# Manual

## MiXBLUP 2.2.0 manual

V2.2 – 2020 – 05

MiXBLUP

# MiXBLUP User's Guide

MiXBLUP, the Mixed-model Best Linear Unbiased Prediction software for PCs for large genetic evaluation systems

This manual is for MiXBLUP version 2.2, builds 2.2.0.00001-2.2.0.00010, released in May 2020

MiXBLUP is developed jointly by LUKE National Resources Institute Finland and Animal Breeding & Genomics, Wageningen University & Research.

**Authors:**
J. ten Napel
J. Vandenplas
M. Lidauer
I. Stranden
M. Taskinen
E. Mäntysaari
M.P.L. Calus
R.F. Veerkamp

Animal Breeding & Genomics, Wageningen University & Research
P.O. Box 338
6700 AH Wageningen
The Netherlands

LUKE National Resources Institute Finland
FI-31600
Jokioinen
Finland

More information on http://www.mixblup.eu/

# CONTENTS

# 1. Introduction

MiXBLUP has been developed for routine breeding value estimation in commercial genetic programmes and supports modern applications, such as random regression models, group selection, the use of genetic markers or haplotypes and the use of genomic information.

## 1.1 Overview

The intention of developing MiXBLUP was to utilize efficient computing strategies for solving mixed model equations. With MiXBLUP it is possible to use sophisticated models in estimation of breeding values in animals, like cattle, pigs, poultry, sheep, horses, goats and dogs, and in plants. The MiXBLUP software also supports many ways to specify genetic similarity between individuals, including pedigree, marker information and genomic information. The statistical method used for genetic evaluation is best linear unbiased prediction (BLUP), which is currently the common methodology for genetic evaluation.

The software was initially developed for classical genetic evaluation without the use of markers or genes by LUKE National resources Institute Finland. The adaptation for the use of marker and genomic information was implemented by Wageningen UR Livestock Research in collaboration with LUKE.

The MiXBLUP parser and kernel have been developed for efficient use of disk space and memory. Due to iteration on data and a very fast algorithm in the kernel (preconditioned conjugate gradient), MiXBLUP is able to solve mixed model equations very fast.

## 1.2 Manual

This manual will guide the user through the use of MiXBLUP. The examples provide a way to test MiXBLUP, to get a feel for the software. A set of examples is provided as an Appendix to the manual. A schematic overview of the input files, output files and instruction file is in Figure 1.

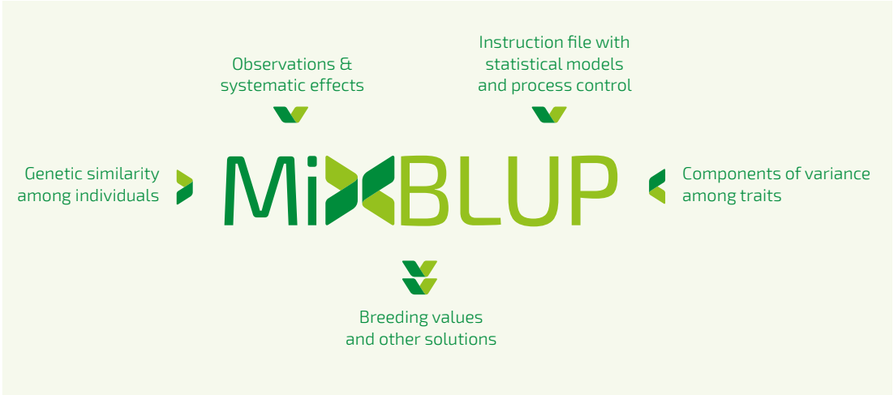


*Figure 1.* Schematic overview of the input and output files of MiXBLUP.

### 1.3 System requirements

MiXBLUP is written in standard Fortran 90 language and is self-contained. The program runs in Windows, Linux and Unix environments and is available in 64-bit version. In Windows, MiXBLUP runs in the command-line interpreter, cmd.exe (DOS box). The MiXBLUP Windows release it is routinely tested in a Windows 10 operating system.

MiXBLUP allocates memory depending on the need. Small applications can be run with a minimum of memory available. Very large applications may need a substantial amount of memory, especially genomic analyses and the calculation of reliabilities. For a reliabilities analysis, the user can increase memory allocation with the !MAXNONZ qualifier in the SOLVING section (see Chapter 9).





# 2. How to start

MiXBLUP is easy to use and easy to install. This chapter describes how to install the software and how to obtain and install a license.

### 2.1 Installing MiXBLUP software

Download the appropriate zip-file from http://www.mixblup.eu and unzip the folder with the executables: 'reliabilities.exe', 'dataprocessor.exe', 'solver.exe', 'calc_grm.exe', 'compute_SNP_effects.exe' and 'MiXBLUP.exe' into the target folder for the MiXBLUP analysis

Alternatively, the executables may be installed in a central folder that can be accessed from other folders. If a central folder is used, the user needs to create a file, named 'SysDir.inp', which contains the path to the executables. This file should be copied to any folder from which MiXBLUP is run. The path to MiXBLUP.exe should be included in the command file that starts up the analysis. MiXBLUP uses SysDir.inp to locate the other executables.

### 2.2 MiXBLUP Licenses

To run MiXBLUP software on your computer you need a license. There are different license types for MiXBLUP. A license can be ordered at http://www.mixblup.eu. The trial license can handle complete datasets and will provide a maximum of 1000 solutions. This will give the user an opportunity to test the software and decide if it suits their needs.

The small commercial license can be used for up to 1 million animal equations. This means that a single trait evaluation could be performed with up to 1 million animals in the pedigree. With multi-trait evaluations (n traits) the number of animals in the pedigree can be 1 million/n. The small commercial license does not allow the use of a genomic relationship matrix.

The full commercial license, has no limit on the number of animal equations and provides access to all functionality that is commercially available in MiXBLUP. The license key of the commercial licenses is computer-specific. Therefore, if executables and the license key 'LICENSE.DAT' are moved to another computer, MiXBLUP will give an error message. Running MiXBLUP with the run-time option –Dl (minus, uppercase D, lowercase L) writes the host name, license type and expiry date in the license file to the screen output.

So if you want to transfer the MiXBLUP software with an existing license to a new computer, you must request a new license from info@mixblup.eu with the LICREQST.DAT attached (how to generate a LICREQST.DAT file see below). You will receive a new license for the remainder of the license period.

*Table 1.* The characteristics of the different license types of MiXBLUP.

| License types | License | Time limit | Limitations |
|---|---|---|---|
| Trial License | Not computer specific | 1 month | 1000 solutions |
| Small commercial License | Computer specific | 1 year | 1 million animal equations, pedigree relationship matrix only |
| Full commercial License | Computer specific | 1 year | Unlimited |

## 2.3 License key

The license key provides the information about the MiXBLUP version, the license type and the expiry date of the license. A trial license can be used for one month and a trial license key is not computer-specific. The small and full commercial license can be used for one year. The license key for these licenses is computer-specific.

### Trial License

Order a trial license at http://www.mixblup.eu. After receiving your order, we send the necessary license key to the e-mail address stated in the order.

### Commercial licenses

Order a commercial license at http://www.mixblup.eu. While entering the order you are asked to upload one or more 'LICREQST.DAT' files. For each computer you need to upload a separate 'LICREQST.DAT' file. This file is required to generate a license key for your computer.

### Generating a 'LICREQST.DAT' file and installing the license 'LICENSE.DAT'

> Run MiXBLUP.exe once without the need for an instruction file. MiXBLUP creates the file LICREQST.DAT in the working directory.
> After payment of the license one or more 'LICENSE.DAT' files will be sent back and should be saved in the bin folder of the corresponding computer(s).
> Store the license key 'LICENSE.DAT' in the C:\MIXBLUP\bin-folder for Windows or in the / usr/bin-folder for Linux.

### Alternative license directory

If the license key cannot be stored in the default directory, the user may create a file, named LicDir.inp, which contains the path to the license file. If this file exists, MiXBLUP will look for the license file in the specified folder.





# 3. Instruction file

The instruction file contains all information that MiXBLUP needs for the analysis. This chapter gives an overview of the instruction file. The various parts of the instruction file are discussed in detail in the chapters 4 to 8.

## 3.1 Parts of the instruction file

The information in the MiXBLUP instruction file is presented in six parts. These parts are:

1. Description of the analysis
2. Observations & systematic effects
3. Genetic similarity among individuals
4. Components of variance and covariance among traits
5. Statistical models
6. Control of analysis and output

These parts may be presented in the instruction file in any order. Sections within a part may also appear in any order.

Below the example instruction file is given for a bivariate animal model for two traits (phen1 and phen2).

*Example.* Parts of the instruction file.

```
TITLE breeding value estimation for phen1 and phen2 using pedigree

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

### 3.1.1 Title of the analysis

The instruction file must start with a specification of the title of the analysis. The TITLE keyword is optional. If omitted, the first line must start with a hash (#). This comment line is then used as the title of the analysis. This line can be used to describe the analysis and distinguish it from other analyses.

### 3.1.2 Observations & systematic effects

The data observations part of the instruction file contains the name of the files with data or covariates, their location and their record layout. The sections that can be used in this part are DATAFILE, CVRTABLE and REGFILE. The syntax of these sections, more advanced options and examples are presented in Chapter 4 of this manual.

### 3.1.3 Genetic similarity among individuals

Genetic similarity among individuals can be specified in many different ways. It may be based on pedigree information only, genomic information only or both sources of information simultaneously. Pedigree information may contain genetic groups for unknown parents or a single code to denote an unknown parent. Inbreeding can be taken into account or ignored. Genomic information may be incorporated through covariances between individuals or through ridge-regression on SNP covariates. Sections that can be used in this part are PEDFILE, ERMFILE, INBRFILE, SNPFILE, IMPFILE, DHFILE, REGFILE and CVMATRIX. The syntax of these sections and examples for the various options are presented in Chapter 5 of this manual.

### 3.1.4 Components of variance and covariance among traits

Genetic and non-genetic random effects have components of variance and covariance among traits in the model. Residual (co)variance components may also vary between groups of data records. Section that can be used in this part are PARFILE, RESFILE, SNPPARFILE and REGPARFILE. The syntax of these sections is presented in Chapter 6.

### 3.1.5 Statistical models

Statistical models are specified by trait. Each trait starts on a new line. The only two sections in this part of the instruction file are MODEL and COMBINE. The syntax of the various statistical models supported by MiXBLUP are presented in Chapter 7.

### 3.1.6 Control of analysis and output

The control part of the instruction file can be used to specify (1) whether to solve the system (i.e. estimate breeding values) or calculate approximate reliabilities, (2) whether or not to use starting values, (3) which resources to use for parts of the process, (4) when to stop the iterative process and write out the solutions, (5) how to present the solutions, (6) which additional output files to create after the solving process has been completed and (7) how to manage temporary files. The sections that can be used in this part are SOLVING, TRAITEBV, PRECON and TMPDIR. Syntax is presented in Chapter 8.

## 3.2 General syntax of the instruction file

- The maximum record length of the instruction file is 5,000 characters
- The instruction file may contain empty lines for the convenience of the user
- Comments may be inserted on a new line or after instructions on the same line, provided that any comment starts with a hash (#). Any text on a line following a hash is ignored by MiXBLUP
- The keyword of any section must be the first word of the line





- Sections may appear in any order
- Qualifiers within a section may generally appear in any order and anywhere in the section, provided that they are not linked to a specific field in the DATAFILE section, a specific file in the SNPFILE or REGFILE section or a specific trait in the MODEL section.
- Qualifiers start with an exclamation mark (!). There must not be a space between the exclamation mark and the qualifier
- A statement may be continued on the next line by using an ampersand (&) as the last word of a line or the last word before a hash (#).
- Section keywords, qualifiers, labels, values and the special characters #, & and ~ must be separated by at least one space.
- Field names, trait names and labels are case-sensitive. Sections and qualifiers are not.

## 3.3 Key to section-specific syntax

The specific syntax is described for each section separately in the next chapters. The key to the paragraphs detailing the section-specific syntax is given below.

- The string <...> is used to indicate a value or text label provided by the user as input.
- The string [...] is used to indicate an optional qualifier or optional input.
- Keywords for sections and qualifiers are in presented in capitals to distinguish them from other text.
- The ampersand (&) is used to continue the syntax on the next line.
- The string ... on a line is used to indicate that similar lines may be present

# 4. Observations & systematic effects

The data observations part of the instruction file is used to specify observed traits and any factors or covariates that cause systematic variation between observations for these traits. This chapter describes the various ways to present observations and systematic effects.

## 4.1 Data file

### 4.1.1 General

Observations and systematic effects are normally presented in the data file. All traits and effects in the statistical model must have a column in the data file, except for covariates in a covariate table file (see chapter 4.2) and covariates in an external covariate file (see chapter 4.3).

The name of the data file is specified in the instruction file. The data file is located by default in the work directory, but it can be in any other folder if this is specified as part of the name of the file (e.g. d:\PerformanceTest\BreedP.txt). The order of the fields in the DATAFILE section must be the same as the order of the columns in the data file.

### 4.1.2 Input file

The data fields (individuals, systematic effects and trait observations) each have their own column in the data file. The data file must be provided in space-separated format, which means that any two columns are separated by at least one space. Data fields can be integer values or alphanumeric labels for class effects or real values for covariates and trait observations. Real values are read with a decimal point.

Details of the layout of the data file:

- The maximum column width in the data file is 25 characters.
- The maximum record length of the data file is 5,000 characters.
- When data is alphanumeric, any of the symbols on the keyboard can be used, including a slash ('/').
- An alphanumeric string must not contain spaces or it will be interpreted as two strings.
- A class effect, regardless of whether it is declared as integer or alphanumerical, must not be zero or negative if it is a number. Data records with a class effect in the model that is zero are omitted from the analysis by the kernel as invalid data points. Therefore, MiXBLUP replaces any classes of zero or a negative number with a 1 for an integer class effect or "NA" for an alphanumerical class effect. This will not affect the results of the evaluation if the invalid classes are not associated with a valid trait observation. It is left to the user to verify that this is indeed the case.
- The default missing-value indicator for traits and covariates is zero. Data records with a covariate in the model that is equal to the missing-value indicator are omitted from the analysis by the kernel. If zero is a valid level for one of the covariates in the model, another missing-value indicator should be used.





*Example.* Columns in data file: animal ID, mean, herd, sex, dam ID, haplotype 1, haplotype 2, common environment, pen mate 1, pen mate 2, age 1, age 2, genotype, body weight at age 1, bodyweight at age 2.

```
A11 1 1 1 A6  1 2 1 A12 A13 100 200 1 1200  2000
A12 1 1 2 A6  2 1 1 A11 A13 101 203 1 1280  2100
A13 1 1 1 A7  1 2 2 A11 A12 99  199 1 1100  1900
A14 1 1 2 A7  2 2 2 A15 A16 102 198 2 1250  1800
A15 1 2 1 A8  2 1 3 A14 A16 90  201 1 1150  2200
A16 1 2 2 A8  2 1 3 A14 A15 95  203 1 1000  2050
A17 1 2 1 A9  1 2 4 A18 A19 103 205 1 1300  1950
A18 1 2 2 A9  2 2 4 A17 A19 105 195 2 1250  2080
A19 1 2 1 A10 1 1 5 A17 A18 110 199 0 1280  1920
```

### 4.1.3 Syntax

```
DATAFILE <filename> [!SKIP <n lines>] [!MISSING <value>] [!SLASH] [!STATS [N][D][H][L]]
!MINMAX <filename> <field 1> <field type: I/R/T/A>
...
[<field i> I !BLOCK]
...
[<field j> I !RESVARCLASS]
...
    [field n] [I/R/T/A]
```

Section:

**DATAFILE**
The DATAFILE section contains all the details of the file with trait observations and systematic effects.

Qualifiers:

**!MISSING <value>**
If the value specified for !MISSING is encountered when reading the data file, it is interpreted as a missing observation for the trait or covariate. A missing covariate invalidates the trait for which the covariate is included in the model.

**!BLOCK**
This field is used as the block variable. If used, the data file and pedigree file both need to contain this column. It is required for the calculation of reliabilities, but might be beneficial in some computationally heavy genetic evaluations. The field must be integer. The !block qualifier must not be specified in the PEDFILE section, but the fourth column in the pedigree file must have the same field name as the block variable in the data file.

**!RESVARCLASS**
This field is used to specify the residual variance class of the data record, in case the residual variance differs for groups of records. The field must be integer. The qualifier !RESVARCLASS must be used if the section RESFILE is specified.

**!SKIP <value>**
With this qualifier, one (!SKIP 1) or more (e.g. !SKIP 2) header lines in a data file can be ignored when reading the data file.

**!SLASH**
The qualifier !SLASH is optional and is used when any of the input files contains a forward slash ('/') as a character. A forward slash is also a control character in certain file formats. If !SLASH is not specified, but MiXBLUP encounters a record with a forward slash, it will re-start reading the file as if !SLASH had been specified. Reading of data with !SLASH specified is slower than normal reading of data.

**!STATS NDHL**
The qualifier !STATS can be used to obtain a summary of descriptive statistics of files in the evaluation, written to Statistics.log. There are four types of statistics that can be produced: **N** for numbers of records in data, pedigree and genotype file; **D** for means and standard deviations of traits and covariates in the data; **H** for grouping class effect levels for each trait by the number of records per class and **L** for a table by trait with number of records for each class effect level. For large evaluations, it is recommended to use !STATS NDH, as the option L might produce a very large output file. Types may be specified in any order. If D, H or L are specified, N is automatically included.

**!MINMAX <filename>**
The qualifier !MINMAX can be used to specify a file with the valid ranges of traits and covariates. The file contains three fields for each record: the name of the field in the data file (case-sensitive!), the minimum and the maximum valid value. Field records may be in any order and may contain field records of other data files, like the parameter file.

More details of the syntax of the DATAFILE section:

- The field specification must start on the line following the line containing the DATAFILE keyword
- The field type indicates whether a field in the data file should be read as an integer value (I), a real value for covariates (R), a real value for a trait (T) or a text string (A).
- Maximum length of field names is 8 characters. A field name may be up to 19 characters long, but only the first 8 characters are used to distinguish fields, so a warning is given to remind the user. Field names longer than 19 characters result in an error.
- Field names are case-sensitive throughout MiXBLUP.
- If !BLOCK is specified for multiple data fields, only the first specification is used. It affects the SORTED-line in the file generated by the parser (dataprocessor.inp or dataprocessor_rel.inp).
- Alphanumerical labels of a class effect (fields coded with A) are converted into integer values for the analysis. Solutions are decoded back to the original alphanumerical labels of the effect.
- Each alphanumerical label in a field in the data file gets a unique numerical value. There is no apparent relation between the alphanumerical label and numerical value, so the numerical value of a string may vary across runs without a restart. The numerical value of a string does not change if !RESTART is specified in the SOLVING section.
- The ID of animal in the data file, and the IDs of animal, its sire and its dam in the pedigree file must all be of the same type, so either alphanumeric (A) or numeric (I).
- The largest integer number that can be used as level of a class effect is approximately 2,100,000,000. For class effects with levels that exceed this number, the field type has to be set to alphanumerical (A).
- The version of the data file with alphanumerical labels converted to integer values is 'data.txt'.
- The use of names reserved as section keywords, qualifiers or functions as field names is not supported.

### 4.1.4 Associated output files

| Output file | Description |
|---|---|
| data.txt | temporary file; data file prepared for analysis by kernel |

## 4.2 Covariate table file

### 4.2.1 General

If the relationship between an independent variable and a dependent trait is modelled as an $n^{th}$ order polynomial, a covariate table file with all levels of the independent variable between its minimum and maximum value in the data and (n+1) columns of covariates may be used for easy presentation of covariates and syntax of the instruction file.





The name of a pre-defined covariate table file is specified in the instruction file. The name may include the path to the covariate table file.

The covariate table file can also be created in MiXBLUP. Currently only a Legendre polynomial is supported. The covariate table is created using the minimum and maximum value of the independent variable and required order of the polynomial. The minimum and maximum value of the independent variable can either be specified by the user or determined from the data.

Only one covariate table can be used, but its columns may be fitted within multiple class effects. Additional polynomials using other independent variables should be added as columns in the data file outside of MiXBLUP.

### 4.2.2 Input file

The covariate table file may be created outside of MiXBLUP, it may have been created in a previous analysis or it may be created at run-time. It consists of the original independent variable and the n+1 covariates derived from it, with n being the order of the polynomial.

If the order is n, the covariate columns in the table are numbered from 0 to n, giving n+1 covariate columns in addition to the original independent variable.

The independent variable has to have an integer field type. The covariate table should contain all levels between the minimum and maximum value with steps of one. It means that an independent variable with decimals must be converted to integer values before a covariate table can be used for it. The independent variable links the record in the data file with the covariate record in the covariate table.

The column in the data file with the independent variable must contain a valid entry for every record.

**Example**. A covariate table file for an independent variable with values in the data between 86 and 115. The order of the Legendre polynomial is 2. The table was created with the line CVRTABLE !CVRMAKE LEG !CVRNUM 2 !CVRMIN 86 !CVRMAX 115 in the instruction file.

