

MiX99

Solving Large Mixed Model Equations

Release I/2022

Command Language Interface Manual (CLIM)

Last update: Feb 2022
©Copyright 2022


Luke
NATURAL RESOURCES
INSTITUTE FINLAND

Preface

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

Disclaimer

MiX99 software is owned by Natural Resources Institute Finland (Luke). When using this program you agree with the following terms. You are not allowed to distribute, copy, give or transfer MiX99, neither under the same nor under a different name. Any decisions based on information given by MiX99 are made at your own responsibility and risk. Only limited technical support can be provided, but vital questions on its use can be directed to the authors (mix99@luke.fi). Please report any bugs to the authors. MiX99 can be referenced by ([MiX99 Development Team, 2022](#)). If you would like to use MiX99, please contact Genetics and Breeding at Natural Resources Institute Finland¹.

MiX99 new (NEW) and development (DEV) features

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

NEW

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

DEV

Authors

Ismo Strandén

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

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

Contents

1	Introduction	1
1.1	Supported statistical models and beta testing features	1
1.2	Organization of the manual	2
1.3	Invoking CLIM and command line options	2
1.4	Simple example	3
2	Theory and notation	4
2.1	Single trait model	4
2.2	Multiple trait model	5
2.3	Solving mixed model equations	6
2.3.1	MiX99 solver: PCG	6
2.3.2	Preconditioner	6
2.3.3	Iteration on data	6
2.3.4	Ordering of equations by blocks	7
3	MiX99 files	7
3.1	Data file	7
3.1.1	Example: Multiple trait data	8
3.2	Pedigree file	8
3.2.1	Example: Pedigree file for the data	8
3.2.2	Phantom parent groups	9
3.2.3	Block code	9
3.3	Variance components file	10
3.3.1	Example: Variance component file	10
3.3.2	Multiple residual (co)variances	10
4	Using the MiX99 solver	11
4.1	Solution files	12
5	Single trait models	13
5.1	Naming of model components	13
5.1.1	Example: Animal model	14
5.1.2	Example: Phantom parent groups in animal model	15
5.1.3	Example: Inbreeding in animal model	15
5.1.4	Example: Repeatability animal model	16
5.1.5	Example: Repeatability animal model in detail	17
5.1.6	Example: Simple sire model	19
5.1.7	Example: Sire model	20
5.1.8	Example: Weights in a model	20
5.2	Random regression and nested effects	21
5.2.1	Nested regression effects	21
5.2.2	Covariable tables	21

	5.2.3	Example: Random regression model	22
	5.2.4	Example: Covariable table and random regression model	24
	5.2.5	Example: Heterogeneous residual variance in test day model	26
	5.3	Maternal effect models	27
	5.3.1	Example: Animal model for a maternal trait	28
6		Multiple trait models	29
	6.1	All traits have the same effects	30
	6.1.1	Example: Simple multiple trait model	30
	6.2	Traits have different effects	30
	6.2.1	Example: Multiple trait model with different effects by trait	31
	6.2.2	Example: Different effects by trait using CLIM beta fea- tures	31
	6.3	Multiple trait random regression model	32
	6.3.1	Example: Multiple trait random regression model	32
	6.4	Combining of trait estimates	34
	6.4.1	Example: Repeatability model by multiple trait model	35
	6.4.2	Example: Reduced rank random regression model	36
	6.4.3	Example: Finnish test-day model	36
	6.5	Multiple trait maternal effects model	37
7		Genomic data models	38
	7.1	SNP-BLUP or genomic effect model	38
	7.1.1	Example: simple genomic marker BLUP	39
	7.1.2	Enhanced formatting of SNP marker information	41
	7.2	Example: simple G-BLUP	42
	7.2.1	Example: G-BLUP with polygenic effect	46
	7.3	Example: single-step method	48
	7.3.1	Large number of genotyped animals in single-step GBLUP: ssGTBLUP	52
	7.3.2	Metafounders	54
8		Special topics	54
	8.1	Trait groups for single trait analysis	54
	8.1.1	Example: Multiple single trait analysis	54
	8.1.2	Example: MACE or Sire model with weights and trait groups	56
	8.2	Deregression	58
	8.2.1	Example: Single trait deregression	59
	8.2.2	Example: Multiple trait deregression	61
9		Summary of all commands	65
	9.1	Required commands	66
	9.1.1	MODEL	66
	9.1.2	DATAFILE	66
	9.1.3	INTEGER	66
	9.1.4	PARFILE	66
	9.1.5	PEDFILE	67
	9.1.6	PEDIGREE	67
	9.1.7	REAL	68
	9.2	Optional commands	68

Command Language Interface Manual (CLIM)

9.2.1	AR	68
9.2.2	COVFILE	68
9.2.3	DATASORT	68
9.2.4	IA22FILE	69
9.2.5	ICFILE	69
9.2.6	IGFILE	69
9.2.7	IHPRECON	70
9.2.8	INBREEDING	70
9.2.9	INBRFILE	70
9.2.10	NORANSOL	71
9.2.11	MISSING	71
9.2.12	RESTARTSOL	71
9.2.13	SCALE	71
9.2.14	PARALLEL	71
9.2.15	PRECON	72
9.2.16	RANDOM	72
9.2.17	REGFILE	73
9.2.18	REGMATRIX	73
9.2.19	REGPARFILE	74
9.2.20	RESIDFILE	74
9.2.21	RESIDUAL	75
9.2.22	TABLEFILE	75
9.2.23	TABLEINDEX	75
9.2.24	TAFILE	75
9.2.25	TEFILE	76
9.2.26	TITLE	76
9.2.27	TMPDIR	76
9.2.28	TRAITGROUP	76
9.2.29	WITHINBLOCKORDER	76
9.2.30	ZCFILE	77
9.2.31	DEFINE	77
10	Appendix: Quick reference card	79
10.1	Reserved and special characters	79
11	References	80
	Index	81

1 Introduction

MiX99 is a software suite for breeding value evaluation. The set has three types of programs: preprocessor (`mix99i`), solver (`mix99s`, `mix99p`), and reliability calculator (`apax99`, `apax99p`). There are some other programs as well to assist use of these programs. For instance, `imake99` for the [parallel computing](#) programs `mix99p` and `apax99p`. The main purpose of this manual is to describe the command language interface (CLIM) to the preprocessor `mix99i`. Some use of `mix99s` is described as well.

The preprocessing program `mix99i` of MiX99 has two ways to give instructions: the original interface, and **command language interface**. In the original interface a **directive file** is given to `mix99i`. The directive file answers questions on data and statistical model. The **command language interface** for MiX99 is called **CLIM**. CLIM helps user in use of the MiX99 preprocessing program `mix99i`. CLIM has the following advantages over the directive file:

- Commands can be given in any order. Thus, the restriction on the strict order on giving commands has been lifted.
- Some commands have default values. Thus, 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 on the statistical model are given in the same model area, not divided into several sections as in directive file.

Current version of CLIM assumes that model effects are given in the same order as explained for MiX99 directive file. Thus, fixed regression effects without nesting are given first, then fixed classification or [nested](#) fixed effects, and then random effect in order of their random effect number.

1.1 Supported statistical models and beta testing features

The MiX99 software can handle many kinds of statistical models. Many of the models have been implemented in CLIM, but not all. In general, the following models are in CLIM:

- linear mixed effect multiple trait model
- random regression ([covariable table file](#))
- reduced rank (combining of traits)
- multiple residual variances
- weights
- least squares models
- large genome information models

There are some specific models, however, that have not been implemented into CLIM. For example, [random regression model](#) with both maternal and [additive genetic effects](#) has not been implemented. Currently **not implemented** in CLIM:

- random regression with multiple [nesting](#) in additive genetics
- MAS-BLUP (combining of effects)
- non-linear models: threshold and Gompertz models

In beta testing:

- order of effect on the model line is free unlike in a [directive file](#).

1.2 Organization of the manual

This manual is organized in the following way.

- [This chapter](#) gives some introductory remarks on use of 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 are given on computational implementation issues in MiX99 needed to understand some of the commands.
- MiX99 files are briefly described in [the third chapter](#).
- [The fourth chapter](#) has brief description of the MiX99 [solution files](#)
- [The fifth chapter](#) has single trait examples. The section on basic models is required reading because basic concepts of the CLIM interface are explained.
- The [sixth](#) and [seventh](#) chapter gives [multiple trait models](#), and some special topics.
- [The last chapter](#) has syntax of all commands.

The manual concentrates on how different statistical models are given to MiX99 using CLIM. However, some options are not much covered. Please study them in the [Chapter 9](#) on summary of all commands. Some of these commands are quite important. For example, [MISSING](#) and [SCALE](#). And, some options can be important when using the programs, such as [TMPDIR](#), and [TITLE](#). But, some are seldom needed, like [NORANSOL](#).

1.3 Invoking CLIM and command line options

CLIM is called by the preprocessing program [mix99i](#). CLIM reads a **command file**, e.g., [mix99.clm](#), and translates these instructions into a [directive file](#) [MiX99_DIR.DIR](#) to be read by [mix99i](#). CLIM is used by [mix99i](#) when an instruction file is given to it:

```
mix99i mix99.clm
```

Note that the old [directive files](#) are read from the standard input. Thus, executing

```
mix99i < mix99.dir
```

would expect the old [mix99i](#) directive file in the file [mix99.dir](#).

It is possible to give some options on the command line of [mix99i](#) (see [Table 1.1](#)). When CLIM instruction information is used, the preprocessor program [mix99i](#) makes a file [MiX99_DIR.DIR](#). This file has the old [directive file](#) format produced from the

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

Option	Effect
-d	CLIM is executed, no preprocessing part in <code>mix99i</code> . File <code>MiX99_DIR.DIR</code> is produced.
-b	allows use of <code>beta version</code> feature(s), see list above.
-h	help
-l	long listing option of <code>mix99i</code> .
--nproc NPROC	Number of parallel processes.
--datafile DATAFILE	Data file name.
--pedfile PEDFILE	Pedigree file name.
--parfile PARFILE	Variance file name.
--checkdata	Enhanced checking of data file.

CLIM instructions. Thus, if/when CLIM cannot make exactly the model you have in mind, a similar model may be feasible. Then, by using the '-d' option (Table 1.1), `directive file` is made, and this directive file can be used as a template:

```
mix99i -d mix99.clm
```

In any case, it is useful to check that the CLIM generated `directive file` is correct.

1.4 Simple example

Here is a simple example just to introduce CLIM. Some notes on the example:

- everything beyond '#' sign on a line is ignored and is considered as a comment.
- all command information are on a line which can be continued by a continuation symbol '&'.
- the parameter file has the old MiX99 format and assumes the same numbering. Because random effects of the given model are animal genetic and random residual, numbering is 1 for animal genetics and 2 for the residual.

A simple `animal model` with one fixed effect (mean) and random effect (animal):

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

PEDFILE   simple.ped  # Name of pedigree file
PEDIGREE  animal am    # Genetics associated with animal code
              # am=animal model
PARFILE    simple.var  # Name of variance components file
MODEL
y = mean animal      # Model
```

The commands can be shortened from the full command names. In addition, the command names are not case sensitive, although in this manual all keywords will be written in capital letters. Thus, for example, the command `INTEGER` can be written `int`. However, all other names are case sensitive, e.g., `herd_year` name in the above example. Thus, the integer number column names must be written on the model lines exactly as they were given for the `INTEGER` command.

2 Theory and notation

Here we introduce some notation and theory for an [animal model](#). The mixed model equations are not described in detail. Differences in concepts such as [animal](#) and [sire model](#), or maternal and paternal effects are not defined. For a clear presentation on these and many other models in animal breeding with examples, see [Mrode and Thompson \(2006\)](#). This manual uses examples from this book.

2.1 Single trait model

A simple single trait animal model has the form

$$y = Xb + Za + e$$

where

- y is $n \times 1$ vector of observations,
- b is $p \times 1$ vector of fixed effects,
- X is $n \times p$ design matrix to link observations to appropriate fixed effects,
- a is $q \times 1$ vector of random additive genetic effects,
- Z is $n \times q$ design matrix to link observations to appropriate random effects,
- e is $n \times 1$ random residual vector.

Hence, there are q animals with n observations, and there are p fixed effect.

In a simple [animal model](#), the matrices X and Z are **incidence matrices**. In other words, these matrices have zeros and ones to indicate which effect corresponds to which observation. However, if model has regression coefficients, these matrices have regression coefficients. Thus, we call these matrices **design matrices** to indicate wider model possibilities.

In MiX99 it is possible to have both regression and classification variables (categories). Classification variables will be sometimes 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, linear function has the linear coefficient in the [design matrix](#). However, [regression effect](#) can be [nested](#) within classification. In practice, difference between regression and classification effects in MiX99 is that a class effect number is in integer number input column, but regression coefficient is in the real number input column of the [data file](#).

Common linear mixed effects assumptions for the expectations are

$$\begin{array}{ll} E(a) &= \mathbf{0} \\ E(e) &= \mathbf{0} \\ E(y) &= Xb \end{array} \qquad \begin{array}{ll} \text{Var}(a) &= A\sigma_a^2 \\ \text{Var}(e) &= I\sigma_e^2 \\ \text{Cov}(a, e) &= \mathbf{0} \end{array}$$

where A is numerator relationship matrix.

For convenience of presentation, it is common to denote the residual covariance matrix by R . Also, it is common to denote the genetic covariance matrix by G . With this

notation, the ***mixed model equations*** to solve are

$$\begin{bmatrix} X'R^{-1}X & X'R^{-1}Z \\ Z'R^{-1}X & Z'R^{-1}Z + G^{-1} \end{bmatrix} \begin{bmatrix} \hat{b} \\ \hat{a} \end{bmatrix} = \begin{bmatrix} X'R^{-1}y \\ Z'R^{-1}y \end{bmatrix}$$

where ' denotes transpose.

The model can be described by giving its effects. For example, if the above model had fixed herd effect (*herd*), and animal 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 model for one individual record, although subscripting to indicate this was not used.

2.2 Multiple trait model

The single trait model can be used to describe multiple trait model as well:

$$y = Xb + Za + e$$

However, for *T* traits the matrices and vectors have traitwise structure. Thus, we can write

$$\begin{aligned} y' &= [y'_1 \quad y'_2 \quad \cdots \quad y'_T] \\ e' &= [e'_1 \quad e'_2 \quad \cdots \quad e'_T] \\ b &= \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_T \end{bmatrix}, \quad X = \begin{bmatrix} X_1 & 0 & \cdots & 0 \\ 0 & X_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X_T \end{bmatrix} \\ a &= \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_T \end{bmatrix}, \quad Z = \begin{bmatrix} Z_1 & 0 & \cdots & 0 \\ 0 & Z_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Z_T \end{bmatrix} \end{aligned}$$

where the vectors and matrices have the same meaning as before and subscripts denote for appropriate trait.

Multiple trait linear mixed effects assumptions are

$$\begin{aligned} E(a) &= 0 & \text{Var}(a) &= G_0 \otimes A \\ E(e) &= 0 & \text{Var}(e) &= R_0 \otimes I \\ E(y) &= Xb & \text{Cov}(a, e) &= 0 \end{aligned}$$

where matrix G_0 is *T* by *T* **genetic covariance matrix**, and R_0 is *T* by *T* **residual covariance matrix**. The **mixed model equations** are

$$\begin{bmatrix} X'R^{-1}X & X'R^{-1}Z \\ Z'R^{-1}X & Z'R^{-1}Z + G_0^{-1} \otimes A^{-1} \end{bmatrix} \begin{bmatrix} \hat{b} \\ \hat{a} \end{bmatrix} = \begin{bmatrix} X'R^{-1}y \\ Z'R^{-1}y \end{bmatrix}$$

2.3 Solving mixed model equations

MiX99 solves mixed model equations using [preconditioned conjugate gradient](#) (PCG) iteration. The following need to be considered when using the program:

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

2.3.1 MiX99 solver: PCG

[PCG](#) is an iterative method to solve linear models. Thus, the coefficient matrix is not inverted or decomposed. In practice, iterative methods will solve the system of equations by updating the latest solutions by some procedure.

An ***iterative solving method*** continues updating solutions until convergence is determined. Convergence criteria value is given before starting iteration. Often this value is a small positive number. In addition, maximum number of iterations is set in order to ensure termination of the iterative method. It has been proved that the number of iterations needed by [PCG](#) is at most the number of unknowns. This has no practical meaning for large problems, but for small problems it is good to know that this limit is used in MiX99.

It is not possible to give convergence criteria or [maximum number of iterations](#) by CLIM. This information is given to the solver program `mix99s` or `mix99p`, not the preprocessor (see Chapter 4).

2.3.2 Preconditioner

[PCG](#) iteration is a flexible method. The CG or ***conjugate gradient*** part of the algorithm is the same (with small variations) but the preconditioner part depends on implementation. Preconditioning usually means transforming the [coefficient matrix](#) to be as diagonal as possible. This is done by approximating inverse of the [coefficient matrix](#). If the approximation is exact, [PCG](#) finds the correct solution in one step. Usually, the approximation is not exact.

MiX99 has different preconditioners available. The reason for different preconditioners is [memory](#) and time. For very large systems of equations, it is not possible to use the most [memory](#) intensive preconditioners implemented. In addition, when only [reliabilities](#) are calculated, no preconditioner is needed. Making the preconditioner may take as much as half of the computing time in the preprocessing program.

2.3.3 Iteration on data

Iteration on data (IOD) means that MiX99 does not make or store the [coefficient matrix](#) of the [mixed model equations](#) to [memory](#). [PCG](#) method needs the coefficient ***matrix times a vector product***. This can be made using the model matrices X and Z , pedigree list, and the [variance component](#) information. In practice, IOD means that reliabilities must be calculated by a separate program because [coefficient matrix](#) is never made explicitly.

2.3.4 Ordering of equations by blocks

We described [mixed model equations](#) in a way that is typical in the literature. Equations are **ordered** by effect. This is convenient when presenting [mixed model equations](#) or theory. But, this is not always computationally optimal. It is better to order equations of animal and its herd close because IOD proceeds one records at a time with a record having animal and herd related classification information. In MiX99, equations can be ordered by common family blocks, e.g., herd. For more information see Chapter [3.2.3](#).

MiX99 orders equations in a different way than by effect even when [block code](#) has not been used to order equations. In [multiple trait model](#), the different trait equations for an animal are always ordered next to each other. This will ensure efficient performance of the [solver](#).

CLIM has command [WITHINBLOCKORDER](#) (Chapter [9.2.29](#)). It can be used to indicate which effects are **within block equations**. For example, if herd is [block code](#), then it is natural that animal effects (such as animal genetics and permanent environment), and herd contemporary effects (such as herd-year-season) are within block. This is not very important when solving the [mixed model equations](#) using [mix99s](#). However, the [parallel computing](#) implementation ([mix99p](#)) depends on good [block ordering](#). When [reliabilities](#) are estimated by ApaX ([apax99](#) or [apax99p](#)), block information is used to determine level of approximation. Only effects within blocks are considered in reliability calculations.

3 MiX99 files

Information on data, pedigree, and variance components for MiX99 are in files. Animals in the [data](#) and [pedigree files](#) must be in the same order. In other words, when data records have been sorted by animal id (or sire id for [sire models](#)), then individuals must be in the same order in the pedigree file. The pedigree can be ordered using RelaX2 program, a separate program for pedigree analysis from Luke ([Strandén and Vuori, 2006](#)).

The MiX99 input files have a certain quite simple format. In the following, formats of these files are described shortly. For a more complete explanation, see MiX99 pre-processor manual [Technical reference guide for MiX99 pre-processor](#).

3.1 Data file

The data file has the observed data to be analyzed. This means observations, and model effect information such as of classification effect numbers and regression coefficients. The default format is 'text' which means **text format data file** where columns are separated by space. A rarely used alternative is **binary format 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 index for heterogeneous residual variance.
- **Real numbers** These are observations, regression coefficients, and [weights](#).

The data file can have columns that are not used in a particular run of MiX99. Because MiX99 accepts only numerical data, alphanumeric data are allowed on the record only after the real number columns in a free format text [data file](#).

All integer numbers are coded using the default machine integer type. Hence, on 32-bit platforms the data file, integer numbers must be positive and at most 2.147.483.648. **Missing integer numbers** must be coded by number zero (0). **Missing real numbers** can be coded with an arbitrary real value which is specified to MiX99 by command **MISSING**.

3.1.1 Example: Multiple trait data

Consider data for a two-trait model. The file has six columns: 4 integer number columns, and 2 real number columns. Note that the [real number columns](#) have integer values. Still, for MiX99 these are real number columns because observations can have any real values. The file named `example.dat` is:

animal 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

This [data file](#) can be described with the following CLIM commands:

