1 LOWER
  1.0
  0.1 0.5
2 LOWER
  2.0
  0.1 0.5
  1.0 0.1 3.0
  0.0 0.3 1.0 3.0
3 LOWER
  7.0
  1.0 10.0
```

We add 356-day weight to make a three-trait maternal effects model but assume that the last trait has only a direct genetic effect, i.e., no maternal genetic effect, and not non-genetic maternal effect.

```
DATAFILE   Beef.dat
INTEGER    BREED HERD id dam sex twin damXage &
           birthXmth HYbirth HY200d HY365d
REAL       BirthWeight 200dWeight 365dWeight age200d age365d

PEDFILE    Beef.ped
PEDIGREE   G am+p

PARFILE    Beef3traits.var
RANDOM     PE G

MODEL
 BirthWeight =     -  twin sex     -    HYbirth -  -    PE(dam) G(dam id)
 200dWeight = age200d twin sex birthXmth  - HY200d -    PE(dam) G(dam id)
 365dWeight = age365d twin sex birthXmth  -   - HY365d PE(-)   G( -  id)
```

The parameter file has now one more genetic variance with covariances to all the others, and a residual variance with covariances:

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	Comment
1	1	1	1.0	non-genetic maternal variance of BirthWeight
1	2	1	0.1	non-genetic cov(BirthWeight, 200dWeight)
1	2	2	0.5	non-genetic maternal variance of 200dWeight
2	1	1	2.0	maternal genetic variance of BirthWeight
2	2	1	0.1	maternal genetic covariance
2	2	2	0.5	maternal genetic variance of 200dWeight
2	3	1	1.0	cov(maternal, direct) of BirthWeight
2	3	2	0.1	cov(maternal 200dWeight, direct BirthWeight)
2	3	3	3.0	direct genetic of BirthWeight
2	4	1	0.0	cov(maternal BirthWeight, direct 200dWeight)
2	4	2	0.3	cov(maternal, direct) of 200dWeight
2	4	3	1.0	direct genetic cov(BirthWeight, 200dWeight)
2	4	4	3.0	direct genetic of 200dWeight
2	5	1	0.0	cov(maternal BirthWeight, direct 365dWeight)

2	5	2	0.1	cov(maternal 200dWeight, direct 365dWeight)
2	5	3	0.5	direct genetic cov(BirthWeight, 365dWeight)
2	5	4	2.0	direct genetic cov(200dWeight, 365dWeight)
2	5	5	3.0	direct genetic of 365dWeight
3	1	1	7.0	residual variance of BirthWeight
3	2	1	1.0	residual co-variance
3	2	2	10.0	residual variance of 200dWeight
3	3	1	0.5	residual cov(BirthWeight, 365dWeight)
3	3	2	1.0	residual co-variance
3	3	3	12.0	residual variance of 365dWeight

The MIXED format type of parameter file can be

```
1 LOWER
  1.0
  0.1 0.5
2 LOWER
  2.0
  0.1 0.5
  1.0 0.1 3.0
  0.0 0.3 1.0 3.0
  0.0 0.1 0.5 2.0 3.0
3 LOWER
  7.0
  1.0 10.0
  0.5  1.0 12.0
```

Currently, the beta testing option (-b) allows this model to be specified without most of the minus (-) characters. Because effects are ordered based on their column position, the resulting directive file may differ from the example above. However, despite the differences in the order of effects on the model line, the solutions from the analysis remain the same. It is important to note that a minus sign (-) is still required for the third trait (365dWeight) due to it being a random effect. This guarantees correct numbering of variance components as defined in the PARFILE command.

```
MODEL
 BirthWeight =          twin sex           HY_birth P(dam)  G(dam id)
 200dWeight  = age200d twin sex birthXmth HY_200d  P(dam)  G(dam id)
 365dWeight  = age365d twin sex birthXmth HY_365d  P(-)    G( -   id)
```

8 CLIM macros and range expansion

Model lines can become wide and difficult to follow. This is particularly true for multi-trait random regession models where several covariates are used for each trait. As already mentioned earlier, macro range expansion can help to reduce size of models and make easier to read (See paragraph 6.4.4). Macros can be used to make the models even shorter. They may also help reduce mistakes when writing models.

Consider the example model in Chapter 7.4.4:

```
MODEL
  milk     = Curve(t1 t2 t3 t4 t96| SEASON) AGE DCC YM HTM  &
                PE(t5  t6  t7  t8  t9  t10| IDcode)@1st     &
                 G(t59 t60 t61 t62 t63 t64| IDcode)@FST
```

```
  protein = Curve(t1 t2 t3 t95 t97| SEASON) AGE DCC YM HTM &
              PE(t11 t12 t13 t14 t15 t16| IDcode)@1st     &
               G(t65 t66 t67 t68 t69 t70| IDcode)@FST

  fat     = Curve(t1 t2 t3 t95 t98| SEASON) AGE DCC YM HTM &
              PE(t17 t18 t19 t20 t21 t22| IDcode)@1st     &
               G(t71 t72 t73 t74 t75 t76| IDcode)@FST
```

These model lines can be shortened using CLIM macro and range abbreviations (command line option `--usemacros` is currently needed when invoking `mix99i`):

```
DEFINE CurveMILK   Curve(t1:3 t4  t96 | SEASON)
DEFINE CurvePROT   Curve(t1:3 t95 t97 | SEASON)
DEFINE CurveFAT    Curve(t1:3 t95 t98 | SEASON)
DEFINE Common      AGE DCC YM HTM

MODEL
  milk    = CurveMILK Common PE( t5:10|IDcode)@1st G(t59:64|IDcode)@FST
  protein = CurvePROT Common PE(t11:16|IDcode)@1st G(t65:70|IDcode)@FST
  fat     = CurveFAT  Common PE(t17:22|IDcode)@1st G(t71:76|IDcode)@FST
```

Macros are user-defined names that represent character strings. When a macro is referenced, its name is replaced by the corresponding character strings. Important considerations:

- Macro replacement is done to all defined macro names after their definition. Thus, it is recommended to give macros just before the `MODEL` command.
- A macro name must not conflict with any predefined CLIM keywords (e.g., `MODEL`) or data column name (e.g., `IDcode` in the example above).
- It is recommended to use macro names with multiple characters to avoid unintended replacements of short names.
- Macros can be nested, i.e., a macro can include other already defined macros.

The following example shows how to use macros to have all fixed effects by macro nesting:

```
DEFINE Common      AGE DCC YM HTM
DEFINE FixedMILK   Curve(t1:3 t4  t96 | SEASON) Common
DEFINE FixedPROT   Curve(t1:3 t95 t97 | SEASON) Common
DEFINE FixedFAT    Curve(t1:3 t95 t98 | SEASON) Common

MODEL
  milk    = FixedMILK PE( t5:10|IDcode)@1st G(t59:64|IDcode)@FST
  protein = FixedPROT PE(t11:16|IDcode)@1st G(t65:70|IDcode)@FST
  fat     = FixedFAT  PE(t17:22|IDcode)@1st G(t71:76|IDcode)@FST
```

In this example, the macro named "Common" is defined first and then reused across three subsequent macros: `FixedMILK`, `FixedPROT`, and `FixedFAT`. These macros are later used in the model lines, simplifying the model. Note that the macro names do not need to be the across traits, as is the case here. However, the contents of each macro must follow the CLIM model rules of the same number of effects and correct column allignment.

9 Genomic data models

There are two widely used approches for incorporating genomic data into statistical models in animal and plant breeding. The first approach includes genomic marker effects directly into the model. The second approach uses genomic data to construct a (co)variance structure, such as a genomic relationship matrix, which is then used to model the (co)variance structure of breeding values. Both types of models are supported in CLIM. The model that includes genomic marker effects is known as the SNP-BLUP model. Models that utilize genomic relationship matrix include GBLUP and the single-step method.

9.1 Genomic marker effects model

Statistical models used in genomic selection often include thousands of SNP markers. In the SNP-BLUP model, each marker is treated as a regression effect. Including genomic data within a data file with the observations and other effects would produce a large file containing both genomic and non-genomic information. Although CLIM macro range expansion can help in writing the model, separating genomic data to different file offers some benefits. For example, it is easy to specify genomic data specific options. CLIM supports the use of covariate matrices, which can have different types of data. These matrices are defined using the `REGMATRIX` and `REGFILE` commands.

Regression covariates or marker genotypes of the regression matrix can be of three types:

Fixed by command `REGMATRIX FIXED`.

Random with common (co)variance by command `REGMATRIX RANDOM`.

Heterogeneous with marker specific (co)variance by command `REGMATRIX HETEROGENEOUS`.

Relevant commands for regression matrices are:

- `REGMATRIX` for defining the type of the matrix, coefficient columns, and some other options.
- `REGFILE` for the name of the file having the marker genotypes.
- `REGPARFILE` for (co)variance component(s).

For syntax and some explanation, please see Chapter 12.2.

Each line in a `REGMATRIX` file, as defined by the `REGFILE` command, contains (marker) values to a single individual. By default, numbers on each line are:

- space separated.
- numbers with any value (also floating point values allowed).
- correspond to a line in the data file specified by the `DATAFILE` command.

There are commands and options that change these defaults.

The correct order of lines in the `REGMATRIX` and data files is crucial. Particularly, if `ID` or `REGINDEX` options are not used. When the `ID` option is used in `REGMATRIX`, the preprocessor verifies that the order of lines in the data and marker files are alligned

correctly by the ID codes. This reduces the risk of errors in genetic evaluations that could arise from mismatch of lines in these files. Furthermore, if the marker file (defined by REGFILE) has individuals not present in the data file, these will be skipped.

Individual ID codes can be located in different columns in the REGFILE and DATAFILE files. In the REGFILE, the column of the individual ID code is specified using the ID option in the REGMATRIX command. For example,

```
REGMATRIX RANDOM mySNPs ID=value ...
```

where value is the column number in the REGFILE file. For the data file, the individual ID code is specifed by the DATASORT command:

```
DATASORT PEDIGREECODE=icol
```

where icol is the integer column name or number in the data file. If either piece of information is missing, the order of the lines cannot be verified and is assumed to be correct. For detailed syntax and explanations of these commands, please refer to Chapters 12.2.4 and 12.2.20.

The ID option in REGMATRIX allows using a marker file that has more genotyped individuals than those with observations in the data file. This is useful when the genotype file includes candidate individuals without observation or when the same marker data is used in the analysis of separate traits, where some individuals may not have observations for some traits.

9.1.1 Example: simple SNP-BLUP model

Consider the model

$$y = \mu + \beta_1 g_1 + \beta_2 g_2 + \beta_3 g_3 + \beta_4 g_4 + \beta_5 g_5 + \beta_6 g_6 + e$$

where the coefficients βs are known, the g terms are unknown random additive marker or allele effects, and e is the random residual. There are six bi-allelic markers, numbered 1 through 6. For each marker, genotypes are coded as 0 for homozygous first allele, 1 for heterozygote, and 2 for homozygous second allele. Each marker effect corresponds to the additive effect of the second allele. Thus, two times the estimated marker effect gives the difference between the two homozygote genotypes.

In general matrix notation, the model can be written as

$$\mathbf{y} = \mathbf{X}\boldsymbol{b} + \mathbf{Z}_g\mathbf{g} + \mathbf{e}$$

where

- $\mathbf{y}$ is the vector of observations,
- $\mathbf{X}$ is the design matrix of fixed effects, now a vector of ones ($\mathbf{1}$),
- $\boldsymbol{b}$ is the vector of fixed effects, now the general mean μ,
- $\mathbf{Z}_g$ is the genotype matrix (marker data),
- $\mathbf{g}$ is the vector of random marker effects, and
- $\mathbf{e}$ is the vector of residuals.

It is assumed that $\mathbf{g} \sim N(\mathbf{0}, \mathbf{I}\sigma_g^2)$ and $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}\sigma_e^2)$. The genetic marker ($\sigma_g^2$) and the residual ($\sigma_e^2$) variances are assumed to be known. We will first use this basic SNP-BLUP model to illustrate a set of commands associated with REGMATRIX. Later, more advanced options are presented, which may improve convergence of the solver.

The input data is provided in two files:

- Data file (command DATAFILE) contains columns for individual ID code, fixed effect (general mean), and the observed phenotype.
- Genotype file (command REGFILE) contains the marker genotype data.

As already mentioned, both files must be aligned such that the records correspond to the same individuals in the same order. Thus, the first record in the data file and in the regression matrix file are from the same individual. It is assumed that all genotypes are known and have been coded by the user. The alignment requirement can be relaxed, please see Chapter 9.2.2.

Let the data file and marker genotype file are

data file gs_obs.dat			genotype file gs_geno.dat						
ID code$_1$	mean$_2$	y$_1$	id$_1$	1	2	3	4	5	6
1	1	5	1	2	1	0	0	0	0
2	1	6	2	1	1	0	1	0	0
3	1	10	3	1	0	2	2	2	1
4	1	15	4	0	1	1	2	2	2
			5	0	0	2	1	1	1
			6	0	0	1	2	2	2

There are six marker genotypes numbered from one to six, which are in columns from two to seven of the genotype file. The file has six rows of which the last two represent candidate individuals having no observations.

Let the variance components be:

- the common genetic SNP marker variance $\sigma_g^2 = \frac{1}{6} = 0.166666666$.
- the residual variance $\sigma_e^2 = 1$.

The variance components for a random REGMATRIX have to be given in a REGPARFILE which is different from the regular PARFILE. The file for the marker genotype variance (gs_gen.par) is

Effect number	Row	Column	Covariance	
1	1	1	0.166666666	marker variance

The regular variance file for PARFILE (gs_res.par) has the residual variance:

Random effect	Row	Column	Covariance	
1	1	1	1	residual variance

The CLIM code is

```
DATAFILE   gs_obs.dat

INTEGER    IDcode mean
REAL       y
```

```
MISSING   -99999.
DATASORT  PEDIGREECODE=IDcode

PARFILE    gs_res.par # residual variance

REGMATRIX  RANDOM SNP ID=1 FIRST=2 LAST=7
REGFILE    gs_geno.dat
REGPARFILE gs_gen.par # marker variance

MODEL
  y = mean
```

The covariate columns in REGMATRIX are specified using the FIRST and LAST options.

Please note that no effect name used to define a REGMATRIX can be given in the model. By default, the effects defined by the REGMATRIX command are included in all traits of the model. This behavior can be changed using the REGTRAITS command as illustrated in Chapter 9.2.3.

The fixed general mean solution (Solf01) is

```
          1          4    7.2604
```

The marker effect solutions (Solreg_mat) are

```
 Trt   Matrix   Effect   Solution     Mat-Name
   1        1        1 -0.66624      SNP
   1        1        2  0.11015      SNP
   1        1        3  0.27294      SNP
   1        1        4  0.55610      SNP
   1        1        5  0.76616      SNP
   1        1        6  0.87631      SNP
```

Estimated genomic breeding values can be computed using the formula:

$$\widehat{\boldsymbol{a}} = \mathbf{1}\hat{\mu} + \boldsymbol{Z}_g\widehat{\boldsymbol{g}}$$

where $\hat{\mu}$ is the estimated general mean, $\widehat{\boldsymbol{g}}$ is the vector of estimated marker effects, and $\boldsymbol{Z}_g$ is the genotype matrix. When the solver is executed with the -p option (e.g., mix99s -p -s), it calculates genomic breeding values ($\widehat{\boldsymbol{a}}$) and writes them to the file yHat.data0. Each line in this file has a breeding value for an individual in the data file. The values are in the same order as the observations in the data file. To combine individual ID codes from the data file with the estimated genomic breeding values, the Unix command paste gs_obs.dat yHat.data0 can be used. This gives output like:

```
1 1  5     6.038078
2 1  6     7.260414
3 1 10     10.66087
4 1 15     12.04063
```

Note that an estimated genomic breeding value is not calculated for an individual without observation when using the solver command line option -p. To compute genomic breeding values for all individuals in the genotype file, including those without observations, use the REGINDEX option (see Chapter 9.2.2). NEW

When there is an additional set of regression effects in a separate file, these can also be included in the model using an additional set of REGMATRIX commands. All associated

commands (REGFILE, REGPARFILE etc.) related to a REGMATRIX command need to appear on consecutive lines in the CLIM file. Please see Chapter 9.2.4. Notes:

1) When the number of regression effects is small, it may be easier and clearer to include them in the data file and specify them in the model line, rather than using a REGMATRIX command.

2) Multiple REGMATRIX commands can refer to the same REGFILE, even if some columns are used for fixed and some for random regression effects.

An example CLIM code having two sets of REGMATRIX blocks is

```
DATAFILE   gs_obs.dat

INTEGER    IDcode mean
REAL       y
MISSING    -99999.
DATASORT   PEDIGREECODE=IDcode

PARFILE     gs_res.par # residual variance

REGMATRIX  RANDOM SNP ID=1 FIRST=2 LAST=7
REGFILE     gs_geno.dat
REGPARFILE gs_gen.par # marker variance

REGMATRIX  FIXED regcov ID=1 FIRST=2 LAST=3
REGFILE     regcov_geno.dat

MODEL
  y = mean
```

The second REGFILE adds covariates of two regression effects from the regcov_geno.dat file. The second and third columns have the covariates, but the first column has the ID code of individual.

9.1.2 Centering and scaling marker data

SNP marker data are typically coded using values 0, 1, or 2, representing the number of a specific allele. To improve solver convergence, the marker values can be centered. Furthermore, scaling of the marker data allows giving marker variance values in the unscaled form. The use of SNP marker format, on the other hand, allows having very large marker files in memory, reduce disk usage, and speed up computations. An optional missing marker value with some a one-digit integer such as 3 or 9 can be used for simple imputation.

The marker value ($\boldsymbol{Z}_{i,m}^{012}$) for marker m of individual i in $\boldsymbol{Z}_g$ can be centered and scaled by subtracting a center value μ_m and multiplying this by a scaling value s_m:

$$\boldsymbol{Z}_{i,m}^{c} = (\boldsymbol{Z}_{i,m}^{012} - \mu_m)s_m$$

Let $\mathbf{Z}_c$ be centered and scaled marker matrix.

Then, the SNP-BLUP model with centering and scaling is

$$\mathbf{y} = \mathbf{X}\boldsymbol{b} + \mathbf{Z}_c\mathbf{g}_c + \mathbf{e}$$

where $\mathbf{g}_c$ has the marker effects of the centered and scaled marker matrix model. Centering has no effect on the marker effect solutions, but scaling changes the solutions.

Furthermore, scaling has to be accounted in the (co)variance of the random marker effect, i.e., in the file defined by REGPARFILE, by dividing the unscaled marker variance by the square of the scaling value.

Centering is specified using the CENTER option in the REGMATRIX command. The centering value μ_m can be:

- average of the markers (default)
- a constant applied to all markers,
- a marker-specific value provided in an external file.

Examples:

```
REGMATRIX RANDOM SNP ID=1 FIRST=2 LAST=7 CENTER
```

centers each marker by its average (i.e., mean across individuals).

```
REGMATRIX RANDOM SNP ID=1 FIRST=2 LAST=7 CENTER=1
```

centers all markers by value of one, i.e., transforms the default “0,1,2 coding” to “-1,0,1 coding”.

```
REGMATRIX RANDOM SNP ID=1 FIRST=2 LAST=7 CENTER=mu.dat
```

reads marker-specific centering values from the file mu.dat. This file must contain one value per marker, either in a single row or column. For example, when all markers are centered by value one as done above with the CENTER=1 option, the mu.dat file is

```
1 1 1 1 1 1
```

or

```
1
1
1
1
1
1
```

It is important that the file has only values for centering. Note that the column numbers used in the marker file are not used in the center file. For example, assume that the markers in the previous example had been

```
REGMATRIX RANDOM SNP ID=1 FIRST=3 LAST=8 CENTER=mu.dat
```

Then, the same file can be used as the number of markers remains six. Naturally, the values in the center file can vary across markers unlike in the given example.

Scaling can be specified using the SCALE parameter in the REGMATRIX command. The scaling value s_m can be:

- a constant for all markers,
- a marker-specific value provided in a file.

Scaling needs to be accounted in the variance specified in the REGPARFILE

$$\sigma_a^2 = \sigma_g^2 / s^2$$

This allows the marker variance to be interpreted in the original (unscaled) form corresponding to the additive genetic variance.

For example:

```
REGMATRIX RANDOM SNP ID=1 FIRST=2 LAST=7 SCALE=0.5
```

scales all markers with a half. In other words, all markers are multiplied by 0.5, i.e., $s = 0.5$. Furthermore, when the SNP marker genotype variance is $1/6$ without scaling, then the 0.5 scaling requires changing the variance in the REGPARFILE to be $1/6/(0.5)^2 = 4/6 = 0.666667$.

The m and m2 options in SCALE multiply the markers by $s = \frac{1}{\sqrt{m}}$ and $s = \sqrt{\frac{2}{m}}$, respectively, where m is the number of markers. The SCALE option 2pq allows using allele frequencies:

```
REGMATRIX RANDOM SNP ID=1 FIRST=2 LAST=7 SCALE=2pq
```

which scales all markers by $s = \frac{1}{\sqrt{2\sum_{i=1}^{m} p_i(1-p_i)}}$ where p_i is the allele frequency of marker i computed in the marker data.

Each marker can be scaled individually by having the scaling values in a file:

```
REGMATRIX RANDOM SNP ID=1 FIRST=2 LAST=7 SCALE=s.dat
```

This assumes six separate scaling values in file s.dat. Format of the file is the same as for centering. In the file, each marker has one value which is used to multiply the (centered) marker value.

Scaling often requires changing the marker variance. When the same scaling factor is used for all markers, there is a choice to either change the variance or use the SCALE option. When marker-specific scaling is used, an alternative is to use marker-specific variances using the REGMATRIX HETEROGENEOUS command. For simplicity, assume the same variance is used across all markers, then the commands can be

```
REGMATRIX HETEROGENEOUS SNP ID=1 FIRST=2 LAST=7
REGPARFILE gs_het.par
```

The file for the marker-specific genotype variances (gs_het.par) is

Marker number	Row	Column	Covariance	
1	1	1	0.166666666	marker 1 variance
2	1	1	0.166666666	marker 2 variance
3	1	1	0.166666666	marker 3 variance
4	1	1	0.166666666	marker 4 variance
5	1	1	0.166666666	marker 5 variance
6	1	1	0.166666666	marker 6 variance

For more complete explanation, please see an example in Chapter 9.2.1. In multi-trait models, this approach allows marker-specific and trait specific variances when scaling is always the same across traits.

9.1.3 Example: SNP-BLUP with centering and scaling

Consider the data for the SNP-BLUP model example from Chapter 9.1.1. However, now the genotypes are centered to be from the “0,1,2 coding” to “-1,0,1 coding”. Furthermore, the marker matrix is scaled by multiplying it with $\sqrt{\frac{1}{3}} = 0.57735$. Consequently, the

marker variance must be multiplied by 3, i.e., squared inverse of the scaling factor, such that the marker matrix times the marker (and eventually the phenotypic) variance is the same as in the model without scaling. The updated SNP variance file (`gs_gen_-scaled.par`) is

Effect number	Row	Column	Covariance	
1	1	1	0.5	scaled marker variance

The new CLIM file remains the same as in the original SNP-BLUP model example except for the extra options (`CENTER` and `SCALE`) for the `REGMATRIX` command:

```
DATAFILE  gs_obs.dat

INTEGER   IDcode mean
REAL      y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

PARFILE    gs_res.par # residual variance

REGMATRIX  RANDOM SNP ID=1 FIRST=2 LAST=7 CENTER=1 SCALE=0.57735
REGFILE    gs_geno.dat
REGPARFILE gs_gen_scaled.par # marker variance

REGMATRIX  FIXED regcov ID=1 FIRST=2 LAST=3
REGFILE    regcov_geno.dat

MODEL
  y = mean
```

Note that the centering above can be achieved by using the `m2` scaling option:

```
REGMATRIX  RANDOM SNP ID=1 FIRST=2 LAST=7 CENTER=1 SCALE=m2
```

because the number of markers m equals 6 and the scaling was by $\sqrt{\frac{1}{3}} = \sqrt{\frac{2}{6}} = \sqrt{\frac{2}{m}}$.

The solutions between the unscaled/non-centered and scaled/centered marker model are different. The fixed general mean solution (`Solf01`) is

```
          1          4    9.1758
```

while earlier this solution was 7.2604.

The marker effect solutions (`Solreg_mat`) are

```
 Trt   Matrix   Effect   Solution     Mat-Name
   1        1        1  -1.1540      SNP
   1        1        2  0.19078      SNP
   1        1        3  0.47274      SNP
   1        1        4  0.96319      SNP
   1        1        5   1.3270      SNP
   1        1        6   1.5178      SNP
```

However, the predicted values remain the same

```
1 1  5      6.038079
2 1  6      7.260415
3 1 10      10.66087
```

```
4 1 15      12.04063
```

The main reason for the changes in the marker solutions is scaling which multiplies the marker matrix by the scaling factor 0.57735. Indeed, by multiplying the marker solutions by the scaling factor gives the same solutions as when no scaling was used (or using SCALE=1). Centering, on the other hand, is responsible for the change in the general mean solution as explained in Strandén and Christensen (2011).

9.1.4 REGMATRIX file format

The general regression matrix defined by the REGMATRIX command assumes by default ***real valued numbers*** as the covariates. However, SNP marker data is usually coded as ***integer values*** 0, 1, or 2, with an option to indicate missing marker value. The preprocessor can be made to check the SNP marker integer values. The format of the REGFILE input file can be specified using the optional FORMAT parameter in the REGMATRIX command. The default file format is "n" (for "normal") but SNP marker data can also be specified using the "m" (for "markers") format. For example:

```
REGMATRIX RANDOM SNP ID=1 FIRST=2 LAST=7 FORMAT=m
```

By default, SNP marker values in the REGFILE file are space separated. Because each marker value (0, 1, or 2) is only one character, the spaces effectively double the file size. To reduce the file size, a more compact "s" (for "squeezed") format has no spaces between the marker values. For example, the file gs_geno_nospaces.dat could contain SNP marker values without space separation:

```
REGMATRIX RANDOM SNP ID=1 FIRST=2 LAST=7 FORMAT=s
REGFILE   gs_geno_nospaces.dat
```

The file could be:

id	SNPs
1	210000
2	110100
3	102221
4	011222
5	002111
6	001222

The REGMATRIX command has an option IMPUTE to specify a ***missing marker value*** in the genotype file. For example:

```
  REGMATRIX ... IMPUTE=3
```

This instructs the preprocessor to replace, or ***impute***, all genotype values equal to 3 with the average of the non-missing values in the corresponding marker column.

In addition to plain text formats, SNP markers can also be provided in the PLINK binary .bed format. This is specified using the FORMAT=pb option:

```
REGMATRIX RANDOM SNP ID=1 FORMAT=pb
REGFILE   gs_geno
```

Note that only the individual-major mode (PLINK2 export format ind-major-bed) is currently supported. The corresponding .bim and .fam files must be available in the same directory as the .bed file. When specifying the file name in the REGFILE command, omit the .bed extension.

9.2 Advanced marker effect models

The REGMATRIX approach has many options and commands for easier and more complex models than the basic SNP-BLUP model. In the following examples we consider the following options:

REGMATRIX HETEROGENEOUS for marker-specific (co)variance matrices.

REGINDEX for estimating breeding values and condense the marker data file.

REGTRAITS to limit REGMATRIX for specified traits.

We also consider the use of many REGMATRIX commands in a CLIM file.

9.2.1 Example: Marker-specific variances in SNP-BLUP

Consider the example in Chapter 9.1.3 but with marker-specific variances. This kind of model can be performed using a vector of scales for the SCALE option as well. We consider both approaches.

Assume the marker-specific scale values for the six markers are:

1	2	3	4	5	6
0.90	0.97	1.00	1.00	1.03	1.10

To compute the marker variance for each marker, the common marker variance is multiplied by the square of the marker-specific scale value. Thus, when the common marker variance is 0.16666666, the marker-specific variances are:

1	2	3	4	5	6
0.135	0.156816666	0.166666666	0.166666666	0.176816666	0.201666666

And so, the marker-specific variance file (gs_gen_heter.par) is

Marker number	Row	Column	Covariance	
1	1	1	0.135	marker 1 variance
2	1	1	0.156816666	marker 2 variance
3	1	1	0.166666666	marker 3 variance
4	1	1	0.166666666	marker 4 variance
5	1	1	0.176816666	marker 5 variance
6	1	1	0.201666666	marker 6 variance

The CLIM code is

```
DATAFILE   gs_obs.dat

INTEGER    IDcode mean
REAL       y
MISSING    -99999.
DATASORT   PEDIGREECODE=IDcode

PARFILE     gs_res.par # residual variance

REGMATRIX  HETEROGENEOUS SNP ID=1 FIRST=2 LAST=7 CENTER=1
REGFILE     gs_geno.dat
REGPARFILE gs_heter.par # snp variance

MODEL
  y = mean
```

The general mean solution is in `Solf01`:

```
          1          4    9.2114
```

The marker effect solutions (`Solreg_mat`) are

```
 Trt   Matrix   Effect   Solution    Mat-Name
   1        1        1 -0.53477     SNP
   1        1        2  0.10753     SNP
   1        1        3  0.25415     SNP
   1        1        4  0.54592     SNP
   1        1        5  0.78174     SNP
   1        1        6   1.0299     SNP
```

The predicted values are

```
1 1  5   6.064950
2 1  6   7.145638
3 1 10   10.68569
4 1 15   12.10372
```

An alternative approach is to use marker-specific scales and common marker variance. The `REGMATRIX` uses the `SCALE` option and the original `REGPARFILE` with the common marker variance of 0.166666666. The change in the above CLIM file is to use the following commands:

```
REGMATRIX  RANDOM SNP ID=1 FIRST=2 LAST=7 CENTER=1 SCALE=scales.dat
REGFILE    gs_geno.dat
REGPARFILE gs_gen.par # snp variance
```

where the `scales.dat` is

1	2	3	4	5	6
0.90	0.97	1.0	1.0	1.03	1.10

The general mean solution in `Solf01` is as above:

```
          1          4    9.2114
```

However, the marker effect solutions (`Solreg_mat`) are altered by the marker-specific scaling factors:

```
 Trt   Matrix   Effect   Solution    Mat-Name
   1        1        1 -0.59418     SNP
   1        1        2  0.11085     SNP
   1        1        3  0.25415     SNP
   1        1        4  0.54592     SNP
   1        1        5  0.75897     SNP
   1        1        6  0.93626     SNP
```

Again, multiplying marker solution by its scaling factor gives the same solution as above.

The predicted values are as above:

```
1 1  5   6.064950
2 1  6   7.145638
3 1 10   10.68569
4 1 15   12.10372
```

9.2.2 Example: Easier SNP-BLUP by REGINDEX

NEW

The REGMATRIX command has option REGINDEX which has several advantages for marker data:

Flexible ordering: The REGFILE file does not need to follow the same order as the data file.

Avoid data duplication: When individuals have repeated records in the data file, REGFILE file only needs to appear once.

Automatically breeding values: An additional solution file will be computed having estimated breeding values for all genotyped individuals.

Suppose individuals have several records in the data file. When no REGINDEX option is used, the marker data needs to be repeated as well, although the genotype records of an individual are the same for all the repeats. Let the data file and the marker genotype file be

data file gs_obs_repeat.dat

IDcode$_1$	mean$_2$	y$_1$	y2$_2$
7	1	4	3
7	1	5	2
7	1	6	1
8	1	5	4
8	1	6	3
8	1	7	2
9	1	10	9
9	1	11	8
9	1	12	7
10	1	14	13
10	1	15	12
10	1	16	11

genotype file gs_geno_repeat.dat

ID$_1$	1	2	3	4	5	6
7	2	1	0	0	0	0
7	2	1	0	0	0	0
7	2	1	0	0	0	0
8	1	1	0	1	0	0
8	1	1	0	1	0	0
8	1	1	0	1	0	0
9	1	0	2	2	2	1
9	1	0	2	2	2	1
9	1	0	2	2	2	1
10	0	1	1	2	2	2
10	0	1	1	2	2	2
10	0	1	1	2	2	2
11	0	0	2	1	1	1
12	0	0	1	2	2	2

Using the REGINDEX option, the genotype file can remain as already shown for gs_geno.dat, except that the ID code is changed to be like in the table above to match the new IDcode number in the data file. Thus, the marker file will be smaller and the computations faster. The gs_geno_new.dat is:

ID$_1$	1$_2$	2$_3$	3$_1$	4	5	6
7	2	1	0	0	0	0
8	1	1	0	1	0	0
9	1	0	2	2	2	1
10	0	1	1	2	2	2
11	0	0	2	1	1	1
12	0	0	1	2	2	2

Consider a simple SNP-BLUP model with a non-genetic permanent environmental effect which is IDcode in the model line. The CLIM code for this SNP-BLUP is

```
DATAFILE   gs_obs_repeat.dat

INTEGER    IDcode mean
REAL       y
```

```
MISSING    -99999.
DATASORT   PEDIGREECODE=IDcode

RANDOM      IDcode
PARFILE     gs_pe_res.par

REGMATRIX   RANDOM SNP ID=1 REGINDEX=IDcode FIRST=2 LAST=7 &
                       FORMAT=m CENTER=1 SCALE=0.57735
REGFILE     gs_geno_new.dat
REGPARFILE gs_gen_scaled.par

MODEL
  y = mean IDcode
```

The new variance file for PARFILE (gs_pe_res.par) has two variances:

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	
1	1	1	0.2	permanent environment variance
2	1	1	1.0	residual variance

The general mean solution is in Solfix:

```
 Fact. Trt       Level     N-Obs   Solution              Factor Trait
   1    1           1        12    9.4831                   mean y
```

The permanent environment effect solutions are in Solr01:

```
          7          3 -0.21007
          8          3 -0.38163
          9          3 -0.14549
         10          3  0.73719
```

The marker effect solutions (Solreg_mat) are

```
 Trt   Matrix   Effect   Solution     Mat-Name
   1        1        1   -1.3672      SNP
   1        1        2   0.21000      SNP
   1        1        3   0.64404      SNP
   1        1        4    1.1572      SNP
   1        1        5    1.7081      SNP
   1        1        6    1.9181      SNP
```

Furthermore, the estimated breeding values are in the SolDGV01 file:

```
          7          3     -3.9229
          8          3     -2.4654
          9          3      1.9049
         10          3      3.5511
         11          0      1.0400
         12          0      3.4298
```

Note that these solutions do not include the general mean solution.

9.2.3 Example: REGMATRIX for some traits

By default, the REGMATRIX effects are included in all traits in the model. The REGMATRIX command has a sub-command called REGTRAITS which allows REGMATRIX marker effects to be included to sepected traits only. To use REGTRAITS, the REGINDEX option must be used in REGMATRIX.

Consider the same dataset used in the previous example but now applied to a two-trait SNP-BLUP model. In this two-trait model:

- trait 1 is modelled using both marker and pedigree information.
- trait 2 is modeled using pedigree information only.

The pedigree file remains the same as used earlier. Both traits have a non-genetic permanent environment effect.

The CLIM code for this SNP-BLUP is

```
DATAFILE   gs_obs_repeat.dat

INTEGER    IDcode mean
REAL       y y2
MISSING    -99999.
DATASORT   PEDIGREECODE=IDcode

PEDFILE    my.ped
PEDIGREE   G am

RANDOM      PE G
PARFILE     gs_pe_res_2tr.par

REGMATRIX  RANDOM SNP ID=1 REGINDEX=IDcode FIRST=2 LAST=7 &
                      FORMAT=m CENTER=1 SCALE=0.57735
REGFILE    gs_geno_new.dat
REGPARFILE gs_gen_scaled_2tr.par
REGTRAITS  y

MODEL
  y  = mean PE(IDcode) G(IDcode)
  y2 = mean PE(IDcode) G(IDcode)
```

where the REGTRAITS command is used to indicate that the REGMATRIX is used only for the first trait.

The PARFILE variance file (gs_pe_res_2tr.par) has (co)variance for three random effects (PE, G, and residual):

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	
1	1	1	0.2	trait 1 permanent environment variance
1	1	1	0.1	permanent environment covariance
1	1	1	0.2	trait 2 permanent environment variance
2	1	1	0.1	trait 1 genetic variance
2	2	1	0.05	genetic covariance
2	2	2	0.4	trait 2 genetic variance
3	1	1	1.0	trait 1 residual variance
3	2	1	0.1	residual covariance
3	1	1	1.0	trait 2 residual variance

The SNP variance file must include values for all traits, even if only a subset of traits is specified in the REGTRAITS command. It is not necessary to provide covariance values for the unused traits, but all specified variances must be positive. For example,

consider a SNP variance file `gs_gen_scaled_2tr.par` which includes variances for both traits, although `REGMATRIX` is applied only in the first:

Effect number$_1$	Row$_2$	Column$_3$	Covariance$_1$	
1	1	1	0.5	trait 1 marker variance
1	2	2	1.0	dummy trait 2 marker variance

Thus, the variance of the second trait is included only to satisfy the format requirements, although the solve will not use the value in the computations.

The general mean solution is in `Solfix`:

```
Fact. Trt      Level    N-Obs  Solution              Factor Trait
  1   1          1        12   9.3485                   mean y
  1   2          1        12   5.8999                   mean y2
```

The permanent environment effect solutions are in `Solr01`:

```
         7         3 -0.79547      -1.3415
         8         3 -0.77131      -1.1045
         9         3  0.19172      0.54262
        10         3   1.3751       1.9035
```

The marker effect solutions (`Solreg_mat`) are

```
Trt  Matrix  Effect  Solution    Mat-Name
  1       1       1  -1.0547     SNP
  1       1       2  0.15318     SNP
  1       1       3  0.50837     SNP
  1       1       4  0.90154     SNP
  1       1       5   1.3231     SNP
  1       1       6   1.4763     SNP
  2       1       1   0.0000     SNP
  2       1       2   0.0000     SNP
  2       1       3   0.0000     SNP
  2       1       4   0.0000     SNP
  2       1       5   0.0000     SNP
  2       1       6   0.0000     SNP
```

Note that the second trait `REGMATRIX` marker effect solutions are given although the trait does not have marker effects. These solutions are zero.

The polygenic additive genetic solutions (`Solani`) are

```
         1       2       0 -0.92551E-07 -0.88344E-06
         2       2       0 -0.92551E-07 -0.88344E-06
         3       2       0  0.38229E-07 -0.53506E-06
         4       2       0  0.38229E-07 -0.53506E-06
         5       3       0  0.29552E-06  0.66125E-06
         6       3       0  0.29552E-06  0.66125E-06
         7       1       3 -0.26985E-01 -0.43328
         8       1       3 -0.20944E-01 -0.14912
         9       2       3  0.47930E-01  0.58241
        10       2       3  0.31980       1.4006
        11       0       0  0.18387      0.99149
        12       0       0  0.18387      0.99149
```

Furthermore, the marker based genetic breeding values are in the `SolDGV01` file:

```
          7          6     -3.0392          0.0000
          8          6     -1.9097          0.0000
          9          6      1.4895          0.0000
         10          6      2.7457          0.0000
         11          0     0.81401          0.0000
         12          0      2.6572          0.0000
```

Also here the second trait solutions are given although this trait has no REGMATRIX marker effects in the model.

9.2.4 Example: Hybrid marker effects model

Multiple regression matrices are supported. Currently, CLIM allows up to five regression matrix files using the REGMATRIX and REGFILE commands.

Consider a hybrid genomic model

$$\mathbf{y} = \mathbf{1}\mu + \mathbf{Z}_f\mathbf{g}_f + \mathbf{Z}_m\mathbf{g}_m + \mathbf{W}_p\mathbf{p}_e + \mathbf{e}$$

where

- $\mathbf{y}$ is vector of observations,
- $\mathbf{1}$ is vector of ones,
- μ is the general mean,
- $\mathbf{Z}_f$ is matrix of centered and scaled female parent marker genotypes,
- $\mathbf{g}_f$ is vector of female marker effects, and
- $\mathbf{Z}_m$ is matrix of centered and scaled male parent marker genotypes,
- $\mathbf{g}_m$ is vector of male marker effects, and
- $\mathbf{W}_p$ is incidence matrix for $\mathbf{p}_e$,
- $\mathbf{p}_e$ is vector of non-genetic permanent environment effects,
- $\mathbf{e}$ is the residual vector.

We consider simple centering by 1 and scaling by the number of markers.

We consider a two-trait hybrid genomic marker effects model and a data with repeated observations. We will use the REGINDEX option to allow giving the genotypes of an individual only once.

Convergence of a marker effect based hybrid model can be slow because the parental effects may try to estimate similar genetics. In some cases convergence can be improved by using the second-level. The second-level preconditioner is invoked by the solver program options. For example, giving value '-sp 100' for the solver mix99s can enhance convergence.

It is assumed that $\mathbf{p}_e \sim N(\mathbf{0}, \mathbf{P}_0 \otimes \mathbf{I})$ and $\mathbf{e} \sim N(\mathbf{0}, \boldsymbol{R}_0 \otimes \mathbf{I})$. For the markers: $\mathbf{g}_f \sim N(\mathbf{0}, \mathbf{G}_f \otimes \mathbf{I})$ and $\mathbf{g}_m \sim N(\mathbf{0}, \mathbf{G}_m \otimes \mathbf{I})$. In the following, we assume

$$\mathbf{P}_0 = \begin{bmatrix} 0.25 & 0.10 \\ 0.10 & 0.25 \end{bmatrix}, \boldsymbol{R}_0 = \begin{bmatrix} 1.00 & 0.10 \\ 0.10 & 1.00 \end{bmatrix}, \mathbf{G}_f = \begin{bmatrix} 1.00 & 0.70 \\ 0.70 & 1.00 \end{bmatrix}, \text{ and } \mathbf{G}_m = \begin{bmatrix} 0.90 & 0.60 \\ 0.60 & 1.10 \end{bmatrix}.$$

The CLIM code for this hybrid marker effects model is

```
DATAFILE gs_obs_repeat_hybrid.dat

INTEGER  id female male mean
REAL     y1 y2
MISSING -99999.

PARFILE     CLIM
1 LOWER     # PE
  0.25
  0.10 0.25
2 LOWER     # residual
  1.00
  0.10 1.00

REGMATRIX   RANDOM SNPfemale ID=1 REGINDEX=female FIRST=2 LAST=7 &
            FORMAT=m CENTER=1 SCALE=m
REGFILE     geno_female.dat
REGPARFILE  female_gvar_2tr_hybrid.par # snp variance

REGMATRIX   RANDOM SNPmale   ID=1 REGINDEX=male   FIRST=2 LAST=7 &
            FORMAT=m CENTER=1 SCALE=m
REGFILE     geno_male.dat
REGPARFILE  male_gvar_2tr_hybrid.par # snp variance

RANDOM     PE

MODEL
  y1 = mean PE(id)
  y2 = mean PE(id)
```

Let the data file file (`gs_obs_repeat_hybrid.dat`) be

IDcode$_1$	ID_female$_2$	ID_male$_3$	mean$_4$	y_1	$y2_2$	x_3
1	11	21	1	4	3	−0.5
1	11	21	1	5	2	0.0
1	11	21	1	6	1	0.5
2	12	22	1	5	4	−0.5
2	12	22	1	6	3	0.0
2	12	22	1	7	2	0.5
3	13	23	1	10	9	−0.5
3	13	23	1	11	8	0.0
3	13	23	1	12	7	0.5
4	14	24	1	14	13	−0.5
4	14	24	1	15	12	0.0
4	14	24	1	16	11	0.5

The SNP variance files (`female_gvar_2tr_hybrid.par` and `male_gvar_2tr_-hybrid.par`) are

Effect number	Row	Column	Value	
1	1	1	1.0	female trait 1 genetic variance
1	1	2	0.7	genetic covariance
1	2	2	1.0	female trait 2 genetic variance

and

Effect number	Row	Column	Value	
1	1	1	0.9	male trait 1 genetic variance
1	1	2	0.6	genetic covariance
1	2	2	1.1	male trait 2 genetic variance

The parent genotype files are:

geno_female.dat for female parent:

female ID	1	2	3	4	5	6
11	0	1	2	0	2	2
12	1	1	1	1	0	2
13	1	2	0	2	0	1
14	2	0	0	2	2	0

geno_male.dat for male parent:

male ID	1	2	3	4	5	6
21	1	2	1	2	2	1
22	1	0	0	1	0	0
23	0	1	0	0	0	1
24	2	1	2	0	2	1

The general mean solution is in `Solfix`:

```
Fact.  Trt       Level     N-Obs   Solution              Factor Trait
   1    1           1        12    9.0649                   mean y1
   1    2           1        12    6.0634                   mean y2
```

The permanent environment effect solutions are in `Solr01`:

```
          1           3 -0.30407     -0.29626
          2           3 -0.43184     -0.41147
          3           3  0.28980      0.28449
          4           3  0.44610      0.42324
```

The marker effect solutions (`Solreg_mat`) are

```
Trt   Matrix   Effect   Solution    Mat-Name
  1        1        1    1.4696     SNPfemale
  1        1        2 -0.29970      SNPfemale
  1        1        3  -2.0412     SNPfemale
  1        1        4   2.0412     SNPfemale
  1        1        5  0.54570     SNPfemale
  1        1        6  -2.3141     SNPfemale
  1        2        1  0.26527     SNPmale
  1        2        2  0.21766     SNPmale
  1        2        3   1.0113     SNPmale
  1        2        4  -1.8028     SNPmale
  1        2        5  0.48293     SNPmale
  1        2        6  0.74604     SNPmale
  2        1        1   1.4446     SNPfemale
  2        1        2 -0.28536     SNPfemale
  2        1        3  -2.0118     SNPfemale
  2        1        4   2.0118     SNPfemale
  2        1        5  0.52111     SNPfemale
  2        1        6  -2.2724     SNPfemale
```

```
  2        2        1   0.28057      SNPmale
  2        2        2   0.23232      SNPmale
  2        2        3    1.1028      SNPmale
  2        2        4   -2.0020      SNPmale
  2        2        5   0.51289      SNPmale
  2        2        6   0.82223      SNPmale
```

The marker based genetic breeding values for both parent groups are in the own files by the `REGMATRIX` number. The female parent genomic breeding value solutions are in the `SolDGV01` file:

```
         11          6      -2.9886          -2.9474
         12          6      -1.1675          -1.1404
         13          6       1.3215           1.3134
         14          6       3.5565           3.4894
```

The male parent genomic breeding value solutions are in the `SolDGV02` file:

```
         21          6     -0.44997         -0.51310
         22          6      -1.0035          -1.0901
         23          6     0.17666E-01      0.43172E-01
         24          6       1.4543           1.5915
```

9.3 Genomic BLUP (GBLUP) model

A simple single trait SNP-BLUP model using matrix notation is

$$\mathbf{y} = \mathbf{X}\boldsymbol{b} + \mathbf{Z}_c\mathbf{g} + \mathbf{e}$$

where

- $\mathbf{y}$ is vector of observations,
- $\mathbf{X}$ is the fixed effect design matrix,
- $\boldsymbol{b}$ is vector of unknown fixed effects,
- $\mathbf{Z}_c$ is a matrix of centered and scaled SNP marker genotypes,
- $\mathbf{g}$ is a vector of unknown random marker effects,
- $\mathbf{e}$ is the random residual.

An equivalent GBLUP model solves the breeding value vector $\mathbf{u} = \mathbf{Z}_c\mathbf{g}$ without the need to solve the marker effects $\mathbf{g}$. The model is

$$\mathbf{y} = \mathbf{X}\boldsymbol{b} + \mathbf{Z}_u\mathbf{u} + \mathbf{e}$$

where $\mathbf{Z}_u$ is an incidence matrix linking breeding values $\mathbf{u}$ to the observations $\mathbf{y}$.

In SNP-BLUP, it is assumed that $\mathbf{g} \sim N(\mathbf{0}, \mathbf{I}\sigma_g^2)$. In the GBLUP model, it is assumed that $\mathbf{u} \sim N(\mathbf{0}, \mathbf{G}\sigma_u^2)$ where $\mathbf{G} = \mathbf{Z}_c\mathbf{Z}_c'$ is the genomic relationship matrix, $\sigma_u^2 = \sigma_g^2/s^2$, σ_g^2 is the marker variance, and s is the scaling value.

Recall that the centered and scaled marker value for marker m of individual i is

$$\boldsymbol{Z}_{i,m}^c = (\boldsymbol{Z}_{i,m}^{012} - \mu_m)s_m$$

where $\boldsymbol{Z}_{i,m}^{012}$ is the original genotype, and μ_m and s_m are the center and scale values, respectively, of marker m. Here we assume that a common scaling factor for all markers

s has been used. Note that $\mathbf{Z}_c$ can include genomic data of the candidate individuals as well. Because these genotypes are included in the genomic relationship matrix, genomic breeding values are estimated for the candidate individuals without observations as well. In order to estimate breeding values using the GBLUP model, MiX99 needs inverse of the genomic relationship matrix, instead of the marker information needed by the SNP-BLUP model.

9.3.1 Example: Genomic relationship matrix

Consider the centered and scaled marker data in the SNP-BLUP model example in Chapter 9.1.3. The scaling factor s is equal to half the number of markers, i.e., 3.

The centered and scaled marker matrix is

$$\mathbf{Z}_c = \sqrt{\frac{1}{3}} \begin{bmatrix} 1 & 0 & -1 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 \\ 0 & -1 & 1 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 & 1 & 1 \\ -1 & -1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

In the SNP-BLUP example, the scaling factor of the matrix matrix was $\sqrt{\frac{1}{3}} = 0.57735$. Thus, the covariance matrix is

$$\mathbf{G} = \mathbf{Z}_c\mathbf{Z}_c' = \begin{bmatrix} 5 & 3 & -3 & -4 & -2 & -4 \\ 3 & 3 & -2 & -2 & -1 & -2 \\ -3 & -2 & 4 & 2 & 2 & 3 \\ -4 & -2 & 2 & 4 & 1 & 4 \\ -2 & -1 & 2 & 1 & 3 & 2 \\ -4 & -2 & 3 & 4 & 2 & 5 \end{bmatrix} /3$$

which is called genomic relationship matrix. Mixed model equations require the inverse of this matrix. MiX99 will not compute it, and the user will have to provide a pre-computed inverse matrix. In our case, the inverse is

$$\mathbf{G}^{-1} = \begin{bmatrix} 57 & -15 & 15 & 75 & 24 & -39 \\ -15 & 6 & -3 & -18 & -6 & 9 \\ 15 & -3 & 6 & 21 & 6 & -12 \\ 75 & -18 & 21 & 105 & 33 & -57 \\ 24 & -6 & 6 & 33 & 12 & -18 \\ -39 & 9 & -12 & -57 & -18 & 33 \end{bmatrix}$$

.

9.3.2 Inverse genomic relationship matrix file format

The inverse of the genomic relationship matrix is provided in a file stored in two alternative formats:

- Co-ordinate (Yale) sparse matrix format.
- Lower triangle dense format.

The file can be in text or in binary format (see Chapter 9.3.3). In the co-ordinate sparse matrix format, only non-zero values of the matrix need to be exist in the file. In the lower

triangle dense matrix format, all lower triangle elements of the matrix are in the file, also the zeros. Because the inverse of the genomic relationship matrix has often mostly non-zero values, the lower triangle dense matrix format is often preferred because it typically gives a smaller file size, is faster to read to RAM by the solver, and allows faster computations.

The default inverse matrix file format is the co-ordinate (Yale) sparse matrix format. In this format, each non-zero element in the lower triangle of the matrix is given its element position. Consider the inverse matrix in Chapter 9.3.1 and store the matrix in file `iG_101.dat`:

```
1 1  57
2 1 -15
2 2   6
3 1  15
3 2  -3
3 3   6
4 1  75
4 2 -18
4 3  21
4 4 105
5 1  24
5 2  -6
5 3   6
5 4  33
5 5  12
6 1 -39
6 2   9
6 3 -12
6 4 -57
6 5 -18
6 6  33
```

Note that the element positions are the individual ID codes. In our case, they are from one to six. The ID codes need not be consecutive or in increasing order.

An alternative is the lower triangle dense matrix format. In the example case, the format (in file `iGL_101.dat`) is:

```
 6 0
 1 2 3 4 5 6
  57
 -15   6
  15  -3   6
  75 -18  21 105
  24  -6   6  33  12
 -39   9 -12 -57 -18 33
```

This matrix file has two header rows:

- The first row has the values 6 and 0:
  - 6 indicates the size of the matrix, i.e., the number of genotyped individuals,
  - 0 indicates that this is a full inverse.
- The second row has the ID codes of the genotyped individuals. Their order define

the row (and column) order of the lower triangle matrix that follows.

The matrix values are listed row by row, corresponditing to the lower triangle of the full matrix.

The ID codes do not have to be sorted in increasing order as given above. The previous matrix can be given as