```
86 0.707106769 -1.22474492 1.58113885
87 0.707106769 -1.14027977 1.26528728
88 0.707106769 -1.05581462 0.971996129
89 0.707106769 -0.971349418 0.701266170
90 0.707106769 -0.886884212 0.453096747
91 0.707106769 -0.802419066 0.227488458
92 0.707106769 -0.717953920 2.44409554E-02
93 0.707106769 -0.633488715 -0.156045854
<lines for 94 to 107 omitted>
108 0.707106769 0.633488715 -0.156045854
109 0.707106769 0.717953920 2.44409554E-02
110 0.707106769 0.802419066 0.227488458
111 0.707106769 0.886884212 0.453096747
112 0.707106769 0.971349418 0.701266170
113 0.707106769 1.05581462 0.971996129
114 0.707106769 1.14027977 1.26528728
115 0.707106769 1.22474492 1.58113885
```

### 4.2.3 Syntax using an existing covariate table

```
DATAFILE <filename>
...
   <field k> I !CVRIND
...
CVRTABLE <filename>
MODEL
<trait> ~ <fixed effects> <Class1>*CVR(n1) !RANDOM <Class2>*CVR(n2) G(Animal*CVR(n3))
...
```

Sections:

**CVRTABLE**
The CVRTABLE section contains the details of the existing or new covariate table.

Qualifiers:

**!CVRIND**
The field marked with !CVRIND is the independent variable used in polynomial regression. Any level of the field specified with !CVRIND must exist in the covariate table file. The field must not contain a missing value indicator for a valid trait observation. The qualifier !CVRIND must be used when the section CVRTABLE is specified. The field must be integer.

**CVR(...)**
The CVR function is used in the MODEL section and is a shorthand for all polynomial terms to be fitted and may be used in the same way as any individual random regression term. The alternative way to specify polynomial random regression is to use the individual columns of the covariate table file. The names of the columns are cvr00, cvr01, cvr02, ..., cvr*nn*.

### 4.2.4 Syntax using a newly created covariate table

```
DATAFILE <filename>
...
<field k> I !CVRIND
...
CVRTABLE !CVRMAKE LEG !CVRNUM <nth order> !CVRMIN <minimum value> !CVRMAX <maximum value>
MODEL
<trait> ~ <fixed effects> <Class1>*CVR(n1) !RANDOM <Class2>*CVR(n2) G(Animal*CVR(n3))
...
```

Additional qualifiers:

**!CVRMAKE**
If !CVRMAKE is specified, MiXBLUP generates a covariate table file using the settings specified with the !CVRNUM, !CVRMIN and !CVRMAX qualifiers. Currently, only a covariate table containing Legendre polynomials can be created, by specifying LEG as the argument of !CVRMAKE. The name of the new covariate table file is 'cvrtable.txt'.

**!CVRNUM**
The qualifier !CVRNUM must be specified and is used to specify the order of the polynomial in the covariate table. The expected number of columns to read is the order + 2, one for the level of the independent variable and one for the order being 0. It is up to the user to make sure that the order specified in the MODEL section is equal to or lower than the order specified with !CVRNUM.





**!CVRMIN and !CVRMAX**
The qualifiers !CVRMIN and !CVRMAX can be used to specify the lowest and highest value of the independent variable that were used to estimate the genetic parameters. Legendre polynomials are dependent on the lowest and highest value of the independent variable and so are the genetic parameters of Legendre polynomials. If !CVRMIN or !CVRMAX is nevertheless omitted, the lowest or highest value of the independent variable in the data is used, instead.

### 4.2.5 Associated output files

| Output file | Description |
|---|---|
| cvrtable.txt | covariate table, if created by MiXBLUP |

## 4.3 General covariate files

### 4.3.1 General

Some covariates are individual-specific: they never change for an individual, but vary across individuals. They are more associated with the individual than with its data records. Examples are breed composition, genetic groups, heterosis and recombination.

Such covariates can be stored in a covariate file, in which all individuals in the analysis have a record. MiXBLUP converts the covariate file with all individuals to a data covariate file that exactly matches the data file, including repeated records.

### 4.3.2 Input file

General covariate files contain at least the ID of the animal and any number of covariates, but all records should have the same number of covariates. General covariate files must be provided in space-separated format. Covariates are read as real numbers, regardless of whether a decimal point is present in the corresponding field.
General covariate files contain at least all individuals with a phenotype for any of the traits in the statistical model. Individuals without any phenotypes will be ignored, except in the case of genetic group covariates (see chapter 5.2.4).

*Example.* Covariate file with breed fractions in a mixed breed population

```
A1 1 0 0
A2 0 1 0
A3 0.5 0.5 0
A4 0 0 1
A5 0.5 0 0.5
A6 0.5 0.25 0.25
<...>
A19 0.5 0.25 0.25
```

### 4.3.3 Syntax

```
REGFILE
<field animal> <field type I or A>
REG01 <file name REG01> !REGTYPE F/R/H [!IDCOL 1] [!STARTCOV 2] [!LASTCOV 7]
REG02 <file name REG02> !REGTYPE F/R/H [!IDCOL 1] [!STARTCOV 2] [!LASTCOV 7]
<...>
REG99 <file name REG99> !REGTYPE F/R/H [!IDCOL 1] [!STARTCOV 2] [!LASTCOV 7]

REGPARFILE
REG01 <file name REG01>
REG02 <file name REG02>
```

```
<...>
REG99 <file name REG99>

MODEL
trait ~ fixed !RANDOM REG(1,2..5)
```

Sections:

**REGFILE**
The REGFILE section specifies the name of one or more general covariate files and its attributes, such as column numbers and whether one variance for all covariates is used or an individual variance for each covariate.

**REGPARFILE**
The REGPARFILE section is used to specify a file with components of variance and covariance among traits associated with general covariates. A general covariate file labelled in REGFILE needs a corresponding entry in a REGPARFILE section if the regression type is R for random or H for heterogeneous variances.

There are no file-independent qualifiers. The file-dependent qualifiers of REGFILE can be specified for each covariate file. These qualifiers are:

**!REGTYPE**
The file-specification line must contain the !REGTYPE qualifier. It specifies how the covariates in the file are fitted in the model.

If 'f' is specified, the covariates in the file are fitted as a fixed regression. Covariates fitted as a fixed effect do not have a variance associated with it, so it is not necessary to specify a parameter file in the REGPARFILE section. If it is present, it is ignored.

If 'r' is specified, the covariates in the file are fitted as a random regression with a single variance for all covariates in the file. The variance is specified in the corresponding parameter file in the REGPARFILE section. If 'h' is specified, the covariates in the file are fitted as a random regression, each with their own variance. The covariate-specific variances are specified in the corresponding parameter file in the REGPARFILE section.

**!IDCOL**
The !IDCOL qualifier is optional and specifies which field in the covariate file contains the ID of the individual. If it is omitted, it is assumed that the ID is in the first field of the record (so the default is !IDCOL 1).

**!STARTCOV**
The !STARTCOV qualifier is optional and specifies which field contains the first covariate. If it is omitted, it is assumed that the covariates start in the second field of the record (so !STARTCOV 2).

**!LASTCOV**
The !LASTCOV qualifier is optional and specifies which field contains the last covariate of the file to include in the model. If it is omitted, it is assumed that all fields after the first covariate contain covariates to include in the model.

**REG(...)**
The REG function is used in the MODEL section and can be used to specify which general covariate files should be fitted in the model of a trait. If a covariate file is specified, then all specified covariates in the file will be fitted simultaneously.

The numbers in the REG(...) function link to the number in the label of the general covariate file in the REGFILE section (and the REGPARFILE section). The numbers may be specified individually as (1, 2, 3, 4) or as a range, indicated by two subsequent full stops, for example (1..4), or a combination of both.

If a covariate file is fitted for any trait through REG(...), the covariates will be fitted for all traits, even the ones for which REG(...) is not specified.





### 4.3.4 Associated output files

| Output file | Description |
|---|---|
| RegCov%%.txt | temporary file; data covariate file |
| RegCov%%NoDat.txt | temporary file; covariates of individuals without any phenotypes |
| Solreg_mat.txt | solutions of all covariates in any general or SNP covariate file |

# 5. Genetic similarity among individuals

Two individuals that have an ancestor in common are more similar than two unrelated individuals. This genetic similarity can be specified in various ways. This chapter describes the various methods in MiXBLUP to specify genetic similarity.

If only a pedigree is available, MiXBLUP will calculate the **expected genetic relationships** between individuals as they appear in the inverse pedigree relationship matrix ($A^{-1}$), without the need to specify this matrix explicitly (chapter 5.1 – 5.4). Alternatively, any existing inverse relationship matrix may be provided (chapter 5.5).

If some individuals were genotyped for many genetic markers, such as SNPs, MiXBLUP can be instructed to call calc_grm, which calculates the **estimated true genetic relationships** in a genomic relationship matrix and inverts it. This inverse genomic relationship matrix may be combined with pedigree information to analyse genotyped and non-genotyped individuals simultaneously (chapter 5.6 – 5.9).

An inverse relationship matrix can be provided in two formats, sparse and dense. Both only contain the lower triangular part of the matrix, as it is symmetrical in the diagonal. The evaluation is much faster if the genomic relationship matrix is stored in dense format. MiXBLUP automatically recognises the format of an existing inverse relationship matrix. When constructing an inverse relationship matrix using calc_grm, the default is to present it in dense format (chapter 5.6).

An equivalent method to use estimated true genetic relationships implicitly, without the need to construct and invert a genomic relationship matrix, is random **regression of all SNPs** simultaneously on the data (chapter 5.10 – 5.11).

## 5.1 Pedigree file, ignoring inbreeding and single code for unknown parents

### 5.1.1 General

Expected genetic similarity between individuals can be based on observed pedigree relationships. MiXBLUP supports analyses using a pedigree that consists of individuals and their parents (animal model). A sire model with sires and maternal grandsires in the pedigree file is currently not supported in MiXBLUP.

Any individual occurring in the data file, regardless whether with a record or as a maternal, paternal or group mate effect (in case of a social interaction model), must be present in the pedigree file. Any individual that does not appear in the data file, but exists as an ancestor in the pedigree file must also have its own record in the pedigree file.

It is inevitable that for at least some individuals in the pedigree, the parents are unknown. When using a single code for unknown parents, code zero (0) must be used.





The name of the pedigree file is specified in the PEDFILE section of the instruction file. The pedigree file is by default expected to be located in the active directory, but it can be in any other folder if the path is specified as part of the filename (e.g. d:\pedigrees\PedigreeBreedP.txt).

### 5.1.2 Input files

The pedigree file consists of the individual identification code (ID) and the IDs of its sire and dam in the first three columns. The columns must be separated by at least one space. The IDs in the pedigree file must be of same type as the IDs in the data file (either numeric or text). The pedigree file may contain other information in any number of additional columns, as long as the number of columns is the same for all records.

Calculating reliabilities requires a block variable to be present in the pedigree file (see Chapter 9). In that case the pedigree file, as well as the data file, will be sorted on the block variable. If a block group variable is added to the pedigree, it must be marked with the qualifier !BLOCK. It does not have to be in the fourth column, as in older versions of MiXBLUP. The pedigree file does not need to be sorted. MiXBLUP takes care of any required sorting.

*Example.* Pedigree file with a single code for unknown parents

```
A1 0 0
A2 0 0
A3 0 0
A4 0 0
A5 0 0
A6 0 0
A7 0 0
A8 0 0
A9 0 0
A10 0 0
A11 A1 0
A12 A2 A6
A13 A3 A7
A14 A4 A7
A15 A5 A8
A16 A1 A8
A17 A2 A9
A18 A3 A9
A19 A4 A10
```

### 5.1.3 Syntax

```
PEDFILE <pedigree file> [!SKIP <n lines>]
<field animal> <field type>
<field sire> <field type>
<field dam> <field type>
[<field block variable> <field type>] !BLOCK
```

Qualifiers:

**!SKIP <n lines>**
The SKIP qualifier may be used to skip the first n lines of the pedigree file. This is useful for ignoring a header.

**!BLOCK**
The BLOCK qualifier specifies the field that contains the equation family block variable (Chapter 9) in case of a reliability calculation. The block variable does not have to be in the fourth column.

### 5.1.4 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| Relani.txt | Approximate reliabilities when the field type of the ID is integer |
| Relani.out | Approximate reliabilities when the field type of the ID is alphanumerical |

## 5.2 Pedigree file with genetic groups for unknown parents

### 5.2.1 General

For pedigrees with a relatively small number of generations or a high proportion of individuals in each generation with unknown parents, it may be desirable to specify that some individuals with unknown parents are more similar than average. For example, in case of genetic selection, two individuals born in the same year are more similar than two individuals born in different years. In case of a large difference in selection differential between males and females, it may be useful to distinguish males and females born in the same year. In case of mixed-breed or mixed-line evaluations, it may be useful to group individuals by breed, line or type of cross. This can be done by assigning individuals with one or two unknown parents to an appropriate genetic (or phantom parent) group.

Genetic groups can be included in the analysis in two ways: (1) Westell grouping and (2) genetic group covariates. Westell grouping augments the pedigree relationship matrix with the number of genetic groups. For genetic group covariates, a covariate matrix Q is set up that contains the proportion of each genetic group for each animal. For both methods, the direct genetic solution includes the genetic group effect.

### 5.2.2 Input files

In the pedigree file, the genetic group of the individual is entered on the position of the unknown parent. Genetic groups must be coded as negative integers. It does not have to be in sequential order.

Genetic groups can be modelled either as fixed, pseudo-random (Westell grouping) or random effects. For Westell grouping, the specified value will be added to the diagonal elements of the genetic group effects in the inverse coefficient matrix. If a value of zero is added, genetic group effects are modelled as fixed effects. For values larger than zero, genetic groups are modelled as pseudo-random effects. The larger the value, the more estimates are regressed towards the mean. For genetic group covariates, a variance component can be specified for each genetic group covariate separately or one for all genetic group covariates. It is also possible to fit the covariates as fixed effects.

*Example.* Pedigree file with genetic groups for unknown parents

```
A1 -1 -1
A2 -1 -1
A3 -1 -1
A4 -2 -2
A5 -2 -2
A6 -35 -35
A7 -35 -35
A8 -17 -17
A9 -17 -17
A10 -17 -17
A11 A1 -2
A12 A2 A6
A13 A3 A7
A14 A4 A7
A15 A5 A8
A16 A1 A8
A17 A2 A9
A18 A3 A9
A19 A4 A10
```





### 5.2.3 Syntax of inclusion of genetic groups through Westell grouping

```
PEDFILE <pedigree file> [!GROUPS <value>]
<field animal> <field type>
<field sire> <field type>
<field dam> <field type>
```

Qualifier:

**!Groups <value>**
The qualifier GROUPS means that genetic groups are included in the pedigree. Genetic groups need to be coded with negative integer values. With <value>, it is possible to specify whether these Genetic group effects should be modelled as fixed (value = 0.0) or as random (value > 0.0). In practice, !GROUPS does not need to be set at a much higher value than about 3.

### 5.2.4 Syntax of inclusion of genetic groups through covariates

```
PEDFILE <pedigree file> !MAKEGGCOV
<field animal> <field type>
<field sire> <field type>
<field dam> <field type>

REGFILE
<field animal> <field type I or A>
    REG01 !GGCOV !REGTYPE F/R/H

[REGPARFILE]
[ REG01 <file name REG01>]

MODEL
    <trait> ~ <fixed effects> !RANDOM REG(1) <other random effects>
```

Qualifier:

**!MakeGGcov**
The qualifier !MakeGGcov triggers MiXBLUP to set up a covariate matrix Q of the number of genetic groups by the number of individuals in the analysis. The covariates are stored in a standard covariate file.

**!GGcov**
The qualifier !Ggcov specifies which external covariate file contains genetic group covariates. If !MakeGGcov is specified, there is no need to specify a file name for the covariate file with !Ggcov

**REG(...)**
The REG function can be used to fit a genetic group covariate file in the model of a trait. If the genetic group covariate file is fitted for any trait through REG(...), the covariates will be fitted for all traits, even the ones for which REG(...) is not specified.

The numbers in the REG(...) function link to the number in the label of the general covariate file in the REGFILE section (and the REGPARFILE section). The numbers may be specified individually as (1, 2, 3, 4) or as a range, indicated by two subsequent full stops, for example (1..4), or a combination of both.

### 5.2.5 Associated output files for Westell grouping

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect including the genetic group effects when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect including the genetic group effects when the field type of the ID is alphanumerical |
| Relani.txt | Approximate reliabilities when the field type of the ID is integer |
| Relani.out | Approximate reliabilities when the field type of the ID is alphanumerical |

### 5.2.6 Associated output files for genetic group covariates

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| SolaniGG.txt | Solutions of the accumulated genetic group effects for each individual |
| Solanitot.txt | Solutions of the direct genetic effect including the genetic group effects when the field type of the ID is integer |
| GeneticGroupsInQ.txt | Original genetic group label by column in the covariate file |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| Solreg_mat.txt | Solutions of genetic group covariates, along with solutions of any other external covariates. |
| SolaniGG.out | Solutions of the accumulated genetic group effects for each individual |
| Solanitot.out | Solutions of the direct genetic effect including the genetic group effects when the field type of the ID is alphanumerical |
| Relani.txt | Approximate reliabilities when the field type of the ID is integer |
| Relani.out | Approximate reliabilities when the field type of the ID is alphanumerical |

## 5.3 Pedigree file accounting for inbreeding

### 5.3.1 General

Inbreeding coefficients are often ignored in breeding value estimation using pedigree relationships only. The internally calculated numerator relationship matrix ($A^{-1}$) is by default set up without taking into account inbreeding. Inbreeding can be included by providing the kernel with a file with the inbreeding coefficient of each individual in the pedigree file. This file may be provided as an existing input file or calculated within MiXBLUP as a preparation step.

Note that inbreeding coefficients do not affect the reliability calculation and will be ignored.

### 5.3.2 Input files

There are no additional requirements of the pedigree file for the calculation of inbreeding coefficients by MiXBLUP.

To use previously calculated inbreeding coefficients, any free-format text file with any number of columns can be used as long as it contains the ID of each individual in the analysis and its inbreeding coefficient. This may be the pedigree file with an additional column of inbreeding coefficients.





*Example.* File with inbreeding coefficients

```
A1 0.00
A2 0.00
A3 0.00
A4 0.00
<..>
A20 0.125
A21 0.0625
```

### 5.3.3 Syntax of calculating inbreeding coefficients in MiXBLUP:

```
PEDFILE <pedigree file> [!CALCINBR]
<field animal> <field type>
<field sire> <field type>
<field dam> <field type>
```

Qualifier:

**!CALCINBR**
The qualifier CALCINBR is optional and is used to indicate that inbreeding coefficients should be calculated and included in the calculation of the inverse pedigree relationship matrix ($A^{-1}$). If !CALCINBR has been specified, the section INBRFILE is ignored. The default setting is that inbreeding coefficients are not taken into account when setting up the inverse pedigree relationship matrix.

### 5.3.4 Syntax of using externally calculated inbreeding coefficients:

```
INBRFILE <inbreeding coefficient file > [!IDCOL <field number>] [!INBRCOL <field number>]
```

Qualifier:

**!IDCOL <value>**
The optional qualifier !IDCOL can be used to specify the field number in the inbreeding coefficient file that contains the animal ID. The default field number is 1.

**!INBRCOL <value>**
The optional qualifier !INBRCOL can be used to specify the field number in the inbreeding coefficient file that contains the inbreeding coefficient. The default field number is 4.

### 5.3.5 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| inbreeding.txt | Inbreeding coefficients calculated from pedigree relationships |
| inbreeding.out | Inbreeding coefficients calculated from pedigree relationships for alphanumerical IDs |

## 5.4 Pedigree file and marker haplotypes (marker-assisted BLUP)

### 5.4.1 General

In a marker-assisted BLUP model, genetic markers of a QTL (quantitative-trait locus) are fitted in addition to a polygenic effect. There are two ways to fit genetic markers of a QTL.

The first method fits the two marker haplotypes of an individual at the QTL. This method is suitable for any number of haplotypes in the population at this QTL. It is described in this section.

The alternative method is only suitable if there are exactly two haplotypes segregating in the population at this QTL. The marker alleles need to be converted to the number of copies of one of the two haplotypes at the QTL. This number can be fitted as a fixed or random covariate. The marker covariate may be placed in the data file or provided in an external SNP covariate file.

### 5.4.2 Input files

For marker-assisted BLUP, both the pedigree file and the data file should contain the marker haplotypes, two columns for each marker. Corresponding haplotype fields in the data and pedigree file have the same field name.

*Example.* Pedigree file with haplotypes of a single marker

```
A1    0   0     2    1
A2    0   0     2    2
A3    0   0     1    1
A4    0   0     2    2
A5    0   0     1    2
A6    0   0     1    2
A7   A1  A3     1    1
A8   A1  A4     1    2
A9   A2  A5     2    1
A10  A2  A6     2    1
```

In addition, a file with the inverse of the variance-covariance matrix between haplotypes should be provided for each marker. This file should contain all non-zero elements and be constructed as: haplotype ID of row, haplotype ID of column, inverse-matrix element. The order and numbers used as row and column numbers should correspond to the haplotype numbers used in the data and pedigree file. Haplotype IDs must be integer. The example below gives the inverse IBD matrix for the general example with only two haplotypes.

*Example.* The inverse variance-covariance relationship matrix (inverse IBD matrix) of two haplotypes that have a relationship of 0.25 amongst each other. Columns: haplotype ID of row, haplotype ID of column, inverse-matrix element.