```
DATAFILE example.dat           # Name of the data file
INTEGER animal sire season ones # Integer column names
REAL tr1 tr2                   # Real column names
```

3.2 Pedigree file

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

1	2	3	4
animal 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.

When [block code](#) is given, the pedigree and observation files need to have the same order by block. The main sort key is [block code](#) (e.g., herd) within which sorting is by animal code. When animal has observations in several data file blocks, in pedigree file the animal must appear only in one of the blocks. This special case will be considered in a separate section on [parallel computing](#).

3.2.1 Example: Pedigree file for the data

Let [pedigree file](#) for the [multiple trait model](#) data in example Chapter 3.1.1 be

animal 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

3.2.2 Phantom parent groups

Missing parents can be replaced by genetic (unknown parent) groups. Then, a **genetic group** code is in place of missing parent. This group code must be a negative integer number in order to distinguish it from an animal code. A genetic group code must not have an own record in the [pedigree file](#).

For example, the pedigree above (Chapter 3.2.1) had two animals (1 and 2) that had unknown parents. The parents could be genetic groups (-1 for unknown sire, -2 for unknown dam). The [pedigree file](#) remains the same for all other animals. The changed part of the pedigree file is:

animal ₁	sire ₂	dam ₃
1	-1	-2
2	-1	-2

3.2.3 Block code

There are some benefits from having a **block code** in the data and pedigree files. Block code is essential in calculation of [reliabilities](#) by [ApaX](#), and when [parallel computing](#) is used by [mix99p](#). Otherwise, use of a block code reduces computing time very little and can be omitted.

Block code of an animal is given on column 4 in the pedigree file. When the pedigree file has a block code column, then every animal must have a block code. In addition, the block code needs to be the same in the [data file](#) as well. Animals with records in different data blocks (e.g. in different herds) have to be coded with the code of one of the different data blocks where it has observations, e.g., block with most of its observations.

If animal does not have an observation, but is parent to an animal having observation(s) in the [data file](#), then it is best that the parent without observation and its offspring have the same block code. An example in dairy cattle is a cow without observations. It should be assigned to a block having most of its daughters.

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

that a group should never be too large. It is advisable to split a large block into several smaller blocks. The solver program reads as many animal blocks at a time as possible, and the largest animal block dictates [memory requirements](#).

3.3 Variance components file

The **variance components** file has variances and covariances for all the random effects in the model. The matrices can be of different size depending on the model specification. The matrices are numbered in the same order as in the **RANDOM** command. However, no random command is needed when the only random effects in the model are animal genetics and residual effects. Residual effect has always the highest random effect number, and the [additive genetic effect](#) the second highest number.

The [variance components](#) file has a line for each (co)variance. Each line has 3 integers followed by a real number (the (co)variance value). The first integer is the random effect number followed by the row-column combination, and, finally, the (co)variance parameter. The row-column combination refers to the element position in the (co)variance matrix. Only the lower (or upper) triangle of the matrix needs to be given.

Order of lines in the file is irrelevant. It is easy to know the random effect number. Correct numbering of (co)variances, i.e., the row-column number, can be more difficult. For example, the row-column numbers have to be checked carefully when random regression effects are missing in a [multiple trait random regression model](#). See examples on multiple trait random regression effects for better explanation (Chapter 5.2).

3.3.1 Example: Variance component file

We illustrate [variance components](#) file for the [multiple trait model](#) data in example Chapters 3.1.1 and 3.2.1. Let the genetic and residual (co)variance matrices be

$$G = \begin{bmatrix} 3.0 & 2.5 \\ 2.5 & 2.5 \end{bmatrix}$$

and

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

respectively. Genetic correlation between the traits is about 91%, and residual correlation about 29%. Heritability of the first trait is 30%, and the second trait about 26%. The parameter file is

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

3.3.2 Multiple residual (co)variances

When multiple residual (co)variances are present, an **additional residual (co)variance file** has to be given. Format of this file is like the regular (co)variance file explained

above. However, the first number on each line is not the random effect number but number of the residual variance class. Numbering of the residual (co)variance classes has to start from one (1), up to 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 given in the **variance components** file. Values of residual (co)variance in the variance components file are ignored by the solver program (**mix99s/mix99p**) but used by the reliability calculation program (**apax99/apax99p**).

4 Using the MiX99 solver

The MiX99 **solver** (**mix99s/mix99p**) assumes user will give some instructions on some aspects of the iteration method, output files produced, and possible special computing to be made (see Chapter 8 on special topics). The instructions can be given either from the standard input or using the **command line options**.

In this manual, usually the **command line option** method has been used. This method is possible only when calculating breeding values. The easiest way to execute solver is to give option **-s** which uses default values in solving breeding values, and produces standard output files. Thus, you give **mix99s -s**. Examples in this manual have been produced with this option, if not otherwise mentioned.

The other command line options are

- **-n** or **-N** for number of iterations
- **-ca** or **-Ca** for Ca convergence criteria
- **-cd** or **-Cd** for Cd convergence criteria
- **-cr** or **-Cr** for Cr convergence criteria
- **-cm** or **-Cm** for Cm convergence criteria

For example, giving **mix99s -n 100 -cr 1e-8** would limit **maximum number of iterations** to 100, and the Cr convergence criteria value to 10^{-8} .

Instructions can be given to the solver in the standard input. This allows much wider set of options and methods than available in the **command line options**. Note, however, that giving command line options will by default lead to not reading the standard input. It is sometimes more convenient to have the instructions in a file than type them every time to the program. This can be achieved by reading them from the standard input, e.g., **mix99s < solver_option_file.slv**. Again, giving command line options will by default lead to not reading the file. Thus, giving

```
mix99s -n 100 -cr 1e-8 < solver_option_file.slv
```

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

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


```
mix99s -i -n 100 -cr 1e-8 < solver_option_file.slv
```

An example of instruction file for breeding value evaluation is

```
H      # RAM: RAM demand: L=large (mix99p only), H=high, M=medium, L=low
# Max. no. iter., Convergence_criterion, Criterion (A/R/M/D)
2000      1.0e-8      R      F
N      # RESID: Calculate residuals? (Y/N)
N      # VALID: N=no
N      # VAROPT: adjust for HV? (N)o
Y      # SOLTYP: Solution files? (N)o, (Y)es
```

The first letter **H** requests 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.

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

- number of iterations in the **PCG** method is limited to 2000 iterations
- convergence value is set to be 10^{-8}
- **convergence criteria** is set to "R"
- the above values are "F"orced to be used.

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

- limit to 5000 iterations in the **PCG** method
- convergence value is 10^{-4}
- **convergence criteria** is "D"

The three options after the **PCG** information are not that important for typical breeding value evaluation, and their values are "n" for no. The chapter on special topics (Chapter 8) will consider some of these options. The last "Y" is important. If the last letter is "N" then instead only binary format solution file is produced.

4.1 Solution files

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

- **Solani**: Solutions for animal effects. (Sol_{mn} in case of a LS-model.)
- **Solfix**: Solutions for all across blocks fixed effects.
- **Solfnn**: Solutions for the n^{th} within block fixed effect. For instance, Sol_{f02} is the solution file for the second **within block fixed effect**.
- **Solrnn**: Solutions for the n^{th} random effect in the model. For example, Sol_{r03} 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).

Structure of the text solution files depends on the model. General form of a particular solution file is the same. However, number of columns in a **Solani** file depends on

the number of traits. Therefore, detailed explanation of the content of those files is given in the printout of the program.

Below are descriptions of the two most common solution file formats. The other solution files have similar formats. Please check solver output for explanation. In this manual column titles are given for `Solani` although they are not present in the files.

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

- 1) Animal ID
- 2) Number of offspring
- 3) Number of observations
- 4) Solution for trait 1
- 5) Solution for trait 2
- 6) ...

When there are random regression effects, solutions are in the numbering order of the random regression effects.

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

5 Single trait models

5.1 Naming of model components

`Data file` has `integer` and `real number columns`. Columns are given names by `INTEGER` and `REAL` commands. The names can have any alphanumeric characters, i.e., letters and numbers. Many other characters such as underscore (`_`) are allowed as well. However, there are some `reserved characters` not allowed in names: `=`, `(`, `)`, `@`, `|`, `!`, `<`, `&` and `#`. These characters have special meaning. For example, `#` starts comment, and `&` marks for `line continuation`. Others are model component separators, and will be discussed in due course in this manual.

Statistical model has effects. The data column names can be used as effect names. If a data column name is an `integer number column` name, then it is assumed to be an effect with classes. If a data column name is a `real number column` name, then it is assumed to be a `regression effect`. CLIM expects that all effect names are different

on a model line. When some model effects refer to the same data column, [component names](#) can be used. See repeatability model example (Chapters [5.1.4](#) and [5.1.5](#)).

5.1.1 Example: Animal model

We consider a simple [animal model](#)

$$tr1 = herd \times year + a + e$$

where

$herd \times year$ is fixed $herd$ times $year$ interaction effect,
 a is random [additive genetic effect](#), and
 e is random residual.

CLIM (nor MiX99) does not make multiplication operations between effects in the model line. Thus, the $herd \times year$ interaction has to be coded in the data as a class effect.

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

```
DATAFILE example.dat
INTEGER animal sire herd_year ones
REAL tr1 tr2

PEDFILE AM.ped # Pedigree file
PEDIGREE animal am # Genetics associated with animal code
# am=animal model
DATASORT PEDIGREECODE=animal

PARFILE AM.var

MODEL
tr1 = herd_year animal
```

The `example.dat` is the same as given earlier (Chapter [3.1.1](#)). The `AM.ped` is the same as given earlier (Chapter [3.2.1](#)). The variance components file (`AM.var`) is for the first trait:

Random effect ₁	Row ₂	Column ₃	Variance ₁
1	1	1	3.0
2	1	1	7.0

First the [preprocessor](#) is executed: `mix99i amodel.clm`. Next 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	herd_yea	tr1
1	1	2	3	122.69	herd_yea	tr1

The `Solani` file is (column names have been added)

Animal	N-Desc	N-Obs	Solution
1	2	0	-.18406E-14
2	2	0	-.18406E-14

3	2	0	0.92308
4	2	1	-.92308
5	3	0	-.37713E-14
6	3	1	1.8462
7	1	0	0.65421
8	1	1	0.64447E-01
9	0	1	2.0506
10	0	1	-.17840

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

5.1.2 Example: Phantom parent groups in animal model

Genetic groups are as easy to give in CLIM as in the MiX99 directive method. Phantom parent groups are signaled in the **pedigree file** by having them as negative numbers. In the CLIM file, a model with genetic groups is used when the **PEDIGREE** command has '+p' for genetic groups. For example, the previous CLIM **instruction file** needs only one change:

```
PEDIGREE animal am+p
```

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

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 same as zero.
- if '+p' was given but an animal has a zero (0) parent (instead of negative number), MiX99 assigns this parent to genetic group -99999999.

5.1.3 Example: Inbreeding in animal model

Relationship matrix in MiX99 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**. MiX99 does not calculate inbreeding coefficients, a separate program such as **RelaX2** (Strandén and Vuori, 2006) needs to be used.

Consider the example for **animal model** (Chapter 5.1.1). **Inbreeding coefficients** calculated using **RelaX2** are (file **AM.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 order to read this file, two additional lines are needed in the CLIM code:

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


The solutions will be slightly different. Fixed effect solutions in the `Solfix` file are

Fact.	Trt	Level	N-Obs	Solution	Factor	Trait
1	1	1	2	99.538	herd_yea	tr1
1	1	2	3	122.61	herd_yea	tr1

Breeding values in the `Solani` file are

Animal	N-Desc	N-Obs	Solution
1	2	0	-.13834E-13
2	2	0	-.13834E-13
3	2	0	0.92308
4	2	1	-.92308
5	3	0	-.12953E-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

5.1.4 Example: Repeatability animal model

Repeatability animal model has usually two effects with the same [incidence matrix](#) but different [covariance structure](#). Hence, in the model line the same [integer number column](#) in [data file](#) is referred by two different effects: permanent environment and direct genetic. However, the same name cannot appear twice on the model line. Because of this, component names can 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 `id` but we want to rename this effect to be `animal`. This is achieved by giving `animal(id)` on the model line.

Consider a repeatability [animal model](#)

$$y = \text{herd} \times \text{year} + p + a + e$$

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

$$\begin{aligned} E(\mathbf{p}) &= \mathbf{0} & \text{Var}(\mathbf{p}) &= \mathbf{I}\sigma_p^2 \\ E(\mathbf{a}) &= \mathbf{0} & \text{Var}(\mathbf{a}) &= \mathbf{A}\sigma_a^2 \\ E(\mathbf{e}) &= \mathbf{0} & \text{Var}(\mathbf{e}) &= \mathbf{I}\sigma_e^2 \end{aligned}$$

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

The model has two random effects which refer to the same class name, named `animal`, 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  animal sire herd_year ones
REAL     tr12

PEDIGREE  animal am
MODEL
  tr12 = herd_year animal animal
```

because `animal` appears twice as an effect.

An alternative would be to have two identical columns with animal id number in both of them. Thus, our model line would be

```
INTEGER    animal pe_animal sire herd_year ones
REAL      tr12

PEDIGREE   animal am # animal for animal genetic
RANDOM     pe_animal # permanent environment

MODEL
  tr12 = herd_year pe_animal animal
```

This model is correct. However, now the [data file](#) is larger, and it is necessary to remember to make two columns having the same content. Instead, component name can be used to name model effects.

The preferred way to give repeatability model in CLIM is to refer an effect by a component name. In the following, name 'G' was given to the animal genetic effect.

```
INTEGER    animal sire herd_year ones
REAL      tr12

PEDIGREE   G am # G for animal genetics
RANDOM     animal # permanent environment

MODEL
  tr12 = herd_year animal G(animal)
```

This is just one way to give repeatability model. Two alternatives are (all other but the changed commands are given):

1:

```
PEDIGREE   G am # G for animal genetics
RANDOM     PE # PE for permanent environment
MODEL
  tr12 = herd_year PE(animal) G(animal)
```

2:

```
PEDIGREE   animal am # animal for animal genetics
RANDOM     PE # PE for permanent environment
MODEL
  tr12 = herd_year PE(animal) animal
```

All these versions will produce the same instructions for `mix99i`. Note that [component names](#) can be given to fixed effects as well. See the next chapter.

5.1.5 Example: Repeatability animal model in detail

We use the multiple trait model data already presented (Chapter [3.1.1](#)) but modify it for repeatability model. The model is

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

where $herd \times year$ is fixed herd-year effect, p is random [permanent environment effect](#), a is random [additive genetic effect](#), and e is 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

Command Language Interface Manual (CLIM)

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 multiple trait model [data file](#) is modified for the repeatability model:

animal ₁	sire ₂	herd×year ₃	ones ₄	tr12 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   animal sire herd_year ones
REAL      tr12

DATASORT  PEDIGREECODE=animal

PEDFILE   AM.ped
PEDIGREE  G am # G for animal genetics
RANDOM     PE # PE for permanent environment

PARFILE   rep.var

MODEL
  tr12 = herd_year PE(animal) G(animal)

```

The fixed effect solutions ([Solfix](#)) are

Fact.	Trt	Level	N-Obs	Solution	Factor	Trait
1	1	11	2	99.833	herd_yea	tr12
1	1	12	3	123.01	herd_yea	tr12
1	1	21	2	194.83	herd_yea	tr12
1	1	22	3	129.68	herd_yea	tr12

Permanent environment solutions ([Solr01](#)) are

Animal	N-Obs	Solution
4	2	-.88889
6	2	0.88889
8	2	1.1867
9	2	-.55846
10	2	-.62828

The breeding values estimates ([Solani](#)) are

Animal	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

5.1.6 Example: Simple sire model

We consider a simple sire model using the data introduced for [animal model](#) (Chapter 5.1.1). The [sire model](#) is

$$tr1 = herd \times year + s + e$$

where

$herd \times year$ is fixed herd-year effect,

s is random sire effect, and

e is random residual.

In [sire model](#), records are associated with sire of the animal having record. The [data file](#) is the same as used for [animal model](#) [example.dat](#) in Chapter 3.1.1.

In this simple [sire model](#), all sires are assumed to be unrelated. Thus, the [pedigree file](#) ([SM.ped](#)) is

animal ₁	sire ₂	maternal grand sire ₃
1	0	0
3	0	0
5	0	0
7	0	0

[Variance components file](#) needs to be changed from the [animal model](#) to [sire model](#). Sire genetics make only quarter of the additive genetics. 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

CLIM code for the [sire model](#) is