```
6 0
3 1 2 4 5 6
  6
 15  57
 -3 -15   6
 21  75 -18 105
  6  24  -6  33  12
-12 -39   9 -57 -18 33
```

Despite the order diffrence, the GBLUP model computations will give the same results.

9.3.3 Binary inverse genomic relationship matrix file

It is recommended to use the Fortran unformatted binary file format or the stream binary file format to achieve fast reading (and writing) of a matrix. When using the `hginv` program to calculate matrix inverse, a file name having a '.bin' suffix will automatically write a Fortran unformatted binary file. When the suffix is '.raw', the file be in the stream binary format. MiX99 reconginizes these suffixes and reads the files correctly provided the files have been written correctly. A lower triangle dense matrix format gives more efficiency as well because it often takes less disk space and is faster to read to memory when the matrix is dense.

In case the `hginv` is unavailable, below is a detailed description of contents of the matrix file for binary file formats. Particularly the Fortran unformatted binary and stream file formats need to be followed exactly and use the given number precision.

- sparse matrix format.

  Each line consists of three numbers: `<i> <j> <value>`

  where `<i>` and `<j>` are 8 byte integer values for matrix position and `<value>` is matrix value in double precision. Individual ID codes are used for matrix position of a value.

- lower triangle dense matrix format.

  The file has three sections:

  - first line: `<i8> <i4> <f4> <f4> <f4>`

    where `<i8>` is an 8 byte integer for the size of the matrix, and `<i4>` is a 4 byte integer for the number of core individuals in APY (Chapter 9.3.7), while the single precision values `<f4>` are ignored.

  - second line: individual ID codes in 8 byte integers.

  - lower triangle dense matrix, where lines and columns are in the order of the ID codes in the second line.

All values are double precision floating point numbers. The form of the matrix follows the format given in the examples, except that in text files, the first line need not include the last 3 values (<f4>).

9.3.4 Defining a GBLUP model

CLIM allows several ways to define a GBLUP model. The easiest is to have inverse of the genomic relationship matrix in a file and assign it to the additive genetic effect. Then, the estimated genomic breeding values are stored to the Solani file. The CLIM commands are:

- use the PEDFILE and PEDIGREE commands. For example,

```
PEDFILE  LOWER iGL_101.dat
PEDIGREE IDcode FILE
```

  Note how the existence of the inverse covariance matrix in a file for the individual genetic effects is indicated by option FILE in the PEDIGREE command.

- use the GBLUP command. For example,

```
GBLUP IDcode LOWER iGL_101.dat
```

In both cases, the LOWER option is used to indicate that the inverse genomic matrix in file iGL_101.dat is in the lower triangle dense matrix format. In addition, IDcode refers to the integer column in the data file having the individual ID code. Both approaches lead to the same model and computational approach. The first approach illustrates that a GBLUP model is a form of a pedigree-based approach where the inverse relationship matrix is in a file. Thus, this approach assumes that the random effect is an additive genetic effect and is the last effect on the model line.

An alternative is to use the COVFILE to associate the inverse genomic relationship matrix to a random effect:

```
COVFILE IDcode LOWER iGL_101.dat
```

Because this can be done to any random effect (other than the residual), the effect has to be defined to be random using the RANDOM command. This is not necessary for the additive genetic effect as it is by definition the second last random effect (the residual being the last).

The inverse genomic relationship matrix can be provided in the co-ordinate format. Then, the file is assumed to have all non-zero elements of the lower triangle of the matrix. Because it may be inconvenient to provide only the lower triangle elements, a mixed format is available which assumes that only upper or lower triangle element is given in the file. Thus, instead of the LOWER option, the MIXED option is used. This option assumes that an off-diagonal value is stored only once, but it can be in either upper or lower triangle of the matrix. For example,

```
GBLUP MIXED iG_101.dat
```

The MIXED option must be used with caution because the preprocessor will not check if a matrix element appears multiple times (e.g., both as an upper and lower triangle element) in the file. If an upper and lower triangle element is given then both will be used to refer to the upper and lower matrix, i.e., the element is used twice.

9.3.5 Example: GBLUP model

We will make the SNP-BLUP model example in Chapter 9.1.3 using GBLUP. In SNP-BLUP, the marker variance was in a separate file due to the markers being defined using the `REGMATRIX` command. When using the `GBLUP` command, the variance(s) of genomic breeding values are included in the random effect variances of the `PARFILE` command. Thus, in this example, the variance components file (`gs_scaled.par`) is

Effect number$_1$	Row$_2$	Column$_3$	Covariance$_1$	
1	1	1	0.5	scaled marker variance
2	1	1	1.0	residual variance

Note that here $\mathbf{Z}_c\mathbf{Z}_c'$ was used the genomic relationship matrix. This is known as VanRaden method 1 (VanRaden (2008)) genomic relationship matrix and it includes scaling of the markers to the scale of the additive genetic variance. In our case, allele frequencies were assumed to be 0.5 for all markers, which leads to the VanRaden method 1 type scaling factor to be $\frac{2}{m}$, where m is the number of markers.

The CLIM code for GBLUP is

```
DATAFILE   gs_obs.dat

INTEGER    IDcode mean
REAL       y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

PARFILE    gs_scaled.par # parameters

GBLUP IDcode LOWER iGL_101.dat

MODEL
  y = mean IDcode
```

The predictions are like in Chapter 9.1.3. The fixed general mean solution (`Solfix`) is

```
 Fact. Trt       Level     N-Obs   Solution              Factor Trait
   1   1           1           4    9.1758                 mean y
```

The breeding value solutions (`Solani`) are

```
           1       1       1   -3.1377
           2       1       1   -1.9154
           3       1       1    1.4850
           4       1       1    2.8648
           5       1       0   0.82903
           6       1       0    2.7547
```

An alternative to using the `GBLUP` command for a GBLUP model, is to use the `COVFILE` command:

```
...
COVFILE IDcode LOWER iGL_101.dat
RANDOM  IDcode
...
```

where the line having `COVFILE` command replaces the line having the `GBLUP` command, and there is an additional line for defining `IDcode` to be the first random effect.

The COVFILE command is useful in models that have many random effects, each with its own genomic relationship matrix, such as the hybrid model (see Chapter 9.3.9).

The results using the COVFILE command are the same. The Solfix file is exactly the same. However, the breeding value solutions are in the Solr01 file:

```
          1          1  -3.1377
          2          1  -1.9154
          3          1   1.4850
          4          1   2.8648
          5          0  0.82903
          6          0   2.7547
```

The file format is different but the solutions are the same.

When the solver is executed with the -p option When the solver program has been executed with the -p option (e.g., mix99s -p -s), the yHat.data0 is produced. Giving the Unix command paste gs_obs.dat yHat.data0 combines individual ID code numbers in the data file with the predicted values which are equal to the estimated genomic breeding values. Result is

```
1 1  5     6.038090
2 1  6     7.260419
3 1 10     10.66087
4 1 15     12.04062
```

Note that these values can be calculated by adding the fixed general mean solution 9.1758 to the breeding value solutions in the Solani (or the Solr01) file.

9.3.6 Example: GBLUP and weights for residual variance

Usually it is assumed that the residual variance is the same for all observations, i.e., a homogeneous residual variance model. However, phenotypic records for genotyped individuals are not always original raw observations but are adjusted or pre-corrected observations where a larger data has been used in a statistical model to account for non-genetic effects. These adjusted observations can have different precision depending on the amount of data to estimate non-genetic effects. Consequently, the adjusted phenotypes have heterogeneous residual variances.

The heterogeneity of residual variances can be accounted for using appropriate weights. This approach is often used in so called two-step genomic evaluation where the first step corrects for non-genetic effects, and the second step uses the corrected observations in a genomic prediction model. For example, VanRaden (2008) defined weights as $r/(1-r)$, where r is the genotyped individual's reliability based on the amount of own or progeny data information. Computation of reliabilities for weights need to be done carefully to avoid double counting of the same information.

Consider the GBLUP example in Chapter 12.2.6. Let the data file named gs_obs_-weights.dat be

ID code	mean	y	weight
1	1	5	0.8
2	1	6	1.0
3	1	10	1.5
4	1	15	4.0

where the data has been augmented with column of weights.

The CLIM code for GBLUP is

```
DATAFILE  gs_obs_weights.dat

INTEGER   IDcode mean
REAL      y weight
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

PARFILE   gs_scaled.par # parameters

GBLUP IDcode LOWER iGL_101.dat

MODEL
  y = mean IDcode | WEIGHT=weight
```

The predictions change slightly. The fixed general mean solution (`Solfix`) is

```
 Fact. Trt       Level     N-Obs   Solution              Factor Trait
   1   1           1           4    10.061                 mean y
```

The breeding value solutions (`Solani`) are

```
          1       1       1  -3.8581
          2       1       1  -2.2830
          3       1       1   1.3043
          4       1       1   3.7427
          5       1       0  0.73231
          6       1       0   3.4015
```

9.3.7 Example: GBLUP with APY

Algorithm for Proven and Young (APY, Misztal et al. (2014)) is an approach for computing inverse genomic relationship matrix having many genotyped individuals. The approach separates the genotyped individuals to two groups: core and non-core. The inverse genomic relationship matrix is made such that the matrix block of the non-core individuals is diagonal. Often the number of core individuals is set to be less than 20,000 allowing for an APY inverse genomic relationship matrix to be build for millions of genotyped individuals. This is possible because the matrix has be made sparse in the non-core block where only the diagonal is stored.

The APY inverse relationship matrix can be stored in the co-ordinate format as given earlier. However, the lower triangle dense format is often faster in computations. A special lower triangle dense format is available for APY where only the diagonal elements of the non-core block are stored.

In the lower triangle dense matrix format, the first line has two numbers. The second number is used to inform the number of core individuals in APY. For example, consider the earlier genomic relationship matrix of six individuals. A core of 2 individuals would give a lower triangle dense matrix format file `iGL_APY.dat`:

```
6 2
 1 2 3 4 5 6
 8.492311
 -3.600001 3.276924
 .6923077 .2307692 1.384615
 4.500002 -1.500001 4.500001
 .6923078 -.2307693 1.384615
```

```
1.800001 -.6000003 1.800000
```

In this format, the core individuals are 1 and 2, and the rest are non-core individuals. The diagonal of the 4 non-core individuals is in the last column of the lower triangle dense matrix format (values on the column starting from 1.384615). This matrix takes less memory than the original full inverse relationship matrix when read to RAM.

The CLIM code for GBLUP is like before

```
DATAFILE   gs_obs.dat

INTEGER    IDcode mean
REAL       y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

PARFILE    gs.par # parameters

GBLUP IDcode LOWER iGL_APY.dat

MODEL
  y = mean IDcode
```

The fixed general mean solution (`Solfix`) is

```
 Fact. Trt       Level     N-Obs   Solution               Factor Trait
   1   1           1           4    9.1466                  mean y
```

The breeding value solutions (`Solani`) are

```
          1       1       1   -3.1217
          2       1       1   -1.8962
          3       1       1    1.6053
          4       1       1    2.8260
          5       1       0    1.2448
          6       1       0    2.4896
```

9.3.8 Example: GBLUP with a polygenic effect

Model is

$$\mathbf{y} = \mathbf{1}\mu + \mathbf{Z}_u\mathbf{u} + \mathbf{Z}_a\mathbf{a} + \mathbf{e}$$

where

- $\mathbf{u}$ is the vector of random additive genetic effects from genomic data
- $\mathbf{a}$ is the vector of random additive polygenic effect using pedigree information
- $\mathbf{e}$ is the vector of random residuals.

Matrix $\mathbf{Z}_u$ is the incidence matrix of the genomic part as in the GBLUP example (Chapter 9.3.5), and matrix $\mathbf{Z}_a$ is the incidence matrix relating observation of an individual in $\mathbf{y}$ to its polygenic (pedigree-based) breeding value in $\mathbf{a}$. The model is the same as in the previous example except for the polygenic $\mathbf{a}$ effect.

The random effects have the following assumptions: $\mathbf{u} \sim N(\mathbf{0}, \mathbf{G}\sigma_u^2)$, $\mathbf{a} \sim N(\mathbf{0}, \mathbf{A}\sigma_a^2)$, and $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}\sigma_e^2)$ where $\mathbf{G}$ is the genomic relationship matrix (see Chapter 9.3.4) and $\mathbf{A}$ is the pedigree-based relationship matrix. The example $\mathbf{A}$ matrix uses the pedigree in file `gs_poly.ped`:

individual$_1$	sire$_2$	dam$_3$
1	2	3
2	4	3
3	5	6
4	5	6
5	0	0
6	0	0

This small pedigree has some non-zero inbreeding coefficients. We account for the inbreeding coefficients in the inbreeding coefficient file named `gs_poly.inbr`, when using the $\mathbf{A}^{-1}$ matrix in MiX99. The file is

individual$_1$	number$_2$	F$_1$
5	1	0.00000
6	2	0.00000
3	3	0.00000
4	4	0.00000
2	5	0.25000
1	6	0.37500

where the first column has the original individual ID code number, and the last column has the inbreeding coefficient. When no inbreeding coefficient file is given, MiX99 builds $\mathbf{A}^{-1}$ assuming all inbreeding coefficients are zero.

The polygenic effect has the pedigree to make the computations due to the pedigree-based numerator relationship matrix. Thus, the polygenic effect takes the additive genetic effect. To include a separate random effect with the genomic relationship matrix in the model, the `COVFILE` command has to be used. Consequently, the `RANDOM` command needs to be used as well. The numbering of the random effects is

1 genomic

2 polygenic

3 residual.

In this example, the variance components are: common scaled marker variance $\sigma^2_u = \frac{1}{2}$, polygenic variance $\sigma^2_a = \frac{1}{2}$, and the residual variance $\sigma^2_e = 1$. The variances are in the file `gs_poly.par`:

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	
1	1	1	0.5	scaled marker genetic variance
2	1	1	0.5	polygenic variance
3	1	1	1	residual variance

The CLIM code is

```
DATAFILE  gs_obs.dat

INTEGER   IDcode mean
REAL      y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode
```

```
PARFILE    gs_poly.par

COVFILE    genomic LOWER iGL_101.dat

PEDFILE    gs_poly.ped
PEDIGREE   polygenic am

INBRFILE    gs_poly.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

RANDOM  genomic polygenic

MODEL
  y = mean genomic(IDcode) polygenic(IDcode)
```

Note that

- the RANDOM command gives the numbering of the random effects (other than residual). The polygenic effect must be the last random effect in this statement.
- the words genomic and polygenic are NOT reserved names but names just to identify different random effects.
- the COVFILE command is used to indicate that the random effect genomic has an inverse (genomic) relationship matrix in the file iGL_101.dat.

The fixed general mean solution (Solfix) is

```
 Fact.  Trt       Level     N-Obs   Solution              Factor Trait
   1    1           1          4    9.4191                   mean y
```

The genomic breeding value estimates (Solr01) are

```
          1          1   -2.8721
          2          1   -1.7818
          3          1    1.4371
          4          1    2.5866
          5          0   0.79525
          6          0    2.5158
```

The additive polygenic solutions (Solani) are

```
          1       1       1   -3.4987
          2       1       1   -2.2013
          3       1       1    1.0060
          4       1       1    3.0177
          5       1       0   0.79525
          6       1       0    2.5158
```

Note that in practice this model may have convergence problems because the polygenic and genomic breeding value effects try to estimate the same entity: genetics. Thus, this model must split the genetic breeding value into its polygenic and marker components where the genomic breeding values have a genomic relationship matrix, and the polygenic effect has a pedigree-based relationship matrix. In this example, these matrices are very different but in practice, they can be similar which increases convergence problems. Furthermore, in this example, the total phenotypic variance was increased by including the polygenic pedigree-based genetic effect having the same variance as the genomic component. This is seldom the case in practice.

An alternative equivalent model to this model would be to make a relationship matrix that combines the genomic and pedigree-based relationship matrices. For example, let $\mathbf{G}_w = (1-w)\mathbf{G} + w\mathbf{A}$ where w is so called residual polygenic proportion. Then, the GBLUP model uses $\mathbf{G}_w^{-1}$, instead of $\mathbf{G}^{-1}$. Thus, the instructions will be the same as for the GBLUP model with the `PEDFILE` replaced by the new inverse covariance matrix $\mathbf{G}_w^{-1}$ where the pedigree-based and the genomic relationship matrices have been combined. This will yield the same estimated complete breeding values. However, in practice, the convergence of the iterative PCG method is likely to be better due to not having to estimate two components of the genetic breeding value for every individual. However, computationally this approach requires computing $\mathbf{A}$ for the genotyped individuals which may be impossible with large number of genotyped.

The residual polygenic proportion w describes the proportion of genetics associated with the polygenic effects. In our example, the ratio of the polygenic variance (0.5) to the sum of genomic and polygenic variances (1.0) gives $w = 0.5$. Thus, the equivalent GBLUP model uses a blended genomic relationship matrix $\mathbf{G}_w = (1-w)\mathbf{G} + w\mathbf{A}$ where $w = 0.5$. In our case, the $\mathbf{G}_w^{-1}$ matrix in the lower triangle dense matrix format is (file `iGLw50.dat`):

```
6 0
 1 2 3 4 5 6
 2.113669
 -1.877159 2.582680
 -.4438857 .3213761 1.842661
 .5862194 -.6652937 -.6519180E-01 2.194285
 .2370449 -.2535169 -.8087914 -.3584017 1.536891
 .3305978 -.8519997E-01 -.9081077 -1.240688 .3753554 2.117931
```

The CLIM code is simple GBLUP model

```
DATAFILE gs_obs.dat

INTEGER IDcode mean
REAL    y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

PARFILE gs_w50.par

GBLUP IDcode LOWER iGL_w50.dat

MODEL
  y = mean IDcode
```

The fixed general mean solution (`Solfix`) is

```
 Fact. Trt      Level     N-Obs   Solution              Factor Trait
   1   1          1          4    9.4191                 mean y
```

The total estimated breeding value will be in the `Solani` file:

```
          1       1       1  -3.4987
          2       1       1  -2.2013
          3       1       1   1.0060
          4       1       1   3.0177
          5       1       0  0.79525
```

```
          6       1       0    2.5158
```

Note that these values are equal to the sum of the genomic and polygenic solutions when these effects were included separately in the model.

9.3.9 Example: Hybrid GBLUP model

Chapter 9.2.4 illustrated a hybrid model with markers. An equivalent GBLUP type model uses genomic relationship matrices instead of markers:

$$\mathbf{y} = \mathbf{1}\mu + \mathbf{W}_f\mathbf{a}_f + \mathbf{W}_m\mathbf{a}_m + \mathbf{W}_p\mathbf{p}_e + \mathbf{e}$$

where

- $\mathbf{y}$ is vector of observations,
- $\mathbf{1}$ is vector of ones multiplying the general mean μ,
- $\mathbf{W}_f$ is incidence matrix for $\mathbf{a}_f$, which is vector of female parent additive genetic effects,
- $\mathbf{W}_m$ is incidence matrix for $\mathbf{a}_m$, which is vector of male parent additive genetic effects,
- $\mathbf{W}_p$ is incidence matrix for $\mathbf{p}_e$, which is vector of non-genetic permanent environment effects, and
- $\mathbf{e}$ is the residual.

The random effects have the following assumptions:

- $\mathbf{a}_f \sim N(\mathbf{0}, \mathbf{G}_f\sigma_f^2)$,
- $\mathbf{a}_m \sim N(\mathbf{0}, \mathbf{G}_m\sigma_m^2)$,
- $\mathbf{p}_e \sim N(\mathbf{0}, \mathbf{I}\sigma_p^2)$,
- $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}\sigma_e^2)$

where $\mathbf{G}_f$ and $\mathbf{G}_m$ are the female and male genomic relationship matrices, respectively.

The marker data can be used to make both of the genomic relationship matrices and invert them. As in the marker effect model, scaling uses the number of markers and centering of all markers is the same. In our case, the two inverse genomic matrices in the lower triangle dense matrix format are

- females in file `iGL_female.dat`):

```
4 0
 11 12 13 14
 5.032259
 1.161291 5.806452
 3.483871 -.5806450 4.258064
 2.903226 2.516129 1.548387 3.290323
```

- males in file `iGL_male.dat`):

```
4 0
 21 22 23 24
 6.000000
 3.000000 3.750000
```

```
   3.000000 .7500001 3.750000
   3.000000 2.250000 2.250000 3.750000
```

The hybrid GBLUP model can use the GBLUP command for only one of the genomic parental effects. Instead, both parental genomic effects are modeled using the COVFILE command. In the current model, the difference between using the GBLUP and COVFILE commands is small. However, it is worth noting that the GBLUP command is just one approach for associating a covariance structure for additive genetic effects. Other approaches include the PEDIGREE command for pedigree-based relationships and the SSGBLUP command for the single-step GBLUP model, among others.

The CLIM code for this hybrid GBLUP model is

```
DATAFILE gs_obs_repeat_hybrid.dat

INTEGER  id female male mean
REAL     y1 y2
MISSING -99999.

PARFILE  CLIM
1 LOWER     # female genomic
  1.0
  0.7 1.0
2 LOWER     # male genomic
  0.9
  0.6 1.1
3 LOWER     # PE
  0.25
  0.10 0.25
4 LOWER     # residual
  1.00
  0.10 1.00

COVFILE  iGf LOWER iGL_female.dat
COVFILE  iGm LOWER iGL_male.dat

RANDOM iGf iGm PE

MODEL
  y1 = mean iGf(female) iGm(male) PE(id)
  y2 = mean iGf(female) iGm(male) PE(id)
```

The solutions are the same except that no marker effect solutions are estimated, and that the solutions are in different files due model differences.

The general mean solutions are as before in Solfix:

```
 Fact. Trt      Level    N-Obs  Solution              Factor Trait
   1   1          1        12   9.0649                   mean y1
   1   2          1        12   6.0634                   mean y2
```

The permanent environment effect solutions are in Solr03 as it is the third random effect:

```
          1         3 -0.30407     -0.29627
          2         3 -0.43184     -0.41146
          3         3  0.28980      0.28449
```

```
          4          3   0.44610       0.42324
```

The female parent genomic breeding value solutions are in the `Solr01` file:

```
         11          3  -2.9886       -2.9474
         12          3  -1.1675       -1.1404
         13          3   1.3215        1.3134
         14          3   3.5565        3.4893
```

The male parent genomic breeding value solutions are in the `Solr02` file:

```
         21          3 -0.44997      -0.51309
         22          3  -1.0035       -1.0901
         23          3  0.17669E-01   0.43165E-01
         24          3   1.4543        1.5915
```

9.3.10 Example: Hybrid random regression GBLUP model

Chapters 9.2.4 and 9.3.9 illustrated a hybrid model using the marker effect and the GBLUP model approaches, respectively. The GBLUP model approch in MiX99 supports more complex model structures (through commands `GBLUP` and `COVFILE`) than those available for the marker effect models through the `REGMATRIX` command. For example, random regression or reaction norm models are supported.

We extend the hybrid GBLUP model presented in the previous chapter. Instead of a single random effect per individual, assume there are two random effects per individual. The first is as in that example, while the second is a linear term using a covariate available in the data file.

The CLIM code for this hybrid random regression GBLUP model is

```
DATAFILE gs_obs_repeat_hybrid.dat

INTEGER  id female male mean
REAL     y1 y2 x
MISSING -99999.

PARFILE  CLIM
1 LOWER     # female genomic
  1.0       # trait 1 direct
  0.7 1.0   # trait 2 direct
  0.5 0.1 1.0      # trait 1 x
  0.0 0.1 0.5 1.0  # trait 2 x
2 LOWER     # male genomic
  0.9       # trait 1 direct
  0.6 1.1   # trait 2 direct
  0.3 0.05 1.0     # trait 1 x
  0.0 0.05 0.4 1.0 # trait 2 x
3 LOWER     # PE
  0.25
  0.10 0.25
4 LOWER     # residual
  1.00
  0.10 1.00

COVFILE  iGf LOWER iGL_female.dat
COVFILE  iGm LOWER iGL_male.dat

RANDOM iGf iGm PE
```

```
MODEL
  y1 = mean iGf(1 x | female) iGm(1 x | male) PE(id)
  y2 = mean iGf(1 x | female) iGm(1 x | male) PE(id)
```

Please note the order of values in the covariance matrices for female and male genetic effects with random regressions, as specified in the PARFILE command. The values are given column-wise, including both traits (first trait 1 and then trait 2). For a more detailed explanation, please see the example of a multi-trait random regression model in Chapter 6.4.

The general mean solutions in the Solfix file are:

```
 Fact. Trt      Level    N-Obs  Solution              Factor Trait
   1   1          1        12   8.9878                   mean y1
   1   2          1        12   6.0636                   mean y2
```

The permanent environment effect solutions are in Solr03 as it is the third random effect:

```
          1          3 -0.32448     -0.29645
          2          3 -0.45702     -0.41063
          3          3  0.29214      0.28629
          4          3  0.48936      0.42079
```

They are similar to the non-random regression model.

The female parent genomic solutions are in the Solr01 file:

```
         11          3  -2.8680      -2.9565     -0.66982      0.54407E-01
         12          3  -1.0719      -1.1511     -0.22152     -0.62597E-01
         13          3   1.3574       1.3105      0.40094     -0.11906
         14          3   3.4951       3.5118      0.87095     -0.11576
```

Here, columns three and four have the direct genetic effect solutions of traits one and two, respectively. Columns five and six have the genetic effect solutions associated with the random covariate x of traits one and two, respectively.

The male parent genomic breeding value solutions are in the Solr02 file:

```
         21          3 -0.44206     -0.51344     -0.28492E-01  0.62189E-01
         22          3 -0.95798      -1.0938      0.64986E-02 -0.14200
         23          3  0.53781E-01  0.42465E-01  0.14224     -0.13997
         24          3   1.4824       1.5956      0.12616     -0.61435E-01
```

The form of this file follows that for the females.

10 Single-step genomic BLUP

A simple single trait single-step genomic BLUP (ssGBLUP) model is

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{Z}_a\mathbf{a}_g + \mathbf{e}$$

where

- $\mathbf{y}$ is the vector of observations,
- the incidence matrices $\mathbf{X}$ and $\mathbf{Z}_a$ relate observations to fixed effects $\mathbf{b}$ and random additive genetic effects $\mathbf{a}_g$, respectively,

- $\mathbf{e}$ is the vector of random residuals.

The model is similar to that in Chapter 9.3.8, except that the relationship matrix of the breeding values $\mathbf{a}_g$ has both pedigree and genomic information.

The random effects have the following assumptions: $\mathbf{a}_g \sim N(\mathbf{0}, \mathbf{H}\sigma_a^2)$, and $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}\sigma_e^2)$ where $\mathbf{H}$ is the combined relationship matrix that integrates both pedigree-based and genomic relationship matrices. The $\mathbf{H}$ matrix is not needed to be given as input to MiX99. Instead, the inverse $\mathbf{H}^{-1}$ matrix is used:

$$\mathbf{H}^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1} \end{bmatrix}$$

where

- $\mathbf{A}$ is the full pedigree-based relationship matrix,
- $\mathbf{A}_{gg}$ is the pedigree-based relationship matrix part for the genotyped individuals,
- $\mathbf{G}$ is the genomic relationship matrix.

There are many approaches available for the $\mathbf{H}^{-1}$ matrix. In practice, the standard ssGBLUP model in CLIM is similar to the use of a purely pedigree-based relationship matrix model supplemented with the inverse genomic relationship matrix $\mathbf{G}^{-1}$. So, the user has to provide the full pedigree used to make the computations involving inverses of $\mathbf{A}$ and $\mathbf{A}_{gg}$, and provide $\mathbf{G}^{-1}$ matrix in a file.

10.1 Example: standard ssGBLUP

standard ssGBLUP

We illustrate solving a standard ssGBLUP model that requires pre-computing the $\mathbf{G}^{-1}$ matrix and providing it in CLIM using the `SSGBLUP` command. Let the full pedigree used for the computations of the pedigree-based relationship matrices be in the file `one_step.ped`:

ID code$_1$	sire$_2$	dam$_3$
1	2	3
2	4	3
3	5	6
4	5	6
5	0	0
6	0	0
7	5	6
8	2	3

Inbreeding coefficients are in file `full_one_step.inbr`:

ID code$_1$	number$_2$	F$_1$
5	1	0.00000
6	2	0.00000
7	3	0.00000
3	4	0.00000
4	5	0.00000
2	6	0.25000
8	7	0.37500
1	8	0.37500

where the first column has the original individual ID code number, and the last column inbreeding coefficient.

It is essential that the genomic and pedigree-based relationship matrices are on the same scale. A commonly used genomic relationship matrix $\mathbf{G}$ is based on VanRaden's method 1. In this example, we appy a weighted combination of $\mathbf{G}$ and the pedigre-based $\mathbf{A}_{gg}$:

$$\mathbf{G}_w = (1-w)\mathbf{G} + w\mathbf{A}_{gg}$$

where w is the residual polygenic proportion. We assume $w = 0.2$, so that 80% of the total genetic variation is explained by markers, and 20% by residual polygenic effects not captured by the markers.

The blended matrix $\mathbf{G}_w^{-1} = ((1-w)\mathbf{G} + w\mathbf{A}_{gg})^{-1}$ matrix has to be pre-computed and provided in a file. The matrix can be in a file either in

- a co-ordinate sparse matrix format (default) or
- a lower triangle dense matrix format.

Assume the $\mathbf{G}_w^{-1}$ matrix has been stored in the lower triangle dense matrix format in the `iGL_w20.dat` file:

```
6 0
 1 2 3 4 5 6
 2.946921
 -2.265682 2.862358
 -.2843413 .4539492 1.718884
 1.293539 -.8104094 .7088089E-01 3.264837
 .4809364 -.4315277 -.6975392 .2673285 1.563790
 .2329393 -.1191732 -.8711874 -2.032240 -.1566584 2.877456
```

The `SSGBLUP` command is used to indicate the standard single-step GBLUP model. The command has the same options as the `GBLUP` command, including:

- `MIXED`, and
- `LOWER`.

In this example, the variance components are $\sigma_a^2 = 0.5$ and $\sigma_e^2 = 1$. These are in file `one_step.par`:

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	
1	1	1	0.5	polygenic variance
2	1	1	1.0	residual variance

The CLIM code is

```
DATAFILE gs_obs.dat

INTEGER  IDcode mean
REAL     y

SSGBLUP  LOWER iGL_w20.dat

PEDFILE  one_step.ped
PEDIGREE IDcode am
```

```
INBRFILE    full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE     one_step.par

MODEL
  y = mean IDcode
```

The fixed general mean solution (`Solfix`) is

```
 Fact. Trt       Level     N-Obs   Solution              Factor Trait
   1   1           1           4    9.2143                 mean y
```

Estimated breeding values (`Solani`) are

```
          1       0       1  -2.9860
          2       2       1  -1.8368
          3       3       1   1.3077
          4       1       1   2.6580
          5       3       0  0.78904
          6       3       0   2.4818
          7       0       0   1.6354
          8       0       0 -0.26452
```

An alternative to the `SSGBLUP` command is to use the `IGFILE` command to specify the inverse genomic relationship matrix. However, this requires giving `IA22FILE PEDIGREE` command as well:

```
IGFILE    LOWER iGL_w20.dat
IA22FILE PEDIGREE
```

Using these commands will lead to exactly the same computations as using the `SSGBLUP` command.

Note that if `IGFILE` command but no `IA22FILE` command is given, it is assumed that the file specified by `IGFILE` has $\mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$. Furthermore, the pedigree and inbreeding coefficient files given to the `PEDFILE` and `INBRFILE` commands, respectively, has to be the same as used to compute the $\mathbf{A}_{gg}^{-1}$ matrix in order to have a consistently defined and correctly computed single-step model.

10.2 Preconditioner in ssGBLUP

The preconditioner is seldom a very good approximation of the coefficient matrix. In MiX99, (trait-block) diagonal preconditioner is often used (see Chapter 2.3.4). In single-step, this requires knowing the diagonal elements of $\mathbf{H}^{-1}$ for all individuals. For non-genotyped individuals, this is easy, as their values the same those in the diagonal of $\mathbf{A}^{-1}$. However, for genotyped individuals, the diagonal values of $\mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$ are needed as well.

When the `SSGBLUP` command is used, the $\mathbf{G}^{-1}$ matrix is provided. However, the $\mathbf{A}_{gg}^{-1}$ matrix is not explictly computed, neither by the preprocessor (`mix99i`) or the solver (`mix99s`). Consequently, the pre-processor does not have directly available the diagonal elements of the $\mathbf{A}_{gg}^{-1}$ matrix to be used for the preconditioner.

MiX99 has three possible approaches to handle this:

1) Monte Carlo estimation by the preprocessor

The preprocessor can estimate the required diagonal elements using a Monte

Carlo method. This is the recommended approach, as it is makes the computation steps simple. The preprocessor stores the diagonal values of $\mathbf{A}_{gg}^{-1}$ to file `TmpiA22.diag` in the directory defined by the `TMPDIR` command.

2) Precompute diagonal values of $\mathbf{A}_{gg}^{-1}$

This can be done using a separate program such as `calc_diag_iA22`. Diagonal values need to be multiplied by the correct value. When the standard ssGBLUP model is used, MiX99 expects contributions of the diagonal of the $-\mathbf{A}_{gg}^{-1}$ matrix (note the minus sign). For other models, a different multiplier may be required. These contributions can be stored in a file and accounted for using the `IHPRECON` command. Each line of the `IHPRECON` file has the ID code of the genotyped individual and the correction value for the preconditioner. This approach can be computationally faster when the same diagonals are used in multiple evaluations.

3) Omit correcting the diagonal values

It is possible to omit the contributions of the diagonals of the $\mathbf{A}_{gg}^{-1}$ matrix by providing an empty `IHPRECON` file. However, this may negatively affect convergence at some stage of the PCG iteration.

While the first approach is generally preferred, the second approach may sometimes be computationally more efficient. For further details, see the `IHPRECON` command.

CLIM file when the second approach is used:

```
DATAFILE gs_obs.dat

INTEGER IDcode mean
REAL    y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

SSGBLUP  LOWER iGL_w20.dat
IHPRECON minus_diA22.dat

PEDFILE  one_step.ped
PEDIGREE IDcode am

INBRFILE   full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE    one_step.par

MODEL
  y = mean IDcode
```

The `minus_diA22.dat` is

```
1 -2.304
2 -2.55686
3 -3.12071
4 -2.505
5 -1.926
6 -2.018
```

where the first column has ID codes of the genotyped individuals and the second column has their diagonal values in $-\mathbf{A}_{gg}^{-1}$.

10.3 Large number of genotyped individuals in single-step

The genomic relationship matrix ($\mathbf{G}$) is a square matrix of size the number of genotyped individuals. As this matrix increases, both the computational and memory demands increase quadratically. When the number genotyped individuals becomes large, the size of the genomic relationship matrix may grow too large to handle efficiently. This can lead to excessive memory use, as well as long computing times. Some alternative approaches to address these problems are:

Algorithm for Proven and Young (APY)
APY approximates the $\mathbf{G}^{-1}$ matrix by making it sparse in large parts of the matrix (Misztal et al. (2014)). This reduces computational and memory demands significantly.

Single-step genomic BLUP with Woodbury identity (ssGTBLUP)
The ssGTBLUP method decomposes the computations of $\mathbf{G}^{-1}$ using the Woodbury matrix identity (Mäntysaari et al. (2017)). This will lead to using a matrix of size the number of genotyped individuals by the number of markers. Thus, the computational and memory demands increase linearly by the number of genotyped individuals.

Single-step genomic ssSNPBLUP (ssSNPBLUP)
The ssSNPBLUP model is an augmented version of the ssGTBLUP model introduced in Liu et al. (2014).

In MiX99, using APY is like using the standard ssGBLUP model with the full $\mathbf{G}^{-1}$ but replacing it with an APY-based inverse matrix. Please see Chapter 9.3.7 where APY was used for GBLUP. We ask readers to consult the literature on the use of APY as its goodness of predictions depends on the chosen approximation strategy.

In the ssGTBLUP approach, the genomic relationship matrix is assumed to have the form: $\mathbf{G} = \mathbf{Z}_c\mathbf{B}\mathbf{Z}_c' + \mathbf{E}$ where

1) $\mathbf{Z}_c$ is the centered marker matrix,
2) $\mathbf{B}$ is a diagonal matrix containing marker scaling (and possibly weighting) factors,
3) $\mathbf{E}$ is a regularization matrix that is easy to invert.

The use of the Woodbury matrix identity allows expressing the inverse as:

$$\mathbf{G}^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{Z}_c(\mathbf{Z}_c'\mathbf{E}^{-1}\mathbf{Z}_c + \mathbf{B}^{-1})^{-1}\mathbf{Z}_c'\mathbf{E}^{-1}$$

In MiX99, two regularization matrices have been implemented:

1) ssGTeBLUP:
   (a) $\mathbf{E} = \epsilon\mathbf{I}$, where the regularization parameter ϵ is a small number (e.g., 0.01),
   (b) $\mathbf{B} = \mathbf{D}$, where $\mathbf{D}$ is the marker scaling matrix.
2) ssGTABLUP:
   (a) $\mathbf{E} = w\mathbf{A}_{gg}$,
   (b) $\mathbf{B} = (1-w)\mathbf{D}$.

   where w is the residual polygenic proportion.

The marker scaling matrix $\mathbf{D}$ often has the form $\mathbf{D} = \mathbf{I}\frac{1}{k}$, where $k = 2\sum_{i=1}^{m} p_i(1-p_i)$ and p_i is the (base population) allele frequency of marker i.

In both ssGTBLUP models, the inverse matrix $\mathbf{G}^{-1}$ can be expressed using a rectangular matrix $\mathbf{T}$, with dimensions equal to the number of SNP markers by the number of genotyped individuals:

$$\mathbf{G}^{-1} = \mathbf{E}^{-1} - \mathbf{T}'\mathbf{T}.$$

where $\mathbf{T} = \mathbf{L}_C'\mathbf{Z}_c'\mathbf{E}^{-1}$ and $\mathbf{C}^{-1} = \mathbf{L}_C\mathbf{L}_C' = (\mathbf{Z}_c'\mathbf{E}^{-1}\mathbf{Z}_c + \mathbf{B}^{-1})^{-1}$. Alternatively,

$$\mathbf{G}^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{Z}_c\mathbf{C}^{-1}\mathbf{Z}_c'\mathbf{E}^{-1}$$

The first expression is the basis of the standard ssGTBLUP approach, and the second equation defines the component-wise ssGTBLUP approach.

MiX99 supports three alternative ways to define an ssGTBLUP model:

1) Standard ssGTBLUP:

   This approach requires precomputing the $\mathbf{T}$ matrix into a file. The file is an argument to the `SSGTBLUP` command which has two options:

   (a) `TEFILE` for ssGTeBLUP,
   (b) `TAFILE` for ssGTABLUP.

2) Basic component-wise ssGTBLUP:

   In this approach, two matrices need to be precomputed and provided in files:

   (a) `ZCFILE` command defines the file having the centered marker matrix $\mathbf{Z}_c$,
   (b) `ICFILE` command defines the file having the $\mathbf{C}^{-1}$ matrix.

   For example, for ssGTABLUP:

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

3) Fully component-wise ssGTBLUP:

   In this approach:

   (a) `ICFILE` command defines $\mathbf{C}^{-1}$ as in basic component-wise ssGTBLUP,
   (b) `SNPMATRIX` and `SNPFILE` commands are used to define the marker data file with centering information.

All three ssGTBLUP approaches require some level of preprocessing before the `mix99i` can be executed. One or two matrices need to be precomputed.

The basic component-wise ssGTBLUP approach is computationally less demanding in the preprocessing than the standard ssGTBLUP approach. Also, its peak RAM is lower during preprocessing. Furthermore, when the basic component-wise ssGTBLUP is used, the solver `mix99s` (or `mix99p`) can easily compute estimated marker effects to the `SolSNP` file, allowing genomic breeding value estimation for the newly genotyped individuals. However, the basic component-wise ssGTBLUP requires computing and storing a centered marker matrix (used by `ZCFILE`) which is as large as the $\mathbf{T}$ matrix. In addition, the $\mathbf{C}^{-1}$ matrix (defined by `ICFILE`) is needed. Thus, although its advantages, the basic component-wise ssGTBLUP may have too large memory requirements in the solver.

The fully component-wise ssGTBLUP model is a memory efficient version of the basic component-wise ssGTBLUP approach. Instead of the centered marker matrix, the original marker matrix is used. This can be stored efficiently leading to the fully component-wise ssGTBLUP to use less memory than either the standard or basic component-wise ssGTBLUP approach, when the number of genotyped individuals exceeds the number of markers. Details of this method are explained in Chapter 10.4.

All ssGT**A**BLUP models require strict consistency in pedigree information. In particular, the same pedigree file used to compute the $\mathbf{T}$ or $\mathbf{C}^{-1}$ matrix must also be provided via the PEDFILE command. If different pedigrees are used, the computations may be incorrect, because the $\mathbf{A}_{gg}^{-1}$ matrix may be computed inconsistently computed in the solver computations and for $\mathbf{T}$ or $\mathbf{C}^{-1}$. In contrast, ssGTeBLUP models do not use pedigree information in the computation of $\mathbf{T}$ or $\mathbf{C}^{-1}$, and so do not have the same pedigree consistency requirements.

An example CLIM file to use the standard ssGTeBLUP approach is

```
DATAFILE gs_obs.dat

INTEGER IDcode mean
REAL    y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

SSGTBLUP TEFILE TE.bin

PEDFILE  one_step.ped
PEDIGREE IDcode am

INBRFILE full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE  one_step.par

MODEL
  y = mean IDcode
```

For the standard ssGTABLUP approach,

```
SSGTBLUP TAFILE TA.bin
```

where the TA.bin file has the $\mathbf{T}_A$ matrix. Thus, the TAFILE option is used instead of TEFILE. Note that the SSGTBLUP command does not include explicit parameter options for ϵ in ssGTeBLUP or w in ssGTABLUP. This is because the $\mathbf{T}$ matrix file already has this information.

The SSGTBLUP command is similar to the SSGBLUP command in that there is an alternative set of commands. For ssGTeBLUP, it is

```
TEFILE   TE.bin
IA22FILE PEDIGREE
```

For ssGTABLUP, it is

```
TAFILE   TA.bin
IA22FILE PEDIGREE
```

As with ssGBLUP (Chapter 10.1), the $\mathbf{A}_{gg}^{-1}$ matrix is not explicitly computed when the command "IA22FILE PEDIGREE" has been given. The same applies to ssGTBLUP models. However, the preconditioner needs diagonal elements of the $\mathbf{A}_{gg}^{-1}$ matrix. Consequently, by default, the mix99i preprocessor uses a Monte Carlo algorithm to estimate the diagonal elements of $\mathbf{A}_{gg}^{-1}$. For more information, see Chapter 10.2.

If the automatic computations by the Monte Carlo approach are not desired, precomputed diagonal values can be supplied using the IHPRECON command (see Chapter 10.2). For the ssGTeBLUP model, the same file as used for ssGBLUP can be used:

```
IHPRECON minus_diA22.dat
```

where each record in the file must contain individual ID code and its negative diagonal in $\mathbf{A}_{gg}^{-1}$, i.e., $-\mathbf{A}_{gg}^{-1}$. For the ssGTABLUP model, the file must contain diagonal values of $(1/w - 1)\mathbf{A}_{gg}^{-1}$ where w is the residual polygenic proportion. In other words, the diagonals of $\mathbf{A}_{gg}^{-1}$ must be multiplied by $(\frac{1}{w} - 1)$ instead of -1.

The basic component-wise ssGTBLUP approach differs from the standard ssGTBLUP approach in that it requires two input files instead of one. An example of commands required for the basic component-wise ssGTABLUP approach:

```
SSGTBLUP ZCFILE Zc.bin
ICFILE          iC.bin
```

where Zc.bin contains the centered marker matrix $\mathbf{Z}_c$ and iC.bin contains the matrix $\mathbf{C}^{-1} = (\frac{1}{w}\mathbf{Z}_c'\mathbf{A}_{gg}^{-1}\mathbf{Z}_c + \mathbf{B}^{-1})^{-1}$. An example for the basic component-wise ssGTeBLUP model:

```
SSGTBLUP ZCFILE Zc.bin
ICFILE  GTe   iCe.bin
```

where iCe.bin conains the matrix $\mathbf{C_e}^{-1} = (\frac{1}{\epsilon}\mathbf{Z}_c'\mathbf{Z}_c + \mathbf{B}^{-1})^{-1}$.

An example CLIM file to use the basic component-wise ssGTBLUP approach is

```
DATAFILE gs_obs.dat

INTEGER IDcode mean
REAL    y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

SSGTBLUP ZCFILE Zc.bin
ICFILE          iC.bin

PEDFILE  one_step.ped
PEDIGREE IDcode am

INBRFILE full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE  one_step.par

MODEL
  y = mean IDcode
```

Again, there is an alternative way to give the basic component-wise model which requires giving the IA22FILE PEDIGREE command. For ssGTABLUP:

```
ICFILE      iC.bin
ZCFILE      Zc.bin
IA22FILE    PEDIGREE
```

and for ssGTeBLUP:

```
ICFILE GTe  iCe.bin
ZCFILE      Zc.bin
IA22FILE    PEDIGREE
```

Efficient computation of the basic component-wise ssGTBLUP model is quaranteed by having the provided matrices in the RAM memory. To enable this, the solver (mix99s) must use the (default) MEL option. However, when the number of genotyped individuals becomes large, the RAM requirements can become too large. In such cases, it is better to use either fully component-wise ssGTBLUP ssSNPBLUP.

The component-wise ssGTBLUP models do not support the inclusion of UPGs in the preprocessing. Thus, the altered QP H-inverse is typically used. The full QP computations are supported in the MiX99 solver for the basic component-wise ssGTABLUP. However, the use of this solver option leads to increased computations. Currently we recommend using the altered QP H-inverse approach in all single-step models due to its ease of use and its approach for accounting missing pedigree information without using genomic information. For more information on the use of UPGs in single-step models, see Chapter 10.6.

10.4 Fully component-wise single-step GTBLUP model (ssGTBLUP)

In the ssGTBLUP approach described above, both computational and memory demainds increase linearly with the number of genotyped individuals. When the number of genotyped individuals is large, the files needed by the above approaches can become very large. The component-wise ssGTBLUP approach can be programmed to use SNP marker data directly, eliminating the need to store the large centered marker file defined by ZCFILE. However, marker scaling information must still be provided in the file specified by the ICFILE command. We call this the fully component-wise ssGTBLUP approach.

In the basic component-wise ssGTBLUP model, the centered marker matrix is stored using double precision numbers (8 bytes per genotype), Because genotypes are integers (0,1,2), the fully component-wise ssGTBLUP can reduce memory use by storng each genotype using 1 byte, reducing RAM need to 1/8th. Further reduction is achieved using marker byte-packing, where five genotypes are packed into a byte (see Table 10.2)).

The fully component-wise ssGTBLUP requires ICFILE. Consequently, the same preprocessing step to compute this file is needed as described in the previous chapter for the basic component-wise ssGTBLUP approach. However, because the centered marker data file is no longer needed, preprocessing has fewer computations and lower disk memory use. Instead, raw marker genotypes are provided along with centering information.

The genotype file is specified using the SNPFILE command and its content is defined

using the SNPMATRIX command. The FIRST and LAST options of SNPMATRIX define the first and last SNP marker columns, respectively, in SNPFILE. Furthermore, the FORMAT option can be used to specify the Fortran format of the SNP marker file for reading. The CENTER option defines marker centering and may require a file specified by the CENTERFILE command. This file has two columns: the marker number and a value. The value is either an allele frequency or direct centering value, depending on the option given to CENTER.

Numbering of the SNP markers requires some care. In CENTERFILE, markers must start from one. However, the SNPMATRIX command uses column indices, which typically start from two. Thus, marker number one in CENTERFILE corresponds to column two in SNPFILE.

The fully component-wise ssGTBLUP model computes marker effect solutions to the SolSNP file just like the basic component-wise ssGTBLUP approach.

An example for fully component-wise ssGTABLUP reads space separated (FORMAT=m) SNP marker data (SNP markers on columns 2 to 7) and centering by allele frequencies (CENTER=p) in file AF.dat:

```
DATAFILE gs_obs.dat

INTEGER IDcode mean
REAL    y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

SNPMATRIX  FIRST=2 LAST=7 CENTER=p FORMAT=m
SNPFILE    SNP.dat
CENTERFILE AF.dat

ICFILE   iC.bin
IA22FILE PEDIGREE

PEDFILE  one_step.ped
PEDIGREE IDcode am

INBRFILE full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE  one_step.par

MODEL
  y = mean IDcode
```

Note that the ICFILE is used which requires precomputing a matrix. The residual polygenic proportion is included in this matrix and the CLIM command file cannot change that value.

Default genotype storage mode is one genotype in a byte. Packing several genotypes to a byte uses less RAM but can increase computing time. On the other hand, this is compensated by the reduced memory access time due to the lower memory use of the marker matrix compared to the centered marker matrix. The genotype storage approach can be chosen by the USE option in the SNPMATRIX command. In the example above, SNP marker byte-packing is taken to use by giving

```
SNPMATRIX USE=PACK FIRST=2 LAST=7 CENTER=p FORMAT=m
```

All the other commands stay the same.

A more realistic example has more markers and no space between the SNP genotypes. For example,

```
SNPMATRIX  FIRST=2 LAST=52001 CENTER=p FORMAT='(i10,1x,52000i1)'
SNPFILE    markers.dat
CENTERFILE baseAF.dat
ICFILE     iC52.bin
```

Centering of the marker genotypes is based on allele frequencies given in the file defined by the CENTERFILE command. Because this is a component-wise ssGTBLUP model, the ICFILE command has to be given.

Note that the fully component-wise ssGTBLUP model available by the SNPMATRIX command expect that the SNP marker genotypes in the SNPFILE are integer numbers. Genotype value can be 0, 1, or 2, and no missing marker code is allowed. The RAM savings in the fully component-wise computations are possible due to making the marker matrix centering on-the-fly. In practice, only a part or a block of the SNP marker matrix is used to make the values of the $\mathbf{Z}_c$ matrix, i.e., only block-wise computations can be performed. Consequently, the MEL option (assuming the full $\mathbf{Z}_c$ matrix) is not allowed in the solver for the fully component-wise models. Instead, use the default MEA option or define the blocking using the MEB option.

10.5 Single-step marker SNPBLUP model (ssSNPBLUP)

The ssSNPBLUP model introduced by Liu et al. (2014) augments the single-step model by explicit marker effects. Despite the increased number of unknowns in the mixed model equations, the computational cost of per PCG iteration is about the same as in the fully component-wise ssGTBLUP approach (Vandenplas et al. (2023)).

In MiX99, many commands used to define the fully component-wise ssGTBLUP are needed for the ssSNPBLUP model as well and are described in Chapter 10.4. A major difference is that no ICFILE command should be given. Instead, marker scaling information has to be provided, and ssSNPBLUP command must be given.

The ssSNPBLUP model uses the same commands as the fully component-wise ssGTBLUP approach:

- SNPFILE: defines the name of the marker genotype file.
- SNPMATRIX: defines the first and last SNP marker columns used by options FIRST and LAST, respectively. Scaling of markers must be defined by the SCALE option. Furthermore, the FORMAT option can be used to define the Fortran format of the SNP marker file for reading. Centering of the marker information can be defined by the CENTER option.
- CENTERFILE: optional marker-specific centering information in a file.

  This file can have either allele frequencies or direct values for centering. This file has two columns: the marker number and the value.
- ssSNPBLUP: defines the regularization type the model and the value of the regularization parameter.

The SCALE option determines that ssSNPBLUP approach is use instead of fully component-wise ssGTBLUP.

The ssSNPBLUP model is used when the SCALE option is defined in the SNPMATRIX command and the ssSNPBLUP command is given. No ICFILE command should be given. However, if the SNPMATRIX command does not have SCALE option, the fully component-wise ssGTBLUP approach with the ICFILE command is expected.

An example using the ssSNPBLUP model is

```
DATAFILE gs_obs.dat

INTEGER IDcode mean
REAL    y
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

SNPMATRIX  FIRST=2 LAST=7 CENTER=p FORMAT=m SCALE=p
SNPFILE    SNP.dat
CENTERFILE AF.dat
ssSNPBLUP  GTA 0.20

IA22FILE PEDIGREE

PEDFILE  one_step.ped
PEDIGREE IDcode am

INBRFILE full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE  one_step.par

MODEL
  y = mean IDcode
```

The ssSNPBLUP command allows choosing the regularization type of the model:

GTA This ssSNPBLUP model is equivalent to a ssGTABLUP model having the genomic relationship matrix $\mathbf{G}_w = (1-w)\mathbf{Z}_c\mathbf{B}^{-1}\mathbf{Z}_c' + w\mathbf{A}_{gg}$ where w is the residual polygenic proportion, $\mathbf{Z}_c$ is the centered SNP marker matrix, $\mathbf{B}$ is the scaling matrix, and $\mathbf{A}_{gg}$ is the pedigree-based relationship matrix for the genotyped.

GTe This ssSNPBLUP model is equivalent to a ssGTeBLUP model having the genomic relationship matrix $\mathbf{G}_e = \mathbf{Z}_c\mathbf{B}^{-1}\mathbf{Z}_c' + \epsilon\mathbf{I}$ where ϵ is a small number such as 0.01.

The necessary w and ϵ values are defined by the ssSNPBLUP command. For example, ssSNPBLUP GTA 0.2 for an equivalent ssGTABLUP model with $w = 0.2$ or ssSNPBLUP GTe 0.01 for an equivalent ssGTeBLUP model with $\epsilon = 0.01$. Thus, the ssSNPBLUP command is used to define the model type while an equivalent ssGTBLUP model (ssGTABLUP or ssGTeBLUP) has this information available in the file defined by the ICFILE command.

In ssSNPBLUP, the scaling $\mathbf{B}$ matrix information has to be defined using the SCALE option of the SNPMATRIX command. The scaling matrix has the form $\mathbf{B} = \frac{1-w}{k}\mathbf{I}$ for GTA and $\mathbf{B} = \frac{1}{k}\mathbf{I}$ for GTe where k is the scaling constant. The scaling constant k can be either given as a value or chosen from the following options:

p $k = 2\sum_i p_i(1-p_i)$ where p_i is the allele frequency of marker i as defined in the file of CENTERFILE.

m k equals the number of markers m.

m2 k equals half of the number of markers m, i.e., $m/2$.

pm k equals $2m\bar{p}_{MAF}(1-\bar{p}_{MAF})$ where $\bar{p}_{MAF}$ is the average MAF of the markers computed using the CENTERFILE information.

no $k = 1$, i.e., no scaling.

dA k computed such that $tr(\mathbf{Z}_c\mathbf{B}\mathbf{Z}_c') = tr(\mathbf{A}_{gg})$.

one k computed such that $tr(\mathbf{Z}_c\mathbf{B}\mathbf{Z}_c')/n$ equals one.

The ssSNPBLUP command is used to set the residual polygenic proportion in this example to 10% and use the scaling constant $k = 2\sum_i p_i(1-p_i)$:

```
SNPMATRIX  FIRST=2 LAST=52001 CENTER=p FORMAT='(i10,1x,52000i1)' SCALE=p
SNPFILE    markers.dat
CENTERFILE baseAF.dat
ssSNPBLUP  GTA 0.10
```

A component-wise ssSNPBLUP model equivalent to a ssGTeBLUP model has commands

```
SNPMATRIX  FIRST=2 LAST=52001 CENTER=p FORMAT='(i10,1x,52000i1)' SCALE=p
SNPFILE    markers.dat
CENTERFILE baseAF.dat
ssSNPBLUP  GTe 0.01
```

The ssSNPBLUP approach computes estimated solutions of the marker effects to file SolSNP just like the component-wise ssGTBLUP approaches.

10.5.1 Second-level preconditioner in ssSNPBLUP

The ssSNPBLUP model uses less RAM than the fully component-wise ssGTBLUP model because the latter reads the $\mathbf{C}^{-1}$ matrix defined by the ICFILE to RAM. However, the convergence of the ssSNPBLUP model is typically poorer if no second-level preconditioner is used for the marker effects. Thus, it is advised that a second-level preconditioner is defined for the solver via the sp option.

An example of defining the second-level preconditioner to have value 100 on the command line:

```
mix99s -sp 100 -MEA -s -cr 1e-6 -n 5000
```

where -s requests solution files, -cr 1e-6 defines the Cr convergence statistic limit of 10^{-6}, and maximum number of iterations to be 5000.

The same instructions in a command file for mix99s:

```
# RAM: RAM options
H MEA sp 100
# STOP:
5000 1.0e-6 r f
...
```

10.5.2 Marker-specific variances or weights in ssSNPBLUP

DEV

Marker-specific variances in the ssSNPBLUP model can be specified using command SNPPARFILE. The file format for the variances has the same structure as for heterogeneous variances in the REGMATRIX command.

In multi-trait models, this approach allows providing marker-specific and trait-specific (co)variances. For example, consider a two-trait model, extending the previous ssSNPBLUP model by including an additional trait. The CLIM commands can include marker-specific variance matrices:

```
DATAFILE gs_obs_2tr.dat

INTEGER IDcode mean
REAL    y1 y2
MISSING -99999.
DATASORT PEDIGREECODE=IDcode

SNPMATRIX  FIRST=2 LAST=7 CENTER=p FORMAT=m SCALE=p
SNPFILE    SNP.dat
CENTERFILE AF.dat
ssSNPBLUP  GTA 0.20
SNPPARFILE gs_gen_heter.par

IA22FILE PEDIGREE

PEDFILE  one_step.ped
PEDIGREE IDcode am

INBRFILE   full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE CLIM
1 LOWER
  0.5
  0.5 1.0
2 LOWER
  1.0
  0.0 1.5

MODEL
  y1 = mean IDcode
  y2 = mean IDcode
```

The marker variance file for this example can be:

Marker number	Row	Column	Covariance	
1	1	1	0.50	marker 1, trait 1 variance
1	2	1	0.50	marker 1, covariance
1	2	2	1.00	marker 1, trait 2 variance
2	1	1	0.02	marker 2, trait 1 variance
2	2	1	0.00	marker 2, covariance
2	2	2	0.30	marker 2, trait 2 variance
3	1	1	0.70	marker 3, trait 1 variance
3	2	1	0.00	marker 3, covariance
3	2	2	0.10	marker 3, trait 2 variance

4	1	1	0.40	marker 4, trait 1 variance
4	2	1	0.40	marker 4, covariance
4	2	2	0.80	marker 4, trait 2 variance
5	1	1	0.01	marker 5, trait 1 variance
5	2	1	0.00	marker 5, covariance
5	2	2	0.02	marker 5, trait 2 variance
6	1	1	0.60	marker 6, trait 1 variance
6	2	1	0.60	marker 6, covariance
6	1	1	1.20	marker 6, trait 2 variance

Sometimes only marker-specific variances have been estimated but no correlations between traits by marker. These variance can be transformed into weights by normalizing them to have an average value of one. These marker-specific weights by trait can be provided to the ssSNPBLUP model. In this approach, it is assumed that the trait correlations are the same as those defined in the PARFILE for the additive genetic effect. CLIM commands are then:

```
SNPMATRIX  FIRST=2 LAST=7 CENTER=p FORMAT=m SCALE=p DWEIGHT=T
SNPFILE    SNP.dat
CENTERFILE AF.dat
ssSNPBLUP  GTA 0.20
WEIGHTFILE weights.dat
```

Note the use of the DWEIGHT option. When this option has 'T', the WEIGHTFILE is expected to have marker weights for each trait in a separate column. In our small example, the marker weight file can be:

Marker number	trait 1 weight	trait 2 weight	
1	0.50	1.00	marker 1
2	0.02	0.30	marker 2
3	0.70	0.10	marker 3
4	0.40	0.80	marker 4
5	0.01	0.02	marker 5
6	0.60	1.20	marker 6

Note that the marker weights in this file do not need to have an average of one per trait. The preprocessor mix99i automatically scales these weights such that their average is one.

A further simplification is possible when the marker-specific variances are not trait-specific. Thus, the same weight is used across traits but the weights can differ by marker. Then the CLIM commands can be:

```
SNPMATRIX  FIRST=2 LAST=7 CENTER=p FORMAT=m SCALE=p DWEIGHT=M
SNPFILE    SNP.dat
CENTERFILE AF.dat
ssSNPBLUP  GTA 0.20
WEIGHTFILE weights_M.dat
```

Now there is only one column of marker weights in WEIGHTFILE. The marker weight file can be:

Marker number	weight	
1	0.70	marker 1
2	0.15	marker 2
3	0.40	marker 3
4	0.60	marker 4
5	0.02	marker 5
6	0.80	marker 6

For further explanation of marker-specific (co)variances or weights in ssSNPBLUP model, please see Strandén and Jenko (2024) where some convergence properties are explained.

10.6 Unknown parent groups in a single-step model

Unknown parent groups (UPGs) can be used to replace missing parents in the pedigree, as described for pedigree-based models (Chapter 3.3.2). Then, the `PEDIGREE` command has `am+p` to include UPGs in the pedigree. When applying single-step models, there are two commonly used approaches for including UPGs:

- Full QP transformed H-inverse (Matilainen et al. (2018)).
- Altered QP transformed H-inverse (Masuda et al. (2021)).

The full QP approach uses both the pedigree-based and genomic relationship matrices for the UPG equations. In contrast, the altered QP approach uses only the pedigree-based relationship matrices for the UPG equations. In MiX99, all single-step models support the altered QP approach, which is the recommended method due to its simplicity and lower bias than the full QP approach (Belay et al. (2022), Himmelbauer et al. (2024)).

The full QP can be used with the standard ssGBLUP and ssGTBLUP models, because the files for the `SSGBLUP`, `TAFILE` or `TEFILE` commands can include the UPG information. These files allow UPG to be augmented:

- standard ssGBLUP

  In the `hginv` program, the '-QP' option augments $\mathbf{G}^{-}1$ with UPGs.

- ssGTBLUP

  In the `T48eig_make`, the '-groups' option augments the $\mathbf{T}$ matrix with UPGs.

However, the component-wise single-step models cannot include UPGs in the external genomic files. The basic component-wise ssGTABLUP model supports the full QP model via the `-fQP` option in `mix99s`, but with an increased computational cost. In MiX99, the fully component-wise ssGTBLUP and ssSNPBLUP models allow only the use of the altered QP H-inverse approach.

Differences between the altered and the full QP transformed H-inverse approaches are illustrated below. Consider the linear mixed model

$$\mathbf{y} = \mathbf{Xb} + \mathbf{WQg} + \mathbf{Wa} + \mathbf{e}$$

where the incidence matrices $\mathbf{X}$ and $\mathbf{W}$ are used to indicate the approriate fixed effects ($\mathbf{b}$ and $\mathbf{Qg}$) and random additive genetic values ($\mathbf{a}$) to the observation vector $\mathbf{y}$, and $\mathbf{e}$

has the residual. The $\mathbf{Q}$ matrix has gene proportions for each unknown parent group per individual according to the pedigree information. Assume: $\text{Var}(\mathbf{a}) = \mathbf{H}\sigma_a^2$ and $\text{Var}(\mathbf{e}) = \mathbf{R}$ where σ_a^2 is the genetic variance and the $\mathbf{R}$ matrix includes the residual variance. The mixed model equations for this model is

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{W}\mathbf{Q} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{Q}'\mathbf{W}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Q}'\mathbf{W}'\mathbf{R}^{-1}\mathbf{W}\mathbf{Q} & \mathbf{Q}'\mathbf{W}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{W}\mathbf{Q} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{W} + \mathbf{H}^{-1}\sigma_a^{-2} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{g}} \\ \hat{\mathbf{a}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Q}'\mathbf{W}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

The estimated breeding values are

$$\hat{\mathbf{u}} = \mathbf{Q}\hat{\mathbf{g}} + \hat{\mathbf{a}}.$$

After the QP transformation of the above mixed model equations, the estimated breeding values $\hat{\mathbf{u}}$ are computed directly without the need for a post-processing step to add the group effect estimates to the estimated additive genetic values.

The full QP approach includes both pedigree and genomic information in the UPG equations. The mixed model equations are:

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{0} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{0} & \mathbf{Q}'\mathbf{H}^{-1}\mathbf{Q}\sigma_a^{-2} & -\mathbf{Q}'\mathbf{H}^{-1}\sigma_a^{-2} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{X} & -\mathbf{H}^{-1}\mathbf{Q}\sigma_a^{-2} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{W} + \mathbf{H}^{-1}\sigma_a^{-2} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{g}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{0} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

where $\mathbf{H}^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1} \end{bmatrix}$.

In the altered QP approach (Masuda et al. (2021)), UPGs are include in the pedigree but not in the external genomic file ($\mathbf{G}^{-1}$ or $\mathbf{T}$ etc.). In practice, UPG effect equations use the pedigree-based relationship matrices but not the genomic relationship matrix. The mixed model equations with altered QP H-inverse for the above model are

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{0} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{0} & \mathbf{Q}'\mathbf{H}_A^{-1}\mathbf{Q}\sigma_a^{-2} & -\mathbf{Q}'\mathbf{H}_A^{-1}\sigma_a^{-2} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{X} & -\mathbf{H}_A^{-1}\mathbf{Q}\sigma_a^{-2} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{W} + \mathbf{H}^{-1}\sigma_a^{-2} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{g}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{0} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

where $\mathbf{H}_A^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{A}_{gg}^{-1} \end{bmatrix}$. Note that the original $\mathbf{H}^{-1}$ is still used for the breeding values but elsewhere in the equations it has been replaced by $\mathbf{H}_A^{-1}$.

10.7 Metafounders

Metafounders (MF) can be used instead of unknown parent groups in any model having pedigree-based relationship matrix (Legarra et al. (2015)). However, it is mostly used in single-step models as it allows alligning the pedigree-based relationship matrix with the genomic relationship matrix. In a MF model, MF are used in place of missing parents in the pedigree file, and included in the pedigree as individuals. A covariance matrix of MF, denoted by $\mathbf{\Gamma}$, has to be estimated before using MFs in a model.

In MiX99, there are two practical differences in the metafounders compared to unknown parent groups:

1) The $\mathbf{\Gamma}^{-1}$ matrix

   MiX99 requires the inverse of the $\mathbf{\Gamma}$ matrix to be provided in co-ordinate sparse matrix format. This can be done using the following command:

```
IGAMMAFILE MIXED iGamma.dat
```

where `iGamma.dat` has the $\mathbf{\Gamma}^{-1}$ matrix.

2) Inbreeding coefficients

   In a MF model, inbreeding coefficients must be computed differently. In particular, the computations use the $\mathbf{\Gamma}$ matrix. The inbreeding coefficients can be computed using the RelaX2 program. The inbreeding coefficients of MFs (negative values) are included in the file having the inbreeding coefficients of all individuals.

MiX99 requires that the metafounders must be positive integer numbers just like any individual ID code in the pedigree, but unlike unknown parent groups. These MF numbers are used as position values in the file defined by `IGAMMAFILE`. Furthermore, each MF must have a record in the pedigree file with both parents unknown (coded as zero), and in the inbreeding coefficients file. This quarantees compatibility in processing the pedigree information. Note that an individual (not a MF) in a pedigree file is not allowed to have any missing parents when MF are present in the pedigree. All other model instructions and commands in MiX99 remain unchanged except the inclusion of the $\mathbf{\Gamma}^{-1}$ matrix using the `IGAMMAFILE` command.

10.8 Summary of single-step models

MiX99 supports many equivalent single-step models. (see Table 10.2). In these models, external files differ in numbers and content.

- Standard ssGBLUP model needs only one external file containing the inverse of the genomic relationship matrix.
- Standard ssGTBLUP models use one external file containing the $\mathbf{T}$ matrix.
- Basic Component-wise ssGTBLUP divides the information in the $\mathbf{T}$ matrix into two files.
- Fully Component-wise ssGTBLUP need multiple external files that include genotype, marker centering and scaling information.
- ssSNPBLUP model requires genotype, marker centering and scaling information.

In general, no single-step model is universally optimal. Their performance depends on factors such as the number of genotyped individuals, the number of markers, variance component values, and data structure. These affect computing time, convergence behaviour and memory requirements, both in the preprocessing and solving steps.

RAM memory use vary significantly across different single-step models (see the RAM column in Table 10.2). Consider a data having 200,000 ($n = 200,000$) or 1 million genotyped individuals ($n = 1,000,000$) and 50,000 SNP markers ($m = 50,000$). Estimated minimum RAM use due to genomic information is (1 million genotyped in parenthesis):

• Standard ssGBLUP:	~ 320 GB	(8 TB)
• Standard ssGTBLUP:	~ 80 GB	(400 GB)
• Basic component-wise ssGTBLUP:	~ 100 GB	(420 GB)
• Fully component-wise ssGTBLUP:	~ 30 GB	(70 GB)
with byte-packed SNPs:	~ 22 GB	(30 GB)
• ssSNPBLUP:	~ 10 GB	(50 GB)
with byte-packed SNPs:	~ 2 GB	(10 GB)

Table 10.2: RAM requirements and equation for the inverse genomic relationship matrix for various single-step approaches.

Single-step model	c[1]	Z[2]	$\mathbf{G}^{-1}$ formula[3]	RAM[4]
Standard ssGBLUP	-	-	$\mathbf{G}^{-1}$	$8n^2$
Standard ssGTABLUP	N	-	$\frac{1}{w}\mathbf{A}_{gg}^{-1}-\mathbf{T}'_A\mathbf{T}_A$	$8nm$
Standard ssGTeBLUP	N	-	$\frac{1}{\epsilon}\mathbf{I}-\mathbf{T}'_e\mathbf{T}_e$	$8nm$
Basic component-wise ssGTABLUP	Y	N	$\frac{1}{w}\mathbf{A}_{gg}^{-1}-\frac{1}{w}\mathbf{A}_{gg}^{-1}\mathbf{Z}_c \times\mathbf{C}_w^{-1}\mathbf{Z}'_c\frac{1}{w}\mathbf{A}_{gg}^{-1}$	$8m(n+m)$
Basic component-wise ssGTeBLUP	Y	N	$\frac{1}{\epsilon}\mathbf{I}-\frac{1}{\epsilon}\mathbf{Z}_c\mathbf{C}_e^{-1}\mathbf{Z}'_c\frac{1}{\epsilon}$	$8m(n+m)$
Fully component-wise ssGTABLUP	Y	Y	$\frac{1}{w}\mathbf{A}_{gg}^{-1}-\frac{1}{w}\mathbf{A}_{gg}^{-1}\mathbf{Z}_c \times\mathbf{C}_w^{-1}\mathbf{Z}'_c\frac{1}{w}\mathbf{A}_{gg}^{-1}$	$nm+8m^2$ or $\frac{1}{5}nm+8m^2$
Fully component-wise ssGTeBLUP	Y	Y	$\frac{1}{\epsilon}\mathbf{I}-\frac{1}{\epsilon}\mathbf{Z}_c\ \mathbf{C}_e^{-1}\mathbf{Z}'_c\frac{1}{\epsilon}$	$nm+8m^2$ or $\frac{1}{5}nm+8m^2$
ssSNPBLUP of ssGTABLUP	Y	Y		nm or $\frac{1}{5}nm$
ssSNPBLUP of ssGTeBLUP	Y	Y		nm or $\frac{1}{5}nm$

[1] = component-wise computations (Yes/No),
[2] = marker matrix packing available (Yes/No),
[3] = where w is the residual polygenic proportion, and ϵ is a small number (e.g., 0.01),
[4] = approximate memory use in bytes due to the genomic model data, n = number of the genotyped, m = number of the SNP markers.

These figures are theoretical estimates based on file sizes used in single-step analysis. Actual RAM use increase depending on the number of unknowns, pedigree complexity, and other data specific factors. When the number of genotyped individuals is smaller, differences in RAM use between models diminish. In particular, if the number of genotyped individuals is less than the number of SNP markers, the standard ssGBLUP model uses less RAM than any ssGTBLUP approach. The ssSNPBLUP model may still use less RAM than the standard ssGBLUP model when marker matrix byte-packing is used.

Number of floating-point operations consumed in the computations of genomic data in single-step methods mostly depends on the number of genotyped individuals (n) and the number of markers (m). For standard ssGBLUP, the computational complexity is about $O(n^2)$. For the standard ssGTBLUP, it is $O(2nm)$ operations. For the component-wise ssGTBLUP and ssSNPBLUP models, it is $O(2nm+m^2)$. When the number of genotyped individuals is large, the component-wise ssGTBLUP and ssSNPBLUP approaches have about the same number of floating point operations within each PCG iteration. Furthermore, because the fully component-wise approaches use less RAM than the standard approaches, memory access and computing times are faster in the fully component-wise than in the standard single-step approaches, when the number of genotyped is large. However, there are differences between the fully component-wise approaches due to the amount of preprocessing needed and convergence of PCG. In general, ssGTBLUP requires more preprocessing than ssSNPBLUP which has often a slower convergence. Neverthelss, for large genotyped data, ssSNPBLUP model is often computationally the most efficient (Vandenplas et al. (2023)).

The practical use of single-step models differ when performing multiple evaluations:

- Standard ssGBLUP allows the reuse of the precomputed inverse genomic relationship matrix.
- ssGTeBLUP is simularly robust allowing the reuse of the same $\mathbf{T}_e$ matrix provided by the `TEFILE` command.
- ssGTABLUP model requires consistent pedigree information across evaluation runs when the same precomputed $\mathbf{T}_A$ (via `TAFILE`) or $\mathbf{C}^{-1}$ (via `ICFILE`) matrix is used. In other words, the solver's computation of the $\frac{1}{w}\mathbf{A}_{gg}^{-1}$ matrix times vector products must align with how the $\mathbf{T}_A$ or $\mathbf{C}^{-1}$ matrix has been computed. Currently, the preprocessor `mix99i` does not verify that the provided pedigree is the same as used in making $\mathbf{T}_A$ or $\mathbf{C}^{-1}$.
- ssSNPBLUP has no preprocessing needs to genomic information making it a robust choice for multiple evaluations.

11 Special topics

11.1 Trait groups for single trait analysis

11.1.1 Example: Multiple single trait analysis

It is common that several single trait analyses use the same pedigree and data file but observations are on different columns. Still, a multi-trait model is not used due to unknown variance components between the traits. Thus, although the data can be presented as for a multi-trait model, all residual and genetic covariance values between the traits are zero. This can be analyzed as a multi-trait model by MiX99 . However, this can be inefficient because the data file may have many missing observations, and the traits have different effects.

Trait groups can be used to make the analysis more efficient. Now, the data is given similarly to the repeatability model. However, instead of a repeatability model, there is a trait group indicator to indicate which model is used. In the example model, the trait group has observations from one trait.

We consider again the multi-trait model example with different effects by trait in Chapter7.2.1). The model is

$$\begin{array}{lcll} trait1 & = & herd \times year & +a+e \\ trait2 & = & \mu & +a+e \end{array}$$

However, now the interest is in analyzing these two traits as separate independent evaluations in the same MiX99 solver run. The multi-trait model way would be to do as in Chapter 7.2.1 but with a parameter file where all covariance values are zero.

The trait group way is to make the data to be similar to the repeatability data (Chapter 6.2.1) but with an additional column to indicate a trait or a trait group. The data file (`example_tr_group.dat`) is

ID code$_1$	sire$_2$	herd year$_3$	ones$_4$	trait$_5$	trait12$_1$
4	1	11	1	1	90
4	1	21	1	2	200
6	3	11	1	1	110
6	3	21	1	2	190
8	5	12	1	1	120
8	5	22	1	2	140
9	5	12	1	1	130
9	5	22	1	2	120
10	7	12	1	1	120
10	7	22	1	2	130

The parameter file (`mt_single.var`) is the same as for the multi-trait model analysis described above:

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$
1	1	1	3.0
1	2	2	2.5
2	1	1	7.0
2	2	2	7.0

Column `trait` is used to indicate the model of the trait. It is referenced by the trait group number in parenthesis on the model line. Command `TRAITGROUP` is needed to indicate which column is the trait group column in the data file. The CLIM file would be

```
DATAFILE   example_tr_group.dat

INTEGER    IDcode sire herdXyear ones trait
REAL       tr

TRAITGROUP trait

PEDFILE    data/my.ped
PEDIGREE   IDcode am

PARFILE    mt_single.var

MODEL
  tr(1) = -     herdXyear IDcode
  tr(2) = ones  -         IDcode
```

Fixed effect solutions (`Solfix`) are

```
 Fact. Trt       Level     N-Obs   Solution            Factor Trait
   1   2           1           5   160.73               ones tr
   2   1          11           2   99.538            herdXyea tr
   2   1          12           3   122.69            herdXyea tr
```

Breeding value estimates in the `Solani` file are

```
    ID code N-Desc  N-Obs     trait1         trait2
          1      2      0 -.29831E-05 0.29868E-05
          2      2      0 -.29831E-05 0.29868E-05
          3      2      0 0.92308      -3.2188
          4      2      2 -.92308       3.2188
          5      3      0 -.12098E-04 -5.8818
```

```
         6       3       2  1.8462      -.55584
         7       1       0 0.65422      -5.0727
         8       1       2 0.64449E-01 -7.4451
         9       0       2  2.0506      -8.9024
        10       0       2 -.17839      -9.9667
```

Solutions can be compared with those for the `trait1` single trait evaluation in Chapter 6.1. Solutions differ slightly. The reason is that the multi-trait model analysis usually makes more iterations. For example, in the example above, the multi-trait model needed 18 PCG iterations but the single trait analysis converged in 12 iterations. There are 12 unknowns in the single trait model, and, in theory, only 12 iterations are needed. However, the multi-trait model has 24 unknowns, although in two separate blocks with 12 unknowns. The PCG method tries to solve the two separate systems at the same time. Solving and convergence in the separate blocks are compromised.

11.1.2 Example: MACE or Sire model with weights and trait groups

We consider a multi-trait sire model where yields of daughters in different countries are considered different traits. This is the MACE example described in Schaeffer (1994). The model is

$$\begin{aligned} y_1 &= \mu_1 + g_1 + s_1 + e_1 \\ y_2 &= \mu_2 + g_2 + s_2 + e_2 \end{aligned}$$

where subscript is for countries 1 and 2, μ is country genetic base, *y* is bull's daughter yield deviation (DYD), *g* is genetic group effect of unknown parents, *s* is sire transmitting ability by country, and *e* is residual.

The sire genetic effects and the residuals have the following assumptions:

$$\begin{aligned} \mathrm{E}(\boldsymbol{s}) &= \boldsymbol{0} \quad \mathrm{Var}(\boldsymbol{s}) = \boldsymbol{G}_0 \otimes \boldsymbol{A} \\ \mathrm{E}(\boldsymbol{e}) &= \boldsymbol{0} \quad \mathrm{Var}(\boldsymbol{e}) = \boldsymbol{R} \end{aligned}$$

These are like the standard multi-trait model assumptions. However, in the MACE model, the residual covariance matrix $\boldsymbol{R}$ is diagonal, i.e., residual correlations are zero and vary by sire. The residual covariance matrix for sire *i* is

$$\boldsymbol{R}_i = \begin{bmatrix} d_{1_i}\sigma^2_{e_1} & 0 \\ 0 & d_{2_i}\sigma^2_{e_2} \end{bmatrix}$$

where *d* equals one over the number of daughters in a bull's DYD, and $\sigma^2_{e_j}$ is the residual variance for country *j*. The d_i values in the residual covariance matrix can be considered as weights. Weights can be defined for each trait separately by option `weight` after the model line. In Schaeffer (1994), the genetic covariance matrix is

$$\boldsymbol{G}_0 = \begin{bmatrix} 100 & 20 \\ 20 & 5 \end{bmatrix}$$

The residual variances are $\sigma^2_{e_1}$ = 1000, and $\sigma^2_{e_2}$ = 80.

A trait group has one or more traits that can be observed together from an individual but cannot be observed with any trait belonging to another trait group. The residual correlation between trait groups is zero. This definition of trait group matches with the MACE model where a country is a trait group. In practice, observation belongs to a trait group specified by an integer number column. The appropriate trait group number is given in parenthesis after the observation name. For example, trait `protein` in trait group 1 is `protein(1)`, but in group 2, `protein(2)`.

The parameter file `MACE.var` is

Table 11.1: The pedigree and data files for the MACE model example.

pedigree file MACE.ped Table 2 in Schaeffer (1994)				data file MACE.dat Table 1 in Schaeffer (1994)			
bull$_1$	sire$_2$	MGS[1]$_3$	MGD[2]$_4$	bull$_1$	country$_2$	protein$_1$	weight[3]$_2$
1	6	7	−5	1	1	56	10
2	8	9	−5	2	1	−23	20
3	10	8	−5	3	1	8	50
4	10	11	−6	1	2	9	100
5	2	6	−6	4	2	3	40
6	−1	−2	−6	5	2	−11	20
7	−1	−2	−6				
8	−1	−2	−6				
9	−3	−4	−6				
10	−3	−4	−6				
11	−3	−4	−6				

[1]maternal grandsire
[2]maternal grandam
[3]daughter yield deviation (DYD)

Random effect$_1$	Row$_2$	Column$_3$	Covariance$_1$	Variance
1	1	1	100	genetic
1	2	1	20	genetic
1	2	2	5	genetic
2	1	1	1000	residual
2	2	2	80	residual