```
1 1 1.06
1 2 -0.26
2 2 1.06
```

If the marker haplotypes are fitted as random effects, some changes to the file with variances and covariances between traits (parameter file, see also Chapter 6) are required, too. It is strongly recommended to use the lower-triangular-matrix format for marker-assisted BLUP. A matrix has to be added for every marker. The label of the matrix is GIV followed by the number of the marker in the analysis. See Example 5.4 in the Appendix.





### 5.4.3 Syntax

```
DATAFILE <data file>
...
<field haplotype 1 marker 1> I
<field haplotype 2 marker 1> I
...

PEDFILE <pedigree file>
...
<field haplotype 1 marker 1> I
<field haplotype 2 marker 1> I

CVMATRIX
    <variance-covariance matrix file of haplotypes of marker 1>

MODEL
    <trait> ~ <fixed effects> !RANDOM GIV(<field haplotype 1 marker 1> AND <field haplotype
2 marker 1>,1) <random effects>
```

**GIV(.., ..)**
The function GIV(...) in the MODEL section links the fields in the data and pedigree file to the variance-covariance matrix of the corresponding marker.
**AND**
The function AND combines the incidence matrices of the two haplotype fields.

### 5.4.4 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| EBVhap<num> | Solutions of the haplotypes at fitted marker <num> |
| EBVtot | Combined solutions of direct genetic effects and marker haplotype solutions |
| EBVhap<num>.out | Solutions of the haplotypes at fitted marker <num> for alphanumerical IDs |
| EBVtot.out | Combined solutions of direct genetic effects and marker haplotype solutions |

## 5.5 Existing external relationship matrix file

### 5.5.1 General

A range of inverse genetic relationship matrix files can be created by MiXBLUP explicitly or are implicitly incorporated in the analysis. It is also possible to use a previously created inverse genetic relationship matrix or one that as yet cannot be created by MiXBLUP itself.

It is not possible to calculate reliabilities with MiXBLUP when using a full external relationship matrix.

### 5.5.2 Input files

The external relationship matrix file name is specified in the instruction file and may be anywhere on a system, provided the full path is part of the file name.

The external relationship matrix file contains all non-zero elements of the matrix. Each line consists at least of three fields: original individual ID of row, original individual ID of column, matrix element. Any other fields on the line are ignored.

*Example.* Columns in external inverse relationship matrix file: animal ID row, animal ID column, matrix element.

```
A1  A1  0.75
A2  A1  -0.5
A2  A2  2
A3  A1  -0.5
A3  A2  -0.999999999
A3  A3  2
<...>
A19 A19 3.97222222
```

### 5.5.3 Syntax

```
ERMFILE <external relationship matrix file> [!SKIP <n lines>] [!ASIS] [!NOORIG]
<field individual ID> <field type>
```

Qualifier:

**!SKIP <n lines>**
The optional !SKIP qualifier may be used to skip the first n lines of the external relationship matrix file. This is useful for ignoring a header line.

**!ASIS**
The !AsIs qualifier is optional. It is used to write the external inverse relationship matrix to the kernel without any checks or sorting. This can be specified if the external relationship matrix file is known to be correct, for example because it was created or checked by MiXBLUP in a previous run. The !AsIs qualifier can only be used if the field type of the individual ID is integer. It is ignored when individual ID has alphanumerical field type.

**!NOORIG**
By default, MiXBLUP creates a copy of the external inverse relationship matrix file that is checked and made lower-triangular, ExtRelMatOrig.txt. If this file is not needed for additional analyses, the !NoOrig qualifier can be specified. Especially for very large analyses, the size of this file can be substantial.

### 5.5.4 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| ExtRelMatOrig.txt | Verified and lower-triangular version of external relationship matrix |

## 5.6 Genomic relationship matrix file calculated from genotype file (GBLUP)

### 5.6.1 General

The external inverse genomic relationship matrix ($G^{-1}$) can also be calculated by MiXBLUP using the calc_grm software. The G-inverse is calculated using only a genetic marker file. For a GBLUP analysis, all individuals with phenotypes must have a genotype record.

An allele frequency file can be specified to use pre-defined allele frequencies instead of data-derived allele frequencies for the entire population.

Alternatively, a breed composition file can be specified to calculate allele frequencies within subpopulations of breeds and crosses.





For a GBLUP analysis, it is possible to estimate the effects of individual genetic markers by backsolving.

### 5.6.2 Input genotype file

The name of the genetic marker file is specified in the instruction file. The path to the file may be specified as part of the file name. The genetic marker file contains the original animal IDs. The genetic marker file must contain the animal ID in the first column and genetic marker data from the second column onwards. The animal ID and the genetic marker data must be separated by at least one space.

A range of formats of the genetic marker file is supported. The file may contain marker alleles or marker genotypes. The pairs of marker alleles may be on the same line or on two different lines. The marker alleles or genotypes may be in space-separated or in dense format. In dense format, every digit is a marker. The genomic data cannot be partly dense and partly space-separated.

The calc_grm software offers flexibility with regard to the method used, the use of pre-defined or data-derived allele frequencies and the use of multiple breeds in the analysis.

*Example.* Genotype file with genomic data presented in six pairs of alleles per animal in space-separated format.

```
A1  2 2 1 2 1 1 1 1 1 1 1 1
A2  1 2 1 2 1 1 1 2 1 1 1 1
A3  1 2 1 1 2 2 2 2 2 2 1 2
A4  1 1 1 2 1 2 2 2 2 2 2 2
A5  1 1 1 1 2 2 1 2 1 2 1 2
<...>
A19 1 1 1 1 1 2 2 2 2 2 2 2
```

*Example.* Genotype file with genomic data presented in two lines of alleles per animal in dense format.

```
A1 211111
A1 221111
A2 111111
A2 221211
A3 112221
A3 212222
<...>
A19 111222
A19 112222
```

*Example.* Genotype file with marker genotype data per animal in dense format. It contains the number of copies per locus of the allele with the highest number (11=0, 12=1 and 22=2).

```
A1 210000
A2 110100
A3 102221
A4 011222
A5 002111
<...>
A19 001222
```

### 5.6.3 Input allele frequency file

If the user does not want to use allele frequencies calculated from the data, then pre-calculated allele frequencies can be supplied as an additional input file, The file specified should contain for each locus the allele frequency of the allele with the highest integer code, if the genetic marker file contains alleles. The file specified should contain for each locus the frequency of

the allele of which the homozygote genotypes are coded as 2. The structure of the file is <locus number in order of the genetic marker file> <allele frequency>.

*Example.* Pre-calculated allele frequency per locus of allele coded as 1 for 6 loci.

```
1 0.234
2 0.452
3 0.178
4 0.842
5 0.541
6 0.609
```

### 5.6.4 Input breed composition file

The breed composition file contains the original animal ID in the first column and contains a number of additional columns that is equal to the number of pure breeds or lines specified. The breed composition may be presented as a number, for example 4 (out of 8 or any other number), as a percentage, for example 50, or as a fraction, for example 0.50. MiXBLUP converts the breed composition of an animal to the value of one breed over the sum of values across breeds. For example in an analysis with four breeds, animal X having 4 0 2 2 as the breed composition will be converted to X 0.500 0.000 0.250 0.250. It is therefore essential that the breed information is complete, so add a column for 'unknown or other', if necessary. All columns must be separated by at least one space.

*Example.* Breed composition file with the percentage of four breeds per animal.

```
A1 100 0 0 0
A2 100 0 0 0
A3 0 100 0 0
A4 0 100 0 0
A5 0 0 100 0
<...>
A19 0 50 0 50
```

*Example.* Breed composition file in parts of one eighth of four breeds per animal.

```
A1 8 0 0 0
A2 8 0 0 0
A3 0 8 0 0
A4 0 8 0 0
A5 0 0 8 0
<...>
A19 0 4 0 4
```

### 5.6.5 Output format of relationship matrix file

The relationship matrix file can be written in sparse or dense output format. In sparse format, each line contains a non-zero element with row identification and column identification. To use this I-J-Value format, specify !USE_IJV.





*Example.* Inverse relationship matrix in sparse format

```
11 11 2.78032302891924
12 11 -0.138473072826274
12 12 2.36875077052943
11 13 0.128605663447062
12 13 -1.49490191616651
13 13 2.26451722336854
11 14 -1.86510841639899
12 14 0.290914677767982
14 13 0.109700495017635
14 14 3.62322893561911
15 11 -0.490544232314569
12 15 -0.534909344684323
15 13 -0.308478256559346
15 14 -0.172777975870294
15 15 7.00033112920426
<...>
11 20 1.27322033536996
12 20 -0.670145770881933
13 20 -0.418678898666513
14 20 -2.40844620890414
15 20 -1.93851549770915
16 20 -1.95915331803544
20 17 -0.204535663379337
18 20 -1.59532493107000
19 20 3.82736754320307
20 20 4.07483496518577
```

In dense format, the first line contains the number of genotyped individuals and the number of individuals in the core of the APY inverse of the genomic relationship matrix, the second line the row identification of each row and the third and subsequent lines contain the elements up to the diagonal element of the row. The dense output format is the default.

```
10 0
11 12 13 14 15 16 17 18 19 20
2.7803230
-0.13847307 2.3687508
0.12860566 -1.4949019 2.2645172
-1.8651084 0.29091468 0.10970050 3.6232289
-0.49054423 -0.53490934 -0.30847826 -0.17277798 7.0003311
-2.9381482 0.13534720 -0.71050168 3.0768518 -0.15904779 3.3119023
-0.81743067 1.4382225 1.8203113 -1.5904384 0.54734675E-01 -1.0537561 4.5018664
<...>
```

### 5.6.6 Syntax

```
ERMFILE <Name file with genetic markers> !CONSTRUCT Ginv
<animal ID> <field type>
!METHOD <Yang, VanRaden or VanRaden2> (optional; default VanRaden2)
!ALFREQ <file name> (optional; default calculated from data)
!CROSSBRED < number of breeds> <file name> (optional; default single breed)
!BREEDS_UNRELATED (optional; default all genomic relationships are considered)
!ALLELES <number of records per animal> (optional; default genotypes)
!INFORMATIVE (optional; default is to use all SNPs)
!DENSE n(optional; default string of markers in free format, starting with column n)
!NMARK <number of markers to be read> (optional; default is all markers)
!MAF <minimum allele frequency> (optional; default is 0.005)
!STORE_GINV (optional; default is no storing)
!NUMPROC <number of processors to be used by calc_grm> (optional; default is 1)
!ZEROG <threshold value> (optional; default is no change of values to zero)
!FORM_IJV (optional; default is to use the dense format to write the relationship matrix)
!SKIP <n lines> (optional; default is reading all lines)
!GFROMDISK (optional; default is to store relationship matrix in memory during solving)
!BACKSOLVE (optional; default is no backsolving)
```

Qualifiers:

**!CONSTRUCT Ginv**
The !CONSTRUCT qualifier is optional and indicates that the external relationship matrix has not been calculated yet and needs to be calculated in the MiXBLUP parser. For a GBLUP analysis, the argument of !CONSTRUCT is Ginv, for an inverse genomic relationship matrix.

**!METHOD <Yang, VanRaden or VanRaden2>**
The !METHOD qualifier is optional and specifies whether the method of Yang (Nat Genet 42:565-569) or the method of VanRaden (J Dairy Sci 91: 4414-4423) is used. The VanRaden2 method is the default.

**!ALFREQ <file name>**
The !ALFREQ qualifier is optional and allows the use of pre-defined allele frequencies per locus from the file specified. By default, the allele frequencies are calculated from the data.

**!CROSSBRED <number of breeds> <name file with breed composition per animal>**
The !CROSSBRED qualifier is optional and can be used for multi-breed analyses that may or may not include crossbred animals. There are two arguments. The one argument is the number of pure breeds in the analysis. The other argument is the name of the file with the breed composition of the animals in the genetic marker file.

The !CROSSBRED option will consider relationships between all animals, regardless of their breed composition, using for each animal allele frequencies that are specific for their breed composition.

**!BREEDS_UNRELATED**
The !BREEDS_UNRELATED option can be used to set relationships between animals of a different breed to zero, despite of any genomic relationship there may be. The default is that all genomic relationships, regardless of the breed or line of origin, are considered. The !BREEDS_UNRELATED option only has a meaning in conjunction with !CROSSBRED.

**!ALLELES <1 or 2 > # records per animal**
The !ALLELES option is optional and is required if the genetic marker data contains alleles. The base pairs A-T and T-A should be coded as the same allele and so should be C-G and G-C. Alleles may be presented in pairs (argument is 1; one line per animal) or on two lines per animal (argument is 2). Alleles must be presented as 1 or 2, and missing alleles must be coded as 0 (zero). SNP alleles must be integer.

The !ALLELES qualifier should be omitted if the genetic marker data is presented as genotypes (the number of copies of one of the alleles). SNPs are presented as genotypes if the number of copies of one of the two alleles is provided, so 11 becomes 0 (or -1), 12 becomes 1 (or 0) and 22 becomes 2 (or 1) if the number of copies of SNP allele 2 is counted. So SNP genotypes must be presented as either -1,0,1 or 0,1,2, and missing genotypes must have a value greater than 2. SNP genotypes may be presented as real values.

**!INFORMATIVE**
The !INFORMATIVE qualifier is used to include only genetic markers with all three genotypes present in the population of genotyped individuals. The default is to include all genetic markers.

**!DENSE [<field number of dense column>]**
The !DENSE qualifier must be specified if the genetic marker data is presented as a sequence of genetic markers without spaces. If the dense column is not the second field in the record, the field number of the dense column needs to be specified after the qualifier, for example !DENSE 4. If !DENSE is not specified for a file with dense genetic marker data, MiXBLUP will give a column-width error, as it attempts to read the dense genetic markers as a single column. By default, MiXBLUP expects space-separated genetic markers.

**!NMARK <number of markers to be included in the analysis>**
The qualifier !NMARK is optional and can be used to analyse a smaller number of genetic markers than are present in the genetic marker file. If a value of N is specified as the argument of NMARK, then the first N genetic markers are included in the analysis. If not specified, all markers are initially included in the analysis, but some may drop out because of being insufficiently informative (less than three genotypes in the population or minor allele frequency below the minimum).

**!MAF <minimum allele frequency>**
The qualifier !MAF is optional and is used to set the minimum allele frequency of genetic markers to be included in the analysis. The default value is 0.005.





**!STORE_GINV**
The qualifier !STORE_GINV is optional and allows the user to save the inverted G matrix in the right format to be re-used by calc_grm. The default name is G_asreml.giv.

**!NUMPROC**
The !NUMPROC qualifier can be used to specify the number of threads to be used by calc_grm.

**!ZEROG <threshold value>**
The !ZEROG qualifier can be used to convert values of the inverse matrix on output to zero if below the threshold value. This increases the sparsity of the inverse matrix somewhat and may be beneficial for reducing time per iteration during solving, when !USE_IJV is specified, as elements of zero are not written or read. If !USE_IJV is not specified, !ZEROG has no effect.

**!FORM_IJV**
Use the sparse output format to write the inverse relationship matrix. The !FORM_IJV qualifier is optional. Dense output format is default.

**!BACKSOLVE**
The !BACKSOLVE qualifier triggers the calculation of the solutions of individual genetic markers from the solutions of genotyped animals. !BACKSOLVE may also be specified in the SOLVING section.

**!GFROMDISK**
The !GFROMDISK qualifier instructs the solver to read the inverse genomic relationship matrix from disk during solving. This was the only option in previous versions of MiXBLUP. The new default is to keep this matrix in memory, which is more demanding for memory requirement, but it saves the time to read this matrix every iteration. It is specified in the SOLVING section of the MiXBLUP instruction file.

### 5.6.7 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| ExtRelMatOrig.txt | Verified and lower-triangular version of external relationship matrix |
| high_grm_coefs.log | Pairs of individuals with a very high genomic relatonship |
| ERMcalc_grm.log | Log file of calc_grm |
| calculated_all_freq.dat | Allele frequencies calculated from the data |
| genomic_inbr_coef.dat | Genomic inbreeding coefficients |
| G.grm | Genomic relationship matrix before inversion |
| ExtRelMat.txt | Inverted genomic relationship matrix |
| genotype_statistics.txt | Descriptive statistics of SNP by position in the genotype file |
| SNP-effects.txt | Solutions of genetic markers calculated from solutions of genotyped animals |

## 5.7 Blended genomic and pedigree relationship matrix from genotype and pedigree file (ssGBLUP)

### 5.7.1 General

A single-step GBLUP (ssGBLUP) model can be used if phenotypes are available for both genotyped and non-genotyped individuals. For a ssGBLUP analysis, genomic and pedigree information is blended. This may be done explicitly by setting up the entire blended inverse genomic and pedigree relationship matrix ($H^{-1}$) or implicitly by allowing the kernel to set up the parts of $H^{-1}$ that relate to non-genotyped animals. In the latter case, only a weighted inverse genomic relationship matrix $\left(\lambda(\alpha G + \beta A_{22})^{-1} - \omega A_{22}^{-1}\right)$ is passed to the kernel, which is more efficient than passing on the entire $H^{-1}$. Passing on the entire $H^{-1}$ is only necessary if the kernel cannot calculate the parts of the pedigree relationship matrix correctly, for example in case of selfing or double haploids. Approximate genomic reliabilities can only be calculated when using a weighted inverse genomic relationship matrix.

### 5.7.2 Input files

Both options require a pedigree file in addition to the genotype file. The requirements of the pedigree file are outlined in chapter 5.1. The requirements of the genotype file are described in the previous section, chapter 5.6.

The method using a weighted inverse genomic relationship also requires a file with inbreeding coefficients to be present. The requirements of this file are outlined in chapter 5.3.

### 5.7.3 Syntax full blended inverse relationship matrix