```
DATAFILE  example.dat

INTEGER   animal sire herd_year ones
REAL      tr1 tr2

PEDFILE   SM.ped
PEDIGREE  G sm

PARFILE   SM.var

MODEL
  tr1 = herd_year G(sire)
```

The fixed effect solutions ([Solfix](#)) are

Fact.	Trt	Level	N-Obs	Solution	Factor Trait
1	1	1	2	100.00	herd_yea tr1
1	1	2	3	123.25	herd_yea tr1

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

5.1.7 Example: Sire model

The previous [sire model](#) example can be analyzed by a [sire model](#) where a sire maternal grand sire relationship matrix is used. The command file does not change, but the [pedigree file](#) is different.

The [pedigree file](#) ([smgms.ped](#)) is

animal 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 solution file [Solfix](#) having fixed effects

Fact.	Trt	Level	N-Obs	Solution	Factor	Trait
1	1	1	2	99.976	herd_yea	tr1
1	1	2	3	123.19	herd_yea	tr1

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

5.1.8 Example: Weights in a model

Weight can be used to indicate that an observation is actually mean from many records. Weight is a real number in the [data file](#). It is indicated in CLIM to be weight by [WEIGHT](#) option in the model line. Model options for a trait come after "!" sign. For example, when column 'weight' has weights then option is '![WEIGHT](#)=weight'.

Consider the previous chapter [sire model](#) example again but with weights. The [data file](#) ([example_w.dat](#)) is now:

animal ₁	sire ₂	herd×year ₃	ones ₄	trait 1 ₁	trait 2 ₂	weight ₃
4	1	1	1	90	200	50
6	3	1	1	110	190	100
8	5	2	1	120	140	60
9	5	2	1	130	120	20
10	7	2	1	120	130	30

CLIM code for weighted sire model is

```
DATAFILE  example_w.dat

INTEGER   animal sire herd_year ones
REAL      tr1 tr2 weight

PEDFILE   smgs.ped
PEDIGREE  G sm
```

```
PARFILE SM.var
```

```
MODEL
```

```
tr1 = herd_year G(sire) ! WEIGHT=weight
```

Fixed effect solutions (**Solfix**) are

Fact.	Trt	Level	N-Obs	Solution	Factor Trait
1	1	1	2	100.68	herd_yea tr1
1	1	2	3	119.33	herd_yea tr1

Breeding value estimates (**Solani**) are

Bull	N-Desc	N-Obs	Solution
1	2	1	-7.0567
3	2	1	7.5018
5	1	2	2.8028
7	0	1	1.6447

5.2 Random regression and nested effects

Random regression models have [regression effects nested](#) within a classification effect. Typical random regression models are [test-day models](#) where lactation curve is fitted for test-day observations.

Class effect has an [integer number column](#) name or a [component name](#) (Chapter 5.1). In the following, we extend the use of [component name](#) as a class effect. Earlier we renamed a class effect such as the [additive genetic effect](#) `G(animal)` and `G(sire)`. The same notation will be used for [nested](#) regression effects.

5.2.1 Nested regression effects

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

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

where `season` and `animal` 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.

The [intercept](#) in the `fixed_curve` can be moved to be a separate class effect. Thus, the above model line can be written also as

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

This moving of `season` effect to be a separate effect is fine for fixed effects. However, for a random effect this cannot be always done because [component names](#) are used to distinguish correlated random regression effects. See below examples for random regression models.

5.2.2 Covariable tables

Regression models in dairy cattle are usually so called test-day models where repeated observations of a cow are modeled during lactation. The regression effects are

functions of days in milk which can have only certain values, e.g., integer values from 1 (one) to 350. Because dairy cattle data sets can be very large, MiX99 allows reducing data file size by using days in milk as an index to a regression coefficient table.

Assume the same fixed effects regression curve as given above. A covariable table file can be given by command **TABLEFILE**. The **data file** must have an integer index column such as days in milk (**DIM**) which is used to indicate regression function coefficients in the table. In our example, the file will have five columns: *DIM*, **intercept** (just ones), linear (equals to *DIM*), quadratic (*DIM*²), and cubic (*DIM*³). The model line can now be written as

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

where *t1*, *t2*, *t3*, *t4* refer to columns two, three, four, five, respectively, in the coefficient table file. In the **covariable table file**, column one has the table index. Data file has an integer number column that has the index to pick the correct line in the coefficient table file. The input column in the data file is indicated by command **TABLEINDEX**. See example in Chapter 5.2.4.

5.2.3 Example: Random regression model

We consider a single trait random regression **animal model**. The example is from Schaeffer and Dekkers (1994).

Cows have **repeated observations** of milk yield. The model is

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

where

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

The random regression function *f* for animal *i* has form

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

Thus, there are three random regression breeding values by animal.

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

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

The parameter file **RRM.var** is

Random effect ₁	Row ₂	Column ₃	Covariance ₁	Comment
2	1	1	44.791	additive genetic: intercept
2	2	1	-0.133	intercept, DIM linear
2	3	1	0.351	intercept, ln(DIM/305)

Command Language Interface Manual (CLIM)

2	2	2	0.073	DIM, DIM
2	2	3	-0.010	DIM, ln(DIM/305)
2	3	3	1.068	ln(DIM/305), ln(DIM/305)
3	1	1	100.000	residual variance

The pedigree and data files for the [random regression model](#) example:

pedigree file RRM.ped				data file RRM.dat					
animal ₁	sire ₂	dam ₃	block ₄	HTD ₁	animal 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

CLIM code for the [random regression model](#) is

```

DATAFILE  RRM.dat
INTEGER   HTD animal blk-var # Integer column names
REAL      DIM ln305DIM &     # Covariables
           milk_yd           # Milk yield

PEDFILE   RRM.ped # Pedigree file
PEDIGREE  G am    # animal model

PARFILE   RRM.var

MODEL
  milk_yd = Lact_curve(DIM ln305DIM) HTD G(1 DIM ln305DIM| animal)

```

Note that the component name `Lact_curve` is informative for the user only, not MiX99, because it is used for fixed regression effects and there is no [nesting](#). Name `Lact_curve` will remind user that these regression effects model the [lactation curve](#).

The fixed effect solutions (`Solfix`) are

Command Language Interface Manual (CLIM)

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

The Lact_curve regression effect solutions ([Solreg](#)) are

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

The random regression effect estimates ([Solani](#)) for each animal are

Bull	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

5.2.4 Example: Covariable table and random regression model

MiX99 allows use of [covariable table files](#). This means using an index in the [data file](#) that indicates a row in a covariable table file having regression coefficients. The table numbers in model lines refer to column numbers in this covariable table file instead of the [data file](#). Note that the first column (index) is not counted as a column in the covariable table file. Use of [covariable table file](#) can reduce size of the [data file](#). Covariable table files are often small because the number of possible indices is small. For example, days in milk in a [data file](#) can be from 1 to 350 days, and, so, only 350 different regression coefficients are needed for one regression effect in the [covariable table file](#).

CLIM commands needed are [TABLEFILE](#) and [TABLEINDEX](#). [TABLEFILE](#) indicates name of the [covariable table file](#). [TABLEINDEX](#) has the [integer number column](#) name having the index in the [data file](#). Only one table file and index is allowed. The indices must be positive numbers greater than zero, and be consecutively numbered. For example, the table file can have indices 1, 2, 3, 4, but having only 1, 2, 4 is unacceptable.

The [covariable table file](#) regression coefficients need to be referenced in the model differently from the other regression coefficients. The covariable table regression coefficients are referenced by letter *t* and a column number: t_n where *n* is column number in the [covariable table file](#). For example, t_3 means column 3 in the [covariable table file](#).

We consider again the single trait random regression animal model example by [Schaffer and Dekkers \(1994\)](#) (Chapter 5.2.3). We use the covariable table approach to reduce number of columns in the [data file](#). The idea is that the [data file](#) has an index to indicate which regression coefficients are used. A natural indicator in dairy cattle

[test-day models](#) is days in milk. For this example, an artificial index was used instead. This was due to MiX99 requiring that the index is consecutively numbered. Thus, the covariable index table file has to have index numbers from some number, say 1, consecutively to a high number, say 305. This would give 305 lines.

Use of [covariable table file](#) leads to changes in the CLIM [instruction file](#), and [data file](#). In addition, there has to be a [covariable table file](#).

The [data file](#) ([RRM_table.dat](#)) is

HTD ₁	animal 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 [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

```

```

DATAFILE RRM_table.dat
INTEGER HTD animal blk-var index
REAL milk_yd

TABLEFILE RRM_table.cov
TABLEINDEX index

PEDFILE RRM.ped
PEDIGREE G am

PARFILE RRM.var

MODEL SCALE
milk_yd = Lact_curve(t1 t2) HTD G(1 t1 t2| animal)

```

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

Trt	Reg-No	Solution	Trait	Covariable
1	1	-.49839E-01	milk_yd	T1
1	2	5.2910	milk_yd	T2

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

5.2.5 Example: Heterogeneous residual variance in test day model

Consider the [random regression model](#) example by [Schaeffer and Dekkers \(1994\)](#) (Chapter 5.2.4). However, assume now that residual variance is different according to the block. There are four blocks. Let residual variance be 100, 110, 105, and 90 in blocks 1, 2, 3, and 4, respectively.

Important commands in CLIM for 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, 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

```

The first column is the residual variance block number. The second and third column refer to matrix position, here scalar. Thus, in our example, matrix position is always (1,1). The last column has the variances.

CLIM code for analysis is

```
DATAFILE RRM_table.dat
INTEGER HTD animal blk-var index
REAL milk_yd

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 blk-var # index for residual variance

MODEL SCALE
milk_yd = Lact_curve(t1 t2) HTD G(1 t1 t2| animal)
```

Fixed effect solutions (**Solfix**) are

Fact.	Trt	Level	N-Obs	Solution	Factor	Trait
1	1	1	3	19.954	HTD	milk_yd
1	1	2	4	20.369	HTD	milk_yd
1	1	3	4	20.589	HTD	milk_yd
1	1	4	4	19.689	HTD	milk_yd
1	1	5	4	18.484	HTD	milk_yd
1	1	6	4	17.755	HTD	milk_yd

Fixed regression effect solutions (**Solreg**) are

Trt	Reg-No	Solution	Trait	Covariable
1	1	-.49291E-01	milk_yd	T1
1	2	5.2951	milk_yd	T2

Random regression effect solutions (**Solani**) are

Bull	N-Desc	N-Obs	Intercept	DIM	ln305DIM
1	1	3	-.43705	0.36362E-01	-.36590E-01
2	1	4	0.26551	-.66395E-01	0.32095E-01
3	0	6	-.69442	0.64368E-02	-.45801E-01
4	0	5	1.0687	-.52282E-02	0.74740E-01
5	0	3	-.15415	0.72484E-02	-.14431E-01
6	0	2	-.48062	0.17272E-01	-.37930E-01
7	2	0	-.97642E-01	0.13161E-01	-.10194E-01
8	2	0	0.44474	-.23875E-01	0.35618E-01
9	2	0	-.60770	0.34699E-01	-.46705E-01
10	2	0	0.44474	-.23875E-01	0.35618E-01
11	2	0	-.18370	-.11773E-03	-.14523E-01

5.3 Maternal effect models

maternal effect model

The [random regression effect models](#) allow quite flexible model description. However, random regression effects have the same [covariance structure](#), e.g., [numerator relationship matrix](#). Random maternal and paternal effects with correlated animal effect have a different structure. [Nesting](#) within a component is now by different class variable. Thus, we have multiple correlated factors within genetic effect.

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

A simple model with a fixed herd effect, random maternal and animal effects is

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

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

Note that the maternal genetic model is different from model

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

Here `dam` is a common dam environment effect for all of its progeny. There is no [relationship matrix](#) involved in this `dam` effect.

5.3.1 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](#) $\sigma_p^2 = 1$, residual variance $\sigma_e^2 = 7$, and [genetic covariance matrix](#)

$$G_0 = \begin{bmatrix} 2.0 & 1.0 \\ 1.0 & 3.0 \end{bmatrix}$$

The parameter file `mat.var` is

Random effect ₁	Row ₂	Column ₃	Covariance ₄	Comment
1	1	1	1.0	maternal permanent env.
2	1	1	2.0	maternal genetic
2	1	2	1.0	cov(maternal, animal)
2	2	2	3.0	animal genetic
3	1	1	7.0	residual variance

We use the previously introduced data (Chapter 3). The [pedigree file](#) (Chapter 3.2.1) 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`):

animal 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

CLIM code is

```
DATAFILE  example_mat.dat

INTEGER   animal dam herd_year ones
REAL      tr1 tr2

PEDFILE   AM.ped
PEDIGREE  G  am
RANDOM     PE

PARFILE    mat.var

MODEL SCALE
  tr1 = herd_year PE(dam) G(dam animal)
```

The herd-year solutions ([Solfix](#)) are

Fact.	Trt	Level	N-Obs	Solution	Factor	Trait
1	1	1	2	99.338	herd_yea	tr1
1	1	2	3	121.71	herd_yea	tr1

The maternal permanent environment solutions ([Solr01](#)) are

2	1	-1.0095
4	1	1.0095
6	2	0.24829
8	1	-.24831

The maternal and animal genetic effect solutions ([Solani](#)) are

Animal	id	N-Desc	N-Obs	Maternal	Animal
	1	2	0	1.0095	0.50476
	2	2	1	-1.0095	-.50475
	3	2	0	0.25237	0.75717
	4	2	1	0.75714	-.25240
	5	3	0	0.38058	0.19030
	6	3	2	1.1337	1.8287
	7	1	0	0.69506	0.82321
	8	1	1	0.22384	0.30441E-01
	9	0	0	1.1042	2.0507
	10	0	0	0.33530	0.54334E-01

Note that 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 animal)` with the maternal effect first, and direct animal effect second. Changing order of the effects changes order in the solution file as well.

6 Multiple trait models

[Multiple traits](#) are defined by separate model lines. Traits are numbered in MiX99. Trait number equals model line number. Thus, the first model line trait is trait number 1, the second is number 2 etc. This has to be kept in mind when making [covariance matrix](#) in `PARFILE`. Sometimes numbering of variance components can be difficult. In particular, when different traits have some (random regression or maternal) [effects missing](#) in another trait. The missing components are signaled by a dash (–) sign in the model lines. In the following, we will consider this in animal models, but sire models work similarly.

6.1 All traits have the same effects

Simple [multiple trait model](#) has several traits that are equal in the sense of having the same effects. For example, both traits have herd-year effect and animal 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](#).

6.1.1 Example: Simple multiple trait model

Consider the multiple trait model data presented in Chapter [3.1.1](#). First consider a simple model where both traits have the same effects:

$$\begin{aligned} tr_1 &= herd \times year_1 + a_1 + e_1 \\ tr_2 &= herd \times year_2 + a_2 + e_2 \end{aligned}$$

where the subscripts 1 and 2 refer to traits 1 and 2. The variance components file (name `mt.var`) was already presented in Chapter [3.3.1](#). CLIM [instruction file](#) is:

```
DATAFILE  example.dat

INTEGER   animal sire herd_year ones
REAL      tr1 tr2

PEDFILE    AM.ped
PEDIGREE   animal am
DATASORT   PEDIGREECODE=animal

PARFILE    mt.var

MODEL SCALE
  tr1 = herd_year animal
  tr2 = herd_year animal
```

The fixed effect solutions ([Solfix](#)) are

Fact.	Trt	Level	N-Obs	Solution	Factor	Trait
1	1	1	2	99.760	herd_yea	tr1
1	2	1	2	194.87	herd_yea	tr2
1	1	2	3	122.93	herd_yea	tr1
1	2	2	3	129.73	herd_yea	tr2

The breeding value estimates ([Solani](#)) are

Bull	N-Desc	N-Obs	tr1	tr2
1	2	0	0.44163E-04	0.10214E-04
2	2	0	0.44163E-04	0.10214E-04
3	2	0	0.47974	0.26924
4	2	1	-.47968	-.26923
5	3	0	-.31037E-04	-.40111E-04
6	3	1	0.95937	0.53844
7	1	0	0.23052	0.78139E-01
8	1	1	0.77389	0.86997
9	0	1	0.43458	-.14051
10	0	1	0.40098E-02	0.91964E-01

6.2 Traits have different effects

Multiple trait models having different effects can be easily handled by MiX99. However, there are some details that need to be remembered when using CLIM. The default

CLIM model line works like the [MiX99 directive file](#): an effects missing in a trait has to be indicated by a dash ('-') sign. Consequently, models are column restricted, i.e., each effect in the model has a column which is present (given model name) or missing (given dash sign) for each trait. Thus, it is important to give effects in a specific order. The [beta testing version of CLIM](#) lifts this restriction, and model effects can be given in any order without missing dash sign indicator. However, this may lead to models that are interpreted incorrectly. Thus, it is very important to check that the model generated by CLIM is correct in the [MiX99_DIR.DIR](#) file.

6.2.1 Example: Multiple trait model with different effects by trait

Consider the multiple trait model data as in Chapter [6.1.1](#) but use model

$$\begin{aligned} tr_1 &= herd \times year + a_1 + e_1 \\ tr_2 &= \mu + a_2 + e_2 \end{aligned}$$

The variance components file is the same as before. CLIM [instruction file](#) is the same except for the model lines:

```
MODEL SCALE
tr1 = -      herd_year animal
tr2 = ones -          animal
```

The fixed effect solutions ([Solfix](#)) are

Fact.	Trt	Level	N-Obs	Solution	Factor	Trait
1	2	1	5	160.28	ones	tr2
2	1	1	2	88.759	herd_yea	tr1
2	1	2	3	137.73	herd_yea	tr1

Breeding values estimates ([Solani](#)) are

Bull	N-Desc	N-Obs	tr1	tr2
1	2	0	-.43878E-05	0.21829E-05
2	2	0	-.43878E-05	0.21829E-05
3	2	0	-2.2842	-2.5112
4	2	1	2.2842	2.5112
5	3	0	-5.8968	-5.8969
6	3	1	1.3284	0.87453
7	1	0	-4.2748	-4.4635
8	1	1	-7.6804	-7.6190
9	0	1	-6.6912	-7.2448
10	0	1	-9.9586	-9.9459

6.2.2 Example: Different effects by trait using CLIM beta features

Order of effects is unimportant in the CLIM [beta version](#). The model lines in example above in Chapter [6.2.1](#) can be given differently using the CLIM [beta version](#) (e.g., giving [mix99i](#) -b model.clm). A natural way of giving would be

```
MODEL SCALE
tr1 =      herd_year animal
tr2 = ones          animal
```

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