```
TITLE       MACE, L.Schaeffer (1994)

DATAFILE    MACE.dat
INTEGER     bull country
REAL        protein weight

TRAITGROUP  country

PEDFILE     MACE.ped
PEDIGREE    bull sm+p 1.0 # sm=sire model

PARFILE     MACE.var

MISSING    -8192.0

MODEL
   protein(1) = country bull ! WEIGHT=weight
   protein(2) = country bull ! WEIGHT=weight
```

Estimates of the country means (`Solfix`) are

```
 Fact. Trt       Level     N-Obs   Solution            Factor Trait
   1   1           1         3     10.497            country protein
```

```
  1   2          2          3  1.2141             country protein
```

Breeding value estimates (`Solani`) by country are

```
                              Country
      Bull N-Desc  N-Obs     1           2
         1      0      2  31.132       7.0211
         2      1      1 -26.538      -5.9504
         3      0      1 -2.4056      -.47722
         4      0      1  4.3014       1.1245
         5      0      1 -29.221      -7.0674
         6      2      0  10.936       2.3169
         7      1      0  8.9970       2.0117
         8      2      0 -12.881      -2.9245
         9      1      0 -8.3029      -1.8438
        10      2      0  1.4727      0.43745
        11      1      0 0.46456E-01 0.69527E-01
        -4      3      0 -.49058      -.76316E-01
        -3      3      0 -.98117      -.15263
        -2      3      0  1.2948      0.27736
        -1      3      0  2.5895      0.55472
        -5      3      0  1.5631      0.39077
        -6      8      0 -3.9756      -.99390
```

11.2 Deregression

Deregression in MiX99 is based on Jairath et al. (1998) and Schaeffer (2001). Calculation of deregressed proofs means solving a non-linear system of equations. The system of equations looks the same as regular mixed model equations. However, it is assumed that solutions for some individuals (for which deregressed proofs are needed) are known but solutions of their ancestors, unknown parent groups, and the general mean in the model are unknown. In addition, the deregressed proofs are unknown. In the mixed model equations, the deregressed proofs are in the right-hand side of the equation.

In MiX99 (or `mix99s`), the non-linear deregression problem is solved by a two-step iterative process:

1) solve ancestral and unknown parent group unknowns given the current solution for the general mean estimate and the known proofs.
2) calculate a new estimate for the general mean.

In practice, the first step means solving mixed model equations which is the core work done in MiX99 . The model to solve ancestral and unknown parent groups has only one fixed effect: general mean. Another important aspect of a deregression model is that the unknown parent groups must be random.

Many methods can be used in solving non-linear systems. MiX99 has the following methods

- Broyden
- Secant
- Gauss-Seidel
- Bisection

The default method is the Broyden method which is often the fastest and most reliable of the implemented methods. Secant method, however, can be better when solving many single-trait models. Note that all the methods should lead to the same deregressed values, but default convergence criterion may terminate the iteration prematurely.

Deregression using `mix99s` needs to be specifically requested. A directive file for the program is convenient to make as was described in Chapter 4. Let the directive file (`dereg.slv`) be

```
H    # RAM:     Demand: H=high
     # STOP:
     # Max.no.iter., Tolerance, Criterion, Force, Max.no.iter. Non-linear
       5000           1.0e-3     D          F      1000
N    # RESID:   Calculate residuals? (Y/N)
R b  # VALID:   Model validation: R=deregression
               # b= Broyden method
               # s= Secant method
               # i= bisection
               # n= Gauss-Seidel
N    # VAROPT: Variance options VCE, PEV, HV?  No
Y    # SOLTYP: Solution file options? Yes
```

Solving a deregression model has some important differences from the regular breeding value estimation. Deregression is requested (letter 'R') with a Broyden method (letter 'b'). In addition, an additional number of `1000` is given on the line for the maximum number of iterations and the convergence criterion. The additional number `1000` is the maximum number of iterations for the non-linear solving method, which is the Broyden method in this example. Solving a deregression model is a non-linear problem. Consequently, two iterative methods are used: PCG iteration to solve a linear problem, and the Broyden method for the non-linear problem. The Broyden method may sometimes fail to converge or is very slow to converge. Then, another method (e.g. secant method) can be used, or the problem needs reformulation (e.g. new definition of unknown parent groups, or different random coefficient value). When deregression is done for uncorrelated traits having different amounts of information, computations may be slower than by single-trait deregression because some trait are much slower to converge than others.

11.2.1 Example: Single trait deregression

The earlier pedigree for the animal and sire model examples is used. However, it is modified for deregression purposes. The sire model pedigree (`sm_dereg.ped`) is

bull$_1$	sire$_2$	maternal grand sire$_3$	maternal grand dam group$_4$
1	-2	-2	-10
3	1	-2	-10
5	3	1	-11
7	5	3	-11

There is a fourth column for the maternal grand dam group. For animal model, this pedigree (`am_dereg.ped`) is

individual$_1$	sire$_2$	dam$_3$
1	−2	11
3	1	2
5	3	4
7	5	6
2	−2	−10
4	1	−11
6	3	−11
11	−2	−10

The data is (`dereg.dat`)

sire$_1$	ones$_2$	proof$_1$	EDC$_2$
1	1	-.35736	50
3	1	0.40621	100
5	1	0.20334	80
7	1	0.24076E-01	20

The variance components file (`SM.var`) is the same as before

Random effect$_1$	Row$_2$	Column$_3$	Variance$_1$
1	1	1	0.75
2	1	1	9.25

Sire model The deregression model is very simple: only the general mean and the sire effect. It is important to use random unknown parent groups. The CLIM code for deregression (`sm_dereg.clm`) is

```
DATAFILE   dereg.dat

INTEGER    sire ones
REAL       ebv EDC

PEDFILE    smgs.ped
PEDIGREE   sire sm+p 1.0

PARFILE    SM.var

MODEL
  ebv = ones sire ! WEIGHT=EDC
```

Calculating deregressed proofs (`mix99s < dereg.slv`) will give a solution for the general mean (`Solfix`)

```
 Fact. Trt       Level     N-Obs   Solution              Factor Trait
   1   1           1           4 0.18958E-01               ones ebv
```

and the deregressed proofs (`Solani`):

```
        sire N-Desc  N-Obs deregressed proof
           1      2      1 -.54488
           3      2      1 0.49919
           5      1      1 0.24384
           7      0      1 -.13403
         -11      2      0  0.0000
         -10      2      0  0.0000
```

```
        -2       3       0   0.0000
```

Solutions for the unknown parent groups (negative id code) can be ignored because these solutions have been set to zero by MiX99 . In general, deregressed proofs are those in the `Solani` file that have an observation, i.e., `N-Obs` is one.

Animal model The CLIM code for the individual animal model is very similar to the sire model case above:

```
DATAFILE   dereg.dat

INTEGER    IDcode ones
REAL       ebv EDC

PEDFILE    am_dereg.ped
PEDIGREE   IDcode am+p 1.0

PARFILE    SM.var

MODEL
  ebv = ones IDcode ! WEIGHT=EDC
```

General mean solution (`Solfix`) is the same as before. Solutions are the same (`Solani`):

```
    ID code N-Desc  N-Obs deregressed proof
          1       2       1 -.54488
          3       2       1 0.49919
          5       1       1 0.24384
          7       0       1 -.13403
          2       1       0  0.0000
          4       1       0  0.0000
          6       1       0  0.0000
         11       1       0  0.0000
        -11       2       0  0.0000
        -10       2       0  0.0000
         -2       3       0  0.0000
```

Naturally, there are now more solutions because the pedigree had more individuals. As before, only solutions with observations (`N-Obs` equal to one) are relevant.

11.2.2 Example: Multiple trait deregression

Multi-trait deregression is done the same way as single-trait deregression. We illustrate multi-trait deregression by the example given by Schaeffer (2001) where a detailed explanation can be found. The example is on multi-trait sire model deregression for international bull evaluation. We consider only the example data for country A. Country B proceeds similarly.

Sire model pedigree (`sch_sm.ped`) is

bull$_1$	sire$_2$	maternal grand sire$_3$	maternal grand dam group$_4$
1	−22	−23	−24
2	−22	−23	−24
3	−22	−23	−24
4	−22	−23	−24

5	−22	−23	−24
6	−25	−26	−27
7	−25	−26	−27
8	−25	−26	−27
9	−25	−26	−27
10	−25	−26	−27
11	−25	−26	−27
12	1	2	−28
13	3	4	−28
14	3	5	−28
15	6	2	−28
16	6	7	−29
17	3	8	−29
18	3	9	−29
19	3	10	−29
20	11	8	−29
21	11	3	−29

As before, there is a fourth column for the maternal grand dam group. The data (`sch_cntry_A.dat`) has three lactations:

		Lactation 1		Lactation 2		Lactation 3	
ones$_1$	sire$_2$	Progeny$_1$	EBV$_2$	Progeny$_3$	EBV$_4$	Progeny$_5$	EBV$_6$
1	12	126	23	0	−999	0	−999
1	12	43	23	43	34	0	−999
1	12	36	23	36	34	36	38
1	13	18	36	0	−999	0	−999
1	13	5	36	5	21	0	−999
1	13	6	36	6	21	6	17
1	14	55	−14	0	−999	0	−999
1	14	21	−14	21	−26	0	−999
1	14	17	−14	17	−26	17	−49
1	15	17	48	0	−999	0	−999
1	15	7	48	7	66	0	−999
1	15	5	48	5	66	5	59
1	16	120	30	0	−999	0	−999
1	16	44	30	44	27	0	−999
1	16	39	30	39	27	39	3

The data has been constructed such that the number of progeny in the third lactation was observed also for the first and second lactation. Consequently, the number of progeny is the same in all lactations when the third lactation is observed. Similarly, the number of progeny observed for the second lactation was assumed to be observed for the first lactation. This data structure is described in Schaeffer (2001).

The variance components file (`sch_cntry_A.var`) is

Random effect$_1$	Row$_2$	Column$_3$	Variance$_1$	
1	1	1	96	Lact. 1 genetic
1	1	2	68	Lact. 1,2 genetic
1	1	3	62	Lact. 1,3 genetic
1	2	2	160	Lact. 2 genetic
1	2	3	110	Lact. 2,3 genetic
1	3	3	190	Lact. 3 genetic
2	1	1	1018	Lact. 1 residual

2	1	2	128	Lact. 1,2 residual
2	1	3	67	Lact. 1,3 residual
2	2	2	1625	Lact. 2 residual
2	2	3	170	Lact. 2,3 residual
2	3	3	1792	Lact. 3 residual

As for the single-trait model, the model for deregression is very simple: only the general mean and the sire effect. It is important to use random unknown parent groups. The CLIM code for deregression (`sch_sm.clm`) is

```
TITLE     Multiple trait model

DATAFILE  sch_cntry_A_2.dat # Data file
INTEGER   ones sire         # Integer column names
REAL      w_1 e_1 w_2 e_2 w_3 e_3

DATASORT  PEDIGREECODE=sire
MISSING   -999

PEDFILE   sch_sm.ped
PEDIGREE  sire sm+p 1.0

PARFILE   sch_cntry_A.var

MODEL
  e_1 = ones sire ! weight=w_1
  e_2 = ones sire ! weight=w_2
  e_3 = ones sire ! weight=w_3
```

Calculating deregressed proofs (`mix99s < dereg.slv`) will give a solution for each general mean (`Solfix`)

```
 Fact. Trt       Level     N-Obs   Solution              Factor Trait
   1   1           1         15    25.011                   mean e_1
   1   2           1         10    24.971                   mean e_2
   1   3           1          5    13.586                   mean e_3
```

and the deregressed proofs (`Solani`):

```
       sire N-Desc  N-Obs deregressed proof
...
         12      0      3    22.510       34.257       45.390
         13      0      3    45.149       9.6328       38.644
         14      0      3   -17.035      -29.146      -75.258
         15      0      3    50.363       85.822       118.94
         16      0      3    30.240       26.829      -3.4327
...
```

Solutions were printed above for only those individuals that have observations, i.e., `N-Obs` more than zero. All other solutions have been set to zero by MiX99 .

11.3 Estimation of variance components

For prediction of breeding values, variance components need to be known. The Monte Carlo (MC) Expectation Maximization (EM) Restricted Maximum Likelihood (REML) (MC EM REML) algorithm has been implemented in the `mix99s` solver for the estimation of variance components (Matilainen et al., 2012).

The Monte Carlo algorithm uses a resampling approach to estimate prediction error variances (PEV) needed in EM REML. In each REML iteration, estimates of location parameters are obtained from the real data, whereas PEVs are obtained by repeatedly simulating data and estimating their location parameters. This approach allows for the calculation of PEVs without inverting the coefficient matrix, leading to memory requirements similar to solving the mixed model equations.

Although the EM algorithm is known to have slow convergence, the MC EM REML approach makes REML feasible for large data sets and complex models, where direc inversion of the coefficient matrix would be too memory and time consuming.

The implementation for the variance component estimation supports the majority of models available in MiX99. However, no metafounders are allowed, and only few single-step models.

11.3.1 Running MC REML

Estimation of variance components requires execution of two programs. First, the preprocessor program `mix99i` is executed with initial values for the variance components. Then, the solver program `mix99s` is executed with specific instructions in the solver option file (see VAROPT option line in Chapter 4.6). When the solver VAROPT option is `'E'` for variance component estimation, thererafter comes the following options:

- STOPE

  Convergence criterion and number of Monte Carlo samples.

- SEED

  Type of the random number generator seed.

- MIXPATH

  Directory path of the preprocessor program `mix99i`.

After these comes the SOLTYP option.

An example of a `mix99s` solver input file for variance component estimation:

```
# RAM:    RAM options
          H
# STOP:   Max.iter, Conv.value BLUP: observed data, Criterion, Enforce
          1000      1.0e-5                         d         f
# RESID:  Residuals calculation
          N
# VALID:  Model validation
          N
# VAROPT: Variance optons VCE, PEV, HV
          E
# STOPE:  REMLrounds, Samples, Conv.value VCE,[Conv.value BLUP][Last samples]
          1000        3        1.0e-10         3.0e-5           100
```

```
# SEED:    Type of seed for the random number generator
           R
# MIXPATH:
           /share/apps/
# SOLTYP: Solution files options
           Y
```

Note the 'E' option in VAROPT, and the subsequent lines for STOPE, SEED, and MIXPATH. In this example, the number of REML iterations is limited to 1000, the number of Monte Carlo samples per iteration is 3, the convergence of REML is 10^{-10}, and the convergence of the BLUP for the sampled data is 3.0×10^{-5}. The last REML round will use 100 Monte Carlo samples for the estimation standard errors of the variance parameters. The seed will be random.

Because models used in MC EM REML estimation are usually complex and the analyses time consuming, it is possible to change some of the iteration parameters during the execution of the mix99s solver to be more suitable. This change can be made using an external file named ITER, which allows modifying parameters regarding both breeding value estimation and variance component estimation.

If the estimation is wanted to be stopped beforehand, this can be done in a controlled way using an external STOP file (see Chapter 4.2).

11.3.2 File with starting values of (co)variance components

The file with the starting values for the (co)variance components must be in the same format as described in Chapter 3.4. The file will be specified by the PARFILE command. The same rules apply also for a file with starting values for the multiple residual (co)variance matrices in the case that a model with multiple residual (co)variances is applied (optional).

11.3.3 CLIM and variance component estimation

There is no need to give CLIM instructions specifically for variance component estimation. This is because the mix99i pre-processor will make the preconditioner matrices using the initial values for the variance components. To improve PCG solver convergence, these matrices are updated with the most recent variance component estimates. In the current implemention, the mix99s solver will periodically restart mix99i through a system call at certain REML iteration intervals. For this purpose, the pre-proprocessor stores its instructions to the files:

- MiX99_IN.DIR (model information) and
- MiX99_IN.OPT (command-line options).

These files must be available and contain the same information used as in the original pre-processing run for the variance component estimation process to work. Alternatively, MiX99 instruction file named as MiX99_DIR.DIR can be used to specify the pre-processor directives for the preconditioner update. This file is automatically created by mix99i when CLIM is used to define the model. File MiX99_DIR.DIR contains the same model information as the CLIM file in instruction file format.

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

```
Updating of preconditioner failed!
```

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

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

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

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

11.3.4 Number of Monte Carlo samples

The second parameter given on the STOPE line is the number of Monte Carlo samples per REML iteration, i.e., the ***number of data samples***. This is a number of data samples generated and analyzed within one round of stochastic MC EM REML.

The number of data samples analyzed within a REML round directly influences the accuracy of the estimated prediction error variances (PEV) needed in the REML algorithm. Increasing the number of samples reduces the Monte Carlo error associated with PEV estimation. However, the size of the Monte Carlo error also depends on the complexity of the specified model, amount of data, and number of individuals included in the analysis.

We have observed that the convergence of the MC EM REML algorithm is not affected by the number of samples specified per REML round. For many models, even one sample per REML round is sufficient. This is an important consideration, because each additional data sample requires solving an additional BLUP model within a REML round, which increases the total computation time. However, one sample per REML round does not allow computing standard errors for the final variance estimates. The number of samples during the last REML round can be set separately. It must be more than the number variance parameters with some margin in order to compute the standard errors.

Our experiences so far suggest that analysis with large amount of data and a sufficient number of individuals (e.g., test-day data with observation from over 10 000 individuals) requires only one Monte Carlo sample per REML round. In contrats, when the amount of data is rather small in relation to the number of parameters to be estimated, a higher number of samples (5, 10 or 20) may be more appropriate. For some analyses with limited data, REML implementation that does not rely on a Monte Carlo method can be more suitable.

11.3.5 Determining convergence of REML parameter estimates

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

k:

$$cc_E^{(k)} = \frac{\left(\widehat{\boldsymbol{s}}^{(k)}(x) - \widehat{\boldsymbol{s}}^{(k)}(x-1)\right)^T \left(\widehat{\boldsymbol{s}}^{(k)}(x) - \widehat{\boldsymbol{s}}^{(k)}(x-1)\right)}{\left(\widehat{\boldsymbol{s}}^{(k)}(x)\right)^T \left(\widehat{\boldsymbol{s}}^{(k)}(x)\right)}$$

After $cc_E^{(k)}$ has reached a value smaller than the specified convergence criterion (see STOPE option line), the REML analysis will perform a sequence of 30 additional MC EM REML rounds, which will reduce the Monte-Carlo error from the parameter estimates by using weighted average with decreasing weights for latest solutions. Depending on the analysis, we have found that values between 10^{-8} to 10^{-9} are suitable convergence limits.

11.3.6 Keeping certain variance components fixed

In some analyses it might be desirable to keep certain pre-defined variance components (starting values) unchanged during the MC EM REML estimation. This can be instructed in the MiX99 solver option file.

For this option, the VAROPT option line must have three entries:

e f *n*,

where

- **f** instructs `mix99s` to keep chosen parameters unchanged.
- *n* is an integer indicating how many variance parameters should remain unchanged.

After the VAROPT option, *n* additional lines must be given. Each line specifies one parameter that should remain unchanged. Each line consists of three integers:

1) The random effect number.
2) The row index.
3) The column index.

These values identify the specific variance component to be unchaged. Note that for a covariance, only the upper or lower triangle matrix position needs to be given. In practice, you can copy the corresponding line from the parameter file, excluding the actual variance component value.

When the model has multiple residual variance matrices, four integers need to be defined for each unchanged residual variance.

1) The random effect number of the residual effect.
2) The residual variance class number.

   This is equal to the first number on the corresponding parameter line in the file containing multiple residual (co)variances.
3) The row index.
4) The column index.

11.3.7 Restarting estimation of variance components

In the case the estimation has stopped or completed, but additional REML rounds are needed, the estimation can be resumed without rerunning the pre-processor `mix99i`. This is done by launching the solver `mix99s` again and specifying a negative value for the maximum number of iterations in the STOPE line of the solver option file. This negative value represents the new total number of REML rounds and must be greater than the REML rounds completed in the previous run.

For example, if the estimation stopped at REML round 1000 and you wish to make 1000 additional rounds, the new maximum number of iterations should be set to -2000. The estimation will resume from the last completed iteration (as recorded in the last row of the `REMLlog` file), and the new estimates will be appended to the `REMLlog` file. Other parameters related to PCG and REML iterations can also be changed for the continued estimation.

Variance component estimation can be restarted as a new run using previously estimated variance components as initial values. To do this, copy the estimated variance components from the `parfile` into the file specified by the `PARFILE` command (and `RESIDFILE`, if applicable). Remember to save the old `REMLlog` file under a different name to keep the previous estimates. At this stage, you may also modify other estimation parameters. Then, run the pre-processor `mix99i` to initialize the estimation with the new starting values for variance components before launching `mix99s` to begin the restarted estimation.

11.3.8 MC EM REML for MACE

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

11.3.9 Standard errors for REML parameter estimates

MiX99 can calculate approximate standard errors for variance component estimates at the last REML round. These are based on variances over sampled gradients as explained for the NR method in Matilainen et al. (2013). Standard errors are calculated automatically when the number of data samples is more than the number estimated variance parameters. Adequate number of samples depends on the data and model, but it is recommended to use at least 50 more samples than the number of estimated variance parameters.

Because using only a small number of data samples (even just one) is computationally fast, a practical approach is to first estimate variance components with a small number of samples. Once the variance components have converged, an additional REML round can be executed to compute standard errors. This can be done in two ways:

- Continue the variance component estimation.

- Restart the estimation using the previously estimated variance components as initial values.

In both cases, the number of Monte Carlo samples should be increased in the solver option file so that the standard errors can be calculated.

After the final REML round, the approximate standard errors and the covariances of REML estimates in the information matrix are written to files. These values can be computed only when the number of Monte Carlo samples is more than the number of variance parameters. It is possible to give the number of Monte Carlo samples in the last REML round separately in the solver directives. Thus, one can use a lower number of samples during the iteration to achive fast computations and only having reached covergence use many Monte Carlo samples in the last REML round. Note, however, that if the REML iteration is terminated due to reaching the maximum number of REML rounds, the separate last REML round value is not used, because reaching maximum number of REML rounds can be a sign of non-convergence.

Standard errors are printed to files:

- `vceSE`: has four columns, similar in structure to the `parfile` file, but has the approximate standard errors.
- `vceI`: has the information matrix.

Example about `vceSE` is for one trait with two random effects and a residual:

vceSE:

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

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

vceI:

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

where the first line has value for the first random effect, the second line has value between the first and the second random effect, the third line has value for the second random effect, and so on.

Because the information matrix contains values for every combination of (co)variance parameters, it can become very large. Let the total number of variance components be c, then the number of lines in a `vceI` file is $c \times (c+1)/2$.

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

vceSE:

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

The information matrix file has $6 \times 7/2 = 21$ lines:

vceI:

```
   1    1    1       1    1    1     0.978512E+10
   1    2    1       1    1    1     0.310664E+09
   1    2    1       1    2    1     0.124959E+08
   1    2    2       1    1    1     0.987140E+07
   1    2    2       1    2    1      483421.
   1    2    2       1    2    2      24303.1
   2    1    1       1    1    1    -0.755926E+10
   2    1    1       1    2    1    -0.238666E+09
   2    1    1       1    2    2    -0.741774E+07
   2    1    1       2    1    1     0.686931E+10
   2    2    1       1    1    1    -0.236382E+09
   2    2    1       1    2    1    -0.937649E+07
   2    2    1       1    2    2     -355953.
   2    2    1       2    1    1     0.216383E+09
   2    2    1       2    2    1     0.849670E+07
   2    2    2       1    1    1    -0.737152E+07
   2    2    2       1    2    1     -356791.
   2    2    2       1    2    2     -17917.6
   2    2    2       2    1    1     0.667658E+07
   2    2    2       2    2    1      319118.
   2    2    2       2    2    2      15877.5
```

For a model with 42 (co)variance parameters, the `vceI` file of the information matrix contains 903 lines.

If model has multiple residual variance matrices, the standard errors for all residual classes are printed after the first residual class (like in `REMLlog`).

11.3.10 Solution files for variance components

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

`parfile` Contains the latest solutions of variance component estimates. The structure of the file is the same as in the file defined by `PARFILE`.

resfile Contains the latest solutions of residual variance component estimates when multiple residual variance matrices are defined. The structure of the file is the same as in the multiple residual variance matrices file defined by RESIDFILE.

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

vceI This file will be produced when approximated standard errors for REML parameter estimates are calculated. The file has seven columns. Seventh column has the value of the information matrix for each pairwise parameter combination. The first parameter is indicated by the first three integers as in vceSE, and the second parameter is indicated by the next three integers as in vceSE.

11.4 Prediction error variances by Monte Carlo

Using mix99s to compute approximate prediction error variances (PEV) by Monte Carlo is similar to the variance component estimation by Monte Carlo REML (Chapter 11.3.1). The same limitations of the models available apply as for variance component estimation.

Computation of PEVs by Monte Carlo differs from using Monte Carlo REML in that several Monte Carlo samples are used and no multiple REML rounds are executed. In the mix99s options file there are two differences to the variance component estimation: VAROPT has 'P' instead 'E', and STOPE has only the number of Monte Carlo samples.

The mix99s options file for MC PEV includes:

- VAROPT has the Monte Carlo estimation method:
  - P1: method 1
  - P2: method 2
  - P3: method 3
- STOPE has the number of Monte Carlo samples.
- SEED is as for MC REML.
- MIXPATH has path for the preprocessor program mix99i.

The methods one to three refer to the Garcìa-Cortés methods (García-Cortés et al. (1995)). An example of a mix99s input file for computing prediction error variances by Monte Carlo using Garcìa-Cortés method 3:

```
# RAM:    RAM options
          H
# STOP:   Max.iter, Tolerance, Convergence criterion, Enforce
          5000      1.0e-5     r                       f
```