```
ERMFILE <Name file with genetic markers> !CONSTRUCT Hinv
<individual ID> <field type>
!LAMBDA <weighting factor G matrix, 0.0-1.0> (optional; default 1.0)
!ALPHA <weighting factor, 0.0-1.0> (optional; default is 1.0)
!BETA <weighting factor, 0.0-1.0> (optional; default is 0.0)
!OMEGA <weighting factor, 0.0-1.0> (optional; default is LAMBDA)
!USE_GINV <file name>

ERMPEDFILE <pedigree file>
<individual ID> <field type>
```

Any qualifier of ERMFILE described in chapter 5.6 can also be used for a full blended inverse relationship matrix. Specific qualifiers:

**!CONSTRUCT Hinv**
The !CONSTRUCT qualifier is optional and indicates that the external relationship matrix has not been calculated yet and needs to be calculated in the MiXBLUP parser. For a ssGBLUP analysis with a full blended inverse relationship matrix, the argument of !CONSTRUCT is Hinv.

**!LAMBDA <weighting factor of G-matrix>**
**!ALPHA <weighting factor of G-matrix>**
**!BETA <weighting factor of G-matrix>**
**!OMEGA <weighting factor of G-matrix>**
The !LAMBDA, !ALPHA, !BETA and !OMEGA qualifiers are the fudge parameters suggested by Misztal and co-workers. They are optional and can be used to give more weight to numerator relationship matrix ($A^{-1}$) in the construction of the blended matrix ($H^{-1}$):

$$H^{-1} = A^{-1} + \begin{bmatrix} 0 & 0 \\ 0 & \lambda(\alpha G + \beta A_{22})^{-1} - \omega A_{22}^{-1} \end{bmatrix}$$

By default, lambda is set to 1, omega to lambda, alpha to 1 and beta to 0.

**!USE_GINV <file name>**
The qualifier !USE_GINV is optional and can be used to avoid the repeated inversion of the G-matrix and use an existing G-inverse in the specified file for calculating the H-inverse. For repeated applications of the same G-inverse, this saves a substantial amount of time.

### 5.7.4 Syntax using a new weighted inverse genomic relationship matrix

```
ERMFILE <Name file with genetic markers> !CONSTRUCT SSmat
<animal ID> <field type>
!SINGLESTEP

[INBRFILE <inbreeding coefficient file> !IDCOL <number> !INBRCOL <number>]

PEDFILE <pedigree file> [!CALCINBR]
<individual ID> <field type>
<sire ID> <field type>
<dam ID> <field type>
```





Any qualifier of ERMFILE described in chapter 5.6 and chapter 5.7.3 can also be used for a weighted inverse genomic relationship matrix. Specific qualifiers:

**!CONSTRUCT SSmat**
The !CONSTRUCT qualifier is optional and indicates that the external relationship matrix has not been calculated yet and needs to be calculated in the MiXBLUP parser. For a ssGBLUP analysis with a weighted inverse genomic relationship matrix, the argument of !CONSTRUCT is SSmat.

**!SINGLESTEP**
The qualifier !SINGLESTEP is optional and can be used to indicate that the MiXBLUP kernel should calculate the H-inverse from a G-inverse, the pedigree file and a file with inbreeding coefficients (see 4.9.3). This option is potentially more efficient on memory requirements and it may yield faster convergence in specific cases. This option requires a matrix that is set up using !CONSTRUCT SSmat in the current or a previous run.

### 5.7.5 Syntax using an existing weighted inverse genomic relationship matrix

```
ERMFILE <Name file with existing matrix >
<animal ID> <field type>
!SINGLESTEP

[INBRFILE <inbreeding coefficient file> !IDCOL <number> !INBRCOL <number>]

PEDFILE <pedigree file> [!CALCINBR]
<individual ID> <field type>
<sire ID> <field type>
<dam ID> <field type>
```

Only the !SINGLESTEP qualifier is meaningful in this context. The name of the file is typically ExtRelMatOrig.txt (or ExtRelMat.txt in case of integer IDs). The original name of this file as created by calc_grm is G_A22.txt.

### 5.7.6 Associated output files full blended inverse relationship matrix

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| ExtRelMat.txt | Full blended inverse relationship matrix |

### 5.7.7 Associated output files weighted inverse genomic relationship matrix

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| Relani.txt | Approximate reliabilities when the field type of the ID is integer |
| Relani.out | Approximate reliabilities when the field type of the ID is alphanumerical |
| ExtRelMat.txt | Weighted inverse genomic relationship matrix |

## 5.8 Blended genomic and pedigree relationship matrix using genetic groups

### 5.8.1 General

Genetic groups can only be used in ssGBLUP models in conjunction with the weighted inverse genomic relationship matrix. Genetic groups can be fitted either in the weighted genomic relationship matrix or as covariates.

### 5.8.2 Input files

The requirements of the genotype, inbreeding and pedigree file are described in chapter 5.7. Genetic groups must be presented as negative integers (see chapter 5.2).

### 5.8.3 Syntax

```
ERMFILE <Name file with genetic markers> !CONSTRUCT SSmat !SINGLESTEP
<animal ID> <field type>

[INBRFILE <inbreeding coefficient file> !IDCOL <number> !INBRCOL <number>]

PEDFILE <pedigree file> [!CALCINBR] !GROUPS <value>
<individual ID> <field type>
<sire ID> <field type>
<dam ID> <field type>
```

The !GROUPS qualifier is used as described in chapter 5.2.3.Either the INBRFILE section or !CALCINBR must be used, but not both. See chapter 5.3.For the syntax of fitting genetic groups as covariates, see chapter 5.2.4.

### 5.8.4 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| Relani.txt | Approximate reliabilities when the field type of the ID is integer |
| Relani.out | Approximate reliabilities when the field type of the ID is alphanumerical |
| ExtRelMat.txt | Weighted inverse genomic relationship matrix |

## 5.9 Using APY to invert genomic relationship matrix

### 5.9.1 General

The use of an inverse genomic relationship matrix requires inverting a matrix with dimensions equal to the number of genotyped individuals. For numbers of genotyped animals exceeding 50,000 to 100,000, this becomes quite a computational burden. The so-called algorithm for proven and young animals (APY) uses genomic recursions to calculate an approximate inverse of the genomic relationship matrix (Fragomeni et al., 2015).

For APY, the genotyped animals are divided into core and non-core animals. A target number of core animals can be the number of eigenvalues that explain at least 98% of the variation. This number of core animals can be chosen at random or supplied in a pre-defined list of core animals. Only the genomic relationship matrix of core animals needs to be inverted. The parts of the inverse genomic relationship matrix that relates to non-core animals are set up using genomic recursions.





The blended inverse genomic and pedigree relationship matrix also requires the inverse of the part of the A matrix that relates to the genotyped animals ($A_{22}$). This matrix has the same dimensions as G and is also demanding to invert. To overcome this issue, the kernel can be instructed to circumvent the need to invert $A_{22}$.

APY can only be used in combination with a pedigree containing genetic groups, if the genetic groups are fitted as covariates (see chapter 5.2.4).

### 5.9.2 Input files

In addition to a genetic marker file (see chapter 5.6) and a pedigree file (see chapter 5.1), the user may opt to supply a pre-defined list of core animals. This file contains at least the original ID of the core animal in the first column. Any other columns are ignored.

### 5.9.3 Syntax using a new APY inverse of genomic relationship matrix using a predefined number of random core animals

```
ERMFILE <Name file with genetic markers> !CONSTRUCT SSmat
<animal ID> <field type>
!SINGLESTEP !APY !APYCoreRan <number>
```

Qualifiers:

**!APY**
The qualifier !APY creates an approximate inverse of the genomic relationship matrix using genomic recursions. The approximation of the inverse matrix passed on to the kernel is $\lambda(\omega G + \beta A_{22})^{-1}_{APY}$. The need to invert $A_{22}$ is circumvented during solving. The weighting factor omega still applies.

**!APYCORERAN <number>**
The !APYCoreRan qualifier is used to randomly choose the specified number of individuals from the population of genotyped individuals to be included in the group of core individuals.

### 5.9.4 Syntax using a new APY inverse of genomic relationship matrix using a predefined list of core animals

```
ERMFILE <Name file with genetic markers> !CONSTRUCT SSmat
<animal ID> <field type>
!SINGLESTEP !APY !APYCoreLis <file name>
```

Qualifier:

**!APYCORELIS <file name>**
The !APYCoreLis qualifier is used to use a predefined list of genotyped individuals as core individuals.

### 5.9.5 Syntax using a new APY inverse of genomic relationship matrix using a number of random core animals determined by PCA

```
ERMFILE <Name file with genetic markers> !CONSTRUCT SSmat
<animal ID> <field type>
!SINGLESTEP !APY !APYPCA <target percentage of variation explained>
```

Qualifier:

**!APYPCA <target percentage of variation explained>**
The !APYPCA qualifier is used to determine the number of random core animals from the number of eigenvalues that explains the target proportion of variation among SNPs. MiXBLUP continues by randomly choosing this number of core animals to calculate the APY-inverse of G.

### 5.9.6 Syntax using an existing APY inverse of genomic relationship matrix

```
ERMFILE <Name file with existing matrix>
<animal ID> <field type>
!SINGLESTEP !APY
```

Only the !SINGLESTEP and !APY qualifiers are meaningful in this context. The name of the file is typically ExtRelMatOrig.txt (or ExtRelMat.txt in case of integer IDs). The original name of this file as created by calc_grm is gapy.dat.

### 5.9.7 Syntax using a new weighted APY inverse of genomic relationship matrix

```
ERMFILE <Name file with genetic markers> !CONSTRUCT SSmat
<animal ID> <field type>
!SINGLESTEP !APY_A22 [!APYCoreRan <value> or !APYCoreLis <file name>]
```

Qualifier:

**!APY_A22**
The !APY_A22 qualifier is used to calculate the APY inverse of the genomic relationship matrix minus the inverse of $A_{22}$. The output matrix is (lambda(alpha*G+beta*Asubscript(22))superscript(-1)subscript(APY) - omega*Asubscript(22)superscript(-1) Formule is als de formule in 5.7.1, maar dan met APY als subscript onder de eerste superscript -1.). The difference between !APY_A22 an !APY is that the inverse of A22 is explicitly calculated prior to solving in the former case and implicitly calculated during solving in the latter case. !APY is considered to be the most efficient option.

### 5.9.8 Syntax using an existing weighted APY inverse of genomic relationship matrix

```
ERMFILE <Name file with existing matrix>
<animal ID> <field type>
!SINGLESTEP !APY_A22
```

Only the !SINGLESTEP and !APY_A22 qualifiers are meaningful in this context. The name of the file is typically ExtRelMatOrig.txt (or ExtRelMat.txt in case of integer IDs). The original name of this file as created by calc_grm is G_apy_A22.dat.

### 5.9.6 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect when the field type of the ID is integer |
| Solani.out | Solutions of the direct genetic effect when the field type of the ID is alphanumerical |
| ExtRelMat.txt | (Weighted) inverse genomic relationship matrix |
| corelist.dat | List of randomly chosen genotyped individuals for the core group |





## 5.10 Using SNP covariates of genotyped animals (RRBLUP)

### 5.10.1 General

RRBLUP (SNP-BLUP) is a means to regress performance of an individual on the number of copies of a specific SNP allele at a very large number of loci. Because of the large number of loci, it is easier to provide the SNP covariates in a separate file to the data file. Two additional types of files are required for RRBLUP. These are the SNP covariate file and the SNP parameter file.

An alternative method to estimate genomic breeding values is using a RRBLUP model. In a RRBLUP model, the direct genetic effect is modelled with a random regression on number of copies of an SNP allele for a large number of loci. There is no need to fit a polygenic direct genetic effect, as well. In such an analysis, only genotyped individuals with data can be included in the analysis. It is possible to estimate the breeding values for genotyped animals without data from the marker effect solutions.

### 5.10.2 Input files

The SNP covariate file contains at least the ID of the genotyped animal and the number of copies per locus of a specific SNP allele, so 0, 1 or 2. All values other than 0, 1 or 2 are treated as missing SNP markers and marked for automatic imputation.

The SNP covariates may be provided in dense format or in space-separated format. If the SNP covariates contain imputed SNPs, the SNP covariates must be in space-separated format, as the imputed genotype will generally consist of more than one digit.

All SNP covariates are read as real numbers, regardless of whether a decimal point is present in the SNP covariate file.

*Example.* SNP covariate file with SNP marker data per animal in dense format. It contains the number of copies per locus of the allele with the highest number (11=0, 12=1 and 22=2).

```
A1 210000
A2 110100
A3 102221
A4 011222
A5 002111
<...>
A19 001222
```

*Example.* SNP covariate file with SNP marker data per animal in space-separated format. It contains the number of copies per locus of the allele with the highest number (11=0, 12=1 and 22=2).

```
A1 2 1 0 0 0 0
A2 1 1 0 1 0.274 0
A3 1 0 2 2 2 1
A4 0 1 1 2 2 2
A5 0 0 2 0.793 1.218 1
<...>
A19 0 0 1 2 2 2
```

### 5.10.3 Syntax

```
SNPFILE [!CENTER] [!NOIMPUTE] [!MISSCOMB 0.01] [!MISSPERLOC 0.01] [!PREDICT] &
[!NOPRUNE] [!VALIDSNP <minimum value> <maximum value>] [!CALCSNPVAR] [!MINGENFREQ] &
[!GBSORTSNP <memory allocation in Gb>] [!SAMEORDER]
<field animal> <field type I or A>
SNP01 <file name SNP01> !REGTYPE R [!IDCOL 1] [!STARTCOV 2] [!LASTCOV 7]
SNP02 <file name SNP02> !REGTYPE R [!IDCOL 1] [!STARTCOV 2] [!LASTCOV 7]
<...>
SNP99 <file name SNP99> !REGTYPE R [!IDCOL 1] [!STARTCOV 2] [!LASTCOV 7]

[SNPPARFILE                  # required only for !REGTYPE H
                             #   or for !REGTYPE R if !CALCSNPVAR is not specified
SNP01 <file name SNP01>
SNP02 <file name SNP02>
<...>
SNP99 <file name SNP99>]

MODEL
trait ~ fixed !RANDOM SNP(1,2,8..15,23)   # so no need for G(...) in the model
```

Sections:

**SNPFILE**
The SNPFILE section specifies the name of one or more SNP covariate files and its attributes, such as column numbers, dense or space-separated SNPs and whether one variance for all SNPs is used or an individual variance for each SNP. The section also has a number of qualifiers that apply to all SNP covariate files.
**SNPPARFILE**
The SNPPARFILE section specifies the name of a parameter file for each SNP covariate file for which the SNP covariates are fitted as a random regression (so !REGTYPE is either 'r' or 'h'). The SNPPARFILE section does not have any associated qualifiers.

The lines of the SNPPARFILE section each contain two columns. The first column is the label that links the parameter file to the SNP covariate file. The second column is the name of the file.

Qualifiers:

The file-independent qualifiers of SNPFILE are typically specified on the first line of the SNPFILE section. These are:

**!CENTER**
The !CENTER qualifier is optional and scales all SNP's to a standard normal distribution N(0,1). For details, see Stranden and Christensen (2011). Centring the SNPs affects the fixed effect solutions, but not the SNP effect solutions. Centering may enhance convergence of the PCG iteration.

**!MISSCOMB <maximum fraction of SNPs missing>**
The MISSCOMB qualifier is optional and can be used to specify the tolerance level of missing combinations of animal and SNP. Above the tolerance level, a warning is printed that the analysis may not yield meaningful results, but the analysis continues. If !MISSCOMB is not specified, the tolerance level is 0.001 of all combinations of animal and SNP marker.

**!MISSPERLOC <maximum fraction of SNPs missing per locus>**
The MISSPERLOC qualifier is optional. It specifies the tolerance level of missing SNPs per locus. Loci with too many missing SNPs are written to CheckDataSNP.log and a warning is printed. If !MISSPERLOC is not specified, the tolerance level is 0.05 of all genotypes for the locus (call rate of 95%).

**!NOIMPUTE**
The !NOIMPUTE qualifier can be used to avoid automatic imputation of missing SNPs with the average SNP value of the locus. If !NOIMPUTE is specified, then animals with one or more missing SNPs get a genomic breeding value of -99999 in the solanigen.txt file. If !NOIMPUTE is not specified, then the average SNP value of the locus is used in the calculation of the genomic breeding value.





**!PREDICT**
The !PREDICT qualifier can be used to calculate genomic breeding values for animals without a record in the data file, but with known SNP covariates. Animals with a genotype record, but no data record, are written to separate files, such as SNPcov01NoDat.txt.

**!NOCHECK**
The !NOCHECK qualifier can be used omit any imputing, pruning, centring or checks of SNP covariates, which must be on the scale 0 to 2.

**!NOPRUNE**
The !NOPRUNE qualifier can be used omit the verification that all SNPs are informative and the exclusion of non- or less-informative SNPs.

**!VALIDSNP**
The !VALIDSNP qualifier is optional and can be used to specify a lowest and highest valid value of a SNP genotype. This can only be specified once for all SNPs. The default values are 0 and 2. The !VALIDSNP qualifier can be used for example to process SNP genotypes coded as -1, 0 and 1 or intermediate imputed values by specifying !VALIDSNP -1 1. These values are used to scale observed and imputed SNP genotypes to a scale of 0 to 2.

Note: With centred SNP genotypes on input, the minimum and maximum valid value are SNP-specific. This is currently not supported in MiXBLUP.

**!MINGENFREQ**
The !MINGENFREQ qualifier is optional and can be used to vary the definition of a less-informative SNP. If the frequency of the minor SNP genotype is below the threshold, it is considered to be less-informative and it will be excluded from the analysis, unless !NOPRUNE has been specified. The default threshold is 0.

**!CALCSNPVAR**
The optional !CALCSNPVAR qualifier can be used to calculate the SNP variance from the direct genetic variance in the parameter file specified in the PARFILE section. The CALCSNPVAR qualifier must not be specified if one or more SNP files have SNP-specific variances (!REGTYPE H). If one or more SNP files are fixed (!REGTYPE F), the SNP variance is calculated using the remaining SNP files with !REGTYPE R. When !CALCSNPVAR is specified, the SNPPARFILE section is ignored.

**!GBSORTSNP <amount of memory in Gb>**
The qualifier !GBSORTSNP is optional and can be used to control the use of memory for sorting SNP covariate records in the order of the data file. SNP covariate records are sorted in blocks of records. !GBSORTSNP determines the size of such a block of records to be sorted simultaneously. The default allocation is 16 Gb. The number of blocks of records is the number of times a SNP covariate file has to be read, so a small memory allocation increase the time needed to sort the SNP covariates.

**!SAMEORDER**
If there are multiple SNP covariate files and records in each file are in the same order of individual ID, then !SAMEORDER can be used. MiXBLUP will sort all SNP covariate files simultaneously. Note that memory allocated with !GBSORTSNP is now used for multiple SNP covariate files instead of a single file.

The second line of the SNPFILE section specifies the animal ID code that can be used to link the data and the SNP covariate files.

The following lines each specify a SNP covariate file. Each line starts with a label. The label links the SNP covariate file to the corresponding SNP parameter file. The label must have the form 'SNPxx' where xx is a number between 01 and 99.

The second field contains the name of the SNP covariate file. The additional fields contain one or more file-specific qualifiers. These are:

**!REGTYPE**
The file-specification line must contain the !REGTYPE qualifier. It specifies how the covariates in the file are fitted in the model.

If 'f' is specified, the covariates in the file are fitted as a fixed regression. Covariates fitted as a fixed effect do not have a variance associated with it, so it is not necessary to specify a parameter file in the SNPPARFILE section. If it is present, it is ignored.

If 'r' is specified, the covariates in the file are fitted as a random regression with a single variance for all covariates in the file. The variance is specified in the corresponding parameter file in the SNPPARFILE section.

If 'h' is specified, the covariates in the file are fitted as a random regression, each with their own variance. The covariate-specific variances are specified in the corresponding parameter file in the SNPPARFILE section.

**!IDCOL**
The !IDCOL qualifier is optional and specifies which field in the SNP covariate file contains the ID of the genotyped animal. If it is omitted, it is assumed that the ID is in the first field of the record (so the default is !IDCOL 1).

**!STARTCOV**
The !STARTCOV qualifier is optional and specifies which field contains the first SNP covariate. If it is omitted, it is assumed that the SNP covariates start in the second field of the record (so !STARTCOV 2).

**!LASTCOV**
The !LASTCOV qualifier is optional and specifies which field contains the last SNP covariate of the file to include in the model. If it is omitted, it is assumed that all fields after the first SNP covariate contain SNP covariates to include in the model.

**SNP(...)**
The SNP function can be used in the MODEL section to specify which SNP covariate files should be fitted in the model of a trait. If a SNP covariate file is specified, then all specified SNP covariates in the file will be fitted. The number in the SNP(...) function links to the number in the label of the SNP covariate file.

The numbers may be specified individually as (1, 2, 3, 4) or as a range, indicated by two subsequent full stops, for example (1..4), or a combination of both.

When SNP(...) is specified, it is not necessary to specify the G(...) function to specify a genetic effect, but it is possible, for example to specify a maternal genetic effect.

Despite what the use of SNP(...) in the MODEL section may suggest, all SNP covariate files used in any trait are fitted for all traits.

### 5.10.4 Associated output files

| Output file | Description |
|---|---|
| Solanigen.txt | Genomic breeding values calculated from the SNP covariate record and the SNP-specific regression coefficients for individuals with a data record of at least one observed trait in the model when the field type of the ID is integer |
| Solanigen.out | Genomic breeding values when the field type of the ID is alphanumerical |
| Solreg_mat.txt | Solutions of all general and SNP covariates in the analysis |





## 5.11 Using SNP covariates of genotyped animals and pedigree information of non-genotyped animals (ssRRBLUP)

### 5.11.1 General

A limitation of the RRBLUP model is that only data records of genotyped animals can be included. Fernando et al. (2014) proposed a method to impute the SNP covariates of non-genotyped individuals from genotyped relatives. For individuals with an imputed SNP covariate record, an additional residual polygenic effect is fitted, to account for the uncertainty of the imputed genotype.

The proposed method has been adapted for BLUP, using a genetic random regression approach on the genotype having-been-imputed. As genotyped individuals will have a zero for the genotype being imputed, effectively no polygenic genetic effect is fitted for genotyped animals.

The inverse relationship matrix for the residual polygenic effect is the identity matrix for the block of genotyped by genotyped individuals, zero for the block of genotyped by non-genotyped individuals and directly obtained from the corresponding block of the inverse pedigree relationship matrix for the block of non-genotyped by non-genotyped individuals.

A total breeding value is calculated for all animals, which consists of the genomic breeding value for individuals that are genotyped and the sum of the genomic and residual polygenic breeding value for individuals with an imputed SNP covariate record.

### 5.11.2 Input files

The requirements of the SNP covariate files are described in chapter 5.10. The SNP covariate files need to contain a record for at least every individual with a phenotype, but preferably for all individuals in the pedigree file. For non-genotyped individuals, the SNP covariates are imputed with PrepareFern software, prior to calling MiXBLUP.

Whether or not an animal is imputed needs to be presented in the data file and a file with all animals in the analysis. An observed genotype records has to be indicated with 0 and a fully-imputed genotype record with 1.

The requirements of the external relationship matrix file are described in 5.4. It has to be calculated prior to calling MiXBLUP using FDGRelMat.exe.

### 5.11.3 Syntax