```
MODEL SCALE
tr1 = animal herd_year
tr2 = ones animal
```

However, this is more difficult to read.

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	herd_yea	tr1
1	1	2	3	137.73	herd_yea	tr1
2	2	1	5	160.28	ones	tr2

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

6.3 Multiple trait random regression model

Multiple trait [random regression models](#) are an extension to the single trait models. Each trait in a multi-trait model has its own model line. It is important to get the numbering of random regression effects correct. Numbering is column-wise from first regression effect of the first trait to the second regression effect of the second trait etc. Thus, in multi-trait models variances for regression effects of the same trait are often not consecutively numbered.

Consider quadratic [random regression function](#) of an animal for two traits:

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

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

$$\begin{aligned} f(\mathbf{a}, x_1) &= a_{1,1} + x_1 \cdot a_{1,2} + x_1^2 \cdot a_{1,3} \\ f(\mathbf{a}, x_2) &= a_{2,1} + x_2 \cdot a_{2,2} + x_2^2 \cdot a_{2,3} \end{aligned}$$

where the first subscript in \mathbf{a} is trait number, and the second is random regression effect number. The random regression effects are numbered

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

where the numbers in parenthesis mean effect number.

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

6.3.1 Example: Multiple trait random regression model

Consider the single trait random regression [test-day model](#) data presented in Chapter 5.2.3. We expand the data by adding 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 trait the [genetic covariance matrix](#) stays the same, and between the traits the correlation

is 95 %. Remember that numbering of regression effects is column-wise. Thus, the **genetic covariance matrix** is

$$\mathbf{G} = \begin{bmatrix} 44.791 & 42.55145 & -0.133 & -0.12635 & 0.351 & 0.33345 \\ 42.55145 & 44.791 & -0.12635 & -0.133 & 0.33345 & 0.351 \\ -0.133 & -0.12635 & 0.073 & 0.06935 & -0.010 & -0.0095 \\ -0.12635 & -0.133 & 0.06935 & 0.073 & -0.0095 & -0.010 \\ 0.351 & 0.33345 & -0.010 & -0.0095 & 1.068 & 1.0146 \\ 0.33345 & 0.351 & -0.0095 & -0.010 & 1.0146 & 1.068 \end{bmatrix} \quad (1)$$

and let the **residual covariance matrix** be

$$\mathbf{R} = \begin{bmatrix} 100.0 & 50.0 \\ 50.0 & 100.0 \end{bmatrix} \quad (2)$$

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
```

CLIM code is

```
DATAFILE RRM_mt.dat
INTEGER HTD animal blk-var
REAL DIM ln305DIM milk_1 milk_2

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

PARFILE RRM_mt.var

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

Fixed effects solutions (**Solfix**) are

Command Language Interface Manual (CLIM)

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	-.50423E-01	milk_1	DIM
1	2	5.2367	milk_1	ln305DIM
2	1	-.50423E-01	milk_2	DIM
2	2	5.2367	milk_2	ln305DIM

Random regression breeding values (**Solani**) are

id		(1)	(2)	DIM(1)	DIM(2)	ln305DIM(1)	
1	1 3	-.54362	-.54362	0.37788E-01	0.37788E-01	-.43693E-01	...
2	1 4	0.34392	0.34392	-.67204E-01	-.67204E-01	0.37731E-01	...
3	0 6	-.85129	-.85129	0.75673E-02	0.75673E-02	-.54810E-01	...
4	0 5	1.2934	1.2934	-.65255E-02	-.65255E-02	0.89009E-01	...
5	0 3	-.18533	-.18533	0.70232E-02	0.70233E-02	-.17046E-01	...
6	0 2	-.57857	-.57857	0.17720E-01	0.17720E-01	-.44841E-01	...
7	2 0	-.12228	-.12228	0.13506E-01	0.13506E-01	-.12222E-01	...
8	2 0	0.54578	0.54579	-.24593E-01	-.24593E-01	0.42246E-01	...
9	2 0	-.75284	-.75284	0.36109E-01	0.36109E-01	-.55656E-01	...
10	2 0	0.54578	0.54579	-.24593E-01	-.24593E-01	0.42246E-01	...
11	2 0	-.21549	-.21549	-.46539E-03	-.46540E-03	-.17023E-01	...

The last column has been omitted due to page width restriction. It is equal to the second last column. Numbers in the parenthesis refer to the trait number of effect.

6.4 Combining of trait estimates

Combining of trait estimates means making effects of different traits to be the same. By default, it is assumed that effects in different traits will be different, and get separate estimated values. Combining of trait estimates or shortly combining of traits allows estimating the same solutions for effects in different traits. In practice, combining of traits is used in reduced rank random regression models. However, the concept is illustrated by a [multiple trait model](#) that is equivalent with [repeatability model](#).

Combining of traits is indicated by the '@' sign in the model lines. After the '@' sign combining group name is given. The name can be any allowed name that has not already been used. Combining of traits can be instructed for any effects that have component name. Thus, it is not possible to use [integer number column](#) names when combining traits. The effect names have to be given a component name. For example, G(animal)@fst.

6.4.1 Example: Repeatability model by multiple trait model

Repeatability model is a [multiple trait model](#) where genetic correlation between traits is one, residual variances are equal, and residual covariances are equal. Residual correlations are equal to $\sigma_p^2 / (\sigma_p^2 + \sigma_e^2)$ where σ_p^2 is [permanent environment variance](#) and σ_e^2 is [residual variance](#). We consider the repeatability model example presented in Chapter [5.1.5](#).

An equivalent two-trait animal model is

$$\begin{aligned} tr_1 &= herd \times year + a + e_1 \\ tr_2 &= herd \times year + a + e_2 \end{aligned}$$

where the fixed effect $herd \times year$ and random animal genetic effect a are common to both traits. Genetic variance is as before $\sigma_a^2 = 2$. [Residual covariance matrix](#) is

$$\mathbf{R} = \begin{bmatrix} 7.0 & 2.0 \\ 2.0 & 7.0 \end{bmatrix} \quad (3)$$

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

The variance components file ([mt_repeat.var](#)) is

Random effect ₁	Row ₂	Column ₃	Covariance ₄	Comment
1	1	1	3.0	animal 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 multiple trait format ([mt_repeat.dat](#)):

animal	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

CLIM code

```
DATAFILE example_mt_repeat.dat

INTEGER animal sire hy_1 hy_2 ones
REAL tr1 tr2

PEDFILE AM.ped
PEDIGREE G am

PARFILE mt_repeat.var

MODEL SCALE
  tr1 = hy_1 - G(animal)@1
  tr2 = - hy_2 G(animal)@1
```

Note that the animal genetic effect `animal` needs a name `G` for combining. Also, there has to be a dash (-) to indicate the use of separate [integer columns](#) for the herd-year effects.

Estimated herd-year solutions (**Solfix**) are

Fact.	Trt	Level	N-Obs	Solution	Factor	Trait
1	1	11	2	99.833	hy_1	tr1
1	1	12	3	123.01	hy_1	tr1
2	2	21	2	194.83	hy_2	tr2
2	2	22	3	129.68	hy_2	tr2

Estimated breeding values (**Solani**) are

Bull	N-Desc	N-Obs	tr12
1	2	0	-.34993E-13
2	2	0	-.34993E-13
3	2	0	0.33333
4	2	1	-.33333
5	3	0	-.76001E-13
6	3	1	0.66667
7	1	0	0.97731E-01
8	1	1	0.98778
9	0	1	-.85515E-01
10	0	1	0.71553E-01

The solutions are the same as by the repeatability model in Chapter 5.1.5. However, no solutions for permanent environment are calculated because no **permanent environment effect** was in this two-trait model.

6.4.2 Example: Reduced rank random regression model

Rank reduction can be used to make **covariance matrices** of highly correlated random regression coefficients more independent. Consequently, size of the **covariance matrix** is reduced. Convergence of the **iterative solver** becomes faster for at least two reasons: equations become less correlated, and there are less unknowns to solve.

In a reduced rank model, coefficients from two or more traits multiply the same solutions. We consider the two-trait random regression example in Chapter 6.3.1.

6.4.3 Example: Finnish test-day model

This example is not a complete presentation of the old Finnish **test-day model**. No data is given, and only the first lactation model. This illustrates potential of MiX99 for solving complex test-day models used currently to solve dairy cattle breeding values.

This was Finnish **test-day model** for first lactation milk, protein, and fat yield. The full model had second, and third with later lactations as additional traits. In this subset model, both **permanent environment** and **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 to one.

```

DATAFILE Ter.dat
INTEGER block animal 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 SCALE

```

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

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

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

```

Note that the model lines of the previous example can be shortened using CLIM [macro and range abbreviations](#) (command line option `--usemacros` is currently needed):

```

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 SCALE

```

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

```

NEW

6.5 Multiple trait maternal effects model

Consider three trait [maternal effect model](#) where the last trait does not have a maternal effect. In addition, some fixed effects are different by trait. Note that spaces between the effect names are only to help reading the model lines.

```

DATAFILE Beef.dat
INTEGER BREED HERD id dam sex twin dam_age &
        brth_mth HY_brth HY_200d HY_365d
REAL Brth_w 200d_w 365d_w age_200d age_365d

PEDFILE Beef.ped
PEDIGREE G am+p

PARFILE Beef.var

MODEL SCALE
Brth_w = - twin sex - HY_brth - - G(dam id)
200d_w = age_200d twin sex brth_mth - HY_200d - G(dam id)
365d_w = age_365d twin sex brth_mth - - HY_365d G( - id)

WITHINBLOCKORDER G HY_365d HY_200d HY_brth

```

Currently in [beta testing](#) (option `-b`) allows this model to be written without the dashes. Because effects are ordered by their column number, the resulting [directive file](#) is different from the example above. However, the analysis will be the same although effects are in different order on the model lines. It is still important to have `-` in the third trait `365d_w` within random effect in order to have correct numbering of variance components.

MODEL

```

Brth_w =          twin sex          HY_brth  G(dam id)
200d_w = age_200d twin sex brth_mth HY_200d  G(dam id)
365d_w = age_365d twin sex brth_mth HY_365d  G( - id)

```

7 Genomic data models

There are two commonly employed alternative ways to use genomic data in statistical models for animal breeding. One way has genomic marker effects directly in the model. The other way uses genomic data to build (co)variance structure such as genomic relationship matrix, and use it as a (co)variance structure to breeding values. Both kinds of models are supported in MiX99 using CLIM. The [genomic marker effect](#) model will be called **SNP-BLUP** model. Models using [genomic relationship matrix](#) include **G-BLUP** and the [single-step method](#).

7.1 SNP-BLUP or genomic effect model

Statistical models for genomic selection have often several thousand [SNP markers](#). In the SNP-BLUP model each marker is a regression effect. It would be tedious to write a model line which has several thousand regression coefficients. Instead, CLIM allows use of regression coefficient matrices in files by commands [REGMATRIX](#) and [REGFILE](#). MiX99 allows use of several matrices but CLIM allows currently up to five [regression coefficient matrix](#) files by [REGMATRIX](#).

The regression coefficients defined by the regression coefficient matrix can be either fixed (command [REGMATRIX FIXED](#) or random (command [REGMATRIX RANDOM](#)). If they are random, the marker effects in MiX99 can have either common variance or, alternatively, each marker can have its individual own variance.

Relevant commands for [regression coefficient matrices](#) are [REGFILE](#) for the name of the file having the regression coefficients, [REGMATRIX](#) for defining type of the matrix as well as coefficient columns, and [REGPARFILE](#) for variance component(s). For syntax and better explanation see Chapter [9.2](#).

All coefficients on a line in a [regression coefficient matrix](#) file belong to only one animal. Each line corresponds to a line in the [data file](#) defined by [DATAFILE](#). It is important to have the lines in the [regression coefficient file](#) in the same order as corresponding observations in the [data file](#). It is possible to instruct MiX99 to check that the files have lines in the same order by animal id code. Then, the genetic evaluation is not as error prone as when no animal id code has been given.

Animal id code can be on different columns in the [regression coefficient file](#) and the [data file](#). Animal id code in the [regression coefficient file](#) is instructed by option [ID](#) in command [REGMATRIX RANDOM ID=value](#) where *value* is column number in the [regression coefficient file](#). In the [data file](#) the command for animal id code is [DATASORT PEDIGREECODE=icol](#) where *icol* is integer number column name (or number) in the [data file](#). If either one information is missing, order of lines cannot be checked, and order is assumed to be correct. Please check summary of commands for syntax and better explanation of these commands (Chapters [9.2.3](#) and [9.2.18](#)).

Use of the [ID](#) option in [REGMATRIX](#) allows giving genotype files that have more genotyped animals than animals with observation. This is convenient when the genotype

data set has genotyped candidate animals without observation. Or, the same marker data is used in analysis of several traits where some animals do not have observations for some traits.

7.1.1 Example: simple genomic marker BLUP

Model is

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

where the β s are regression coefficients, the g s are random additive marker or allele effects, and e is random residual. There are six markers, numbered 1, 2, ..., 6. The markers have bi-allelic loci. For each marker, the genotypes are coded as 0 for homozygous first allele, 1 for heterozygote, and 2 for homozygous second allele. Marker effect is additive allele effect of the second allele. Thus, two times the marker effect solution gives difference between the homozygotes for the first and the second allele.

The variance components are: common **SNP marker** genetic variance $\sigma_g^2 = \frac{1}{6} = 0.166666666$, and residual variance $\sigma_e^2 = 1$. The variances are in two separate files. The file for the marker genotype variance (**gs_gen.par**) is

Matrix number ₁	Row ₂	Column ₃	Covariance ₄	
1	1	1	0.166666666	marker variance

The other parameter file (**gs_res.par**) has the residual variance:

Random effect ₁	Row ₂	Column ₃	Covariance ₄	
1	1	1	1.0	residual variance

The data has been divided into two separate files. The **data file** (command **DATAFILE**) is the standard MiX99 observation file having columns for animal id code, fixed effect (general mean), and observation. The other file (command **REGFILE**) has the genomic information, i.e., regression coefficients defined by genotype. As mentioned, records in these files must be in the same order. Thus, the first record in **data file** and in the **regression coefficient file** are from the same animal. It is assumed that all genotypes are known, and have been coded by user.

The **data files** are

data file gs_obs.dat			genotype file gs_geno.dat						
animal 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

CLIM code is

```
DATAFILE  gs_obs.dat

INTEGER   animal mean
REAL      y
```

```
MISSING -99999.
DATASORT PEDIGREECODE=animal

PARFILE    gs_res.par # residual variance

REGMATRIX  RANDOM SNP ID=1 FIRST=2 LAST=7
REGFILE    gs_gen0.dat
REGPARFILE gs_gen.par # snp variance

PRECON    d

MODEL
  y = mean
```

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 calculated as

$$\hat{\mathbf{a}} = \mathbf{1}\hat{\mu} + \mathbf{Z}\hat{\mathbf{g}} \quad (4)$$

where $\mathbf{1}$ is vector of ones, $\hat{\mu}$ is estimate of the general mean, \mathbf{Z} is the [regression coefficient matrix](#), and $\hat{\mathbf{g}}$ is 6×1 vector of estimated marker effects. MiX99 (specifically `mix99s`) calculates [genomic breeding values](#) ($\hat{\mathbf{a}}$) for this model when option `-p` is given. Thus, `mix99s -p -s` writes file `yHat.data0` where each line has a breeding value for an animal. The values are in the same order as observations in the [data file](#). Unix command `paste gs_obs.dat yHat.data0` combines animal id numbers in the data file with the estimated [genomic breeding values](#). Result is

1	1	5	6.0380783
2	1	6	7.2604141
3	1	10	10.660874
4	1	15	12.040634

Note that estimated [genomic breeding value](#) is not calculated to animals without observation using option `-p`.

Assuming there is an additional set of fixed regression effects in a file, these can be included as well. All commands to the same `REGMATRIX` need to be on successive rows. Notes:

- 1) when the number of regression effects is low, it may be easier and clearer to have them in the data file and include them on the model line than use the `REGMATRIX` command
- 2) it is possible for several `REGMATRIX` commands to refer to the same `REGFILE` although some columns are used for fixed and some random regression effects.

An example CLIM code having two sets of `REGMATRIX` blocks is

```

DATAFILE    gs_obs.dat

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

PARFILE     gs_res.par # residual variance

REGMATRIX   RANDOM SNP ID=1 FIRST=2 LAST=7
REGFILE     gs_gen0.dat
REGPARFILE  gs_gen.par # snp variance

REGMATRIX   FIXED regcov ID=1 FIRST=2 LAST=3
REGFILE     regcov_gen0.dat

PRECON      d

MODEL
  y = mean

```

The second **REGFILE** adds covariates of two regression effects from the **regcov_gen0.dat** file. The second and third column have the covariates, but the first column has the ID code of animal.

7.1.2 Enhanced formatting of SNP marker information

SNP marker information is typically coded with values 0, 1, or 2, and optional **missing marker value** with some other one digit integer such as 3 or 9. The coding can also include separate centering and scaling information so that the possibly very large marker information can be kept in integer form to save disk space and memory.

REGMATRIX command has optional parameter **IMPUTE** for specifying **missing marker value**. For example:

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

instructs to replace, or **impute**, all marker values 3 in the regression coefficient matrix by averages of the (non-missing) marker values.

The marker value ($Z_{i,m}^{012}$) can be optionally centered and scaled by subtracting a center value (μ_m) from the marker value and multiplying this by a scaling value (s_m):

$$Z_{i,m} = (Z_{i,m}^{012} - \mu_m) * s_m$$

The centering can be specified with optional **CENTER** parameter of the **REGMATRIX** command. The centering value μ_m can be either average of the markers, a given constant for all markers, or separate value for each marker stored in a file. For example:

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

centers the markers around the averages,

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

uses “-1,0,1 coding” instead of the “0,1,2 coding”, i.e. centers around one (1), and

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

reads six ($6 = 7 - 2 + 1$) separate centering values from file **mu.dat**.

Similarly, optional scaling can be specified with **SCALE** parameter of the **REGMATRIX** command. The scaling value s_m can be either a given constant for all markers or separate value for each marker stored in a file. For example:

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

scales all markers with a half (0.5),

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

scales all markers with $\frac{1}{\sqrt{2 \sum pq}}$, and

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

uses six separate scaling values stored in file **s.dat**.

By default, the general regression coefficient matrix values are **real valued numbers**. The SNP marker information is typically coded with **integer values** 0, 1, or 2, and optional **missing marker value**. MiX99 can be made to check the SNP marker integer values. The file format of the **REGFILE** file can be specified with optional **FORMAT** parameter of the **REGMATRIX** command. Default file format is “n” (or “normal”) and the SNP marker information can be specified with format “m” (or “markers”):

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

By default, the SNP marker values are assumed to be separated by spaces in the **REGFILE** file. The marker values (0, 1, and 2) are only one character wide. Thus, the spaces separating the markers effectively double the file size. The spaces between the marker values can be removed by specifying file format “s” (or “squeezed”). For example, file **gs_genos_nospaces.dat** could contain marker values without spaces as:

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

and the file format can be specified with:

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

```
REGFILE gs_genos_nospaces.dat
```

The SNP marker values can also be given using PLINK binary **.bed** file format:

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