```
# RESID:  Residuals calculation
          N
# VALID:  Model validation
          N
# VAROPT: Prediction error variances by Monte Carlo
          P3
# STOPE:  Number of Monte Carlo samples
          50
# SEED:   Type of seed for the random number generator
          R
# MIXPATH:
          /share/apps/
# SOLTYP: Solution files options
          Y
```

The Garcia-Cortes method 1 uses expection in its computation formula while method 2 relies on Monte Carlo samples only. Method 3 pools these two methods using weights that try to optimize the best result. Thus, on average over all prediction error variances, method 3 may require the least number of Monte Carlo samples to get most accurate values.

11.5 Prediction error variances by direct inversion

MiX99 allows the computation of prediction variances and accuracies using the `exa99` program via inversion of the MME coefficient matrix. The `exa99` needs to be specifically requested. `exa99` is only useful for small problems (up to 200 000 equations) due to it taking a lot of memory and becoming slow for larger equation systems. Furthermore, models not supported include those using the `REGMATRIX` command and most single-step GBLUP models. Approximate reliabilities can be computed using `apax99` and `apax99p` which are intended for large-scale computations.

11.5.1 Command options

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

There are four matrix formats:

F Full dense storage

```
F
```

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

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

```
F 1.0001
```

Fs Full dense storage

```
Fs
```

This is like the `F` option except that memory is used less by lower numerical precision. Thus, in some cases, this option may have much more deviation

to the correct results. This can happen when the variance component of a random effect is very high or low either in proportion to total variance or in absolute value.

V Packed vector storage

```
V
```

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

S Sparse matrix storage

```
S
```

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

```
S 1234567
```

Only the lower triangle of the matrix is stored. Thus, memory need can be much less than for the packed vector storage. If the number non-zeros given is not enough, the program will increase sparse matrix size. However, this will use more memory than is optimal. Computationally the sparse matrix storage form can be very slow.

The total memory need depends on the order of equations and can be even higher than for the full dense storage. When specifying the model in the MODEL line(s), it is important to order the effects by the number of their levels. For the within block effects, the effect with the most levels should get the lowest block ordering number and the effect with the least levels the highest. Similarly, for the across block fixed effects, the effect with most levels should be specified first and the effect with the least levels should be specified last in the model line. This helps keeping the memory requirements as low as possible when inverting the coefficient matrix.

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

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

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

The three values have two purposes. First, these values are used to detect singularities in the coefficient matrix, i.e., dependencies. Singularity detection can help making the computations more reliable because rounding errors due to almost zero values are not propagated. Second, the values

allow preserving some sparsity in the matrix. The operational values mean that when during inversion calculations absolute value of added element is less than operational value, the value is not added to the coefficient matrix. Too high operational value leads to too high approximation of computations. The matrix sparsity value is used to make a matrix element value zero when absolute value of an element is less than the sparsity value. Thus, the operational zero values are used to effectively zero values calculated during inversion calculations but the sparsity value is used to neglect elements for these computations even before these computations are made.

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

The coefficient matrix of mixed model equations can be non-inverable due to dependencies between model effects. In order to have an invertable matrix, a small value is added to the diagonal of all fixed effects. A number can be given after the 'F','Fs' or 'V' matrix formats to change the value added to the diagonal. Default value is 0.001. The format is

```
F 0.001
```

If the model has only one fixed effect, such as the general mean, there is no need to add this value. Thus, the value can be changed to be zero:

```
F 0.0
```

This is not recommended as a general approach because models often have dependencies and with dependencies the inversion fails and terminates the program. The error message is either that the coefficient matrix is singular or that the matrix is not positive definite. The sparse matrix approach makes always a generalized inverse where dependies are taken care automatically.

After the matrix format information has been given, name of the file having diagonal values of the genetic relationship matrix is asked. If accuracies of breeding values are of no interest or the model has no additive genetic effect, then answering NONE allows computations without this information.

The diagonal matrix file has format:

```
<ID code> <Diagonal value>
```

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

An example of an exa99 option file:

```
F
Adiag.dat
```

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

For sparse matrix storage the command lines can be:

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

11.5.2 Output files

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

ACCani PEV and accuracies for the individual additive animal genetic effects.

SEfix Standard errors for all across blocks fixed effects.

SEf*nn* Standard errors for the n^{th} within blocks fixed effect. For example, SEf02 is the solution file for the 2^{nd} within block fixed effect.

PEVr*nn* Prediction error variance for the n^{th} random effect in the model. For example, PEVr03 is the solution file for the random effects with the random effect number 3.

SEreg Standard errors for the across whole data regression effects.

12 Summary of all commands

CLIM has optional commands, and options to the required commands. If an optional command is not given then default values are used for this command. In general, the commands are quite self-explanatory. Below they are divided into groups. There are chapter numbers after the short description. Required commands are explained in Chapter 12.1 and optional commands in Chapter 12.2.

Data file commands	
DATAFILE	name of data file, 3.1
DATASORT	information on how the data file was sorted, 12.2.4 (optional)
INTEGER	integer number column names in the data file, 12.1.3
MISSING	code for missing observations, 12.2.14 (optional)
REAL	real number column names in the data file, 12.1.7
REGFILE	regression matrix file, 12.2.19 (optional)
TABLEFILE	name of the separate covariable table file, 12.2.35 (optional)
TABLEINDEX	integer number column name of the covariable table file number in the data file, 12.2.36 (optional)
TRAITGROUP	integer number column name of the trait group number, 12.2.41 (optional)

Pedigree file commands	
PEDFILE	name of pedigree file, 3.3
PEDIGREE	effect/component name in the model to which the pedigree is attached. Also, type of pedigree information, i.e., model type (animal or sire model). If random unknown parent groups, then coefficient value as well, 12.1.6 (optional)

Variance component information	
PARFILE	MiX99 variance components file, 3.4
RESIDFILE	name of residual (co)variance parameter file, 12.2.24 (optional)
RESIDUAL	integer number column name of the residual (co)variance number, 12.2.25 (optional)
REGPARFILE	name of random regression matrix parameter file, 12.2.21 (optional)

Model commands	
MODEL	statistical model
RANDOM	random effects in the model, 12.2.18 (optional if additive genetic and residual effects are the only random effects)
REGMATRIX	regression matrix information, 12.2.20 (optional)

Solving	
NORANSOL	random effects for which no solution files are to be written, 12.2.15 (optional)
PRECON	preconditioning information, 12.2.17
TITLE	title of the analysis, 12.2.39 (optional)
TMPDIR	directory for the MiX99 temporary files, 12.2.40 (optional)
WITHINBLOCKORDER	ordering of effects in the blocks (optional)

Macros and range abbreviations	
DEFINE	defines a text macro replacement

12.1 Required commands

Following are explanation and syntax of all commands.

12.1.1 MODEL

MODEL is considered in Chapters 6, 7, 9, 10.5, and 11.

12.1.2 DATAFILE

Name of data file. Optional information: file type of text or binary can be given. The default is text file. So, for a binary file, file type must be always specified.

Syntax:

```
DATAFILE [TEXT/BINARY] <filename>
```

Example. Data file Beef_MiX.dat has standard text data.

```
DATAFILE ../data/Beef_MiX.dat
```

12.1.3 INTEGER

Names of integer number columns in the data file. These are used to give names to data file columns.

Syntax:

```
INTEGER <names of integer variables>
```

Example. Data file having 8 columns of integer data information. The first column is named block, second is id, the third is trt_group etc.

```
INTEGER block id trt_group HTM AGE DCC DIM
```

12.1.4 PARFILE

Name of (co)variance parameter file. Values of variance components can be given using several ways and formats. Please see Chapters 3.4 and 3.4.1.

Syntax:

```
PARFILE <filename>
```

Example. Name of the parameter file is Beef.par.

```
PARFILE ../data/Beef.par
```

12.1.5 PEDFILE

Name of the pedigree file. This pedigree file is read by mix99i. The PEDFILE command can have a specifier or file type. When the specifier is DYD, then DYDs can be computed by the solver program. When the pedigree type is FILE in the command PEDIGREE, the file has the inverse of the co-variance matrix for the breeding values. The default format for the matrix is the lower triangle co-ordinate sparse matrix format. Option MIXED can be used to relax the requirement of the lower triangle matrix (see Ch. 9.3.2). However, this means that the file can have an element (1,2) of the matrix, i.e., an upper triangle element, but the file should not have a corresponding lower triangle element (2,1). This condition is not checked. Alternatively the LOWER option allows the use of the lower triangle dense matrix format or LOWER format. This option allows faster computations than the co-ordinate sparse matrix format.

Syntax:

```
PEDFILE [DYD | LOWER | MIXED] <filename>
```

Example 1. Name of the pedigree file is Beef_phantom.ped.

```
PEDFILE ../data/Beef_phantom.ped
```

Example 2. Name of the pedigree file is Beef_phantom.ped and DYDs will be requested.

```
PEDFILE DYD ../data/Beef_phantom.ped
```

Example 3. Name of co-ordinate format matrix with upper and lower triangle elements:

```
PEDFILE MIXED ../data/iG_matrix.dat
```

Example 4. Name of lower triangel dense matrix:

```
PEDFILE LOWER ../data/iGL_matrix.dat
```

12.1.6 PEDIGREE

Pedigree type and other information. Three types of pedigrees are accepted: am for animal model, sm for sire model, and ar for autoregressive model. Genetic groups or unknown parent groups are indicated by the suffix +p, e.g., am+p for an animal model with unknown parents groups. Pedigree has been stored in the file given by the command PEDFILE.

When the pedigree type is FILE, the inverse of the relationship co-variance structure (e.g. $\mathbf{G}^{-1}$ in GBLUP) is in the file specified by the PEDFILE command. See example in Ch. 9.3.

Syntax:

```
PEDIGREE <effect name> <pedigree type> &
         [<optional coefficient for random unknown parent groups>]
```

Example 1. Pedigree is for an animal model with random unknown parent groups. Pedigree is associated with model effect name G. The unknown parent groups are random with the inverse genetic variance (matrix) multiplied by the coefficient 0.5.

```
PEDIGREE G am+p 0.5
```

Example 2. GBLUP model where the inverse of the genomic relationship matrix G^{-1} is in file iGL.dat stored in the lower triangle dense matrix format.

```
PEDFILE  LOWER iGL.dat
PEDIGREE G FILE
```

12.1.7 REAL

Names of real number columns in the data file. Integer number columns are always before real number columns. See MiX99 manual *Technical reference guide for MiX99 pre-processor* for more information.

Syntax:

```
REAL <names of real variables>
```

Example. There are 3 real number columns (after the integer number columns) in the data file. The first is B_WEIGHT, the second is W_WEIGHT, and the third is W_AGE.

```
REAL      B_WEIGHT W_WEIGHT W_AGE
```

12.2 Optional commands

The following commands have default values that are used if the command is not given.

12.2.1 AR

Define the autoregressive values for each trait in autoregressive model. When this command is given, the PEDIGREE command type must be ar. The default is no autoregressive model.

Syntax:

```
AR <values for each trait>
```

Example. Autoregressive values for 2 traits

```
AR 0.8 0.9
```

12.2.2 CENTERFILE

The file defined by this command has values for centering marker data defined in SNPMATRIX and SNPFILE. The file has two columns: marker number and centering value. The marker numbers are from 1 onwards corresponding to the columns in SNPFILE after the first column (ID values). Thus, number 1 is the second column in SNPFILE. When the CENTER option in SNPMATRIX has 'p', CENTERFILE is assumed to have allele frequencies, i.e., values from zero to one. The values will be multiplied by two before centering the genotype values in SNPFILE. When the CENTER option is 'f' then the values are used as such.

Syntax:

```
CENTERFILE  <filename>
```

Example. Base population allele frequencies are in file baseAF.dat

```
CENTERFILE baseAF.dat
```

12.2.3 COVFILE

Attaches the given inverse correlation matrix to the specified random effect. The given matrix is in a file which is by default in the co-ordinate (Yale) sparse matrix format. The LOWER option allows the use of the lower triangle dense matrix format or LOWER format (see Ch. 9.3.2).

Syntax:

```
COVFILE <random effect name> [LOWER] <filename>
```

Example. Inverse correlation matrix for the random effect named `genomic` is in file `covfile.dat` in the lower triangle dense matrix format:

```
COVFILE genomic LOWER covfile.dat
```

12.2.4 DATASORT

Names of the integer number columns for the block sorting variable (`BLOCK`), and the pedigree individual sorting variable (`PEDIGREECODE`). By default, none are needed.

See MiX99 manual *Technical reference guide for MiX99 pre-processor* for more explanation.Syntax:

```
DATASORT BLOCK       =<block in INTEGER column> &
         PEDIGREECODE=<code  in INTEGER column>
```

Example. The block code is integer number column `block`, and the pedigree code is column `IDcode` in the data file

```
DATASORT BLOCK=block PEDIGREECODE=IDcode
```

12.2.5 DEFINE

Defines a ***macro*** identifier to be used as an abbreviation for its replacement text:

```
DEFINE <macro name> <replacement text>
```

Instructs to replace all successive occurrences of the identifier with the replacement. Allows to shorten repeatedly occurring text, for example, directory names and common effects in complex CLIM models.

In addition to macros, a ***range expansion*** can be used as an abbreviation. Every occurrence of

```
<identifier><N>:<M>
```

is replaced by a space separated list of

```
<identifier><N> <identifier><N+1> ... <identifier><M-1>  <identifier><M>
```

where the identifier is replicated M-N+1 times with integers from N to M. For example, `het1:5` is replaced by

```
het1 het2 het3 het4 het5
```

Range expansion can be used, for example, to name `INTEGER` and `REAL` columns, or to specify covariable table columns in complex models.

Example:

```
DEFINE HomeDIR /home/user
DEFINE WorkDIR .

DATAFILE HomeDIR/mydata.dat
PEDFILE  HomeDIR/mydata.ped

INTEGER  block IDcode HTM YM SEASON AGE DCC DIM
REAL     milk protein fat x1:2 het1:5

DEFINE CurveMILK  Curve(t1:3 t4  t96 | SEASON)
DEFINE CurvePROT  Curve(t1:3 t95 t97 | SEASON)
```

```
DEFINE CurveFAT   Curve(t1:3 t95 t98 | SEASON)
DEFINE Common     AGE DCC YM HTM

MODEL
  milk    = CurveMILK Common PE( t5:10|IDcode)@1st G(t59:64|IDcode)@FST
  protein = CurvePROT Common PE(t11:16|IDcode)@1st G(t65:70|IDcode)@FST
  fat     = CurveFAT  Common PE(t17:22|IDcode)@1st G(t71:76|IDcode)@FST

TMPDIR WorkDIR/tmpMiX
```

is replaced by

```
DATAFILE /home/user/mydata.dat
PEDFILE  /home/user/mydata.ped

INTEGER  block IDcode HTM YM SEASON AGE DCC DIM
REAL     milk protein fat x1 x2 het1 het2 het3 het4 het5

MODEL
  milk    = Curve(t1  t2  t3  t4  t96     | SEASON)     AGE DCC YM HTM &
               PE(t5  t6  t7  t8  t9  t10 | IDcode)@1st                &
                G(t59 t60 t61 t62 t63 t64 | IDcode)@FST
  protein = Curve(t1  t2  t3  t95 t97     | SEASON)     AGE DCC YM HTM &
               PE(t11 t12 t13 t14 t15 t16 | IDcode)@1st                &
                G(t65 t66 t67 t68 t69 t70 | IDcode)@FST
  fat     = Curve(t1  t2  t3  t95 t98     | SEASON)     AGE DCC YM HTM &
               PE(t17 t18 t19 t20 t21 t22 | IDcode)@1st                &
                G(t71 t72 t73 t74 t75 t76 | IDcode)@FST

TMPDIR ./tmpMiX
```

Currently, the preprocessor (`mix99i`) command line option `--usemacros` is needed to activate the CLIM macro and range expansion.

12.2.6 GBLUP

This allows for a GBLUP model by attaching the given inverse genomic relationship matrix in a file to the effect specified to be the random additive genetic effect. The matrix format in the file is by default in the co-ordinate (Yale) sparse matrix format. Lower triangle dense matrix is allowed with the `LOWER` option. Note that the random genetic effect has to be the last one on the model line. See `COVFILE` for having several external correlation matrices associated with other random effects than the genetic.

Syntax:

```
  GBLUP <random effect name> [LOWER|MIXED] <filename>
```

Example. Inverse genomic relationship matrix for the random effect named `IDcode` is in file `iGL.dat` in the lower triangle dense matrix format:

```
  GBLUP IDcode LOWER iGL.dat
```

12.2.7 IA22FILE

This command partly depricated, only allowing `IA22FILE PEDIGREE`.

Giving `PEDIGREE` for `IA22FILE` indicates MiX99 that the computations for $\mathbf{A}_{gg}^{-1}$ are done using pedigree information available in the file defined by `PEDFILE`. The computations are typically fast because there is no need to make $\mathbf{A}_{gg}^{-1}$ and the solver uses fast sparse matrix routines for the computations.

Syntax:

```
IA22FILE PEDIGREE
```

12.2.8 ICFILE

Gives the file for ssGTABLUP in the single-step method having the matrix $\mathbf{C}^{-1} = (\frac{1}{w}\mathbf{Z}_c'\mathbf{A}_{gg}^{-1}\mathbf{Z}_c + \mathbf{B}^{-1})^{-1}$ matrix in

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

where w is the residual polygenic proportion, $\mathbf{Z}_c$ has the centered marker matrix, and $\mathbf{B}$ has the scaling information. The command requires the ZCFILE command. The $\mathbf{C}^{-1}$ matrix file can be made by a special preprocessing program such as T48eig_make. The approach requires the $\mathbf{A}_{gg}^{-1}$ matrix to be given separately using the command IA22FILE with option PEDIGREE, i.e., IA22FILE PEDIGREE.

Syntax:

```
ICFILE  <filename>
```

Example. Matrix $\mathbf{C}^{-1}$ is in the iC.dat file:

```
ICFILE iC.dat
```

12.2.9 IGAMMAFILE

Gives the name of the file having the Γ^{-1} matrix for models having metafounders. The matrix has to be in co-ordinate sparse matrix format. Index values are metafounder numbers.

Example. Matrix $\mathbf{\Gamma}^{-1}$ is in the iGamma.dat file:

```
IGAMMAFILE MIXED iGamma.dat
```

12.2.10 IGFILE

This command can be used for the single-step method, but please use the SSGBLUP command instead. The IGFILE command allows giving an additional inverse covariance matrix for a subset of individuals as in the single-step method. An approach is to provide the $\mathbf{C}_{GA} = \mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$ matrix. Typically, IA22FILE command is give as well ('IA22FILE PEDIGREE'), and then the file given for IGFILE must contain $\mathbf{G}^{-1}$ only.

The default matrix format is the co-ordinate (Yale) sparse matrix format, which is easiest to use with the MIXED option. Often more effient is the lower triangle dense matrix format (see Ch. 9.3). For this matrix, the LOWER option ('IGFILE LOWER') need to be used.

Syntax:

```
IGFILE [LOWER | MIXED] <filename>
```

Example. Matrix $\mathbf{C}_{GA}$ is in the iH.dat file:

```
IGFILE iH.dat
```

12.2.11 IHPRECON

Gives the file having changes to the The preconditioner matrix in the diagonal of the inverse relationship matrix. For example, consider an inverse relationship matrix for the single-step method: $\mathbf{H}^{-1} = \mathbf{A}^{-1} - \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1} \end{bmatrix}$ where $\mathbf{A}^{-1}$ is the inverse of the pedigree relationship matrix for all individuals, $\mathbf{G}^{-1}$ is the inverse genomic relationship matrix, and $\mathbf{A}_{gg}^{-1}$ is the inverse pedigree relationship matrix of the genotyped individuals. When command IA22FILE is used with the PEDIGREE option, the preprocessor program uses a Monte Carlo approach to approximate the diagonal element values of $\mathbf{A}_{gg}^{-1}$ for the preconditioner. However, this step takes some time, and repeated computations for the same set of genotyped needs the same values. Thus, precomputed values can be used instead. These values can be given using the IHPRECON command. One diagonal element of $-\mathbf{A}_{gg}^{-1}$ is given on a line: <ID code> <value>. Note that these are NOT diagonal elements of $\mathbf{A}_{gg}$ nor $\mathbf{A}_{gg}^{-1}$ but diagonal elements of $-\mathbf{A}_{gg}^{-1}$. Furthermore, single-step GTBLUP needs different values. Because there is a high change of providing incorrect values, the use of the IHPRECON can be error-prone. Alternatively, no precomputed values nor the Monte Carlo computations can be requested. This can be achieved with command "IHPRECON NONE", but this may lead to poor convergence.

Syntax:

```
IHPRECON <filename>
```

Example. Diagonal of matrix $-\mathbf{A}_{gg}^{-1}$ is in file minus_diA_genotyped.dat

```
IHPRECON minus_diA_genotyped.dat
```

12.2.12 INBREEDING

Column numbers of the individual ID code and inbreeding coefficient in the inbreeding coefficient file (see 12.2.13). The default is that all inbreeding coefficients are zero. The column number of the individual ID code must be before the inbreeding coefficient.

Syntax:

```
INBREEDING PEDIGREECODE=<individual ID code column> &
           FINBR=<inbreeding coefficient column>
```

Example. The first column has the individual ID code, and the third column has the inbreeding coefficient:

```
INBREEDING PEDIGREECODE=1 FINBR=3
```

12.2.13 INBRFILE

Name of the inbreeding coefficient file. The default is that all inbreeding coefficients are zero, and, thus, no inbreeding coefficient file is read.

Syntax:

```
INBRFILE <filename>
```

Example. Inbreeding coefficient file is my.inbr:

```
INBRFILE my.inbr
```

12.2.14 MISSING

Number indicating missing information for data in the real number columns. The default is zero.

Syntax:

```
MISSING <number for missing>
```

Example. Set missing value to `-99999.0`

```
MISSING       -99999.0
```

12.2.15 NORANSOL

Give random effects for which no solution files are made. The default is that solutions are written for all random effects.

Syntax:

```
NORANSOL <random effects>
```

Example. No solutions are written of effects `HTM` and `PE`.

```
NORANSOL HTM PE
```

12.2.16 PARALLEL

Information on parallel computing: number of processors and number of common area blocks. Alternatively, common area blocks can be defined by specifying the block code of the first common area block instead and adding option `FIRST` after it.

Optional additional information is method of the workload division, and total maximum size of I/O buffers. The default is no parallel computing, i.e., number of processors is zero. Work division is either by number of records (default), or number of equations. Giving letter E or e will use number of equations in work division to the processors.
Total size of I/O buffers is given as an integer number in megabytes but by default is determined by the preprocessor program. See PARALLEL in MiX99 manual *Technical reference guide for MiX99 pre-processor* for more information.

Syntax:

```
PARALLEL <N processors> <N common blocks> [<work division> <buffer size>]
PARALLEL <N processors> <first common block> FIRST [...]
```

Example. Parallel computing with 6 processors. The last 10 blocks in the data file belong to the common area blocks.

```
PARALLEL       6   10
```

12.2.17 PRECON

Information on the preconditioning. Format is the same as given in MiX99 manual for `mix99i` *Technical reference guide for MiX99 pre-processor*. Thus, the first characters (one for each effect) are for the within block effects, and then one common for all across blocks effects. The default is block diagonal preconditioner for all effects.

effect type	available preconditioners
within block	d=diagonal, b=block diagonal.
across blocks fixed	d=diagonal, b=block diagonal, f= full block, m=mixed block

Note that giving PRECON n will lead to use of no preconditioner.

Syntax:

```
PRECON <preconditioning information>
```

Example. Block diagonal preconditioner is used for the 4 within block effects. The across blocks fixed effects have mixed block preconditioner where the first effect is in block of its own and the others are in another block.

```
PRECON b b b b   m
```

12.2.18 RANDOM

This command gives order numbers to the random effect names in the model line(s) other than the additive genetic effect associated to pedigree. If this command is not given, the only random effects are the additive genetic and the residual effects. Order of the effects after this command gives numbering of the random effects.

Syntax:

```
RANDOM <effect names>
```

Example. There are four random effects: herd-test-month, permanent environment, additive genetic, and residual. Data file has integer columns HTM for herd-test-month and IDcode for individual ID code. In the model line, there are random effects HTM for herd-test-month, PE(IDcode) for permanent environment, and G(IDcode) for additive genetics. The numbers for random effects are 1 for HTM, 2 for PE, 3 for G, and 4 for the residual. These numbers are used in the (co)variance file defined by PARFILE. The commnd has the random effects in their numbering order:

```
RANDOM HTM PE G
```

or alternatively

```
RANDOM HTM PE
```

In the latter case, G is known to be a random effect due to it been defined in the PEDIGREE command to be associated with a relationship matrix, and such a random effect has always random effect number just before the residual.

12.2.19 REGFILE

Regression covariable matrix file contains covariates of regression effects defined by the REGMATRIX command. Commonly SNP genotype matrix is given as a regression covariable matrix file.

Syntax:

```
REGFILE <filename>
```

Example. Regression covariates in file snp.dat

```
REGFILE snp.dat
```

12.2.20 REGMATRIX

Regression covariable matrix information. The information given:

- effect type:

  FIXED Fixed effects,

RANDOM Random effects with single common variance,

HETEROGENEOUS Random effects, each with its own variance

- name of the effect
- column numbers of covariates in the file defined by REGFILE:

  ID = a Column number of individual code is *a* (optional),

  FIRST = b The first column of covariates is *b*,

  LAST = c The last column is *c*.
- SNP marker information (optional):
  - RegIndex:

    REGINDEX = <col> Defines integer column name <col> or REGMATRIX index in data file used to associate a data record with a line in a REGMATRIX file.
  - Imputation:

    IMPUTE = <i> Missing (integer) marker value <i> to be imputed.
  - Centering:

    CENTER Average marker value.

    CENTER = <r> Constant real value <r>, e.g., CENTER = 1.

    CENTER = <file> Separate centering for markers in file <file>.
- Scaling (optional, default is no scaling):

  SCALE = <r> Constant real value <r>, e.g., SCALE = 0.5.

  SCALE = 2pq Scaling with $\frac{1}{\sqrt{2\sum_i p_i(1-p_i)}}$ where $p_i = \frac{\mu_i}{2}$ are half of the mean marker values (μ_i).

  SCALE = m Scaling with $\frac{1}{\sqrt{m}}$ where m is the number of markers.

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

  SCALE = <file> Separate scaling for markers in file <file>.
- File format (optional, default is *n*):

  FORMAT = n|m|s|pb File format is *n* (or *normal*) for normal real valued regression coefficients with spaces separating columns, *m* (or *markers*) for SNP markers of integer values '0', '1', '2' and optional missing value of IMPUTE (with spaces separating columns), *s* (*squeezed* for "squeezed" SNP marker values without separating spaces, or *pb* for binary PLINK .bed file format. Optional minus sign prevents byte-packing of SNP matrices.
- preconditioner type (optional):

PRECON = n|d|b Preconditioner type is *n* for none, *d* for diagonal (default), or *b* for block diagonal. Block diagonal means trait block by REGMATRIX column.

See also commands REGFILE, REGPARFILE, REGTRAITS.

Syntax:

```
REGMATRIX <type> <name> [ID=<column>] FIRST=<column> LAST=<column>
          [REGINDEX=<INTEGER column name>] [IMPUTE=<missing>]
          [SCALE={<real value>|<filename>}] [FORMAT=<file format>]
          [CENTER[={<real value>|<filename>}]] [PRECON=<preconditioner>]
```

Example. Regression matrix information: common random variance, name of the effects is snp, covariates in columns 3 to 10, ID code on column 1. Block diagonal preconditioner.