```
DATAFILE <file name>
<...>
<not-being-genotyped>  R
<...>

ERMFILE <file name FDG relationship matrix>
<field individual ID> <field type I or A>

SNPFILE [!INCLNONGENO]
<field individual ID> <field type I or A>
SNP01 <file name SNP01> !REGTYPE R [!IDCOL <value>] [!STARTCOV <value>] [!LASTCOV <value>]
<...>

IMPFILE <file name> [!ImpIDcol 1] [!ImpCol 4]

MODEL
<trait> ~ <...> !RANDOM SNP(1..10) G(<not-being-genotyped>*<Individual ID>
```

Section:

**IMPFILE**
The IMPFILE section must be used to specify a file that indicates for each individual in the FDG relationship matrix whether it was genotyped (0) or not (1). IMPFILE is specified when the qualifier !InclNonGeno is specified in the SNPFILE section. IMPFILE has two qualifiers, !ImpIDcol and !ImpCol.

Qualifiers:

**!ImpIDcol**
This qualifier is optional and only needs to be specified if the individual ID is not in the first column.

**!ImpCol**
This qualifier is also optional and needs to be specified if the imputation information is in another than the fourth column.

**!INCLNONGENO**
The !InclNonGeno qualifier is used in the SNPFILE section and triggers ssRRBLUP. When it is specified, the data file has to contain a column that indicates whether the SNP covariate record is observed (indicated with 0) or imputed (indicated with 1). This column in the data file has to be used as covariate (so field type R) in the genetic random regression of each trait in the model. If the covariate has value 0, then effectively the residual polygenic effect is not fitted for the individual.

If !InclNonGeno is used, then section IMPFILE has to be specified, too. It is used to calculate a total breeding value for all individuals in the analysis. It is up to the user to verify that the information on not-being-genotyped is identical between the data and imputation file for each individual.

The **ERMFILE section** must be used to specify the relationship matrix for the residual polygenic effect, which contains the A-inverse for the diagonal block of non-genotyped animals, an identity matrix for the diagonal block of the genotyped animals and zeroes for the off-diagonal blocks linking the genotyped and non-genotyped animals.

**!CalcSNPvar**
If !CalcSNPvar is specified in a ssRRBLUP analysis, the G matrix in the parameter file needs to include a line for both <trait>(<ID>) and <trait>(<ID>*<not-being-genotyped>), for example

```
G
Weight(Animal)         425.34
Weight(Animal*NonG)      0.00   425.34
```

The <trait>(<ID>) line is included for !CalcSNPvar and the <trait>(<ID>*<not-being-genotyped>) for the genetic effect in the model. The variance components are typically the same. The covariance components between the two effects is not used, so may be zero. For multiple traits, it is recommended to keep the <trait>(<ID>) lines together in a diagonal block, the <trait>(<ID>*<not-being-genotyped>) lines in a separate, but identical block and the off-diagonal block of zeroes. For example

```
G
Weight1(Animal)         425.34
Weight2(Animal)         284.59   528.93
Weight3(Animal)         230.21   304.47   673.85
Weight1(Animal*NonG)      0.00     0.00     0.00   425.34
Weight2(Animal*NonG)      0.00     0.00     0.00   284.59   528.93
Weight3(Animal*NonG)      0.00     0.00     0.00   230.21   304.47   673.85
```





### 5.11.4 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the residual polygenic effect for all individuals in the analysis |
| Solanigen.txt | Genomic breeding values calculated from the SNP covariate record and the SNP-specific regression coefficients for individuals with a data record of at least one observed trait in the model when the field type of the ID is integer |
| Solanitot.txt | Genomic breeding values for genotyped individuals and the sum of the genomic and residual polygenic breeding values for individuals with an imputed SNP covariate record. It contains only individuals with a data record of at least one observed trait in the model. |
| Solani.out | As Solani.txt when the field type of the ID is alphanumerical |
| Solanigen.out | As Solanigen.txt when the field type of the ID is alphanumerical |
| Solanitot.out | As Solanitot.txt when the field type of the ID is alphanumerical |
| Solreg_mat.txt | Solutions of all general and SNP covariates in the analysis |

## 5.12 Using a pedigree with double-haploid individuals

### 5.12.1 General

Double-haploid individuals are used in plant breeding and are formed from two identical gametes. Note that this is virtually always different from selfing. With selfing, an individual is formed from two gametes of the same individual. Therefore, information on being double-haploid must be specified in a separate column or file.

All gametes of a double-haploid individual are identical. Offspring of two double-haploids are genetically identical. Such individuals are not included in the relationship matrix. This requires special attention for preparing the data file.

### 5.12.2 Input files

The presence of double-haploid individuals requires a double-haploid information file and changes to the data file.

The **double-haploid information file** may have various formats. It may contain just a list of double-haploid individuals, either just the individual ID or the individual ID in a record with multiple fields. It may also be specified in a file with double-haploid individuals marked as 1 and others as 0.

*Example.* DH information file with just individual IDs.

```
D1123
D1319
D1326
<...>
D0980
```

*Example.* DH information file with just individual IDs as part of a larger record.

```
D1123 2011 A0078
D1319 2013 A0103
D1326 2013 A0111
<...>
D0980 2009 A0013
```

*Example.* DH information file containing all individuals in the pedigree.

```
A0001     0     0 0
<...>
A0078 A0013 A0029 0
<...>
D1123 A0078 A0078 1
D1319 A0103 A0103 1
D1326 A0111 A0111 1
<...>
D0980 A0013 A0013 1
```

The **data file** must not contain offspring of two double-haploid individuals. Instead, such a data record should be assigned to each parent, weighted with 0.5. See example below. The breeding values of these individuals can be calculated as the parental average.

*Example.* Offspring of two DH individuals in the data file.
Pedigree record

```
<...>
A2704 D1123 D1319
<...>
```

Available data records

```
<...>
A2702 201508_0105 103 2392
A2703 201508_0105 107 2459
A2704 201508_0105 102 2412
<...>
```

Data file

```
<...>
A2702 201508_0105 103 2392 1.0
A2703 201508_0105 107 2459 1.0
D1123 201508_0105 102 2412 0.5
D1319 201508_0105 102 2412 0.5
<...>
```





### 5.12.3 Syntax

**DHFILE <file name DH information> [!DHMAKE] [!DHID <field number>] [!DHCOL <field number>] <field individual ID> <field type I or A>**

The section DHFILE is used to specify the double-haploid information file. If the file just contains double-haploid individuals, there is no need for the qualifiers !DHMAKE, !DHID and !DHCOL. These qualifiers are used for alternative formats.

**!DHMAKE**
The !DHMAKE qualifier is optional and must be used when the file with double-haploid information contains all individuals and a column of 0 and 1 to distinguish double-haploid individuals.

**!DHID**
The !DHID qualifier is optional and is used to specify the field number of the individual ID. The default value is field number 1.

**!DHCOL**
The !DHCOL qualifier is optional and used to specify the field number of being-double-haploid. The default value is field number 4. !DHCOL only has a meaning in combination with !DHMAKE.

The file with DH information may be the same file as specified for the pedigree file.

**Warning:** When using double-haploid individuals in combination with genomic information, one must use !CONSTRUCT Hinv to blend pedigree and genomic information. Using !CONSTRUCT SSMAT will ignore information on double-haploid individuals.

### 5.12.4 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | Solutions of the direct genetic effect for all individuals in the analysis |
| Solani.out | As Solani.txt, but for alphanumerical ID |

# 6. Components of variance and covariance among traits

Components of variance and covariance among traits are normally specified in the general parameter file. Additional covariance components for covariates in a covariate file need to be specified in separately labelled parameter files. Heterogeneous residual variances also need to be specified in a separate file. This chapter describes how to specify components of variance and covariance among traits.

## 6.1 General parameter file

### 6.1.1 General

The trait (co)variance components file contains the between-trait variance-covariance matrices of any random effects in the statistical model.

There are two options for the format of the general parameter file: (1) in lower-triangular-matrix form and (2) in sparse-matrix form. It is strongly recommended to use the lower-triangular-matrix format.

The instruction file specifies the name of the trait (co)variance components file. The trait (co) variance components file is located by default in the work directory, but can be in another folder if specified in the name of the file.

### 6.1.2 Input file in lower-triangular-matrix format

The lower-triangular-matrix form is the default option and strongly recommended. In this form, the trait covariance components file can be specified as a lower-triangular matrix using trait names to identify the components. This is the most user-friendly way. The name of the random effect is given at the top of the matrix and the names of the traits are given at the start of each line of the matrix.

- The lower triangular matrices and the traits within a matrix can be specified in any order. It means that the order given in the MODEL section of the instruction file is not leading.
- The number of traits in the matrices can be larger than the number of traits specified in the model section. Only the lines for which the name has been specified in the model section will be used.
- The order of the column names must be the same as the order of row names, so variance components are on the diagonal.
- Restriction: in case of a marker-assisted BLUP model with the use of haplotype variance-covariance matrices, each matrix needs to be named and numbered, e.g. GIV1, GIV2, etc. The name GIV refers to the use of the General Inverse Variance (GIV) function in the model. The order of matrices must be the same as the order of haplotypes given in the model lines of the instruction file. See Example 5.4 in the Appendix.





- In the case of multiple genetic effects (e.g. animal, dam, mate), it should be specified immediately after the trait name and within brackets whether it is the genetic variance of animal, dam or mate.
- In case of non-genetic random regression, the name of the class effect is specified at the top of the matrix and a line for each combination of trait and the full random regression term in the model of the trait should be specified. The syntax in previous versions of MiXBLUP with a separate matrix for each random regression term is still supported, but not recommended, as it ignores covariance components between different random regression terms of the same trait.
- If the model contains genetic random regression, then all regression terms should be specified (e.g. animal*covar1 and animal*covar2).
- If a covariate table file is used for random regression, then the columns should be referred to as cvr00 for the first covariate column, cvr01 for the second column and so on. The name should be lowercase: the use of CVR00 will give an error.
- In case of a social interaction model, with multiple mate effects in the model, the first group mate effect in the model should be specified (e.g. mate2*mate2_x).

*Example.* The lower triangular trait (co)variance components file with two traits (body weight 1 and body weight 2) for non-genetic random regression, animal genetic and residual effects.

```
sex
bw1(sex*age1)   100
bw2(sex*age1)   0 150
bw2(sex*age2)   0   50 200

G
bw1(animal) 3000
bw2(animal) 2939 4500

residual
bw1 7000
bw2 1715 10500
```

### 6.1.3 Input file in sparse-matrix format

In the sparse matrix form, the order of the matrices must be the same as the order of random effects in the model, with the restriction that the genetic effect should be the last random effect in the model and the elements of its (co)variance matrix should appear in the sparse matrix file just before the elements of the residual (co)variance matrix. The residual (co) variance matrix should be specified at the end of the sparse matrix file.

In summary, the order of matrices is:

- Non-genetic random effects in the same order as specified in the model
- Genetic effects as specified in the model
- Residual effect

The matrix elements must be specified as the random effect number, row number, column number and the value of the (co)variance. To avoid mistakes, it is recommended to provide the elements of the lower triangle of the matrix, in other words, any column number is smaller than or equal to the row number. Off-diagonals only need to be specified if they are non-zero.

When haplotypes are used in the model for marker-assisted BLUP with the use of an inverse IBD matrix, both haplotypes are counted as effects, but the same variance components are used for

the first and the second haplotype, when haplotypes are combined with the AND function, so the variance components should not be repeated for the second haplotype. Effectively, the effect number corresponding to the second haplotype is skipped from the list of inverse matrix elements. See Example 5.4 in the Appendix.

*Example.* The trait (co) variance components file in sparse-matrix format with two traits for the animal genetic and residual effects Columns: random effect number, trait row number, trait column number and variance or covariance component.

```
1 1 1 3000 #animal
1 2 1 2939
1 2 2 4500
2 1 1 7000 #residual
2 2 1 1715
2 2 2 10500
```

### 6.1.4 Syntax

```
PARFILE <filename> [!SPARSE]
```

Qualifier:

**!Sparse**
If !SPARSE is specified, the variance and covariance components are read in sparse matrix form. If omitted, the matrix is read in lower triangular form.

## 6.2 Parameter files for general covariates

### 6.2.1 General

The regression parameter file is specified for each general covariate file that is fitted as random regression. The file may contain a single set of variances and covariances between traits that apply to all covariates or a set for each covariate separately.

The MiXBLUP shell checks whether scaling is necessary to avoid an error that the matrix is not positive-definite and applies any required scaling automatically.

### 6.2.2 Input file

The format of the files with parameters of general covariates is the lower-triangular-matrix format of the general parameter file.

*Example.* Regression parameter file with a single set of variances and covariances between traits for all covariates. A regression parameter with covariate-specific variances and covariances contain such a set for each covariate. The *number* in the label of the matrix is linked with the *position* of the covariate in the record.

```
REG001
bw1  0.0347
bw2  0              0.0619
```





### 6.2.3 Syntax

```
REGPARFILE
REG01 <file name REG01>
REG02 <file name REG02>
<...>
REG99 <file name REG99>
```

The REGPARFILE section must contain the name of a parameter file for each covariate file for which the covariates are fitted as a random regression (so !REGTYPE is either 'r' or 'h'). If the regression type is fixed, the corresponding file in REGPARFILE is ignored. The REGPARFILE section does not have any associated qualifiers.

The lines of the REGPARFILE section each contain two columns. The first column is the label that links the parameter file to the covariate file. The second column is the name of the file.

## 6.3 Parameters for SNP covariate files

### 6.3.1 General

The SNP parameter file is specified for SNP covariate files that are to be fitted for random regression. The file may contain a single set of variances and covariances between traits for all SNP covariates or a set for each SNP covariate separately.

For a RRBLUP model without a direct genetic effect and SNP genotypes presented as 0, 1 and 2, the SNP variance can be calculated from the direct genetic variance with

$$var_{SNP} = var_G / \sum_{i=1}^{N} 2p_i(1-p_i)$$

where *N* is the number of informative SNPs and $p_i$ is the allele frequency of the SNP allele counted on locus *i*. Non-informative SNPs must not be included in this calculation.

If variances smaller than 1.0E-06 are specified, then the MiXBLUP kernel may give an error that the variance-covariance matrix is not positive-definite. This can be resolved by scaling the phenotypes with 10 or 100 and the variances with 100 or 10,000 accordingly. The MiXBLUP shell checks whether scaling is necessary and applies any required scaling automatically.

### 6.3.2 Input file

The format of the files with parameters of general covariates is the lower-triangular-matrix format of the general parameter file.

If a single set of variances and covariances between traits is to be used for all SNP covariates (so !REGTYPE is 'r'), then only one matrix needs to be specified. The matrix label needs to start with 'SNP', but the number is ignored.

If SNP-specific variances and covariances are to be used (so !REGTYPE is 'h'), then a matrix has to be specified for every SNP covariate separately. Depending on the number of SNP covariates in a file, this could be many thousands. The label has to start with 'SNP'. The *number* in the label of the matrix is linked with the *position* of the SNP covariate in the record of the corresponding file. The number must be sequential and may be an integer between 1 and 2.1 billion.

The label of a matrix in a SNP parameter file refers to a SNP covariate in the corresponding covariate file and should not be confused with the label linking the SNP covariate and parameter files.

*Example.* SNP parameter file with a single set of variances and covariances between traits for all SNP covariates.

```
SNP00001
bw1  1.07667E-05
bw2  0             1.29534E-05
```

### 6.3.3 Syntax

```
SNPPARFILE
SNP01 <file name SNP01>
SNP02 <file name SNP02>
<...>
SNP99 <file name SNP99>
```

The SNPPARFILE section specifies the name of a parameter file for each SNP covariate file for which the SNP covariates are fitted as a random regression (so !REGTYPE is either 'r' or 'h'). If the regression type is random and !CalcSNPvar has been specified, the corresponding file in SNPPARFILE is ignored. The SNPPARFILE section does not have any associated qualifiers.

The lines of the SNPPARFILE section each contain two columns. The first column is the label that links the parameter file to the SNP covariate file. The second column is the name of the file.

## 6.4 Parameters in case of heterogeneous residual variances

### 6.4.1 General

The residual variance may not be the same for all observations. If this is the case, observations can be grouped by their residual variance prior to the analysis. A column in the data file links the observation to the correct residual variance matrix.

Modelling data with a random regression approach often requires the use of multiple residual variance classes.

### 6.4.2 Input file

The file contains a matrix for every class number in the linking column in the data file. The name of the matrix is Res followed by the class number between brackets. The class number has to be an integer.

The example below gives the series of residual matrices for a situation with observations being linked to one of three residual variances classes.

*Example.* The residual covariance file with three residual variance-covariance matrices.

```
Res (1)
bw1 5000
bw2 1264 8000

Res (2)
bw1 6000
bw2 1587 10500

Res (3)
bw1 10000
bw2 2280 13000
```





### 6.4.3 Syntax

```
DATAFILE <filename>
...
[<field j> I !RESVARCLASS]
...