```
REGFILE gs_genos
```

Currently, only the individual-major mode (PLINK2 export format **ind-major-bed**) is supported. Corresponding **.bim** and **.fam** files need to be available. The file name must be given without the extension (**.bed**).

7.2 Example: simple G-BLUP

Consider the same data as in the previous example. The **SNP-BLUP** model using matrix notation is

$$\mathbf{y} = \mathbf{1}_4\mu + \mathbf{Z}\mathbf{g} + \mathbf{e}$$

NEW

where \mathbf{y} is 4×1 vector of observations, $\mathbf{1}_4$ is 4×1 vector of ones, \mathbf{Z} is 4×6 matrix of **SNP marker** genotypes, \mathbf{g} is 6×1 vector of random marker effects, and \mathbf{e} is random residual. An equivalent model or G-BLUP model solves breeding value vector $\mathbf{u} = \mathbf{Z}\mathbf{g}$ without need to solve the marker effects \mathbf{g} . The model is

$$\mathbf{y} = \mathbf{1}_4\mu + \mathbf{Z}_u\mathbf{u} + \mathbf{e}$$

where \mathbf{Z}_u is 4×6 **incidence matrix** linking breeding values \mathbf{u} to the observations. In our case, $\mathbf{Z}_u = [\mathbf{I}_4 \quad \mathbf{0}]$ where $\mathbf{0}$ is 4×2 matrix of zeros. In the **SNP-BLUP** model it was assumed that $\mathbf{g} \sim N(\mathbf{0}, \mathbf{I}_6\sigma_g^2)$. In the **G-BLUP** model, it is assumed that $\mathbf{u} \sim N(\mathbf{0}, \mathbf{G}\sigma_g^2)$ where $\mathbf{G} = \mathbf{Z}\mathbf{Z}'$ is a 6×6 matrix. Note that \mathbf{Z} has marker coefficients of the candidate animals as well.

The \mathbf{Z} matrix is

$$\mathbf{Z} = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 2 & 2 & 2 & 1 \\ 0 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 2 & 1 & 1 & 1 \\ 0 & 0 & 1 & 2 & 2 & 2 \end{bmatrix}$$

and the **covariance matrix** is

$$\mathbf{G} = \mathbf{Z}\mathbf{Z}' = \begin{bmatrix} 5 & 3 & 2 & 1 & 0 & 0 \\ 3 & 3 & 3 & 3 & 1 & 2 \\ 2 & 3 & 14 & 12 & 9 & 12 \\ 1 & 3 & 12 & 14 & 8 & 13 \\ 0 & 1 & 9 & 8 & 7 & 8 \\ 0 & 2 & 12 & 13 & 8 & 13 \end{bmatrix}$$

Mixed model equations need inverse of **covariance matrix** of a random effect. In our case, \mathbf{G}^{-1} is needed. MiX99 will not compute it: user has to calculate it. In our case the inverse is

$$\mathbf{G}^{-1} = \begin{bmatrix} 0.7500 & -0.5000 & -0.5000 & -0.2500 & 0.3333 & 0.5833 \\ -0.5000 & 2.0000 & -1.0000 & -1.5000 & 1.0000 & 1.5000 \\ -0.5000 & -1.0000 & 2.0000 & 1.5000 & -1.6666 & -2.1666 \\ -0.2500 & -1.5000 & 1.5000 & 2.7500 & -1.3333 & -3.0833 \\ 0.3333 & 1.0000 & -1.6666 & -1.3333 & 1.8888 & 1.5555 \\ 0.5833 & 1.5000 & -2.1666 & -3.0833 & 1.5555 & 3.9722 \end{bmatrix}$$

The inverse matrix is given to MiX99 in a file stored in either co-ordinate (Yale) sparse matrix format or lower triangle dense format.

The default inverse matrix format is the co-ordinate sparse matrix format, also called the Yale format. In this format, each element in the lower triangle of the matrix is given with its element position. In our case, the matrix in file `iG_raw.dat` is

```
1 1 0.75
2 1 -0.5
2 2 2
3 1 -0.5
3 2 -0.9999999999999999
3 3 2
4 1 -0.25
4 2 -1.5
```

```

4 3 1.5
4 4 2.75
5 1 0.3333333333333333
5 2 0.9999999999999999
5 3 -1.6666666666666667
5 4 -1.3333333333333333
5 5 1.8888888888888889
6 1 0.5833333333333333
6 2 1.5
6 3 -2.1666666666666667
6 4 -3.0833333333333333
6 5 1.5555555555555555
6 6 3.9722222222222222

```

Note that the element positions are the animal id codes. In our case they are from one to six. In practice, the numbers need not be consecutive or in increasing order.

The inverse co-variance file `iG_raw.dat` is given by command `PEDFILE` as an inverse co-variance file to the breeding values. An important requirement is that all elements of the given matrix are from the lower triangle only. Option '`MIXED`' can be used to read file that has both lower and upper triangle elements of a symmetric matrix, e.g.,

```
PEDFILE MIXED iG_raw.dat
```

The mixed option should be used with caution because MiX99 will not check if a matrix element appears as an upper and lower triangle element.

An altogether different matrix format is lower triangle dense matrix format. Command option '`LOWER`' indicates it:

```
PEDFILE LOWER iGL_raw.dat
```

In the lower triangle dense matrix format, all lower triangle elements of the matrix are in the file. There are two header rows having the size of the matrix and animal id codes. The previous matrix in a lower triangle dense format is

```

6 0
1 2 3 4 5 6
0.75
-0.5 2
-0.5 -1 2
-0.25 -1.5 1.5 2.75
0.33 1 -1.67 -1.33 1.89
0.58 1.5 -2.17 -3.08 1.55 3.97

```

where the matrix values have been limited to two decimals for output reasons. Note that the order of id codes is irrelevant to MiX99, i.e., they do not have to be in increasing order as given here. However, the order of id codes on the second row gives order of rows below that. In other words, for MiX99 the previous matrix can be given as

```

6 0
3 1 2 4 5 6
2
-0.5 0.75
-1 -0.5 2
-0.25 -1.5 1.5 2.75
0.33 1 -1.67 -1.33 1.89
0.58 1.5 -2.17 -3.08 1.55 3.97

```

Note that the first row has two number: 6 and 0. The second number (0) can be used

to inform number of core animals when APY matrix is given. For example, core of 2 would mean matrix:

```
6 2
1 2 3 4 5 6
0.75
-0.5 2
-0.5 -1 2
-0.25 -1.5 2.75
0.33 1 1.89
0.58 1.5 3.97
```

where the same values have been taken as above for presentation purposes. In practice, APY inverse matrix would have different values.

The variance components file (`gs.par`) is

Matrix number ₁	Row ₂	Column ₃	Covariance ₁	
1	1	1	0.166666666	marker variance
2	1	1	1.0	residual variance

Note that here $(Z_u Z_u')^{-1}$ was used. This allowed using marker variance directly. Typically, [genomic relationship matrix](#) is build using some of the methods by VanRaden.

CLIM code for **G-BLUP** is

```
DATAFILE gs_obs.dat

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

PARFILE gs.par # parameters

PEDFILE iG_raw.dat
PEDIGREE animal FILE

PRECON d d

MODEL
y = mean animal
```

Note how existence of the co-variance structure of animal genetic effect is indicated by the `PEDIGREE` command with option `FILE`.

The fixed general mean solution (`Solfix`) is

Fact.	Trt	Level	N-Obs	Solution	Factor Trait
1	1	1	4	7.2604	mean y

The breeding value solutions (`Solani`) are

1	1	1	-1.2223
2	2	1	-.11178E-05
3	3	1	3.4005
4	4	1	4.7802
5	5	0	2.7444
6	6	0	4.6701

Again giving Unix command `paste gs_obs.dat yHat.data0` combines animal id numbers in the data file to the predicted values which are equal to the estimated

genomic breeding values. Result is

1	1	5	6.0380797
2	1	6	7.2604151
3	1	10	10.660873
4	1	15	12.040632

Note that these values can be calculated by adding the fixed general mean solution 7.2604 to the breeding value solutions in the `Solani` file.

7.2.1 Example: G-BLUP with 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 vector of random additive genetic effects from genomic data, vector \mathbf{a} has random additive polygenic effect from pedigree information and vector \mathbf{e} is random residual. Matrix \mathbf{Z}_u is incidence matrix as in G-BLUP example, and matrix \mathbf{Z}_a is incidence matrix relating observation in \mathbf{y} to proper 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: $\mathbf{u} \sim N(\mathbf{0}, \mathbf{G}\sigma_g^2)$, $\mathbf{a} \sim N(\mathbf{0}, \mathbf{A}\sigma_a^2)$, and $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}\sigma_e^2)$ where \mathbf{G} is genomic co-variance matrix (see previous example), and \mathbf{A} is pedigree based [relationship matrix](#). The following pedigree for the [relationship matrix](#) \mathbf{A} is in file (`gs_poly.ped`):

animal 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 will account inbreeding coefficients in the building of the \mathbf{A}^{-1} in MiX99 by an [inbreeding coefficient file](#), named `gs_poly.inbr`. The file is

animal 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 original animal id code, and the last column **inbreeding coefficient**. When no **inbreeding coefficient file** is given, MiX99 builds A^{-1} assuming all inbreeding coefficients are zero.

The variance components are: common SNP marker variance $\sigma_g^2 = \frac{1}{6}$, polygenic variance $\sigma_a^2 = 1$, and residual variance $\sigma_e^2 = 1$. The variances are in file `gs_poly.par`:

Random effect ₁	Row ₂	Column ₃	Covariance ₁	
1	1	1	0.1666666666	SNP genetic variance

Command Language Interface Manual (CLIM)

2	1	1	1.0	polygenic variance
3	1	1	1.0	residual variance

CLIM code is

```
DATAFILE gs_obs.dat

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

PARFILE gs_poly.par

COVFILE 1 iG_raw.dat

PEDFILE gs_poly.ped
PEDIGREE polygenic am

INBRFILE gs_poly.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

RANDOM genomic polygenic

MODEL
y = mean genomic(animal) polygenic(animal)
```

Note that the **RANDOM** command gives numbering of the random effects (other than residual). The **polygenic effect** must be the last random effect in this statement. Note also that the words **genomic** and **polygenic** are NOT reserved names but names just to identify different random effects.

The fixed general mean solution (**Solfix**) is

Fact.	Trt	Level	N-Obs	Solution	Factor Trait
1	1	1	4	7.9564	mean y

The **genomic breeding value** estimates (**Solr01**) are

1	1	-.99014
2	1	-.93297E-06
3	1	3.0507
4	1	4.0358
5	0	2.4286
6	0	3.9911

The polygenic animal solutions (**Solani**) are

1	0	1	-1.1307
2	1	1	-.79141
3	2	1	-.73879
4	1	1	0.73879
5	2	0	0.12569E-12
6	2	0	0.12569E-12

Again giving Unix command `paste gs_obs.dat yHat.data0` combines animal id numbers in the data file to the predicted values which are equal to the estimated **genomic breeding values**. Result is

```
1 1 5      5.8356075
2 1 6      7.1650143
3 1 10     10.268371
4 1 15     12.731008
```

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 genetic breeding value to its polygenic and marker components which may be sometimes difficult. In particular, the **genomic breeding values** have a **genomic relationship matrix**, and the **polygenic effect** has pedigree based **relationship matrix**. In this example, these matrices are very different but in practice they can be similar.

An alternative equivalent model to this model would be to make a **relationship matrix** that combines the **genomic** and pedigree **relationship matrices**. For example, let $G_A = G + A \frac{\sigma_g^2}{\sigma_g^2 + \sigma_a^2}$. Then, instead of using G^{-1} in the **G-BLUP** model, use G_A^{-1} . Thus, the instructions will be the same as for the **G-BLUP** model with the **PEDFILE** replaced by the new inverse covariance matrix G_A^{-1} where pedigree based **relationship matrix** and genomic information have been combined. This will yield the same estimated complete breeding values. However, in practice, convergence of the iterative method is likely to be better due to not having to estimate two genetic breeding values for every animal.

The equivalent **G-BLUP** model needs G_A^{-1} in a file. In our case, the G_A^{-1} matrix (lower triangle) in co-ordinate sparse matrix format is (file **iGa.dat**):

```
1 1 0.212360759655864
2 1 -0.173168009265463
2 2 0.302462455536769
3 1 -0.0820544909187614
3 2 -0.0175160071940054
3 3 0.284302642414021
4 1 0.0222724767716754
4 2 -0.10603222546145
4 3 0.0440272119204682
4 4 0.265976547255634
5 1 0.0343385639825208
5 2 0.0289958157919094
5 3 -0.169030129597629
5 4 -0.126868150312209
5 5 0.247839050926183
6 1 0.0436056268940923
6 2 0.0386568862841335
6 3 -0.172788965328849
6 4 -0.180933888034668
6 5 0.122875534464136
6 6 0.272614251165761
```

As mentioned, the CLIM-instructions will be the same as for the **G-BLUP** example with only one change: **PEDFILE** **iG_raw.dat** changed to **PEDFILE** **iGa.dat**. The predicted breeding values using this approach are as before

```
1 1 5 5.8356066
2 1 6 7.1650133
3 1 10 10.268371
4 1 15 12.731009
```

7.3 Example: single-step method

Model is

$$\mathbf{y} = \mathbf{1}\mu + \mathbf{Z}_a\mathbf{a}_g + \mathbf{e}$$

where \mathbf{a}_g has random additive genetic effect using pedigree and genomic information, and \mathbf{e} is random residual. Matrix \mathbf{Z}_a is [incidence matrix](#) relating observation in \mathbf{y} to correct breeding value in \mathbf{a} as in previous example.

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 pedigree [relationship matrix](#) blended with genomic data. The \mathbf{H} is not needed by MiX99 but its inverse \mathbf{H}^{-1} . The inverse is

$$\mathbf{H}^{-1} = \mathbf{A}^{-1} + \begin{bmatrix} \mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

where \mathbf{A} is the full pedigree based [relationship matrix](#), \mathbf{A}_{gg} is pedigree based relationship part for the genotyped animals, and \mathbf{G} is [genomic relationship matrix](#). User has to provide matrix $\mathbf{C}_{GA} = \mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$ and the full pedigree used to calculate \mathbf{A} to MiX99. In practice, the single-step method in MiX99 is similar to using animal model with an additional covariance matrix \mathbf{C}_{GA} in lower triangle co-ordinate sparse matrix format (Yale format) by command [IGFILE](#).

Let the full pedigree for [relationship matrix](#) \mathbf{A} be in file [one_step.ped](#):

animal 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

It is the same as in the previous example but with two added animals (numbers 7 and 8). [Inbreeding coefficients](#) are in file [full_one_step.inbr](#):

animal 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 original animal id code, and the last column [inbreeding coefficient](#).

In the [single-step method](#) it is important to use properly made [genomic relationship matrix](#) \mathbf{G} such that its scale is the same as the pedigree based [relationship matrix](#) \mathbf{A} . A common approach is [VanRaden method 1](#). In the example we will use this method, and add 20% of pedigree based [relationship matrix](#) \mathbf{A}_{gg} to assume that the markers explain 80% of genetic variation, and 20% is polygenic variation not explained by the available markers. The same pedigree information will be used as in the previous example. The $\mathbf{C}_{GA} = \mathbf{G}^{-1} - \mathbf{A}_{gg}^{-1}$ matrix is provided in a file. As for the GBLUP model,

the matrix can have either the co-ordinate sparse matrix format (default) or the lower triangle dense matrix format.

Assume the matrix has been stored in the default format, i.e., lower triangle co-ordinate sparse matrix format, in file `iH.dat`:

```
1 1 -0.263096445605446
2 1 -0.659942436053408
2 2 0.244046491072819
3 1 0.870441978527262
3 2 0.998787679967059
3 3 -1.40380798986209
4 1 0.526879611011652
4 2 0.476171586907099
4 3 -0.469768966568885
4 4 -0.0336084181419456
5 1 0.332614778332254
5 2 -0.119449290320484
5 3 0.469298887947602
5 4 1.10970591785976
5 5 -0.351471648428734
6 1 0.595748584829456
6 2 0.150394111665596
6 3 0.405173304876264
6 4 -0.697147095891039
6 5 -0.745222104850557
6 6 1.21570834901989
```

Command `IGFILE` is used to indicate `single-step method`. The command has option `MIXED` to allow giving mixed lower/upper triangle form matrix:

```
IGFILE MIXED iH.dat
```

The lower triangle dense matrix format has option `LOWER` and can be given as:

```
IGFILE LOWER iHL.dat
```

The `MIXED` option works as explained for the `PEDFILE` command earlier in `G-BLUP` example: mixed co-ordinate sparse matrix format file has both lower and upper triangle elements of a symmetric matrix but only one of them is assumed to be present. Thus, in a symmetric matrix it is assumed that only lower or upper triangle element is present in the file. The mixed option should be used with caution because MiX99 will not check if a matrix element given as an upper and lower triangle element.

Variance components are $\sigma_a^2 = 1$ and $\sigma_e^2 = 1$. These are in file `one_step.par`:

Random effect ₁	Row ₂	Column ₃	Covariance ₄	
1	1	1	1.0	polygenic variance
2	1	1	1.0	residual variance

CLIM code is

```
DATAFILE gs_obs.dat

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

IGFILE iH.dat
```

```
PEDFILE one_step.ped
PEDIGREE animal am

INBRFILE full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE one_step.par

MODEL
  y = mean animal
```

The fixed general mean solution (**Solfix**) is

Fact.	Trt	Level	N-Obs	Solution	Factor Trait
1	1	1	4	9.8195	mean y

Estimated breeding values (**Solani**) are

1	0	1	-4.2873
2	2	1	-2.8378
3	3	1	0.85603
4	1	1	2.9909
5	3	0	0.32826
6	3	0	2.6379
7	0	0	1.4831
8	0	0	-.99089

Again giving Unix command `paste gs_obs.dat yHat.data0` combines animal id numbers in the data file to the predicted values which are equal to the estimated **genomic breeding values**. Result is

1	1	5	5.5322323
2	1	6	6.9817309
3	1	10	10.675563
4	1	15	12.810474

It is possible to give the matrices G^{-1} and A_{gg}^{-1} that form the C_{GA} separately in different files by commands **IGFILE** and **IA22FILE**, respectively. In practice, it can be computationally more efficient to let the solver program to do the required computations to make A_{gg}^{-1} computations using pedigree information. Then, no A_{gg}^{-1} need to be computed before calling the preprocessor. This can be done by giving **PEDIGREE** as file name to **IA22FILE**.

A further way for faster computations is to use Fortran unformatted binary file format. When **hginv** program is used to calculate the inverse, having '.bin' suffix to the file name will automatically write a binary file. A lower triangle dense 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. So, an alternative command file to do single-step would have

```
IGFILE LOWER iGL.bin
```

which will be more efficient than the former commands when there are many genotyped animals. First, the G^{-1} matrix in file **iGL.bin** has been stored in the lower triangle dense format. Second, the A_{gg}^{-1} matrix is not precomputed but the computations are done using pedigree information by the solver. Third, binary format is faster to read and write.

Both the co-ordinate format and lower triangle dense format G^{-1} matrices can be computed using the **hginv** program.

NEW

When the ssGBLUP instructions have the PEDIGREE option, i.e.,

```
IA22FILE PEDIGREE
```

for the solver to take into account the A_{gg}^{-1} matrix computations, the pre-processor has not calculated diagonal of the A_{gg}^{-1} matrix contribution to the preconditioner. This may affect convergence at some stage of the PCG iteration. A separate program called `calc_diag_iA22` can be used to calculate the diagonal elements of A_{gg}^{-1} . In practice, MiX99 expects contributions of diagonal of the $-A_{gg}^{-1}$ matrix (note the minus sign). These contributions can then be accounted using command `IHPRECON` by MiX99.

So, combining all these together would give