```
REGMATRIX RANDOM snp ID=1 FIRST=3 LAST=10 PRECON=b
```

12.2.21 REGPARFILE

Variance components for a random regression matrix. See also commands REGFILE and REGMATRIX.

Syntax:

```
REGPARFILE <filename>
```

Example. Variance components in file reg.par

```
REGPARFILE rep.par
```

12.2.22 REGTRAITS

Define the traits for this regression matrix (REGMATRIX).

Syntax:

```
REGPTRAITS <traits>
```

Example. Model has traits y1, y2 and y3, but only y1 has this REGMATRIX:

```
REGTRAITS y1
```

12.2.23 RESTARTSOL

Make the restart solution file Solunf. The restart solution file allows the solver to start iteration using old solutions. The default is no file is written. Command RESTARTSOL is an option within MODEL command. The option is given on the same line as command MODEL.

Syntax:

```
MODEL RESTARTSOL
```

Example. Restart solution files requested.

```
MODEL RESTARTSOL
```

12.2.24 RESIDFILE

Residual variance covariance matrix file in case of different residuals for different observations. If residual file is given then data file must have an integer number column associated with the residual matrix number. See command RESIDUAL. The default

is that no additional residual variance file is used, i.e., the same residual (co)variance defined in PARFILE is used for all observations.

Syntax:

```
RESIDFILE <filename>
```

Example. Residuals are in file mix99par.respar

```
RESIDFILE ./data/mix99par.respar
```

12.2.25 RESIDUAL

Name of integer number column in the data file indicating number of residual variance used for this observation (see RESIDFILE). The default is that same residual (co)variance is used for all observations.

Syntax:

```
RESIDUAL <integer column name>
```

Example. Residual variance number is on integer data column ResidualNumber.

```
RESIDUAL ResidualNumber
```

12.2.26 SCALE

The SCALE option is used under the MODEL command normalize or scale all observation by ther residual standard deviation. When this option is used, all trait observations are on a comparable scale. This can be useful when solving multi-trait models, as it may lead to more stable and numerically efficient computations. Operation:

- each observation is divided by the residual standard deviation of its trait
- after the model is solved, all solutions are automatically rescaled back to their original units before writing solutions to the files.

By default, no scaling is applied. To enable scaling, include the SCALE option on the same line as the MODEL command.

Syntax:

```
MODEL SCALE
```

Example. Scaling is requested:

```
MODEL SCALE
```

12.2.27 SNPFILE

Defines a file having the SNP marker data where each marker is coded by the number of a counted allele, i.e., having values 0, 1, or 2. The first column has the ID code. The format for reading and the relevant columns having SNP markers are set in the SNPMATRIX command. The SNP file is assumed to be a text file.

Syntax:

```
SNPFILE  <filename>
```

Example. SNP marker data is in file SNP.dat

```
SNPFILE SNP.dat
```

12.2.28 SNPMATRIX

Instructs to use the fully component-wise ssGTBLUP or the ssSNPBLUP model. A marker file needs to be defined by the SNPFILE command. Options for the SNPMATRIX command :

- column number information:

  FIRST = b First SNP marker column is *b*,

  LAST = c Last SNP marker column is *c*.

- SNP marker data file format in SNPFILE:

  FORMAT = m space separated SNP markers (default).

  FORMAT = <format> Fortran format for reading data.

- Centering of SNP markers:

  CENTER = <r> Constant real value <r>, e.g., CENTER = 1.

  CENTER = 'p' Centering using allele frequencies of markers in the CENTERFILE.

  CENTER = 'f' Centering using values in the CENTERFILE.

- Used SNP marker memory storage mode:

  USE = BYTE Store one genotype to a byte (default).

  USE = PACK Pack several genotypes to a byte.

- Scaling information (ssSNPBLUP model only): Assume the genomic relationship matrix part is $\mathbf{G} = \mathbf{Z}_c\mathbf{B}\mathbf{Z}_c'$. The diagonal scaling matrix $\mathbf{B}$ depends on the model (GTA or GTe) defined by the ssSNPBLUP command. The matrix is $\mathbf{B} = \frac{1-w}{k}\mathbf{I}$ for GTA, and $\mathbf{B} = \frac{1}{k}\mathbf{I}$ for GTe where the scaling constant k is computed:

  1 **SCALE = <r>** Constant real value <r>, e.g., SCALE = 15432.1.

  2 **SCALE = p** Scaling with $2\sum_i p_i(1-p_i)$ where p_i are allele frequencies in the CENTERFILE.

  3 **SCALE = m** Scaling by m.

  4 **SCALE = m2** Scaling by $\frac{m}{2}$.

  5 **SCALE = pm** Scaling by $2p_{MAF}(1-p_{MAF})m$ where p_{MAF} is average minor allele frequency in the CENTERFILE.

  6 **SCALE = no** No scaling.

  7 **SCALE = dA** Scaling such that $tr(\mathbf{Z}_c\mathbf{B}\mathbf{Z}_c') = tr(\mathbf{A}_{gg})$.

  8 **SCALE = one** Scaling such that $tr(\mathbf{Z}_c\mathbf{B}\mathbf{Z}_c')/n$ equals one.

  where $\mathbf{Z}_c$ is the centered marker matrix, m is the number of markers and n is the number of genotyped.

- Marker weights (optional, ssSNPBLUP model only): Assume the genomic relationship matrix part is $\mathbf{G} = \mathbf{Z}_c\mathbf{B}\mathbf{Z}_c'$ where $\mathbf{Z}_c$ is the centered marker matrix. The diagonal scaling matrix $\mathbf{B}$ depends on the weight matrix $\mathbf{D}$ and has the form

DEV

$\mathbf{B} = \frac{1-w}{k}\mathbf{D}$ for GTA, and $\mathbf{B} = \frac{1}{k}\mathbf{D}$ for GTe where k is the scaling constant. The weight matrix $\mathbf{D}$ can be:

1 DWEIGHT = M WEIGHTFILE has marker weights common to all traits.

2 DWEIGHT = T WEIGHTFILE has trait-specific marker weights.

When the SCALE option is not specified, then the ssGTABLUP approach is used, and it is assumed that an ICFILE is given as well. When the SCALE option is given, the ssSNPBLUP model is used and no ICFILE command should be given.

Syntax:

```
SNPMATRIX FIRST=<column> LAST=<column> FORMAT=<file format>
          CENTER={<real value>|'p'|'f'} [USE={BYTE|PACK}]
          [SCALE={<real value>|'p'|'m'|'m2'|'pm'|'no'|'dA'|'one'}]
```

Example. ssGTALUP model with 1000 space separated SNP markers in the file defined by the SNPFILE command and using centering as if all allele frequencies were 0.5.

```
SNPMATRIX  FIRST=2 LAST=1001 FORMAT='m' CENTER=1
```

Example. ssGTABLUP model without space between SNP markers by a Fortran reading format and using allele frequencies for centering in file defined by the CENTERFILE command.

```
SNPMATRIX  FIRST=2 LAST=1001 FORMAT='(i2,1x,1000i1)' CENTER=p
```

Example. ssSNPBLUP model like the previous ssGTBLUP model but scaling uses a constant computed using allele frequencies from the CENTERFILE.

```
SNPMATRIX  FIRST=2 LAST=1001 FORMAT='(i2,1x,1000i1)' CENTER=p SCALE=p
```

12.2.29 SNPPARFILE

DEV

This command defines the file having heterogeneous marker variances in a ssSNPBLUP model. By default, all variances are assumed to have the same variance as defined for the additive genetic effect. In SNPPARFILE, each marker has sparse matrix variances with the usual notation of marker number, two matrix positions, and the (co)variance.

```
SNPPARFILE VC_het.dat
```

12.2.30 SSGBLUP

Instructs to use the standard single-step ssGBLUP model where the inverse genomic relationship matrix $\mathbf{G}^{-1}$ is given in a file. Note that the PEDIGREE and PEDFILE commands are needed as well in the single-step model. The computations due to the $\mathbf{A}_{gg}^{-1}$ matrix are performed by the solver. Thus, this command is the same as giving IGFILE with the $\mathbf{G}^{-1}$ file and IA22FILE by option PEDIGREE.

The file formats available for the $\mathbf{G}^{-1}$ matrix. The default format is co-ordinate sparse matrix format having lower triangle elements only. Options include MIXED to allow both upper and lower triangular elements in the sparse co-ordinate matrix, and LOWER for the lower triangle dense matrix.

Syntax:

```
SSGBLUP  [LOWER|MIXED] <filename>
```

Example. Matrix $\mathbf{G}^1$ is in file iGL.bin

```
SSGBLUP LOWER iGL.bin
```

12.2.31 SSGTBLUP

Instructs to use the ssGTBLUP model where the inverse genomic relationship matrix $\mathbf{G}^{-1}$ has a genomic part and a regularization matrix $\mathbf{C}$.

Note that the PEDIGREE and PEDFILE commands are needed as well as in any single-step model. The computations due to the $\mathbf{A}_{gg}^{-1}$ matrix are done by the solver.

Syntax:

```
SSGTBLUP  [TAFILE|TEFILE] <filename>
```

Example. The $\mathbf{T}$ matrix of ssGTABLUP is in file TA.bin

```
SSGTBLUP TAFILE TA.bin
```

12.2.32 SSGTABLUP

Instructs to use the standard ssGTABLUP model of single-step as explained for TAFILE. This command is the same as giving TAFILE with the $\mathbf{T}_A$ file and IA22FILE by option PEDIGREE. This is the same as using command SSGTBLUP with option TAFILE.

Syntax:

```
SSGTABLUP  <filename>
```

Example. Matrix $\mathbf{T}_A$ is in file TA.bin

```
SSGTABLUP TA.bin
```

12.2.33 SSGTEBLUP

Instructs to use the standard ssGTeBLUP model of single-step as explained for TEFILE. This command is the same as giving the TEFILE command with the $\mathbf{T}_e$ file and the IA22FILE command with the option PEDIGREE. This is the same as using command SSGTBLUP with option TEFILE.

Syntax:

```
SSGTEBLUP  <filename>
```

Example. Matrix $\mathbf{T}_e$ is in file Te.bin

```
SSGTEBLUP Te.bin
```

12.2.34 ssSNPBLUP

This command is used to inform the constant used in the ssSNPBLUP model and requires the SNPMATRIX and SNPFILE commands. Two models equivalent to the ssGTBLUP model are available

- The ssGTABLUP model needs the residual polygenic proportion.
- The ssGTeBLUP model needs the small value (e.g., 0.01) for the regularization.

The models are indicated by GTA for ssGTABLUP and GTe for ssGTeBLUP. The ssSNPBLUP command can be optionally used to give the name of the marker genotype file such that the SNPFILE command need not be given. Note that this command requires giving the SCALE option in the SNPMATRIX command.

Syntax:

```
ssSNPBLUP [GTA|GTe] <value> [<SNP marker file>]
```

Example. Define the residual polygenic proportion to be 20%.

```
ssSNPBLUP GTA 0.20
```

12.2.35 TABLEFILE

Name of covariable table file. See TABLEINDEX command. The default is no table index file is needed.

Syntax:

```
TABLEFILE <filename>
```

Example. Covariable table file is FinTDMpara.cov.

```
TABLEFILE FinTDMpara.cov
```

12.2.36 TABLEINDEX

Name of integer number column in the data file indicating column for table index. The default is no table index. See TABLEFILE.

Syntax:

```
TABLEINDEX <integer column name>
```

Example. Index for the covariable table file is on the integer number column DIM. in the data file.

```
TABLEINDEX DIM
```

12.2.37 TAFILE

Gives file having the $\mathbf{T}_A$ matrix in $\mathbf{G}^{-1} = ((1-w)\mathbf{Z}_c\mathbf{B}\mathbf{Z}_c' + w\mathbf{A}_{gg})^{-1} = \frac{1}{w}\mathbf{A}_{gg}^{-1} - \mathbf{T}_A'\mathbf{T}_A$ used by the ssGTABLUP model of the single-step method where w is the residual polygenic proportion, $\mathbf{Z}_c$ is the centered marker matrix, and $\mathbf{B}$ has the scaling information. The approach requires the $\mathbf{A}_{gg}^{-1}$ matrix to be given separately using the command IA22FILE, e.g., by IA22FILE PEDIGREE.

The format of the $\mathbf{T}_A$ matrix file is like the lower triangle dense matrix format. However, the $\mathbf{T}_A$ matrix is a rectangular matrix. The $\mathbf{T}_A$ matrix file has to be made by a special preprocessing program such as T48eig_make.

Syntax:

```
TAFILE  <filename>
```

Example. Matrix $\mathbf{T}_A$ is in file TA.bin

```
TAFILE TA.bin
```

12.2.38 TEFILE

Gives file having the $\mathbf{T}_e$ matrix in $\mathbf{G}^{-1} = (\mathbf{Z}_c\mathbf{B}\mathbf{Z}_c' + \epsilon\mathbf{I})^{-1} = \frac{1}{\epsilon}\mathbf{I} - \mathbf{T}_e'\mathbf{T}_e$ used by the ssGTeBLUP model of the single-step method where ϵ is a small number such as 0.01, $\mathbf{Z}_c$ is the centered marker matrix, and $\mathbf{B}$ has the scaling information. The approach requires the $\mathbf{A}_{gg}^{-1}$ matrix to be given separately using the command IA22FILE, e.g., by IA22FILE PEDIGREE.

The format of the $\mathbf{T}_e$ matrix file is like the lower triangle dense matrix format. However, the $\mathbf{T}_e$ matrix is a rectangular matrix. The $\mathbf{T}_e$ matrix file needs to be made by a special preprocessing program such as `T48eig_make`.

Syntax:

```
TEFILE  <filename>
```

Example. Matrix $\mathbf{T}_e$ is in file `Te.bin`

```
TEFILE Te.bin
```

12.2.39 TITLE

Line for title of the analysis. The default title is:
`MiX99 preprocessing time&date: <current time and date>`

Syntax:

```
TITLE <Title of the analysis>
```

Example.

```
TITLE The new multi-trait model for yield
```

12.2.40 TMPDIR

Directory for temporary files. The default is the current directory.

Syntax:

```
TMPDIR <directory>
```

Example.

```
TMPDIR ./tmpMiX
```

12.2.41 TRAITGROUP

Name of integer number column for the trait group.

Syntax:

```
TRAITGROUP <integer column name>
```

Example. Trait group is in integer number column `trait`.

```
TRAITGROUP trait
```

12.2.42 WEIGHTFILE

DEV

This command allows using marker-specific weights in a file for an ssSNPBLUP model when `DWEIGHT` option has been given in the `SNPMATRIX` command. By default, all variances are assumed to have the same weight of one.

```
WEIGHTFILE VR_weights.dat
```

12.2.43 WITHINBLOCKORDER

Ordering of effects within block. Order of effect after the command name gives the ordering. The default is that only the individual additive genetic effect is a within block effect. If the model has no such, then the last model effect.

Syntax:

```
WITHINBLOCKORDER <effect names>
```

Example. There are three effects within block. Order of effects within block: 1=G, 2=PE, 3=HerdXyear.

```
WITHINBLOCKORDER G PE HerdXyear
```

12.2.44 ZCFILE

Gives file having the centered marker matrix $\mathbf{Z}_c$ matrix in the basic component-wise ssGTBLUP model of the single-step method. Note: the fully component-wise ssGTBLUP can be computationally more efficient as it does not require this file. For example, in ssGTABLUP, $\mathbf{G}^{-1} = ((1-w)\mathbf{Z}_c\mathbf{B}\mathbf{Z}_c'+w\mathbf{A}_{gg})^{-1} = \frac{1}{w}\mathbf{A}_{gg}^{-1} - \frac{1}{w^2}\mathbf{A}_{gg}^{-1}\mathbf{Z}_c\mathbf{C}^{-1}\mathbf{Z}_c'\mathbf{A}_{gg}^{-1}$ where w is the residual polygenic proportion, $\mathbf{B}$ has scaling information and $\mathbf{C}^{-1} = (\frac{1}{w}\mathbf{Z}_c'\mathbf{A}_{gg}^{-1}\mathbf{Z}_c+\mathbf{B}^{-1})^{-1}$. The command requires giving the ICFILE command having $\mathbf{C}^{-1}$. The $\mathbf{Z}_c$ matrix file can be made by a special preprocessing program such as T48eig_make. The approach requires the $\mathbf{A}_{gg}^{-1}$ matrix to be given separately using the command IA22FILE within option PEDIGREE.

Syntax:

```
ZCFILE  <filename>
```

Example. Matrix $\mathbf{Z}_c$ is in file Zc.dat

```
ZCFILE Zc.dat
```

13 Appendix: Quick reference card

The following commands are necessary. Options are in square brackets [ ]. Syntax and short explanation of the required commands are in Chapter 12.1 except for command MODEL which is considered separately in Chapters 6, 7, 9 and 11. Note that in CLIM and in the following, symbol '&' is continuation to the next line.

```
DATAFILE [TEXT | BINARY] <FileName>
INTEGER  <column names>
REAL     <column names>

PARFILE  <FileName>

DATASORT   BLOCK=<block in INTEGER> PEDIGREECODE=<code in INTEGER>
MISSING    <value for missing real number data>
NORANSOL   <random effect numbers without solution file>
PARFILE    <filename>
RANDOM     <random effect names>
REGFILE    <filename>
REGPARFILE <filename>
REGMATRIX  <type> <name> [ID=<column>] FIRST=<column> [LAST=<column>]
           [REGINDEX=<INTEGER column name>] [IMPUTE=<missing>]
           [SCALE={<real value>|<filename>|2pq|m|m2}] [FORMAT=<file format>]
           [CENTER[={<real value>|<filename>}]] [PRECON=<preconditioner>]
RESIDFILE  <filename>
RESIDUAL   <INTEGER column name of the residual variance number>
SSGBLUP    [LOWER|MIXED] <filename>
TABLEFILE  <filename>
TABLEINDEX <table index INTEGER column name>
TITLE      <title of analysis>
TMPDIR     <directory>
TRAITGROUP <trait group INTEGER column name>
WITHINBLOCKORDER <Effect names in the order>
DEFINE     <macro name> <replacement text>
```

13.1 Reserved and special characters

Special symbols that can't be used in user defined names like data file column names:

#	start for comment
&	symbol for line continuation
" "	string in between the apostrophes is read unchanged
!	options follow (on the model line)
( )	parenthesis used on model line(s)

14 References

Belay, T.K., Eikje, L.S., Gjuvsland, A.B., Nordbø, Ø., Tribout, T., and Meuwissen, T. (2022). "Correcting for base-population differences and unknown parent groups in single-step genomic predictions of Norwegian Red cattle". In: *J. Anim. Sci.* 100.9, skac227. DOI: 10.1093/jas/skac227 (cit. on p. 121).

García-Cortés, L.A., Moreno, C., Varona, L., and Altarriba, J. (1995). "Estimation of prediction-error variances by resampling". In: *J. Anim. Breed. Genet.* 112.1-6, pp. 176–182. DOI: 10.1111/j.1439-0388.1995.tb00556.x (cit. on p. 142).

Himmelbauer, J., Schwarzenbacher, H., Fuerst, C., and Fuerst-Waltl, B. (2024). "Exploring unknown parent groups and metafounders in single-step genomic BLUP: Insights from a simulated cattle population". In: *J. Dairy Sci.* 107.10, pp. 8170–8192. DOI: 10.3168/jds.2024-24891 (cit. on p. 121).

Jairath, L., Dekkers, J.C.M., Schaeffer, L.R., Liu, Z., Burnside, E.B., and Kolstad, B. (1998). "Genetic evaluation for herd life in Canada". In: *J. Dairy Sci.* 81.2, pp. 550–562. DOI: 10.3168/jds.S0022-0302(98)75607-3 (cit. on p. 129).

Legarra, A., Christensen, O.F., Vitezica, Z.G., Aguilar, I.i, and Misztal, I. (2015). "Ancestral relationships using metafounders: finite ancestral populations and across population relationships". In: *Genetics* 200.2, pp. 455–468. DOI: 10.1534/genetics.115.177014 (cit. on p. 122).

Lidauer, M., Matilainen, K., Mäntysaari, E. A., Pitkänen, T. J., Taskinen, M., and Strandén, I. (2023a). *Technical Reference Guide for MiX99 Solver*. Release X/2023. Natural Resources Institute Finland (Luke) (cit. on p. 6).

Lidauer, M., Matilainen, K., Mäntysaari, E. A., Pitkänen, T. J., Taskinen, M., and Strandén, I. (2023b). *Technical reference guide for MiX99 pre-processor*. Release X/2023. Natural Resources Institute Finland (Luke) (cit. on pp. 11, 27, 30, 150, 151, 155).

Liu, Z., Goddard, M. E., Reinhardt, F., and Reents, R. (2014). "A single-step genomic model with direct estimation of marker effects". In: *J. Dairy Sci.* 97.9, pp. 5833–5850. DOI: 10.3168/jds.2014-7924 (cit. on pp. 110, 116).

Mäntysaari, E. A., Evans, R. D., and Strandén, I. (2017). "Efficient single-step genomic evaluation for a multibreed beef cattle population having many genotyped animals". In: *J. Anim. Sci.* 95.11, pp. 4728–4737. DOI: 10.2527/jas2017.1912 (cit. on p. 110).

Masuda, Y., Tsuruta, S., Bermann, M., Bradford, H. L., and Misztal, I. (2021). "Comparison of models for missing pedigree in single-step genomic prediction". In: *J. Anim. Sci.* 99.2, skab019. DOI: 10.1093/jas/skab019 (cit. on pp. 121, 122).

Matilainen, K., Mäntysaari, E. A., Lidauer, M., Strandén, I., and Thompson, R (2012). "Employing a Monte Carlo algorithm in expectation maximization restricted maximum likelihood estimation of the linear mixed model". In: *J. Anim. Breed. Genet.* 129.6, pp. 457–468. DOI: 10.1111/j.1439-0388.2012.01000.x (cit. on p. 135).

Matilainen, K., Mäntysaari, E. A., Lidauer, M., Strandén, I., and Thompson, R. (2013). "Employing a Monte Carlo algorithm in Newton-type methods for restricted maximum likelihood estimation of genetic parameters". In: *PLoS ONE* 8.12. e80821. DOI: 10.1371/journal.pone.0080821 (cit. on p. 139).

Matilainen, K., Strandén, I., Aamand, G.P., and Mäntysaari, E. A. (2018). "Single step genomic evaluation for female fertility in Nordic Red dairy cattle". In: *J. Anim. Breed. Genet.* 135.5, pp. 337–348. DOI: 10.1111/jbg.12353 (cit. on p. 121).

Meuwissen, T. H. E., De Jong, G., and Engel, B. (1996). "Joint estimation of breeding values and heterogeneous variances of large data files". In: *J. Dairy Sci.* 79.2, pp. 310–316. DOI: 10.3168/jds.S0022-0302(96)76365-8 (cit. on pp. 21, 30).

MiX99 Development Team (2025). *MiX99: A software package for solving large mixed model equations*. Release IX/2025. Natural Resources Institute Finland (Luke). Jokioinen, Finland. URL: http://www.luke.fi/mix99 (cit. on p. ii).

Misztal, I., Legarra, A., and Aguilar, I. (2014). "Using recursion to compute the inverse of the genomic relationship matrix". In: *J. Dairy Sci.* 97.6, pp. 3943–3952. DOI: 10.3168/jds.2013-7752 (cit. on pp. 97, 110).

Mrode, R. A. and Swanson, G. J. T. (2004). "Calculating cow and daughter yield deviations and partitioning of genetic evaluations under a random regression model". In: *Livest. Prod. Sci.* 86.1–3, pp. 253–260. DOI: 10.1016/j.livprodsci.2003.09.001 (cit. on p. 27).

Mrode, R.A. and Thompson, R. (2006). *Linear models for the prediction of animal breeding values*. CABI. DOI: 10.1079/9780851990002.0000 (cit. on p. 4).

Pitkänen, T. J. et al. (2022). "From data to genomic breeding values with the MiX99 software suite". In: *Proc. 12th World Congr. Genet. Appl. Livest. Prod.* Wageningen Academic Publishers. Rotterdam, The Netherlands, pp. 1534–1537 (cit. on p. ii).

Schaeffer, L. R. and Dekkers, J. C. M. (1994). "Random regressions in animal models for test-day production in dairy cattle". In: *Proc. 5th World Congr. Genet. Appl. Livest. Prod.* Vol. 18, pp. 443–446 (cit. on pp. 49, 52, 54).

Schaeffer, L.R. (1994). "Multiple-country comparison of dairy sires". In: *J. Dairy Sci.* 77.9, pp. 2671–2678. DOI: 10.3168/jds.S0022-0302(94)77209-X (cit. on pp. 127, 128).

Schaeffer, L.R. (2001). "Multiple trait international bull comparisons". In: *Livest. Prod. Sci.* 69.2, pp. 145–153. DOI: 10.1016/S0301-6226(00)00255-4 (cit. on pp. 129, 132, 133).

Strandén, I. and Christensen, O.F. (2011). "Allele coding in genomic evaluation". In: *Genet. Sel. Evol.* 43.25. DOI: 10.1186/1297-9686-43-25 (cit. on p. 80).

Strandén, I. and Jenko, J. (2024). "A computationally feasible multi-trait single-step genomic prediction model with trait-specific marker weights". In: *Genet. Sel. Evol.* 56.58. DOI: 10.1186/s12711-024-00926-2 (cit. on p. 121).

Strandén, I. and Vuori, K. (Aug. 2006). "RelaX2: pedigree analysis program". In: *Proc. 8th World Congr. Genet. Appl. Livest. Prod.* Belo Horizonte, MiG, Brazil, pp. 27–30 (cit. on pp. 11, 41).

Tyrisevä, A.-M. et al. (2011). "Principal component approach in variance component estimation for international sire evaluation". In: *Genet. Sel. Evol.* 43.21. DOI: 10.1186/1297-9686-43-21 (cit. on p. 139).

VanRaden, P.M. (2008). "Efficient methods to compute genomic predictions". In: *J. Dairy Sci.* 91.11, pp. 4414–4423. DOI: 0.3168/jds.2007-0980 (cit. on pp. 95, 96).

Vandenplas, J. et al. (2023). "Efficient large-scale single-step evaluations and indirect genomic prediction of genotyped selection candidates". In: *Genetics Selection Evolution* 55.1, pp. 1–17. DOI: 10.1186/s12711-023-00808-z (cit. on pp. 116, 124).

Index

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

Page numbers:
- primary definition: **169**
- also referred: 169
- ☞ see example on page