RESFILE <filename>
```

Section:

**RESFILE**
The RESFILE section specifies the name of the file with residual between-trait variance-covariance matrices. The RESFILE section does not have specific qualifiers.

Qualifier:

**!RESVARCLASS**
The qualifier !ResVarClass in the DATAFILE section links a data record to the appropriate residual variance class, in case the residual variance differs for groups of records. The field is integer. The qualifier !RESVARCLASS must be used if the section RESFILE is specified.

# 7. Statistical models

Observed traits on two individuals may be similar due to genetic effects and systematic non-genetic effects. The statistical model contains all such effects known to explain variation in observed traits. This chapter describes various statistical models to estimate genetic effects on traits with as little bias as possible.

## 7.1 Basic models

### 7.1.1 General

The MODEL section specifies the start of the statistical models for the traits in the analysis. Traits and statistical models start immediately below the line with the MODEL keyword. For each trait, the statistical model is specified on a separate line. MiXBLUP supports up to 63 traits to be analysed simultaneously, if computer resources permit this.

The basic statistical model for a breeding value evaluation contains fixed effects, uncorrelated, non-genetic random effects and a direct genetic random effect. Uncorrelated, non-genetic random class effects are assumed to have an identity relationship matrix between levels of the effect.

Each model line contains trait name, fixed effects, non-genetic random effects and genetic random effects. Fixed effects may be class effects, covariates or covariates nested within a class effect. Similarly, random effects may be class effects or covariates nested within a class effect. The residual random effect does not need to be specified. The minimum statistical model contains one fixed effect and one genetic random effect.

### 7.1.2 Syntax

```
MODEL
<trait1>  ~ <class effects> [covariates] [class effect * covariates]
&!RANDOM G(<direct genetic effect>) [<non-genetic random effects>]
[<trait2> ...]
...
[<traitN> ...]
```





Section:

**MODEL**
The MODEL section specifies the statistical model for the traits in the analysis.

Qualifiers:

**~ (tilde)**
The tilde separates the trait from the statistical model.

*** (star)**
The star is used for nesting a covariate within a class effect, to yield a regression coefficient for each level of the class effect. This can be used for both fixed and random nested covariates. The star must not be used to model an interaction of class effects.

**!RANDOM**
The !RANDOM qualifier separates the fixed effects from the random effects in the model.

**G(...)**
The G(...) function links a random effect to the inverse genetic or genomic relationship matrix.

### 7.1.3 Associated output files

| Output file | Description |
|---|---|
| Solfix.txt | Solutions of fixed effects by trait (class effects, covariates and nested covariates) |
| Solfix.out | As Solfix.txt, but for alphanumerical class labels |
| Solr00.txt | Solutions of non-genetic, uncorrelated random effect 00 by trait, where 00 ranges from 1 to the number of non-genetic, uncorrelated random effects across traits (class effects, covariates and nested covariates) |
| Solr00.out | As Solr00.txt, but for alphanumerical class labels |

## 7.2 Repeatability models

### 7.2.1 General

Certain traits are measured just once in the lifetime of an individual. Other traits may be measured repeatedly. Two observations on a single individual are more similar than expected from having the same genotype. A permanent environmental effect is usually added to the model to account for non-genetic similarity of records of the same individual.

Such a permanent environmental effect has the same label as the direct genetic effect, but with an identity relationship matrix between levels. This permanent environmental effect should have its own column in the data file and must not be the column with the direct genetic effect (although identical).

### 7.2.2 Syntax

```
MODEL
<trait1>  ~ <fixed effects> !RANDOM G(<direct genetic effect>) &
<permanent environmental effect>
...
[<traitN> ...]
```

### 7.2.3 Associated output files

| Output file | Description |
|---|---|
| Solr00.txt | Solutions of non-genetic, uncorrelated random effect 00 by trait, where 00 ranges from 1 to the number of non-genetic, uncorrelated random effects across traits (among which the permanent environmental effect) |
| Solr00.out | As Solr00.txt, but for alphanumerical class labels |

## 7.3 Maternal genetic models

### 7.3.1 General

Some traits are affected by the genotype of the animal itself and the genotype of its dam at the same time. An example is weaning weight in beef cattle. For such traits, a maternal genetic model should be used. The inverse numerator relationship matrix is applied both to the direct genetic effect (animal) and the maternal genetic effect (dam).

The maternal genetic effect must be a separate field in the data file and each individual in this field must exist as an individual in pedigree, genotype file or other resource of genetic similarity. For biological dams, this is self-evident, but for foster dams, this requires special attention.

### 7.3.2 Syntax

```
MODEL
<trait1>  ~ <fixed effects> !RANDOM G<direct genetic effect>,&
<maternal genetic effect>)
...
[<traitN> ...]
```

The maternal genetic effect is placed within the brackets of function G to link it with the relationship matrix between individuals.

### 7.3.3 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | The solutions of the maternal genetic effect are included as additional columns in Solani.txt. The exact layout of Solani.txt is printed at the end of solver.log. |
| Solani.out | As Solani.txt, but for alphanumerical ID labels |

## 7.4 Social interaction models

### 7.4.1 General

The social interaction model (or group selection) is used to estimate the effects of an animal's genotype on its own performance and on the performance of its pen mates simultaneously. It should be used for groups of a single group size, but a slightly varying group size is also supported.

For a single group size, a genetic effect for social interaction is fitted for each pen mate. This effect can be interpreted as the genetic value for supporting or inhibiting the expression of the direct genetic merit of pen mates. The genetic variance of this social effect is dependent on the size of the group, so performance in small and large groups by design should not be combined as one trait.

If the size of groups is the same by design, but it varies slightly due to death or removal from





the pen, it is still possible to fit a social interaction model by adding a covariate of either 0 (not present) or 1 (present) and apply a nested regression of this covariate within pen mate. The ID label used for not-present pen mates must appear in the pedigree or genotype file, too, but no information is added to its social genetic value when the covariate is zero (not present).

### 7.4.2 Syntax of the social interaction model with one group size for all groups

```
DATAFILE
    Animal I
    ...
    mate1 I
    mate2 I
    ...
    mateN I

MODEL
    <trait1> ~ ... !RANDOM G(Animal,mate1 AND mate2 ... AND mateN) ...
    [<trait2> ...]
    ...
    [<traitN> ...]
```

The pen mates need to be defined in the data file. The number of additional columns is equal to the number of pen mates (mate1, mate2, ..., mateN).

Qualifier:

**AND**

The function AND combines the effects of different mates to one design matrix. There are a few constraints when combining effects; all of them are rare:

- When 'AND' is used for group selection, it cannot be used for IBD-haplotypes. In other words, haplotype effects cannot be included in a social interaction model.
- Combining of effects needs to be the same for any traits for which effects are combined. In other words, traits measured when the animal was in a different group need to be analyzed in a separate analysis.
- The parser supports an evaluation in which some traits have social genetic effects and other traits do not.
- There can be only one group of combined genetic effects, so no commas must be placed between the effects of mates. It means that the genetic effects can only be combined to one other effect and not more.
- The order of genetic effects should be kept the same across traits with a social interaction model. So for all traits involved, the order should be either G(animal,mate2 and mate3) or G(mate2 and mate3, animal), but not for some traits G(animal,mate2 and mate3) and for others G(mate2 and mate3, animal).

### 7.4.3 Syntax of the social interaction model with slightly varying group sizes

```
DATAFILE
    Animal I
    ...
    mate1 I
    present1 R
    mate2 I
    present2 R
    ...
    mateN I
    presentN R

MODEL
<trait1> ~ ... !RANDOM G(Animal,present1*mate1 AND present2*mate2 ... AND
presentN*mateN) ...
[<trait2> ...]
...
[<traitN> ...]
```

Both the pen mates and their presence need to be defined in the data file. The number of additional columns is therefore equal to two times the number of pen mates (mate1, mate2, ..., mateN, present1, present2, ..., presentN).

### 7.4.4 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | The solutions of the social genetic effects are included as additional columns in Solani.txt. The exact layout of Solani.txt is printed at the end of solver.log. |
| Solani.out | As Solani.txt, but for alphanumerical ID labels |

## 7.5 Random regression models

### 7.5.1 General

There are two types of random regression models supported by MiXBLUP, the non-genetic and genetic random regression model. Both original covariates and polynomials derived from an independent variable may be used in the model.

In a non-genetic random regression model, the regression of the observations on an independent covariate is fitted as a random effect. Random regression in MiXBLUP has to be specified as regression nested within a class variable. If no nested regression is required, the user needs to add a column of ones to the data file, and fit the covariate within this class effect of a single level. The internal structure of MiXBLUP requires that any random effect be associated with a class effect. This can be seen in the parameter files, too, where variance-covariance matrices are all specified by class effect.

In a genetic random regression model, trait observations are regressed on the covariate within animals, taking into account the genetic relationships between animals. The estimated breeding values from such an analysis concern the animal-specific parameters of the line or curve fitted. The user needs to convert these estimates to estimated breeding values at a given level of the covariate or for a function of levels of the covariate.

If the relationship between an observed trait and an independent variable is non-linear, it may still be possible to model the relationship with polynomials, as a special case of multiple





linear regression. Polynomial regression is a form of linear regression in which the relationship between the independent variable x and the observed trait is modelled as an $n^{th}$ degree polynomial in x by fitting (n+1) covariates derived from x. Polynomials may be provided by the user either in the data file or in a covariate table, or may calculated during the preparations for the analysis and stored in a covariate table. Polynomials calculated by MiXBLUP are Legendre polynomials.

### 7.5.2 Syntax of a non-genetic random regression model

```
MODEL
<trait1> ~ ... !RANDOM <class>*<covariate> [<class>*<covariate> ...]
...
[<traitN> ...]
```

The random regression term consists of a class effect with field type integer (I) or alphanumerical (A) and a covariate with field type real (R). Each random regression term has to be present in the variance-covariance matrix of the class effect in the parameter file (see chapter 6.1).

Qualifier:

*** (star)**
The star is used for nesting a covariate within a class effect, to yield a regression coefficient for each level of the class effect. There is no specific order of class effect and covariate.

### 7.5.3 Syntax of a genetic random regression model

```
MODEL
<trait1> ~ ... !RANDOM G(<ID>*<covariate>[,<ID>*covariate>]) ...
...
[<traitN> ...]
```

The regression terms nested within the individual's ID are placed within the function G(...) to indicate that the relationship matrix of individuals should be used.

### 7.5.4 Syntax of a polynomial regression model using a covariate table

```
CVRTABLE <name covariate>
MODEL
<trait1> ~ ... <class>*CVR(<n1>) !RANDOM <class>*CVR(<n2>) G(<ID>*CVR(<n3>)) ...
...
[<traitN> ...]
```

For the use of covariate table files, see chapter 4.2.

Qualifier:

**CVR(...)**
The CVR function is used in the MODEL section and is a shorthand for all polynomial terms to be fitted and may be used in the same way as any individual random regression term. The alternative way to specify polynomial random regression is to use the individual columns of the covariate table file. The names of the columns are cvr00, cvr01, cvr02, ..., cvrnn. The label is lowercase and has exactly two digits ranging from 00 to 99.

### 7.5.5 Associated output files

| Output file | Description |
|---|---|
| Solani.txt | The solutions of the genetic nested regression effects are included as additional columns in Solani.txt. The exact layout of Solani.txt is printed at the end of solver.log. |
| Solani.out | As Solani.txt, but for alphanumerical ID labels |
| Solr00.txt | The solutions of the non-genetic nested regression effects are included as additional columns in Solr00.txt. The exact layout of Solr00.txt is printed at the end of solver.log. |
| Solr00.out | As Solr00.txt, but for alphanumerical class labels |

## 7.6 Weighting residuals by record

### 7.6.1 General

If the common assumption of constant standard deviation of the residuals (i.e. homogeneous residual variance) is not met, it is possible to weight individual records. Less precise measurements get less weight and more weight is given to more precise measurements when estimating breeding values.

An example is the use of de-regressed breeding values as a pseudo-phenotype. The standard deviation of the residual depends on the reliability of the original estimated breeding value. A weighting factor based on the reliability can be used to give more weight to pseudo-phenotypes based on a relatively large amount of information.

Another example is variation in residual variances within contemporary groups. Observations in contemporary groups with a large residual variance can be given a proportionally lower weighting factor.

### 7.6.2 Syntax

```
MODEL
<trait1> [!WEIGHT <weighting factor>] ~ <fixed effects> !RANDOM G(<ID>) [<non-genetic
random effects>]
...
[<traitN> ...]
```

Qualifier:

**!WEIGHT**
A field in the data file can be specified as a weighting factor for a specific trait using the !WEIGHT qualifier.

### 7.6.3 Associated output files

The standard output files are used for a weighted analysis.

## 7.7 Combining effects across traits

### 7.7.1 General

If a trait measured in different cycles or parities or on individuals of different strains and crosses is modelled as multiple traits, it may be desirable to estimate fixed effects across these traits, in order to increase the precision of the solutions of the model.

Random effects can easily be combined by specifying covariances between the traits that are equivalent to a correlation close to unity.

For fixed effects, it has to be specified across which traits the effect should be estimated.





### 7.7.2 Syntax

```
MODEL
<trait1> ~ <fixed1> !RANDOM G(<ID>) [<random1>]
<trait2> ~ <fixed1> !RANDOM G(<ID>) [<random1>]
...
<traitN> ~ <fixed> !RANDOM G(<ID>) [<random>]

COMBINE
<fixed1> ~ <trait1> <trait2>
```

Section:

**COMBINE**
The section COMBINE allows to specify across which traits a fixed effect should be estimated. It supports class effects, covariates and nested covariates.

### 7.7.3 Associated output files

The standard output files are used for an analysis with fixed and random effects estimated across several traits.

## 7.8 Correction of heterogeneous residual variances

### 7.8.1 General

If residual variance within contemporary groups varies (heterogeneous residual variance), the user may specify appropriate weighting factors in the data file and weight records accordingly (see chapter 7.6).

MiXBLUP also offers the possibility to calculate appropriate weighting factors in a three-step approach. In the first step, the traits are analysed with the assumption of homogeneous residual variance. The residuals (ê) are read from the output of step 1 and the linearized squared residuals (z) for trait i and animal j are calculated as

$$z_{ij} = LOG\left(Var(e_i)\right) + \frac{(\hat{e}_{ij}^2 - Var(e_i))}{Var(e_i)}$$

$Var(e_i)$ is the residual variance of trait i used in the first step and is obtained from the res(idual) matrix in the parameter file or, if residual variance classes are used, the residual variance of the corresponding class of the record.

In the second step, these linearised squared residuals are analysed using a suitable model.
The predicted phenotypes of this second model are used to calculate weighting factors.
The weighting factor for trait i and individual j is calculated from the predicted value of the linearised squared residual ($\hat{Z}_{ij}$) as

$$W_{ij} = \frac{1}{e^{(\hat{z}_{ij} - \bar{z}_i)}}$$

($\hat{Z}_i$) where is the average predicted value of the linearized squared residual for trait i across all individuals.

In the third step, the analysis of the first step is repeated, but with a weighting factor added to account for heterogeneous residual variance.

MiXBLUP can run these three steps in a single process.

### 7.8.2 Syntax

```
MODEL
<trait1>  ~ <fixed1> !RANDOM <random1> G(<ID>)
<trait2>  ~ <fixed2> !RANDOM <random2> G(<ID>)
<trait3>  ~ <fixed3> !RANDOM <random3> G(<ID>)

VARMODEL
LSR1  ~ <fixed> !RANDOM <random> G(<ID>) !VARTRAIT <trait1>
LSR2  ~ <fixed> !RANDOM <random> G(<ID>) !VARTRAIT <trait2>

SOLVING
!DHGLM !HETCOR
```

Section:

**VARMODEL**
The VARMODEL section specifies the statistical model for the second step for the linearized squared residuals.

Qualifiers:

**!VARTRAIT**
The qualifier !VARTRAIT in the VARMODEL section is mandatory and links the linearized squared residual to the original trait. Original traits do not have to be all represented in the VARMODEL section.

**!DHGLM**
The option !DHGLM in the SOLVING section prepares MiXBLUP for multiple calls of the kernel.

**!HETCOR**
The qualifier !HETCOR in the SOLVING section creates the data file and instruction file for each step.

### 7.8.3 Associated output files

The standard output files are used for an analysis with correction for heterogeneous variances.

## 7.9 Using a threshold model for a categorical trait

### 7.9.1 General

The linear model used in MiXBLUP to estimate breeding values is based on the assumptions that a trait has a continuous normal distribution, its components of variance are homogeneous and residuals are uncorrelated with genetic and non-genetic random effects.

There are traits that are recorded as categories. A binary trait has only two possible categories, for example, present or absent, true or false, all or none. Traits with more than two categories may be ordered, for example small-medium-large, or unordered, such as red-yellow-blue. For categorical traits, the usual assumptions of a linear model are violated.

It has been proposed to overcome this by assuming a continuous trait underlying the categorical trait. Thresholds on the underlying scale determine the recorded category on the observed scale. The threshold model is more demanding than the linear model.

There are methods available for the threshold model: Newton-Raphson (NR) and Expectation-Maximisation (EM). Both methods give the same results but use a different route. Both methods have two levels of iterations. The outer level iterates on the thresholds, given the set of estimated solutions of the previous iteration. The inner level iterates on the solutions given the current estimates of the threshold.





Currently only one categorical trait can be analysed with a threshold model in a multi-trait evaluation of any number of traits analysed with a linear model.

Although theoretically incorrect, assuming a linear model for a categorical trait often yield solutions that rank selection candidates largely in correct order. This is especially the case for intermediate prevalence of categories.

### 7.9.2 Input files

Category labels must be numbered 1 to the number of categories for the MiXBLUP kernel. MiXBLUP can rename category labels for this purpose from a file with ordered labels by trait. The file is specified with !CONVERTCAT. The first field is the trait name in the data file. Subsequent fields contain the category labels. The position in the sequence determines the new sequential integer code, 1..n. Although MiXBLUP currently supports only a single trait with a threshold model in combination with any number of traits with a linear model, multiple traits may be specified for use across evaluations. In the example below, the stature categories Small, Medium and Large are converted to 1, 2 and 3, according to their positions in the record. The diseased categories 1 and 0 are converted to 1 and 2. Note that a binary trait coded as 0/1 has to be converted to 1 and 2.

Example. Category conversion table.

```
Stature Small Medium Tall
Diseased 1 0
```

It is also possible to fix thresholds at a predefined value, using !THRFIXED. Thresholds must be taken from a N(0,1) underlying distribution. The file may contain any number of categorical traits, but in any evaluation, only one categorical trait can be analysed with a threshold model, currently. The number of thresholds to be specified is the number of categories minus 1. Thresholds may be taken from a previous analysis or calculated from the observed prevalence in a larger set of data.

Example. Table with fixed thresholds.

```
Stature -0.34 0.56
Diseased 0.0
```

### 7.9.3 Syntax

```
DATAFILE <filename> [!CONVERTCAT <filename>]
...