```
DATAFILE gs_obs.dat

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

IGFILE LOWER iGL.bin
IA22FILE PEDIGREE
IHPRECON minus_diA22.dat

PEDFILE one_step.ped
PEDIGREE animal am

INBRFILE full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE one_step.par

MODEL
  y = mean animal
```

7.3.1 Large number of genotyped animals in single-step GBLUP: ssGTBLUP

Large number of genotyped animals lead to a large genomic relationship matrix. This can lead to too high disk and core memory requirements and too long computing times when some threshold is reached. An alternative approach is to use APY (Algorithm for Proven and Young) which leads to making the G^{-1} matrix sparse in large parts of the matrix by an approximation. In MiX99, use of APY is like using ssGBLUP with the full G^{-1} but instructing the `hginv` program to make an APY based G^{-1} matrix.

Another alternative is ssGTBLUP which is based on Woodbury matrix identity. It assumes that the genomic relationship matrix has form $G = Z_c B Z_c' + E$ where

- 1) Z_c is the centered marker matrix,
- 2) B is the diagonal matrix having weights and scaling for the markers,
- 3) E is an easily inverted regulation matrix.

Use of the Woodbury matrix identity allows expressing the inverse as

$$G^{-1} = E^{-1} - E^{-1} Z_c (Z_c' E^{-1} Z_c + B^{-1})^{-1} Z_c' E^{-1}$$

In practice, two regulation matrices have been in use and implemented in MiX99:

- 1) ssGTeBLUP: $\mathbf{E} = \epsilon \mathbf{I}$ and $\mathbf{B} = \mathbf{D}$ is the marker scaling matrix.
- 2) ssGTABLUP: $\mathbf{E} = w \mathbf{A}_{gg}$, $\mathbf{B} = (1-w) \mathbf{D}$ and w is the residual polygenic proportion.

Often the marker scaling matrix has the form $\mathbf{D} = \mathbf{I}_k^{\frac{1}{k}}$ where $k = 2 \sum_{i=1}^m p_i(1 - p_i)$ and p_i is (base population) allele frequency of marker i .

In both ssGTBLUP models, the inverse matrix can be expressed using a rectangular matrix, denoted \mathbf{T} , which has size number of SNP markers times number of genotyped animals. So, the inverse can be expressed $\mathbf{G}^{-1} = \mathbf{E}^{-1} - \mathbf{T}'\mathbf{T}$.

In MiX99, use of ssGTBLUP is similar to using ssGBLUP. Instead of giving \mathbf{G}^{-1} using command **IGFILE**, other commands are used to define the needed matrices. There are two approaches:

- 1) The \mathbf{T} matrix is given using command **TEFILE** for ssGTeBLUP and **TAFILE** for ssGTABLUP.
- 2) for ssGTABLUP, there is an alternative where the centered marker matrix \mathbf{Z}_c is given using command **ZCFILE** and matrix $\mathbf{C}^{-1} = (\frac{1}{w} \mathbf{Z}_c' \mathbf{A}_{gg}^{-1} \mathbf{Z}_c + \mathbf{B}^{-1})^{-1}$ using command **ICFILE**.

An example command file to do ssGTeBLUP using the first approach is

```
DATAFILE gs_obs.dat

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

TEFILE TE.bin
IA22FILE PEDIGREE

PEDFILE one_step.ped
PEDIGREE animal am

INBRFILE full_one_step.inbr
INBREEDING PEDIGREECODE=1 FINBR=3

PARFILE one_step.par

MODEL
y = mean animal
```

For ssGTABLUP, only one line is changed:

```
TAFILE TE.bin
```

None of the commands include information on the ϵ value for ssGTeBLUP or residual polygenic proportion w for ssGTABLUP. This is because the \mathbf{T} matrix file has to have this information.

An example command file to do ssGTABLUP using the second approach would not have the TAFILE but instead

```
ICFILE iC.bin
ZCFILE Zc.bin
```

The following practicalities need to be accounted when using this model. First, convergence can be improved by adjusting the preconditioner, see below. Second, the solver (mix99s) needs to use the 'MEL' option. Third, the MPI parallel solver mix99p does not yet have this approach.

The ssGTBLUP instructions given above may work but convergence can become poor when the convergence statistic starts to be low. This is because the preconditioner is unable account contributions due to the A_{gg}^{-1} matrix like in the case for ssGBLUP and the PEDIGREE command. For the ssGTBLUP model, the same file as used for ssGBLUP can be used:

```
IHPRECON minus_diA22.dat
```

which has diagonals of $-A_{gg}^{-1}$.

For ssGTBLUP, the diagonal of A_{gg}^{-1} values need to be multiplied by value $(\frac{1}{w} - 1)$ instead of -1 where w is the residual polygenic proportion.

7.3.2 Metafounders

Metafounders can be used instead of genetic groups. In MiX99, there are two practical difference in metafounders compared to genetic groups. First, the inbreeding coefficients need to be computed differently. The inbreeding coefficients for a metafounder model include the covariance matrix information between the metafounders or the so called Γ matrix. These can be computed using [RelaX2](#) program. Second, the inverse of Γ matrix needs to be given to MiX99.

MiX99 requires that the metafounders are positive integer numbers just like any individual. So, the metafounders need to be in the pedigree like any individual with both parents unknown (zero). Otherwise, all the instructions in MiX99 are as before for a model, except that there is an additional command line:

```
IGAMMAFILE MIXED iGamma.dat
```

The Γ^{-1} matrix must be given only in co-ordinate format.

8 Special topics

8.1 Trait groups for single trait analysis

8.1.1 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, multiple trait model is not used due to unknown variance components between the traits. Thus, although the data can be presented as for a multiple trait model, all residual and genetic covariances between the traits are zero. This can be analyzed as a [multiple 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.

NEW

We consider again the multiple trait model example with different effects by trait in Chapter 6.2.1). The model is

$$\begin{aligned} tr1 &= herd \times year + a + e \\ tr2 &= \mu + a + e \end{aligned}$$

However, now the interest is in analyzing these two traits as separate independent evaluations in the same MiX99 solver run. The multiple trait model way would be to do as in Chapter 6.2.1 but with a parameter file where all covariances are zero.

Trait group way is to make the data to be similar to the repeatability data (Chapter 5.1.5) but with an additional column to indicate trait. The data file (example_tr_group.dat) is

animal ₁	sire ₂	herd	year ₃	ones ₄	trait ₅	tr12 ₁
4	1	11	1	1	1	90
4	1	21	1	2	2	200
6	3	11	1	1	1	110
6	3	21	1	2	2	190
8	5	12	1	1	1	120
8	5	22	1	2	2	140
9	5	12	1	1	1	130
9	5	22	1	2	2	120
10	7	12	1	1	1	120
10	7	22	1	2	2	130

The parameter file (mt_single.var) is the same as would be for the multiple 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 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 animal sire herd_year ones trait
REAL tr

TRAITGROUP trait

PEDFILE data/AM.ped
PEDIGREE animal am

PARFILE mt_single.var

MODEL SCALE
tr(1) = - herd_year animal
tr(2) = ones - animal
```


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	herd_yea	tr
2	1	12	3	122.69	herd_yea	tr

Breeding values estimates (**Solani**) are

Bull	N-Desc	N-Obs	tr1	tr2
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 `tr1` single trait evaluation in Chapter 5.1.1. Solutions differ slightly. The reason is that the multi-trait model analysis usually makes more iterations. For example, in the example above, the multiple 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 **multiple 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 is compromised.

8.1.2 Example: MACE or Sire model with weights and trait groups

We consider a multiple trait sire model where yields of daughters in different countries are considered as different traits. This is MACE example as 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:

$$\begin{aligned} E(s) &= \mathbf{0} & \text{Var}(s) &= \mathbf{G}_0 \otimes \mathbf{A} \\ E(e) &= \mathbf{0} & \text{Var}(e) &= \mathbf{R} \end{aligned}$$

These are like the standard multi-trait model assumptions. However, in the **MACE model** the **residual covariance matrix** \mathbf{R} is diagonal, i.e., residual correlations are zero, and varies by sire. **Residual covariance matrix** for sire i is

$$\mathbf{R}_i = \begin{bmatrix} d_{1i}\sigma_{e_1}^2 & 0 \\ 0 & d_{2i}\sigma_{e_2}^2 \end{bmatrix} \quad (5)$$

where d equals one over the number of daughters in a bull's **DYD**, and $\sigma_{e_j}^2$ is residual variance for country j . The d_i values in **residual covariance matrix** can be considered

Table 8.1: The pedigree and data files for the MACE model example.

pedigree file MACE.ped				data file MACE.dat			
Table 2 in Schaeffer (1994)				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	56.0	10.0
2	8	9	-5	2	1	-23.0	20.0
3	10	8	-5	3	1	8.0	50.0
4	10	11	-6	1	2	9.0	100.0
5	2	6	-6	4	2	3.0	40.0
6	-1	-2	-6	5	2	-11.0	20.0
7	-1	-2	-6				
8	-1	-2	-6				
9	-3	-4	-6				
10	-3	-4	-6				
11	-3	-4	-6				

¹maternal grandsire

²maternal grandam

³daughter yield deviation (DYD)

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

$$G_0 = \begin{bmatrix} 100 & 20 \\ 20 & 5 \end{bmatrix}$$

The residual variances are $\sigma_{e_1}^2 = 1000$, and $\sigma_{e_2}^2 = 80$.

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. Residual correlation between trait groups is zero. This definition of trait group matches with the **MACE model** where country is 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

Random effect ₁	Row ₂	Column ₃	Covariance ₁	Variance
1	1	1	100.0	genetic
1	2	1	20.0	genetic
1	2	2	5.0	genetic
2	1	1	1000.0	residual
2	2	2	80.0	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 values estimates (**Solani**) by country are

Bull	N-Desc	N-Obs	Country	
			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

8.2 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 animals (for which deregressed proofs are needed) are known but solutions of their ancestors, [unknown parent groups](#), and 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 current solution for mean estimate, and known proofs.
- 2) calculate new general mean estimate

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 deregression model is that the [unknown parent groups](#) are random.

Many methods can be used in solving non-linear systems. MiX99 has the following methods

- Gauss-Seidel
- Bisection
- Secant
- Broyden

Default method is [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.

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: RAM demand: H=high, M=medium, L=low
# Max. no. iterations, Stopping criterion, Criterion (A/R/M/D)
5000          1.0e-3          D   F 1000
N # RESID: Calculate residuals? (Y/N)
R b # R=deregression
    # b= Broyden method
    # s= Secant method
    # i= bisection
    # n= Gauss-Seidel
N # adjust for HV? No
Y # Solution files? Yes
```

There are some most important difference to the regular breeding value estimation. Deregression is requested (letter 'R') with Broyden method (letter 'b'). In addition, an additional number 1000 on the line for [maximum number of iterations](#) and [convergence criteria](#). The additional number is [maximum number of iterations](#) for the non-linear solving method, which is [Broyden method](#) in this example. Deregression is a non-linear problem. Consequently, two [iterative methods](#) are used: [PCG](#) iteration to solve a linear problem, and [Broyden method](#) for the non-linear problem. In case the [Broyden method](#) does not converge, the [maximum number of iterations](#) is reached. Then, another method (e.g. [secant method](#)) can be used, or the problem needs re-formulation (e.g. new definition of genetic groups, or different random genetic group value).

8.2.1 Example: Single trait deregression

Pedigree earlier used for the animal and sire model examples are 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 maternal grand dam group. This pedigree for animal model (`am_dereg.ped`) is

animal 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 model for `deregression` is very simple: only general mean and the sire effect. It is important to use random `unknown parent groups`. 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 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`):

Command Language Interface Manual (CLIM)

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 CLIM code for animal model is very similar to the sire model case above:

```
DATAFILE   dereg.dat

INTEGER    animal ones
REAL       ebv EDC

PEDFILE    am_dereg.ped
PEDIGREE    animal am+p 1.0

PARFILE    SM.var

MODEL SCALE
  ebv = ones animal ! WEIGHT=EDC
```

General mean solution (**Solfix**) is exactly the same as before. Solutions are the same (**Solani**):

animal	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 animals. As before, only solutions with observations (N-Obs equal to one) are relevant.

8.2.2 Example: Multiple trait deregression

Multiple trait deregression is done the same way as single trait deregression. We illustrate multiple trait deregression by example given in [Schaeffer \(2001\)](#) where detailed explanation can be found. The example is on **multiple 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

Command Language Interface Manual (CLIM)

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 maternal grand dam group. The data (`sch_cntry_A.dat`) has three lactations:

ones ₁	sire ₂	Lactation 1		Lactation 2		Lactation 3	
		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 number of progeny in the third lactation also were observed for first and second lactation. Consequently, number of progeny is the same in all lactations when third lactation is observed. Similarly, the number of progeny observed for second lactations were assumed to also be observed for 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 general mean and the sire effect. It is important to use random **unknown parent groups**. 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

PRECON     b f # Preconditioner: b=block

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 solution for the 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 given above for only those animals that have an observations, i.e.,

Command Language Interface Manual (CLIM)

N-Obs more than zero. All other solutions have been set to zero by MiX99.

9 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 [9.1](#) and [optional commands](#) in Chapter [9.2](#).

Data file commands

DATAFILE	name of data file , 3.1
DATASORT	information on how the data file was sorted, 9.2.3 (optional)
INTEGER	integer number column names in the data file , 9.1.3
MISSING	code for missing observations, 9.2.11 (optional)
REAL	real number column names in the data file , 9.1.7
REGFILE	regression coefficient matrix file, 9.2.17 (optional)
TABLEFILE	name of the separate covariable table file , 9.2.22 (optional)
TABLEINDEX	integer number column name of the covariable table file number in the data file, 9.2.23 (optional)
TRAITGROUP	integer number column name of the trait group number, 9.2.28 (optional)

Pedigree file commands

PEDFILE	name of pedigree file , 3.2
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 genetic groups, then coefficient value as well, 9.1.6 (optional)

Variance component information

PARFILE	MiX99 variance components file, 3.3
RESIDFILE	name of residual (co)variance parameter file, 9.2.20 (optional)
RESIDUAL	integer number column name of the residual (co)variance number, 9.2.21 (optional)
REGPARFILE	name of random regression matrix parameter file, 9.2.19 (optional)

Model commands

MODEL	statistical model
RANDOM	random effects in the model, 9.2.16 (optional if additive genetic and residual effects are the only random effects)
REGMATRIX	regression coefficient matrix information, 9.2.18 (optional)

Solving

NORANSOL	random effects for which no solution files are to be written, 9.2.10 (optional)
PARALLEL	number of processors used in parallel computing, 9.2.14 (optional)
PRECON	preconditioning information, 9.2.15
TITLE	title of the analysis, 9.2.26 (optional)
TMPDIR	directory for the MiX99 temporary files, 9.2.27 (optional)
WITHINBLOCKORDER	ordering of effects in the blocks, 9.2.29 (optional)

Macros and range abbreviations

DEFINE	defines a text macro replacement
--------	----------------------------------

9.1 Required commands

Following are explanation and syntax of all commands.

9.1.1 MODEL

MODEL is considered in Chapters 5, 6, 7 and 8.

9.1.2 DATAFILE

Name of data file. Optional information: file type of text or binary can be given. 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
```

9.1.3 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
```

9.1.4 PARFILE

Name of (co)variance parameter file. Information in the file has the same format as described in the MiX99 manual for mix99i *Technical reference guide for MiX99 pre-processor*.

Syntax:

```
PARFILE <filename>
```

Example. Name of the parameter file is `Beef.par`.

```
PARFILE ../data/Beef.par
```

9.1.5 PEDFILE

Name of **pedigree file**. This **pedigree file** is read by `mix99i`. When type of pedigree is **FILE** in command **PEDIGREE**, the file has inverse of the co-variance matrix for breeding values. Default format for the matrix is lower triangle co-ordinate sparse matrix format (see Ch. 7.2). Option **MIXED** can be used to relax requirement of lower triangle matrix (see Ch. 7.2). However, this means that there can be element (1,2) of matrix, i.e., upper triangle element, but there cannot be corresponding lower triangle element (2,1) in the file. Another optional format is lower triangle dense matrix or **LOWER** (see Ch. 7.2).

Syntax:

```
PEDFILE [LOWER | MIXED] <filename>
```

Example 1. Name of the **pedigree file** is `Beef_phantom.ped`.

```
PEDFILE ../data/Beef_phantom.ped
```

Example 2. Name of co-ordinate format matrix with upper and lower triangle elements:

```
PEDFILE MIXED ../data/iG_matrix.dat
```

Example 3. Name of lower triangle dense matrix:

```
PEDFILE LOWER ../data/iGL_matrix.dat
```

9.1.6 PEDIGREE

Pedigree type and other information. See MiX99 manual *Technical reference guide for MiX99 pre-processor* for the pedigree types. Common pedigree types are **am** for animal model and **sm** for sire model. Autoregressive model has type **ar**. Genetic groups are indicated by suffix **+p**, e.g., **am+p** for animal model with unknown parent or genetic groups. Pedigree has been stored in file given by command **PEDFILE**.

When pedigree type is **FILE**, the inverse of the relationship co-variance structure (e.g. G^{-1} in **G-BLUP**) is in file specified by **PEDFILE** command. See example in Ch. 7.2.

Syntax:

```
PEDIGREE <effect name> <pedigree type> &  
[<optional number for random genetic groups>]
```

Example 1. Pedigree is for an animal model with random genetic groups. Pedigree is associated with model effect name G. The **unknown parent groups** are random with genetic variance coefficient 1.0.

```
PEDIGREE G am+p 1.0
```

Example 2. GBLUP model where the inverse of the genomic relationship matrix G^{-1} is in file `iGL.dat` stored in lower triangle dense matrix format.

```
PEDFILE LOWER iGL.dat  
PEDIGREE G FILE
```

9.1.7 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
```

9.2 Optional commands

The following commands have default values that are used if command is not given.

9.2.1 AR

Define the autoregressive values for each trait in autoregressive model. When this command is given, the [PEDIGREE](#) type must be [ar](#). Default is no autoregressive model.

Syntax:

```
AR <values for each trait>
```

Example. Autoregressive values for 2 traits

```
AR 0.8 0.9
```

9.2.2 COVFILE

Attaches given inverse correlation matrix to specified random effect. The given matrix is in a file in co-ordinate (Yale) sparse matrix format.

Syntax:

```
COVFILE <random effect number> <filename>
```

Example. Inverse correlation matrix for random effect number 1 is in file [covfile.dat](#)

```
COVFILE 1 covfile.dat
```

9.2.3 DATASORT

Names of the [integer number columns](#) for the block sorting variable ([BLOCK](#)), and the animal sorting variable ([PEDIGREECODE](#)). See MiX99 manual [Technical reference guide for MiX99 pre-processor](#) for more explanation. By default, none are needed.