MODEL
<trait1>  ~ <fixed1> !RANDOM <random1> G(<ID>)
...
<trait3>  ~ <fixed3> !RANDOM <random3> G(<ID>) !THRESHOLD <number of thresholds>
```

SOLVING

[!THRMAXIT <maximum number of NR or EM iterations at outer level>]
[!THRMAXPCG <maximum number of iterations at inner level>]
[!THRMETHOD <EM or NR>]
[!THRFIXED <filename>]

Qualifiers:

**!CONVERTCAT**
The qualifier !CONVERTCAT in the DATAFILE section is optional. It can be used to specify a file with category labels.

**!THRESHOLD**
The option !THRESHOLD in the MODEL section is mandatory and specifies which trait is to be analysed with a threshold model.

**!THRMAXIT**
The qualifier !THRMAXIT in the SOLVING section can be used to specify the maximum number of NR or EM iterations. The default number is 5,000.

**!THRMXPCG**
The qualifier !THRMAXPCG in the solving section is optional and specifies the maximum number of iterations within each NR or EM iteration. The default number is 100.

**!THRMETHOD**
The qualifier !THRMETHOD is optional and specifies the method to implement the threshold model. The default method is NR. The alternative method is EM.

**!THRFIXED**
The qualifier !THRFIXED is optional and can be used to specify a file with fixed thresholds per trait.

### 7.9.4 Associated files

The standard output files are used for an analysis with a threshold model.





# 8. Control of analysis and output

This chapter describes the control part of the instruction file, which can be used to control the analysis and the output generated from the analysis.

## 8.1 Control of the analysis

### 8.1.1 General

Control of an iterative process like solving a linear system to estimate breeding values involves setting the convergence criterion and the maximum number of iterations, specifying whether the run is a continuation or a new start and defining the starting values for a new evaluation.

An more advanced option is to specify the type of preconditioner to be used for solving the system. Generally, the default type of preconditioner is optimal. The default varies across models to specify genetic similarity between individuals.

### 8.1.2 Syntax

```
SOLVING
[!MAXIT <number of rounds>]
[!STOPCRIT <convergence criterion>]
[!NOPEEK]
[!PEEKFIRST <iteration number>]
[!PEEKEVERY <number of rounds>]
[!PEEKKEEP]
[!RESTART]
[!GFROMDISK]

PRECON <n, b, d>
[!WITHINBL <b, d>]
[!ACROSSBL <f, m, b, d>]
```

Sections:

**SOLVING**
The SOLVING section is used to control the process and the output of the analysis.

**PRECON <option>**

The PRECON section can be used to change the type of preconditioner. Possible options are n (no preconditioner), b (block-diagonal preconditioner) and d (diagonal preconditioner).

Qualifiers:

**!MAXIT <number of iterations>**
The optional !MAXIT qualifier in the SOLVING section can be used to set the maximum number of iterations to be used. If !MAXIT is not specified, the default maximum number of iterations is 5,000.

**!STOPCRIT <convergence criterion>**
If the convergence criterion needs to be different from 1.0E-04, it can be set with the optional !STOPCRIT qualifier in the SOLVING section.

**!NOPEEK**
MiXBLUP stores intermediate results by default every 100th iteration. All solutions files are created and starting values for a restart are stored as if solutions have converged. By default, only the last set of preliminary results is kept. The name of each of the file is the normal file name extended with _PEEK, so for example Solani_PEEK.txt and solunf_PEEK. The last set of preliminary results will be removed when convergence has been attained or the maximum number of iterations reached The process of storing preliminary results can be avoided by specifying !NOPEEK.

**!PEEKFIRST <iteration number>**
The iteration number at which the preliminary results are stored for the first time can be specified with !PEEKFIRST.

**!PEEKEVERY <number of iterations>**
The number of iterations between storing two subsequent sets of preliminary results can be specified with !PEEKEVERY.

**!PEEKKEEP**
Instead of only keeping the last set of preliminary results, MiXBLUP can also retain each set of preliminary results. In this case, the name of each file is the normal file name extended with the iteration number, so for example Solani_100.txt and solunf_100.txt. This option can be specified with !PEEKKEEP. This option is useful for investigating the causes of unexpected convergence behaviour.

**!RESTART**
The optional qualifier !RESTART can be used to specify that preliminary solutions of an interrupted analysis or old solutions of the previous analysis are to be used as starting values for the new evaluation.

**!GFROMDISK**
The !GFROMDISK qualifier instructs the solver to read the inverse genomic relationship matrix from disk during solving. This was the only option in previous versions of MiXBLUP. The new default is to keep this matrix in memory, which is more demanding for memory requirement, but it saves the time to read this matrix every iteration.

**!WITHINBL <option>**
The optional qualifier !WITHINBL is used in the PRECON section and can be used to use a different preconditioner for the within-block effects than the default preconditioner type. Valid options are b (block-diagonal) and d (diagonal).

**!ACROSSBL <option>**
The optional qualifier !ACROSSBL is used in the PRECON section and can be used to use a different preconditioner for the across-block effects than the default preconditioner type. Valid options are f (full), m (mixed), b (block-diagonal) and d (diagonal).

## 8.2 Control of output

### 8.2.1 General

A successful analysis produces at least a log file and files with solutions to all effects in the model. In some cases, additional results may be required for development or evaluation purposes. Various options are available to specify these additional files when required.

### 8.2.2 Syntax

```
SOLVING
[!BASEANIMALZERO]
[!YHAT]
[!EHAT]
[!YIELDDEV]
[!IDD]
[!DYD]
[!KEEPTMP]
[!SELINDEX <filename>]

TMPDIR <work directory>
```





Sections:

**TMPDIR**
The TMPDIR section can be used to specify an existing folder to store the temporary files of the kernel.

Qualifiers:

**!BASEANIMALZERO**
The estimated breeding values of each individual in Solani.txt or Solani.out can be presented as a deviation of the average of a specified group of individuals, which are referred to as base animals. The animal ID of the base animals should be given in the file BaseAnimals.dat (one animal ID per row). Genetic groups can also be included in the group of base animals by including the genetic group ID (a negative number) in BaseAnimals.dat. The average per trait of the group of base animals is included in the log file MiXBLUP.log. If not all animals or phantom groups are present in the data file, then a warning is given in the log file MiXBLUP.log. The file BaseAnimals.dat must contain the original animal IDs, as they appear in the pedigree file.

**!YHAT**
The optional qualifier !YHAT creates a prediction for each observed trait and each animal in the data file. The predictions are stored in Yhat.txt, which is a text file that contains the animal ID in the first column and predicted observations for each trait in the model in subsequent columns. Missing observations in the data file get the code -8192.0 in the file with predictions.

**!EHAT**
The optional qualifier !EHAT stores the residual term of each observed trait and each animal in the data file. The residuals are stored in Ehat.txt, which is in text format and contains the animal ID in the first column and residuals for each trait in the model in subsequent columns. Missing observations in the data file get the code -8192.0 in the file with residuals.

**!YIELDDEV**
The optional qualifier !YIELDDEV stores the observation corrected for fixed effects and non-genetic random effects for each observed trait and animal in the data file. The yield deviations are stored in YD.txt, which is in text format and contains the animal ID in the first column and yield deviations for each trait in the model in subsequent columns. Missing observations in the data file get the code -8192.0 in the output file.

**!IDD**
The optional qualifier !IDD ("individual daughter deviation") stores the yield deviation corrected for the genetic contribution of the dam for each observed trait and animal in the data file. The individual sire progeny deviations are stored in IDD.txt, which is in text format and contains the animal ID in the first column and yield deviations for each trait in the model in subsequent columns. Missing observations in the data file get the code -8192.0 in the output file.

**!DYD**
The optional qualifier !DYD ("daughter yield deviation") stores the individual yield deviations averaged by sire. The daughter yield deviations are stored in soldyd.txt, a text file that contains sire in the first column, the number of progeny included in the progeny yield deviation in the fourth column, the progeny yield deviations by trait followed by the same number of fields to indicate whether the progeny yield deviation of the corresponding trait is valid (value is 1) or not (value is 0).

**!KEEPTMP**
The optional qualifier !KEEPTMP can be used to stop the removal of temporary files at the end of an analysis, for example to check for possible errors. The default is that all large temporary files are deleted as soon as they are no longer requird.

**!SELINDEX <filename>**
The qualifier !SELINDEX can be used to automatically calculate a selection index value as the sum of weighted genetic solutions (weighted EBV). The selection index value is added as an additional column in the Solani output file. The file specified after the qualifier contains the selection index weighting factor for each combination of genetic effect and trait in the model. The syntax is <trait>(<genetic effect>) <selection index weighting factor>, for example: phen1(animal) 1.0.

# 9. Reliabilities

Besides estimating genetic effects (or breeding values), MiXBLUP supports a second type of analysis to quantify the amount of information available to estimate the genetic effect of each individual. This is expressed as the reliability of the estimated breeding value. This chapter describes how approximate reliabilities can be calculated with MiXBLUP.

## 9.1 General

A reliability is a measure of the information that is available for the estimate of a breeding value. The reliability is dependent on the heritability and the presence of observations for the individual itself. The biggest impact on the reliability comes from the number of progeny with observations.

MiXBLUP supports the calculation of the approximate reliability of the EBVs of animals for most statistical models.

| Type of evaluation | Reliabilities | Remarks |
|---|---|---|
| Basic statistical model (chapter 7.1) | direct | |
| Maternal genetic model (chapter 7.2) | direct & maternal | |
| Social interaction model (chapter 7.3) | direct & indirect | |
| Random regression model (chapter 7.4) | direct | Only for calculated totals |
| Weighted residuals (chapter 7.5) | not supported | |
| Pedigree relationships (chapter 5.1 – 5.3) | direct & indirect | |
| Relationships based on pedigree & marker haplotypes | not supported | |
| Genomic relationships (chapter 5.5-5.9) | pedigree + genomic | |
| SNP covariates (chapter 5.10-5.11) | pedigree + genomic | Using the method of genomic relationships |

Approximate pedigree and genomic reliabilities are calculated for families within blocks. It is a completely different process than estimating breeding values. Approximate reliabilities are calculated in a separate analysis.

## 9.2 The concept of blocks in the reliability calculation in MiXBLUP

### 9.2.1 Block variable

The calculation of reliabilities in MiXBLUP requires the use of a family block variable. The objective of using a block variable is to minimise the use of memory during the calculation by ordering of equations in equation family blocks (Lidauer and Strandén, 1999). Using a block variable has no benefit for breeding value estimation with MiXBLUP.

To take advantage of this concept, it is important that each equation family block contains as many closely connected equations as possible, and that the number of equations connecting equation family blocks is as low as possible. A family consists of an individual, its parents and its progeny.





### 9.2.2 Common-block variable

Equations that connect all (or many) equations, such as individuals with progeny in many different blocks, can be grouped into one or several common blocks. Common blocks refer to blocks of equations, which will be kept in memory for all families. Equations of common blocks must be ordered to appear at the bottom of the MME, i.e. animals of common blocks must have largest block sorting variables. This ordering of mixed model equations yields for many animal breeding problems a structured coefficient matrix that has nearly doubly-bordered block diagonal form. The number of common blocks can be specified by the user. The default is that no common blocks are used. MiXBLUP will solve the mixed model equations even if such a structure cannot be achieved. However, pre-processing time may be longer than usual. As a consequence, it is important to keep this concept in mind when editing the data for the approximation of reliabilities.

### 9.2.3 Sorting data and pedigree file on block variable

The concept of equation family blocks requires that data and pedigree records be sorted on the block variable. MiXBLUP automatically checks whether files are sorted and sorts them if necessary.

### 9.2.4 Strategies for block definition

For example in dairy cattle, equations for animals in the same herd represent an equation family. Many models across species contain such effects and are therefore suitable block variables to be used to group equations into equation families.

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

The MiXBLUP kernel reads the data by blocks with one or several blocks at a time. If there is only one block in a large data file, all iteration files are read into the random access memory at the same time, which might exhaust computer resources. If the data can be read into the memory then this may be sensible.

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

When an animal does not belong to any equation family (no observations to give block code), or it is in many different families through relationship information (e.g. dairy sires have progeny in many herds), a new block code should be given. It is recommended to use a separate block code for animals with links to many different equation family blocks. For instance, sires in a dairy cattle population can be assigned to one block. These blocks should be defined as common blocks and largest block code variables have to be given to these blocks. Thus, sorting by the block code

variable will ensure that animals of common blocks will appear at the bottom of the MME.

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

## 9.3 Differences between the syntax of reliability calculation and breeding value estimation

### 9.3.1 Data file

For a reliability calculation, every animal in the evaluation has to be uniquely assigned to a level of the block variable. This block variable must be present in the data file.

### 9.3.2 Genetic similarity between individuals

The level of the block variable of an animal is also required in its record in the pedigree file. Genetic groups are ignored in the reliability calculation and replaced with unknown parents.

If genomic information is available, too, the external relationship matrix must be set up, which can be done by specifying !CONSTRUCT SSMAT !SINGLESTEP.

### 9.3.3 Statistical model

The statistical model for reliability calculation contains a single fixed effect that is treated as nested within blocks, even if it is an across-block effect. It should be the fixed effect with the largest impact on the reliability, so the lowest average number of observations within a class. The use of !WEIGHT is not supported.

### 9.3.4 Control of analysis

A reliability calculation is triggered with !RELIABILITY in the SOLVING section.





## 9.4 Syntax

```
DATAFILE <file name>
<ID> I/A
...
<block code> I !BLOCK

[ERMFILE <file name> !CONSTRUCT SSMAT !SINGLESTEP
<ID> I/A
<qualifiers>]

PEDFILE <file name>
<ID> I/A
...
<block code> I !BLOCK

MODEL
<trait1> ~ BL(<largest fixed class effect trait1>) !RANDOM <random effects> G(<ID>)
<trait2> ~ BL(<largest fixed class effect trait2>) !RANDOM <random effects> G(<ID>)

SOLVING
!RELIABILITY
[!MAXNONZ <number>]
[!NCOMBLK <number>]
[!KEEPTMP]
```

**!MAXNONZ**
The optional qualifier !MAXNONZ can be used to set the maximum number of non-zeroes, which effectively determines the memory allocated to an evaluation of reliabilities. It has no effect if !RELIABILITY is not specified. The default value is 9,000,000. The maximum value that can be entered for !MAXNONZ is determined by the maximum integer of the operating system or the memory available, whichever is limiting.

**!NCOMBLK <number>**
If common blocks are used, the qualifier !NCOMBLK should be used to specify how many common blocks there are. Common blocks should have a block identification that positions the block at the end of the list of blocks. The specified number of blocks at the end of the list of blocks are considered common blocks.

Qualifiers other than !MAXNONZ, !NCOMBLK and !KEEPTMP have no meaning when !RELIABILITY is specified, and will be ignored.

## 9.5 Associated output files

| Output file | Description |
|---|---|
| Relani.txt | The reliability of direct genetic effects of traits in the model. The exact layout of Relani.txt is printed at the end of reliabilities.log. |
| Relani.out | As Relani.txt, but for alphanumerical ID labels |
| Relani_indirect.txt | The reliability of indirect genetic effects of traits in the model. The exact layout of Relani_indirect.txt is printed at the end of reliabilities.log. |
| Relani_indirect.out | As Relani_indirect.txt, but for alphanumerical class labels |

# 10. Running MiXBLUP

This chapter describes practical issues when analyzing data with MiXBLUP.

## 10.1 Starting a MiXBLUP evaluation

Solving mixed model equations using MiXBLUP involves execution of three out of four programs. The main executable is MiXBLUP.exe. This is the parser and calls dataprocessor.exe and either solver.exe or reliabilities.exe. MiXBLUP is started with the command

```
MiXBLUP <name instruction file>
```

Or in Linux:

```
MiXBLUP.exe <name instruction file>
```

The command may be given in two ways:

1. by entering the command on the command line
2. in batch mode for single or multiple analyses

If MiXBLUP.exe is run from the command line and no instruction file is specified, MiXBLUP will ask for the name of instruction file. If MiXBLUP.exe is run in a batch mode, make a batch file (mixblup.bat) that contains one or more commands as specified above.

## 10.2 Choosing a breeding value evaluation or a reliability calculation

MiXBLUP either estimates breeding values, using solver.exe, or calculates approximate reliabilities, using reliabilities.exe. The type of analysis is controlled with the !RELIABILITY qualifier in the SOLVING section in the instruction file. If it is omitted, a breeding value analysis is started. If it is specified, a reliabilities calculation is started. See Chapter 9 for the additional changes in the instruction file when a reliabilities analysis is required.

## 10.3 A breeding value analysis with previous solutions as starting values

In case of large evaluations of breeding values, there may be a substantial saving in time to convergence of 10-30% by using the previous solutions as starting values for the current evaluation. This is activated by specifying the !RESTART qualifier in the SOLVING section. It does not have an effect in case of a reliabilities analysis.

The only additional file necessary for using previous solutions is the Solunf file. If !RESTART is specified, MiXBLUP renames the file Solunf to Solold. If the file Solold is present, the pre-processor dataprocessor will create a file Solvec. This file will be used to initialise the solution vector of the mixed-model equations before the start of the iteration process (in solver).

If any of the effects in the statistical model has been defined with field type A (alphanumerical labels), then the file Code.inp of the previous analysis must be present, too. The file Code_index.inp of a previous analysis is not used for a restart.





## 10.4 Monitoring and checking the process

When developing new analyses, it may be useful to monitor the progress of the analysis. This can be specified with –Ds on the command line, for example