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 [animal](#) in the [data file](#)

```
DATASORT BLOCK=block PEDIGREECODE=animal
```

9.2.4 IA22FILE

Gives file having matrix A_{gg}^{-1} used in the [single-step method](#). In the single-step method, matrix $C_{GA} = G^{-1} - A_{gg}^{-1}$ is needed. The two matrices can be given in a single file using the [IGFILE](#) command, or in two separate files giving G^{-1} to the [IGFILE](#) and A_{gg}^{-1} to the [IA22FILE](#).

This approach of giving two separate filenames for the components of C_{GA} allows separation of the needed computations for the two matrices. In particular, giving file name [PEDIGREE](#) for [IA22FILE](#) indicates MiX99 that the computations for A_{gg}^{-1} are done using pedigree information without need to give this matrix explicitly in a file.

Default format is the co-ordinate (Yale) sparse matrix format. The preprocessor checks that each row is in lower triangle part of the matrix. This check is simple (the second column value (j) must be less or equal to the first column value (i)), and not all programs produce this format. Thus, option [MIXED](#) allows reading the file without the check. Another format allowed is the lower triangle dense format (see Ch. 7.2). which can be prepared using the [hginv](#) program. For this matrix option [LOWER](#) need to be used.

Syntax:

```
IA22FILE [LOWER | MIXED] <filename>
```

If the filename is [PEDIGREE](#) then no file of that name is read but instead the solver will use pedigree information to do the required computations.

Example. Matrix A_{gg}^{-1} is in file [iA22L.dat](#)

```
IA22FILE LOWER iA22L.dat
```

9.2.5 ICFILE

Gives file having the matrix $C^{-1} = (\frac{1}{w}Z_c'A_{gg}^{-1}Z_c + B^{-1})^{-1}$ matrix in $G^{-1} = ((1 - w)Z_cBZ_c' + wA_{gg})^{-1} = \frac{1}{w}A_{gg}^{-1} - \frac{1}{w^2}A_{gg}^{-1}Z_cC^{-1}Z_c'A_{gg}^{-1}$ used by the [single-step method](#) where w is the residual polygenic proportion and B has scaling information. The command requires command [ZCFILE](#) as well. The C^{-1} matrix file can be made by a special preprocessing program such as T48eig_make. The approach requires the A_{gg}^{-1} matrix to be given separately using the command [IA22FILE](#) with option [PEDIGREE](#).

Syntax:

```
ICFILE <filename>
```

Example. Matrix C^{-1} is in file [iC.dat](#)

```
ICFILE iC.dat
```

9.2.6 IGFILE

Gives file having matrix $C_{GA} = G^{-1} - A_{gg}^{-1}$ used by the [single-step method](#). In case [IA22FILE](#) is given as well, then the file given by [IGFILE](#) contains G^{-1} for the single-step and [IA22FILE](#) has A_{gg}^{-1} .

Default matrix format is the co-ordinate (Yale) sparse matrix format. The preprocessor checks that each row is a lower triangle part of the matrix. This check is simple (the second column value (j) must be less or equal to the first column value (i)), and not all programs produce this format. Thus, option [MIXED](#) allows reading the file without the check. Another format allowed is the lower triangle dense format (see Ch. 7.2). which

DEV

can be prepared using the `hginv` program. For this matrix option `LOWER` need to be used.

Syntax:

```
IGFILE [LOWER | MIXED] <filename>
```

Example. Matrix C_{GA} is in file `iH.dat`

```
IGFILE iH.dat
```

9.2.7 IHPRECON

Gives file having changes to the preconditioner matrix in the diagonal of inverse relationship matrix. For example, consider inverse relationship matrix for the [single-step method](#): $H^{-1} = A^{-1} - \begin{bmatrix} 0 & 0 \\ 0 & G^{-1} - A_{gg}^{-1} \end{bmatrix}$ where A^{-1} is inverse of pedigree relationship matrix for all animals, G^{-1} is inverse genomic relationship matrix, and A_{gg}^{-1} is inverse pedigree relationship matrix of genotyped animals. When command `IA22FILE` is used with the `PEDIGREE` option, no pre-calculated diagonal element values of A_{gg}^{-1} are available for the preconditioner. These values can be given using `IHPRECON` in a file. One diagonal element of $-A_{gg}^{-1}$ is given on a line: <ID code> <value>. Note that these are NOT diagonal elements of A_{gg} nor A_{gg}^{-1} but diagonal elements of $-A_{gg}^{-1}$.

Syntax:

```
IHPRECON <filename>
```

Example. Diagonal of matrix $-A_{gg}$ is in file `minus_diA_genotyped.dat`

```
IHPRECON minus_diA_genotyped.dat
```

9.2.8 INBREEDING

Column numbers of individual id code and [inbreeding coefficient](#) in the [inbreeding coefficient file](#) (see 9.2.9). Default is that inbreeding coefficients are all zero. 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
```

9.2.9 INBRFILE

Name of inbreeding coefficient file. Default is that all inbreeding coefficients are zero, and, thus, no inbreeding coefficient file is read.

Syntax:

```
INBRFILE <filename>
```

Example. Inbreeding coefficient file is `AM.inbr`.

```
INBRFILE AM.inbr
```

9.2.10 NORANSOL

Give random effects for which no solution files are made. 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
```

9.2.11 MISSING

Number indicating missing information for data in the real number columns. Default is zero.

Syntax:

```
MISSING <number for missing>
```

Example. Set missing value to -99999.0

```
MISSING -99999.0
```

9.2.12 RESTARTSOL

Make restart solution file. The restart solution file allows the solver `mix99s` to start iteration using old solutions. 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
```

9.2.13 SCALE

Scaling of observation by residual standard deviation. Scaling of observations by the residual standard deviation can be important for multiple trait models. Scaling makes observations from different traits to be on the same residual scale which may lead to numerically better behaving computations. Before any output is generated, all solutions are scaled back to the original units. Default is no scaling. Command `SCALE` is an option within `MODEL` command. The option is given on the same line as command `MODEL`.

Syntax:

```
MODEL SCALE
```

Example. Scaling is requested.

```
MODEL SCALE
```

9.2.14 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. 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
```

9.2.15 PRECON

Information on the [preconditioning](#). Format is the same as given in MiX99 manual for [mix99i](#) *Technical reference guide for MiX99 pre-processor*. Thus, first characters (one for each effect) are for the [within block effects](#), and then one common for all [across blocks effects](#). 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. See MiX99 manual for details.

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
```

9.2.16 RANDOM

Random effect names other than the [additive genetic effect](#) associated to pedigree. If command is not given, the only random effects are additive genetic and residual effects. Order of the effects give numbering of the random effects.

Syntax:

```
RANDOM <effect names>
```

Example. There are four random effects: HTM, PE, additive genetic, and residual. Thus, HTM is random effect number 1, PE is number 2, additive genetics is number 3, and residual is number 4. These numbers are used in the (co)variance file defined by **PARFILE**.

```
RANDOM HTM PE
```

or alternatively

```
RANDOM HTM PE animal
```

9.2.17 REGFILE

Regression matrix coefficient file. Commonly SNP coefficient matrix is given as a [regression coefficient matrix](#) file. See also commands [REGPARFILE](#) and [REGMATRIX](#).

Syntax:

```
REGFILE <filename>
```

Example. Regression matrix in file `snp.dat`

```
REGFILE snp.dat
```

9.2.18 REGMATRIX

Regression matrix information. The information given:

- type:

FIXED Fixed coefficient matrix,

RANDOM Random effects with single common variance,

HETEROGENEOUS Each effect has its own variance

- name: name of the effect
- column number information:

ID = **a** Column number of individual code is *a* (optional),

FIRST = **b** First column of regression coefficients is *b*,

LAST = **c** Last column number is *c*.

- SNP marker information (optional):

- 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:

SCALE = **<r>** Constant real value **<r>**, e.g., **SCALE** = 0.5.

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

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>**.

NEW

– File format:

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.

NEW

- 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 regression coefficient.

See also commands **REGFILE** and **REGPARFILE**.

Syntax:

```
REGMATRIX <type> <name> [ID=<column>] FIRST=<column> LAST=<column>
[IMPUTE=<missing>] [CENTER[={<real value>|<filename>}]]
[SCALE={<real value>|<filename>}] [FORMAT=<file format>]
[PRECON=<preconditioner>]
```

Example. Regression matrix information: common random variance, name of effects is *snp*, regression coefficients in columns 3 to 10. No column for id code. Block diagonal preconditioner.

```
REGMATRIX RANDOM snp FIRST=3 LAST=10 PRECON=b
```

9.2.19 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
```

9.2.20 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**. 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 *mix99pat.respar*

```
RESIDFILE ./data/mix99par.respar
```

9.2.21 RESIDUAL

Name of [integer number column](#) in the [data file](#) indicating number of residual variance used for this observation (see [RESIDFILE](#)). 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
```

9.2.22 TABLEFILE

Name of [covariable table file](#). See [TABLEINDEX](#) command. Default is no table index file is needed.

Syntax:

```
TABLEFILE <filename>
```

Example. Covariable table file is [FinTDMpara.cov](#).

```
TABLEFILE FinTDMpara.cov
```

9.2.23 TABLEINDEX

Name of [integer number column](#) in the [data file](#) indicating column for table index. 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
```

9.2.24 TAFILE

Gives file having the \mathbf{T}_A matrix in $\mathbf{G}^{-1} = ((1-w)\mathbf{Z}\mathbf{Z}' + w\mathbf{A}_{gg})^{-1} = \frac{1}{w}\mathbf{A}_{gg}^{-1} - \mathbf{T}_A'\mathbf{T}_A$ used by the [single-step method](#) where w is the residual polygenic proportion. The approach requires the \mathbf{A}_{gg}^{-1} matrix to be given separately using the command [IA22FILE](#), or by option PEDIGREE.

Format of the \mathbf{T}_A matrix file is like the lower triangle dense 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](#). The \mathbf{T}_A matrix file needs to be made by a special preprocessing program.

Syntax:

```
TAFILE <filename>
```

Example. Matrix \mathbf{T}_A is in file [TA.dat](#)

```
TAFILE TA.dat
```

DEV

9.2.25 **TEFILE**

Gives file having the T_e matrix in $G^{-1} = (ZZ' + \epsilon I)^{-1} = \frac{1}{\epsilon}I - T_e' T_e$ used by the [single-step method](#) where ϵ is a small number such as 0.01. The approach requires the A_{gg}^{-1} matrix to be given separately using the command **IA22FILE**, or by option **PEDIGREE**.

Format of the T_e matrix file is like the lower triangle dense format. However, the T_e matrix is a rectangular matrix. The T_e matrix file needs to be made by a special preprocessing program such as [T48eig_make](#).

Syntax:

```
TEFILE <filename>
```

Example. Matrix T_e is in file [Te.dat](#)

```
TEFILE Te.dat
```

9.2.26 **TITLE**

Line for title of the analysis. Default title is:

MiX99 analysis time: <current time and date>

Syntax:

```
TITLE <Title of the analysis>
```

Example.

```
TITLE New model for Simmental birth weight
```

9.2.27 **TMPDIR**

Directory for temporary files. Default is current directory.

Syntax:

```
TMPDIR <directory>
```

Example.

```
TMPDIR ./tmpMiX
```

9.2.28 **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
```

9.2.29 **WITHINBLOCKORDER**

Ordering of [effects within block](#). Order of effect after the command name gives the ordering. Default is that only animal genetic effect is within block.

Syntax:

```
WITHINBLOCKORDER <effect names>
```

Example. There are three effects within block. Order of effects within block: 1=G, 2=PE, 3=HerdYear.

```
WITHINBLOCKORDER G PE HerdYear
```

9.2.30 ZCFILE

Gives file having the centered marker matrix Z_c matrix in $G^{-1} = ((1 - w)Z_c B Z_c' + w A_{gg})^{-1} = \frac{1}{w} A_{gg}^{-1} - \frac{1}{w^2} A_{gg}^{-1} Z_c C^{-1} Z_c' A_{gg}^{-1}$ used by the [single-step method](#) where w is the residual polygenic proportion, B has scaling information and $C^{-1} = (\frac{1}{w} Z_c' A_{gg}^{-1} Z_c + B^{-1})^{-1}$. The command requires command [ICFILE](#) as well. The Z_c matrix file can be made by a special preprocessing program such as [T48eig_make](#). The approach requires the A_{gg}^{-1} matrix to be given separately using the command [IA22FILE](#) within option PEDIGREE.

DEV

Syntax:

```
ZCFILE <filename>
```

Example. Matrix Z_c is in file [Zc.dat](#)

```
ZCFILE Zc.dat
```

9.2.31 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.

NEW

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
```

```
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 SCALE
```

```
milk = CurveMILK Common PE(t5:10|animal)@1st G(t59:64|animal)@FST
```

```
protein = CurvePROT Common PE(t11:16|animal)@1st G(t65:70|animal)@FST
```

```
fat = CurveFAT Common PE(t17:22|animal)@1st G(t71:76|animal)@FST
```



```
TMPDIR WorkDIR/tmpMiX
```

is replaced by

```
DATAFILE /home/user/mydata.dat
```

```
PEDFILE /home/user/mydata.ped
```

```
REAL milk protein fat x1 x2 het1 het2 het3 het4 het5
```

```
MODEL SCALE
```

```

milk      = Curve(t1  t2  t3  t4  t96      | SEASON)      AGE DCC YM HTM &
              PE(t5  t6  t7  t8  t9  t10   | animal)@1st      &
              G(t59 t60 t61 t62 t63 t64   | animal)@FST
protein = Curve(t1  t2  t3  t95 t97      | SEASON)      AGE DCC YM HTM &
              PE(t11 t12 t13 t14 t15 t16  | animal)@1st      &
              G(t65 t66 t67 t68 t69 t70   | animal)@FST
fat      = Curve(t1  t2  t3  t95 t98      | SEASON)      AGE DCC YM HTM &
              PE(t17 t18 t19 t20 t21 t22  | animal)@1st      &
              G(t71 t72 t73 t74 t75 t76   | animal)@FST

```

```
TMPDIR ./tmpMiX
```

Currently, preprocessor ([mix99i](#)) [command line option](#) `--usemacros` is needed to activate the CLIM macro and range expansion.

10 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 9.1 except for command **MODEL** which is considered separately in Chapters 5, 6, 7 and 8. 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>

PEDFILE  <FileName>
PEDIGREE <Effect name> [ FILE | <type of relationship matrix> &
                        [<coefficient for random genetic group>] ]

MODEL [SCALE] [RESTARTSOL]
      <model lines>
```

Optional commands (see “[Optional commands](#)” for more details):

```
DATASORT  BLOCK=<block in INTEGER> PEDIGREECODE=<code in INTEGER>
IGFILE    [LOWER|MIXED] <filename>
IA22FILE  [LOWER|MIXED] <filename>
MISSING   <value for missing real number data>
NORANSOL  <random effect numbers without solution file>
PARALLEL  <number of processors> <number/first of common blocks> [FIRST]
PRECON    <preconditioning information>
RANDOM     <random effect names>
REGFILE   <filename>
REGPARFILE <filename>
REGMATRIX <type> <name> [ID=<column>] FIRST=<column> [LAST=<column>]
          [IMPUTE=<missing>] [CENTER[={<real value>|<filename>}]]
          [SCALE[={<real value>|<filename>|2pq|m|m2}]] [FORMAT=<file format>]
          [PRECON=<preconditioner>]
RESIDFILE <filename>
RESIDUAL  <INTEGER column name of the residual variance number>
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>
```

10.1 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) |

11 References


- 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](https://doi.org/10.3168/jds.S0022-0302(98)75607-3) (cit. on p. 58).
- Lidauer, M., Matilainen, K., Mäntysaari, E. A., Pitkänen, T., Taskinen, M., and Strandén, I. (2022). *Technical reference guide for MiX99 pre-processor*. Release I/2022. Natural Resources Institute Finland (Luke) (cit. on pp. 7, 66–68, 72).
- MiX99 Development Team (2022). *MiX99: A software package for solving large mixed model equations*. Release I/2022. Natural Resources Institute Finland (Luke). Jokioinen, Finland. URL: <http://www.luke.fi/mix99> (cit. on p. ii).
- Mrode, R.A. and Thompson, R. (2006). *Linear models for the prediction of animal breeding values*. CABI. DOI: [10.1079/9780851990002.0000](https://doi.org/10.1079/9780851990002.0000) (cit. on p. 4).
- 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. 22, 24, 26).
- 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](https://doi.org/10.3168/jds.S0022-0302(94)77209-X) (cit. on pp. 56, 57).
- 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](https://doi.org/10.1016/S0301-6226(00)00255-4) (cit. on pp. 58, 61, 62).
- 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. 7, 15).

Index

Index entry styles:

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

Page numbers:

- primary definition: [81](#)
- also referred: [81](#)
-  see example on page

+p, [15](#), [15](#), [67](#)


across blocks effects, [12](#), [72](#)

additional residual (co)variance file, [10](#)


additive genetic effect, [1](#), [4](#), [5](#), [10](#), [14](#),
[16](#), [17](#), [21](#), [36](#), [46](#), [72](#)

additive genetic variance, [16](#)

am, [67](#)

 [3](#), [14](#), [16–18](#), [23](#), [26–30](#), [33](#), [35](#),
[47](#), [51–53](#), [55](#)

am+p, [67](#)

 [15](#), [36](#), [37](#), [61](#), [67](#)

animal model, [4](#), [4](#), [14–16](#), [19](#), [22](#)

ApaX, [7](#), [9](#)

apax99, [1](#), [7](#), [11](#)

apax99p, [1](#), [1](#), [7](#), [11](#)

AR, [68](#)

 [68](#)

ar, [67](#), [68](#)

beta version, [2](#), [3](#), [31](#), [37](#)


BINARY

 [66](#), [79](#)

binary format data file, [7](#)

binary format solution file, [12](#)

BLOCK, [68](#)

 [68](#), [79](#)

block code, [7](#), [8](#), [9](#), [68](#), [71](#)

block diagonal preconditioner, [72](#), [72](#)


block ordering, [7](#), [68](#)

blocks, [7](#)

Broyden method, [59](#), [59](#)

calc_diag_iA22, [52](#)

CENTER, [41](#), [73](#)

 [41](#), [74](#), [79](#)

centering of SNP markers, [41](#)

CLIM, [1](#)

coefficient matrix, [6](#), [6](#)

combining traits, [34](#)

command file, [2](#)

command language interface, [1](#)

command line options, [3](#), [11](#), [37](#), [78](#)

commands, [65](#)

AR, [68](#)

COVFILE, [68](#)

DATAFILE, [65](#), [66](#)

DATASORT, [65](#), [68](#)

DEFINE, [66](#), [77](#)

IA22FILE, [69](#)

ICFILE, [69](#)

IGFILE, [69](#)

IHPRECON, [70](#)

INBREEDING, [70](#)

INBRFILE, [70](#)

INTEGER, [65](#), [66](#)

MISSING, [65](#), [71](#)

MODEL, [65](#), [66](#)

NORANSOL, [66](#), [71](#)

optional, [68](#)

PARALLEL, [66](#), [71](#)

PARFILE, [65](#), [66](#)

PEDFILE, [65](#), [67](#)

PEDIGREE, [65](#), [67](#)

PRECON, [66](#), [72](#)

RANDOM, [65](#), [72](#)

REAL, [65](#), [68](#)

REGFILE, [65](#), [73](#)

REGMATRIX, [65](#), [73](#)

REGPARFILE, [65](#), [74](#)

required, [66](#)

RESIDFILE, [65](#), [74](#)

RESIDUAL, [65](#), [75](#)

RESTARTSOL, [71](#)

SCALE, [71](#)

TABLEFILE, [65](#), [75](#)

- TABLEINDEX, [65](#), [75](#)
- TAFILE, [75](#)
- TEFILE, [76](#)
- TITLE, [66](#), [76](#)
- TMPDIR, [66](#), [76](#)
- TRAITGROUP, [65](#), [76](#)
- WITHINBLOCKORDER, [66](#), [76](#)
- ZCFILE, [77](#)
- comment sign, [3](#)
- common blocks, [71](#)
- component names, [14](#), [16](#), [17](#), [21](#)
- convergence criteria, [6](#), [12](#), [59](#)
 - Ca, [11](#)
 - Cd, [11](#)
 - Cm, [11](#)
 - Cr, [11](#)
- covariable table file, [1](#), [22](#), [22](#), [24](#), [25](#), [65](#), [75](#)
- covariance matrix, [22](#), [29](#), [36](#), [43](#)
- covariance structure, [16](#), [16](#), [27](#)
- COVFILE, [68](#)
 - ☞ [47](#), [68](#)
- data file, [4](#), [7](#), [7–9](#), [11](#), [13](#), [16–26](#), [30](#), [32](#), [35](#), [38–40](#), [54](#), [55](#), [57](#), [65](#), [66](#), [68](#), [72](#), [74](#), [75](#), [79](#)
- DATAFILE, [38](#), [39](#), [65](#), [66](#)
 - ☞ [3](#), [8](#), [14](#), [18–20](#), [23](#), [26](#), [27](#), [29](#), [30](#), [33](#), [35–37](#), [39](#), [41](#), [45](#), [47](#), [50](#), [52](#), [53](#), [55](#), [57](#), [60](#), [61](#), [63](#), [66](#), [77–79](#)
- DATASORT, [65](#), [68](#)
 - ☞ [14](#), [18](#), [30](#), [40](#), [41](#), [45](#), [47](#), [50](#), [52](#), [53](#), [63](#), [68](#), [79](#)
- daughter yield deviation, [56](#)
- DEFINE, [66](#), [77](#)
 - ☞ [37](#), [77](#), [79](#)
- deregressed proofs, [58](#), [61](#)
- deregression, [58](#), [59](#), [60](#), [63](#)
- design matrix, [4](#), [4](#)
- Development features (DEV), [ii](#), [53](#), [69](#), [75](#), [77](#)
- directive file, [1](#), [2](#), [3](#), [31](#), [37](#), [59](#)
- DYD, daughter yield deviation, [56](#), [56](#), [57](#)
- executing CLIM, [2](#)
- FILE, [45](#), [67](#), [67](#)
 - ☞ [45](#), [67](#), [79](#)
- files, [7](#)
 - covariable tables, [21](#), [77](#)
 - data, [7](#)
 - directive, [2](#)
 - multi-trait data, [8](#)
 - multiple residual (co)variances, [10](#)
 - pedigree, [8](#)
 - regression coefficient matrix, [38](#)
 - solution files, [12](#)
 - Sol_mn, [12](#)
 - Solani, [13](#)
 - Solfnn, [12](#)
 - Solfix, [13](#)
 - Solrnn, [12](#)
 - Solreg, [12](#)
 - Solreg_mat, [40](#)
 - variance components, [10](#)
- FINBR
 - ☞ [15](#), [47](#), [51–53](#), [70](#)
- FIRST
 - ☞ [40–42](#), [72](#), [74](#), [79](#)
 - PARALLEL, [71](#)
 - REGMATRIX, [73](#)
- FIXED, [38](#), [73](#)
 - ☞ [41](#)
- FORMAT, [42](#), [74](#)
 - ☞ [42](#), [74](#), [79](#)
- G-BLUP, [38](#), [43](#), [43](#), [45](#), [46](#), [48](#), [50](#), [67](#)
- genetic covariance matrix, [4](#), [5](#), [28](#), [32](#), [33](#), [57](#)
- genetic group, [9](#), [15](#)
- genomic breeding value, [40](#), [40](#), [46–48](#), [51](#)
- genomic marker effect, [38](#), [38](#)
- genomic relationship matrix, [38](#), [38](#), [45](#), [48](#), [49](#)
- HETEROGENEOUS, [73](#)
- heterogeneous residual variance, [26](#)
- hginv, [51](#), [51](#), [52](#), [69](#), [70](#)
- I/O buffer size, [72](#)
- IA22FILE, [51](#), [69](#), [69](#), [70](#), [75–77](#)
 - ☞ [52](#), [53](#), [69](#), [79](#)
- ICFILE, [53](#), [69](#), [77](#)
 - ☞ [53](#), [69](#)
- ID, [38](#), [73](#)

- ☞ 40–42, 74, 79
- IGAMMAFILE, [54](#)
- ☞ 54
- IGFILE, [49–51](#), [53](#), [69](#), [69](#)
- ☞ 50–52, 70, 79
- IHPRECON, [52](#), [70](#), [70](#)
- ☞ 52, 54, 70
- imake99, [1](#)
- imputation, [41](#), [73](#)
- IMPUTE, [41](#), [73](#)
- ☞ 41, 74, 79
- INBREEDING, [70](#)
- ☞ 15, 47, 51–53, 70
- inbreeding coefficient, [15](#), [15](#), [46](#), [49](#), [70](#)
- inbreeding coefficient file, [46](#), [70](#), [70](#)
- INBRFILE, [70](#)
- ☞ 15, 47, 51–53, 70
- incidence matrix, [4](#), [16](#), [43](#), [46](#), [49](#)
- instruction file, [1](#), [2](#), [14](#), [15](#), [25](#), [30](#), [31](#)
- INTEGER, [3](#), [11](#), [13](#), [65](#), [66](#), [77](#)
- ☞ 3, 8, 14, 16–20, 23, 26, 27, 29, 30, 33, 35–37, 39, 41, 45, 47, 50, 52, 53, 55, 57, 60, 61, 63, 66, 68, 79
- integer number column, [7](#), [13](#), [16](#), [21](#), [24](#), [26](#), [34](#), [35](#), [65](#), [66](#), [68](#), [74–76](#)
- intercept, [21](#), [21](#), [22](#)
- IOD, iteration on data, [6](#)
- iteration on data, [6](#)
- iterative method, [6](#), [36](#), [59](#)
- lactation curve, [21](#), [21](#), [23](#)
- lactation model, [36](#)
- LAST, [73](#)
- ☞ 40–42, 74, 79
- line continuation, [3](#), [13](#)
- LOWER
- ☞ 44, 50–52, 67, 69, 70, 79
- IA22FILE, [69](#)
- IGFILE, [50](#), [70](#)
- PEDFILE, [44](#), [67](#)
- LS-model, [12](#)
- MACE model, [56](#), [56](#), [57](#)
- macro, [77](#)
- macro and range abbreviations, [37](#), [77](#)
- maternal effect model, [27](#), [37](#)
- maximum number of iterations, [6](#), [6](#), [11](#), [59](#)
- memory requirements, [6](#), [10](#), [12](#)
- metafounders, [54](#)
- MISSING, [2](#), [8](#), [65](#), [71](#)
- ☞ 39, 41, 45, 47, 50, 52, 53, 58, 63, 71, 79
- missing effect, [29](#), [31](#)
- missing marker value, [41](#), [41](#), [42](#)
- missing observation, [54](#)
- missing parents, [9](#)
- missing value
 - integer number column, [8](#)
 - real number column, [8](#), [71](#)
- missing variance component, [29](#)
- Mix99_DIR.DIR, [2](#), [2](#), [3](#), [31](#), [32](#)
- mix99i, [1](#), [1–3](#), [17](#), [66](#), [67](#), [72](#), [78](#)
- mix99p, [1](#), [6](#), [7](#), [9](#), [11](#), [11](#)
- mix99s, [1](#), [6](#), [7](#), [11](#), [11](#), [40](#), [58](#), [59](#), [71](#)
- MIXED
- ☞ 44, 50, 54, 67, 69, 70, 79
- IA22FILE, [69](#)
- IGFILE, [50](#), [69](#)
- PEDFILE, [44](#), [67](#)
- mixed model equations, [5](#), [5–7](#), [43](#), [58](#), [59](#)
- solving, [6](#)
- MODEL, [65](#), [66](#), [71](#), [79](#)
- ☞ 3, 14, 16–19, 21, 23, 26–31, 33, 35–38, 40, 41, 45, 47, 51–53, 55, 58, 60, 61, 63, 71, 77–79
- models
 - animal, [3](#), [4](#), [14](#)
 - Finnish test-day, [36](#)
 - G-BLUP, [38](#)
 - genomic data models, [38](#)
 - genomic evaluation, [38](#)
 - maternal effect, [27](#)
 - multi-trait, [37](#)
 - multi-trait, [5](#), [29](#)
 - random regression, [21](#)
 - reduced rank random regression, [34](#)
 - repeatability animal, [16](#)
 - single trait, [4](#)
 - sire, [19](#)
 - SNP-BLUP, [38](#)
- multiple trait model, [2](#), [5](#), [7](#), [8](#), [10](#), [29](#),

- 30, 34, 35, 54–56
- multiple trait sire model, [56](#), [61](#)
- nesting, [1](#), [2](#), [4](#), [21](#), [21](#), [23](#), [27](#)
- New features (NEW), [ii](#), [37](#), [42](#), [51](#), [54](#),
[73](#), [74](#), [77](#)
- NORANSOL, [2](#), [66](#), [71](#)
 - ☞ [71](#), [79](#)
- numerator relationship matrix, [4](#), [27](#)
- old solutions, [71](#)
- optional commands, [65](#), [68](#)
- PARALLEL, [66](#), [71](#)
 - ☞ [72](#), [79](#)
- parallel computing, [1](#), [7–9](#), [66](#), [71](#)
 - workload division, [72](#)
- PARFILE, [29](#), [32](#), [65](#), [66](#), [72](#), [74](#)
 - ☞ [3](#), [14](#), [18–20](#), [23](#), [26](#), [27](#), [29](#), [30](#),
[33](#), [35–37](#), [40](#), [41](#), [45](#), [47](#),
[51–53](#), [55](#), [58](#), [60](#), [61](#), [63](#), [67](#),
[79](#)
- PCG, [6](#), [6](#), [12](#), [56](#), [59](#)
- PEDFILE, [44](#), [48](#), [50](#), [65](#), [67](#), [67](#)
 - ☞ [3](#), [14](#), [18–20](#), [23](#), [26](#), [27](#), [29](#), [30](#),
[33](#), [35–37](#), [44](#), [45](#), [47](#), [50](#), [52](#),
[53](#), [55](#), [58](#), [60](#), [61](#), [63](#), [67](#),
[77–79](#)
- PEDIGREE, [15](#), [45](#), [65](#), [67](#), [67](#)
 - ☞ [3](#), [14–20](#), [23](#), [26–30](#), [33](#), [35–37](#),
[45](#), [47](#), [51–53](#), [55](#), [58](#), [60](#), [61](#),
[63](#), [67](#), [79](#)
- IA22FILE, [51](#), [69](#), [69](#)
- IHPRECON, [70](#)
- pedigree file, [7](#), [8](#), [8](#), [9](#), [15](#), [19](#), [20](#), [23](#),
[28](#), [57](#), [65](#), [67](#)
- PEDIGREECODE, [68](#)
 - ☞ [14](#), [15](#), [18](#), [30](#), [40](#), [41](#), [45](#), [47](#),
[50–53](#), [63](#), [68](#), [70](#), [79](#)
- permanent environment effect, [16](#), [17](#),
[28](#), [36](#)
- permanent environment variance, [16](#),
[28](#), [35](#)
- polygenic effect, [46](#), [47](#), [48](#)
- polygenic variance, [46](#)
- PRECON, [66](#), [72](#)
 - ☞ [40](#), [41](#), [45](#), [63](#), [72](#), [74](#), [79](#)
- REGMATRIX, [74](#)
- preconditioned conjugate gradient
method, [6](#), [6](#)
- preconditioning, [6](#), [66](#), [72](#)
 - block diagonal, [72](#)
- preprocessor, [1](#), [14](#)
- RANDOM, [10](#), [47](#), [65](#), [72](#)
 - ☞ [17](#), [18](#), [28](#), [29](#), [36](#), [40–42](#), [47](#),
[72–74](#), [79](#)
- REGMATRIX, [38](#), [73](#)
- random regression function, [22](#), [32](#)
- random regression model, [1](#), [10](#), [21](#),
[23](#), [26–28](#), [32](#)
 - multiple trait, [10](#), [32](#)
- range expansion, [77](#)
- rank reduction, [36](#)
- REAL, [13](#), [65](#), [68](#), [77](#)
 - ☞ [3](#), [8](#), [14](#), [16–20](#), [23](#), [26](#), [27](#), [29](#),
[30](#), [33](#), [35–37](#), [39](#), [41](#), [45](#), [47](#),
[50](#), [52](#), [53](#), [55](#), [57](#), [60](#), [61](#), [63](#),
[68](#), [77–79](#)
- real number column, [7](#), [8](#), [13](#), [21](#), [68](#)
- reduced rank model, [36](#)
- reduced rank random regression
models, [34](#)
- REGFILE, [38–42](#), [65](#), [73](#), [74](#)
 - ☞ [40–42](#), [73](#), [79](#)
- REGMATRIX, [38](#), [40–42](#), [65](#), [73](#), [73](#), [74](#)
 - ☞ [40–42](#), [74](#), [79](#)
- REGPARFILE, [38](#), [65](#), [73](#), [74](#), [74](#)
 - ☞ [40](#), [41](#), [74](#), [79](#)
- regression coefficient file, [38](#), [39](#), [73](#)
- regression coefficient matrix, [38](#), [38](#),
[40](#), [65](#), [73](#)
- regression effect, [4](#), [4](#), [12](#), [13](#), [21](#)
- relationship matrix, [4](#), [15](#), [28](#), [46](#), [48](#),
[49](#)
- RelaX2, [7](#), [15](#), [54](#)
- repeatability model, [21](#), [34](#), [35](#), [54](#)
- repeated observations, [21](#), [22](#)
- required commands, [65](#), [66](#), [79](#)
- reserved characters, [13](#), [79](#)
- RESIDFILE, [26](#), [65](#), [74](#), [75](#)
 - ☞ [27](#), [74](#), [79](#)
- RESIDUAL, [26](#), [65](#), [74](#), [75](#)
 - ☞ [27](#), [75](#), [79](#)
- residual covariance matrix, [4](#), [5](#), [11](#), [33](#),
[35](#), [56](#)

residual variance, [16](#), [22](#), [35](#)

restart solution file, [71](#)

RESTARTSOL, [71](#)

☞ [71](#), [79](#)

SCALE, [2](#), [71](#)

☞ [26](#), [27](#), [29–31](#), [33](#), [35](#), [37](#), [42](#),
[55](#), [61](#), [71](#), [74](#), [77–79](#)

REGMATRIX, [42](#), [73](#)

scaling of SNP markers, [41](#)

secant method, [59](#), [59](#)

single-step method, [38](#), [49](#), [49](#), [50](#), [69](#),
[70](#), [75–77](#)

sire maternal grand sire relationship
matrix, [20](#)

sire model, [4](#), [7](#), [19](#), [19](#), [20](#)

sm, [67](#)

☞ [19](#), [20](#)

sm+p

☞ [58](#), [60](#), [63](#)

SNP marker, [38](#), [39](#), [39](#), [41](#), [43](#)

SNP marker variance, [46](#)

SNP-BLUP, [38](#), [38](#), [42](#), [43](#)

Sol_mn, [12](#)

Solani, [12](#), [13](#), [13](#), [14](#), [16](#), [18–21](#), [24](#),
[27](#), [29–32](#), [34](#), [36](#), [45–47](#), [51](#),
[56](#), [58](#), [60](#), [61](#), [63](#)

Solfnn, [12](#), [40](#)

Solfix, [12](#), [13](#), [14](#), [16](#), [18–21](#), [23](#), [27](#),
[29–33](#), [36](#), [45](#), [47](#), [51](#), [56](#), [58](#),
[60](#), [61](#), [63](#)

Solrnn, [12](#), [18](#), [29](#), [47](#)

Solreg, [12](#), [24](#), [26](#), [27](#), [34](#)

Solreg_mat, [40](#)

solution files, [2](#), [12](#)

solver, [7](#), [11](#), [12](#), [14](#)

T48eig_make, [69](#), [75–77](#)

TABLEFILE, [22](#), [24](#), [65](#), [75](#), [75](#)

☞ [26](#), [27](#), [36](#), [75](#), [79](#)

TABLEINDEX, [22](#), [24](#), [65](#), [75](#), [75](#)

☞ [26](#), [27](#), [36](#), [75](#), [79](#)

TAFILE, [53](#), [75](#)

☞ [53](#), [75](#)

TEFILE, [53](#), [76](#)

☞ [53](#), [76](#)

temporary files, [66](#), [76](#)

test-day model, [21](#), [21](#), [25](#), [32](#), [36](#)

TEXT

☞ [66](#), [79](#)

text file, [7](#)

TITLE, [2](#), [66](#), [76](#)

☞ [57](#), [63](#), [76](#), [79](#)

TMPDIR, [2](#), [66](#), [76](#)

☞ [76–79](#)

trait group, [54](#), [54](#), [57](#), [76](#)

TRAITGROUP, [65](#), [76](#)

☞ [55](#), [57](#), [76](#), [79](#)

unknown parent group, [9](#), [56](#), [58–61](#),
[63](#), [67](#)

VanRaden method 1, [45](#), [49](#)

variance components, [6](#), [10](#), [10](#), [11](#), [19](#)

WEIGHT, [20](#)

☞ [21](#), [58](#), [60](#), [61](#)

weighted sire model, [20](#)

weights, [7](#), [20](#)

within block effect, [7](#), [72](#), [76](#)

within block fixed effect, [12](#), [12](#)

WITHINBLOCKORDER, [7](#), [66](#), [76](#)

☞ [37](#), [76](#), [79](#)

yHat.data0, [40](#)

ZCFILE, [53](#), [69](#), [77](#)

☞ [53](#), [77](#)