```
MiXBLUP TestRun.inp –Ds
```

After the run is finished, it is worth to look through the various log-files. In MiXBLUP.log some information is given about pre- and post-processing of the data. It also lists error messages, if any.

If a mistake is made somewhere or the variance-covariance matrix, variance-covariance matrix, is not positive definite, it is likely that at least the solver.exe does not run. Check in that case the dataprocessor.log, because it may give some indications for errors.

If all programs have run successfully, it is worth to check the solver.log, to see how the convergence was reached. In cases with poor convergence, it will give a warning and some model checking may be appropriate. When reliabilities are calculated, one can check the reliabilities.log, or reliabilities_direct.log and reliabilities_indirect.log when reliabilities are calculated for both direct and indirect genetic effects, such as maternal genetic effects.

## 10.5 Interrupting a process of the kernel

A process of the MiXBLUP kernel can be interrupted by placing an empty file with file name STOP in the folder with the executables and instruction file. After every iteration, the MiXBLUP kernel checks whether this file is present. If so, it will start producing the output files as if convergence had been attained, instead of the next iteration, and it will stop afterwards.

# 11. Output files

This chapter gives an overview of the most useful output files of MiXBLUP.

## 11.1 Solution files

The solution files are split up into standard output files and post-processed solution files. Post-processed solution files are generated only in specific cases, such as alphanumeric data, use of a genetic-base population (!BASEANIMALZERO) or marker-assisted breeding value estimation with IBD-matrices. The tables in this section refer to the layout of the output files of the general example used throughout this manual.

### 11.1.1 Standard output files

The exact layout of the standard output files is listed at the end of solver.log, reliabilities.log or reliabilities_indirect.log.

| | |
|---|---|
| Solfix.txt | Contains solutions of fixed effects across blocks |
| Solani.txt | Contains solutions of the direct and indirect genetic effects. These solutions are the estimated breeding values or EBVs. If maternal genetic effects or social genetic effects are present, the indirect EBV can be found after the columns of the direct EBV. |
| Relani.txt | Contains reliabilities of direct EBVs |
| Relani_indirect.txt | Contains for reliabilities of indirect EBVs |
| Solr01.txt | Contains solutions for the first non-genetic random effect |
| Solf01.txt | Contains solutions for the first fixed effect within blocks (using BL(... )) |
| Solreg.txt | Contains estimated regression coefficients for covariates in the data file |
| Solreg_mat.txt | Contains estimated regression coefficients for covariates in a separate covariate file or regression design matrix file. |
| Solunf | Contains all solutions to the mixed-model equations (MME) in binary format. Solutions in this file can be used as starting values for a subsequent evaluation when more data has been added. If !RESTART is used, the file Solunf is renamed to Solold. If Solold is present at the start of the pre-processor dataprocessor.exe, it will create a file with a solution vector called 'Solvec'. This file will be used to pre-set the solution vector of the MME before the start of the iteration process (by solver.exe). |

### 11.1.2 Post-processed output files

If alphanumeric labels exist in data and/or pedigree file, they will be recoded to integer numbers to be able to run dataprocessor.exe and solver.exe. After solver.exe has finished, all solutions are decoded back to their original alphanumeric codes in MiXBLUP. Solution files with the integer codes have file extension **.txt**. The solution files decoded to the original labels have extension **.out**. For example, the file Solani.out contains the EBVs with the original individual ID, if the ID was specified to have field type A. If fixed effects contain alphanumeric data, Solfix.out is created giving the solutions of the fixed effects recoded back to alphanumeric codes. The format is the same as that of Solfix.txt. If other random effects are alphanumeric, the Solr#.out contains the solutions for the levels of the random effect recoded back to alphanumeric codes. The format is the same as that of Solr#.txt.

When the option !BASEANIMALZERO is used, Solani.out contains the EBV after deducting the average of the genetic base population. In this case, the solutions in Solani.out and Solani.txt are different.





When marker-assisted breeding value estimation with IBD-matrices is performed, MiXBLUP creates EBVhap# with the estimated haplotype effects for each animal for QTL#. The total EBV is given in EBVtot and is calculated per animal as the sum of the polygenic effect and the haplotype effects. The polygenic EBV and QTL-EBV are equally weighted. The format is simply the animal ID followed by the total EBV for all traits. The order of the traits is the same as in the Solani.txt file.

## 11.2 Log files

| | |
|---|---|
| MiXBLUP.log | Log-file of MiXBLUP parser |
| OK_dataprocessor | Dataprocessor ran successfully |
| OK_solver | Solver ran successfully |
| OK_reliabilities | Reliabilities ran successfully |
| Warning.log | Log-file with warnings, if any |
| MiXBLUP.lst | Contains a short summary of dataprocessor and summary statistics of the data |
| Dataprocessor.log | Extensive log-file of Dataprocessor, gives also errors |
| Solver.log | Log-file of Solver. It gives the convergence and a description of output files |
| Reliabilities.log | Log file of reliabilities for direct genetic effects |
| Reliabilities_indirect.log | Log file of reliabilities for maternal genetic effects |
| Memory.txt | Contains information about the amount of random access memory used during the execution of the programs. In practice, the amount actually is often larger due to memory overhead and depends on the computer. |
| Modlog.txt | Contains model and data parameters. |

In some cases, the log files cannot be read by Notepad. In that case, use of other text editor software programs, such as ConTEXT or Programmer's File Editor (both freely available), are recommended.

## 11.3 Temporary files

| | |
|---|---|
| Instruction.inp | Temporary file used – contains an updated version of the input file for the parser |
| Data99.tmp | Tempory data file used when converting alphanumeric data |
| Data.txt | Transformed data file used by dataprocessor |
| Ped99.tmp | temporary pedigree file when converting alphanumerical IDs |
| Pedigree.txt | Transformed pedigree file used by dataprocessor |
| Covar.txt | Parameter file with variance-covariance matrices in dataprocessor format |
| Dataprocessor.inp | Instruction file for dataprocessor |
| Stopping_criteria.inp | File with stopping criteria for solver |
| Solani.tmp | Intermediate version of Solani.txt file (only in case of alphanumerical coding or when the baseanimalszero option is used) |
| Code.inp | In case of alphanumerical coding, the line number of the string is the code that corresponds to the value. |
| Code_index.inp | Contains the hash value of each label. The first number is the maximum number of alphanumerical labels. Line+1 corresponds with line in Code.inp. This file is only used during a run and obsolete after a run has completed. |

## 11.4 Reserved filenames

The names of the files in the previous sections cannot be used as a name of a file that the user provides. In most cases, MiXBLUP will check for the use of reserved filenames and stop the analysis, if a reserved filename is used as a data file, pedigree file, trait covariance matrix file or haplotype covariance matrix file.

# 12. Tuning MiXBLUP

MiXBLUP may give warnings and errors. This chapter describes the most commonly observed problems and potential solutions, as well as suggestions to improve the performance of MiXBLUP.

## 12.1 Trouble-shooting

### 12.1.1 Problems related to the license

If the license cannot be found, is out of date or inappropriate for the type of analysis, an error message will appear in the screen log and in MiXBLUP.log. It can be resolved by specifying the correct path in LicDir.inp or obtaining a current and appropriate license via MiXBLUP@wur.nl.

### 12.1.2 Underlying executables not found

If one or more of the underlying executables are not found, an error message will appear in the screen log and in MiXBLUP.log at the start of the analysis. It can be resolved by specifying the correct path in SysDir.inp.

### 12.1.3 Problems with the syntax of the instruction file

Incorrect syntax of the instruction file generally results in an error message in the screen log and in MiXBLUP.log. It can be resolved by looking up the corresponding chapters in this manual and correcting the syntax.

### 12.1.4 Problems of reading and writing input files

If one or more lines in an input file do not match the specification in the instruction file, MiXBLUP generally produces an error message in the screen log and in MiXBLUP.log. Not all types of errors, however, are always detected. If a record has more columns than specified it may go unnoticed, as redundant columns are generally ignored. A pedigree file containing both a code for unknown parents and genetic groups will not be detected, but may give a seemingly unrelated error. It is up to the user to avoid this to happen.

### 12.1.5 Problems in calc_grm.exe

If calc_grm.exe encounters a problem, the error message will be printed in ERMcalc_grm. log. MiXBLUP only detects that ExtRelMat.txt or another expected output file does not exist and this message will be printed in the screen log and in MiXBLUP.log. This can be resolved by addressing the problem detected by calc_grm.exe.

### 12.1.6 Problems in dataprocessor.exe

Dataprocessor performs a wide range of checks. Any detected problems are written to dataprocessor.log. MiXBLUP will point the user to this file if dataprocessor.exe does not complete normally.

### 12.1.7 Problems in solver.exe or reliabilities.exe

Errors in solver.exe are unlikely and generally related to the system environment such as insufficient memory or disk space. The same is true for reliabilities.exe, but too low a maximum number of non-zeroes specified may also give an error. The latter is easily resolved by





specifying a larger number for !MAXNONZ. If solver.log or reliabilities.log do not specify an error message, the problem is likely to be system-related.

### 12.1.8 Feedback on ease to resolve encountered errors

Feedback on the helpfulness (or lack of it) of an error message to resolve an encountered problem is welcomed by the developers of MiXBLUP. Please send any comments and suggestions to MiXBLUP@wur.nl.

## 12.2 Variance covariance matrix not positive definite

The dataprocessor.exe checks whether the used variance-covariance matrices given in the variance-covariance file are positive definite. If the matrix is not positive definite, the dataprocessor.exe will stop immediately and gives in dataprocessor.log an indication which variance –covariance matrix is not positive definite. The user should bend this matrix, e.g. with existing methods as presented in literature (Hayes and Hill, 1981; Jorjani et al., 2003). The best practice is to check beforehand the eigenvalues of all matrices.

## 12.3 Convergence problems

In all cases it is wise to check the solver.log to see the convergence characteristics. It will give an indication whether the convergence was poor or slow. In some cases MiXBLUP does not converge easily or does not converge at all. In those cases, a number of things should be checked:

- Check whether fixed effects are confounded amongst each other
- Check whether fixed effects and phantom parent groups are confounded
- Does the run include traits that are highly correlated (genetic correlation > 0.9)?
- A mismatch between used variance-covariances and the current values, e.g. when using an old set of variances and covariances.
- A useful comparison is to compare estimates of EBV and fixed effects of different programs, e.g. MiXBLUP and ASREML/DMU.

**A few solutions to problems**

- Simplification of the model, e.g. remove some confounded fixed effects.
- Add a value to diagonal of phantom parent groups to regress poorly estimable phantom parents groups back to mean (!groups value). This value could be like 1.0, 3.0 or 5.0. The higher the value the more the estimate is regressed towards the mean (Schaeffer, 1994).
- Re-estimate all variances and covariances.
- If traits are highly correlated, traits might be combined to one trait and observations might be used as repeated observations (repeatability model).
- Very high correlations (> 0.90) might be bended to values of 0.90.
- Split up the evaluation into a few evaluations, e.g. of groups of traits that are more correlated amongst each other because of biological similarity.

## 12.4 Optimisation of memory and time

In very large genetic evaluations with millions of records and animals in the pedigree, it may help to put fixed effects with a large number of equations within block with the function BL(fixed effect). Herd-year-season effect is an example of an effect that is best placed in blocks of herd or groups of herds.

Especially for a reliability analysis involving a large number of animals, it is worth investigating the optimal blocking strategy for faster completion of the analysis.

In some cases the memory allocation is limited by the maximum integer in the 32-bit version of the software. The maximum integer is much larger in the 64-bit version of the software. When 64-bit computers are available, the 64-bit version of the software should be used so that a larger amount of memory can be utilized.

Very large genetic evaluations can be split into a number of smaller evaluations, if the analysis time of a single analysis is too long, especially if there are groups of traits with relatively low genetic correlations between groups or if there are unrelated populations not sharing fixed or random effect classes.





# 13. References

Fernando, RL, and M. Grossman, 1989. Marker assisted selection using best linear unbiased prediction. Genet. Sel. Evol., 21: 467-477.

Fernando, R.L., J.C. Dekkers, D.J. Garrick. 2014. A class of Bayesian methods to combine large numbers of genotyped and non-genotyped animals for whole-genome analyses. Genet Sel Evol 46.

Fragomeni, BO, DAL Lourenco, S Tsuruta, Y Masuda, I Aguilar, A Legarra, TJ Lawlor, I Misztal, 2015. Hot topic: Use of genomic recursions in single-step genomic best linear unbiased predictor (BLUP) with a large number of genotypes. J. Dairy Sci. 98: 4090-4094.

Hayes, J.F., and W.G. Hill, 1981. Modification of estimates of parameters in the construction of genetic selection indices ("Bending"). Biometrics 37: 483-493.

Lidauer, M. and Strandén, I. (1999). "Fast and flexible program for genetic evaluation in dairy cattle". In: INTERBULL Bulletin. 20. Tuusula, Finland, pp. 20–25.

Jorjani, H., L. Klei, and U. Emanuelson. 2003. A simple method for weighted bending of genetic (co) variance matrices. J. Dairy Sci. 86:677-679.

Stranden, I. and O. Christensen. 2011. Allele coding in genomic evaluation. Genet Sel Evol. 43:25

Schaeffer, L.R. 1994. Multiple-Country Comparison of Dairy Sires. J. Dairy Sci.77:2671-2678

# 14. Acknowledgments

**Release 1.0 – April 2010**
The authors of the manual and software would like to acknowledge the financial support of SABRE, CRV, IPG and HG and the European Commission, within the 6th Framework project SABRE, contract No. FOOD-CT-2006-016250. The text represents the authors' views and does not necessarily represent a position of the Commission who will not be liable for the use made of such information.

In addition the authors would like to thank a number of people that have helped at different stages and different tasks in developing the software and manual. First of all, we would like thank Robin Thompson for his guidance, technical knowledge and support during the whole project.

In addition, we would like to thank Kaarina Matilainen, Rudi de Mol, Wijbrand Ouweltjes and Marco Pool for programming different parts in the MiXBLUP software.

John Voskamp, Noelle Hoorneman, Lucia Kaal and Paul Goethals are acknowledged for their contributions to the manual.

Addie Vereijken, Dieuwke Roelofs-Prins, Egbert Knol, Marc Rutten, Rob Bergsma, Saskia Bloemhof, Abe Huisman and Chris Schrooten are acknowledged for testing the software in a commercial environment and discussing implementation of new features.

Finally, Debbie Bohte-Wilhelmus and Evert van Steenbergen are acknowledged for testing early versions of the MiXBLUP software for various applications.

**Release 1.3 – October 2013**
Annemieke Sprangers is acknowledged for testing new features of the software and critically reading the draft of the manual.

**Release 2.0 – January 2016**
The authors acknowledge the financial support from the Breed4Food consortium. Breed4Food is dedicated to be the leading research consortium in animal breeding, genetics and genomics enabling the Breed4Food partners to breed better products to benefit society's needs. Hendrix Genetics, Topigs Norsvin, CRV, Cobb Europe and LUKE (formerly MTT) are actively involved in this research programme.

Birgit Zumbach, Susan Wijga, Annemieke Sprangers and Addie Vereijken are acknowledged for testing the various new features in this release and making suggestions for improvements.

Jérémie Vandenplas is acknowledged for making substantial improvements in efficiency to the calc_grm software and for helpful technical discussions.





**Release 2.1 – June 2017**
Ghyslaine Schopen and Coralia Manzanilla are acknowledged for testing new features of the software. Ghyslaine Schopen and Dianne van der Spek are acknowledged for critically reading the draft of the manual.

**Release 2.1 - November 2018**
Ilse van Grevenhof is acknowledged for helpful discussions. Kaarina Matillainen is acknowledged for programming the threshold model in the MiXBLUP kernel and Topigs Norsvin and Hendrix Genetics are acknowledged for testing it.

**Release 2.2 - May 2020**
Rianne van Binsbergen is acknowledged for reading the draft of the manual.

# Appendix. Examples

Example numbers refer to the paragraph describing the feature.

### Example 4.1 Data file specification

```
TITLE breeding value estimation for phen1 and phen2 using pedigree

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99 !SLASH
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





### Example 4.2.3 Existing covariate table file in addition to data file

```
TITLE random regression of phen1 on indep using pedigree and new covariate table

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    indep   I  !CVRIND
    phen1   T
    phen2   T
    blk     I

CVRTABLE ExampleCVR.txt

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1*CVR(5) !RANDOM ran*CVR(3) G(animal*CVR(2))

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

## Example 4.2.4 Covariate table file in addition to data file

```
TITLE random regression of phen1 on indep using pedigree and new covariate table

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    indep   I  !CVRIND
    phen1   T
    phen2   T
    blk     I

CVRTABLE !CVRMAKE LEG !CVRNUM 5 !CVRMIN 2 !CVRMAX 43

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1*CVR(5) !RANDOM ran*CVR(3) G(animal*CVR(2))

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





## Example 4.3 General covariate file in addition to data file

```
TITLE analysis of phen1 and phen2 using pedigree and general covariate file

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

REGFILE
    REG01 ExampleCov.txt !REGTYPE R  # default column animal is 1; use all covariates

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

REGPARFILE ExampleCovPar.txt

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran REG(1) G(animal)
    phen2   ~ fix2 cov !RANDOM ran REG(1) G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

## Example 5.1 Pedigree file, single code for unknown parents & ignoring inbreeding

```
TITLE breeding value estimation using pedigree with single code for unknown parents

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt  # unknown parents coded as 0
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





## Example 5.2 Pedigree file with genetic groups for unknown parents & ignoring inbreeding

```
TITLE breeding value estimation using pedigree with genetic groups

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
#   unknown parents coded as negative integers for genetic groups
PEDFILE ExamplePedGG.txt !GROUPS 1.0
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

### Example 5.3.3 Pedigree file accounting for newly calculated inbreeding coefficients

```
TITLE breeding value estimation using pedigree and calculated inbreeding coefficients

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt !CALCINBR
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





### Example 5.3.4 Pedigree file accounting for inbreeding using existing file

```
TITLE breeding value estimation using pedigree and existing inbreeding file

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

INBRFILE ExampleInbr.txt !IDCOL 1 !INBRCOL 2

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

## Example 5.4 Pedigree file and marker haplotypes

```
TITLE breeding value estimation using pedigree and marker haplotypes

# Observations & systematic effects
DATAFILE ExampleDatMA.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I
    mrk1_1 I
    mrk1_2 I
    mrk2_1 I
    mrk2_2 I

# Genetic similarity among individuals
PEDFILE ExamplePedMA.txt
    animal  A
    sire    A
    dam     A
    mrk1_1 I
    mrk1_2 I
    mrk2_1 I
    mrk2_2 I
    blkped  I

CVMATRIX
    ExampleCVmat1.txt
    ExampleCVmat2.txt

# Components of variance and covariance among traits
PARFILE ExampleParMA.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran GIV(mrk1_1 AND mrk1_2,1) GIV(mrk2_1 AND mrk2_2,2) &
      G(animal)
    phen2   ~ fix2 cov !RANDOM ran GIV(mrk1_1 AND mrk1_2,1) GIV(mrk2_1 AND mrk2_2,2) &
      G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

Additional matrices in parameter file ExampleParMA.txt (GIV**1** corresponds with GIV(... AND ..., **1**) and the **first** CVmatrix file, etc.):

```
GIV1
phen1 0.1
phen2 0.0 0.1

GIV2
phen1 0.1
phen2 0.0 0.1
```





## Example 5.5 Existing external relationship matrix file

```
TITLE breeding value estimation using existing external relationship matrix file

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
ERMFILE ExampleERM.txt
    animal  A
# !ASIS     # optional; no checks, but cannot be used if animal has field type A
!NOORIG     # optional; does not create ExtRelMatOrig.txt

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

## Example 5.6 Genomic relationship matrix calculated from genotype file (GBLUP)

```
TITLE genetic evaluation using genomic relationship matrix set up from genotypes
#     GBLUP: all animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDatGeno.txt !MISSING -99  # data reduced to genotyped animals only
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
ERMFILE ExampleGeno.txt !CONSTRUCT Ginv
   animal  A
!METHOD VanRaden2               # optional; default VanRaden2
!ALFREQ ExampleAlFreq.txt       # optional; default calculated from data
# !CROSSBRED 3 ExampleBreeds.txt # optional; default single breed
# !BREEDS_UNRELATED   # optional; default all genomic relationships are considered
# !ALLELES 1                    # optional; default genotypes
# !DENSE                        # optional; default string of markers in free format
!NMARK 1000                     # optional; default is all markers
!MAF 0.01                       # optional; default is 0.005
!STORE_GINV                     # optional; default is no storing
!NUMPROC 12                     # optional; default is 1
# !BACKSOLVE                    # optional; default is no backsolving

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```





## Example 5.7.3 Full blended inverse relationship matrix calculated from pedigree and genotype file (ssGBLUP)

```
TITLE Full blended inverse relationship matrix set up from genotypes
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
ERMFILE ExampleGeno.txt
   animal  A
!CONSTRUCT Hinv                 # full H-inverse prior to solving
!METHOD VanRaden2               # optional; default VanRaden2
# !ALFREQ ExampleAlFreq.txt       # optional; default calculated from data
# !CROSSBRED 3 ExampleBreeds.txt   # optional; default single breed
# !BREEDS_UNRELATED   # optional; default all genomic relationships are considered
# !ALLELES 1                      # optional; default genotypes
# !DENSE                          # optional; default markers in free format
# !NMARK 1000                     # optional; default is all markers
!MAF 0.01                       # optional; default is 0.005
# !STORE_GINV                     # optional; default is no storing
!NUMPROC 12                     # optional; default is 1
!LAMBDA 0.85                    # optional; default 1.0
!ALPHA 0.95                     # optional; default is 1.0
!BETA 0.05                      # optional; default is 1.0 - ALPHA
!OMEGA 0.90                     # optional; default is LAMBDA
!USE_GINV G_asreml.giv          # optional; default is calculating G-inverse anew

ERMPEDFILE ExamplePed.txt
   animal A

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```

## Example 5.7.4 New weighted inverse genomic relationship matrix calculated from genotype file (ssGBLUP)

```
TITLE New weighted inverse genomic relationship matrix calculated from genotype file
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
ERMFILE ExampleGeno.txt
   animal  A
!CONSTRUCT SSMAT                 # lambda*inv(alpha*G+beta*A22)-omega*inv(A22)
!SINGLESTEP                      # add A-inverse during solving
!METHOD VanRaden2                # optional; default VanRaden2
# !ALFREQ ExampleAlFreq.txt        # optional; default calculated from data
# !CROSSBRED 3 ExampleBreeds.txt   # optional; default single breed
# !BREEDS_UNRELATED   # optional; default all genomic relationships are considered
# !ALLELES 1                       # optional; default genotypes
# !DENSE                           # optional; default markers in free format
# !NMARK 1000                      # optional; default is all markers
!MAF 0.01                        # optional; default is 0.005
!NUMPROC 12                      # optional; default is 1
!LAMBDA 0.85                     # optional; default 1.0
!ALPHA 0.95                      # optional; default is 1.0
!BETA 0.05                       # optional; default is 1.0 - ALPHA
!OMEGA 0.90                      # optional; default is LAMBDA

PEDFILE ExamplePed.txt !CALCINBR
   animal A
   sire   A
   dam    A
   blkped I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```





## Example 5.7.5 Existing weighted inverse genomic relationship matrix calculated from genotype file (ssGBLUP)

```
TITLE New weighted inverse genomic relationship matrix calculated from genotype file
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
#   use an ExtRelMatOrig.txt created with !CONSTRUCT SSMAT !SINGLESTEP
ERMFILE ExtRelMatOrig.txt
   animal  A
!SINGLESTEP                      # add A-inverse during solving
!LAMBDA 0.85                     # optional; default 1.0
!OMEGA 0.90                      # optional; default is LAMBDA

PEDFILE ExamplePed.txt !CALCINBR
   animal A
   sire   A
   dam    A
   blkped I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```

## Example 5.8 Weighted inverse genomic relationship matrix (ssGBLUP) using genetic groups

```
TITLE New weighted inverse genomic relationship matrix using genetic groups
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
ERMFILE ExampleGeno.txt
   animal A
!CONSTRUCT SSMAT                 # lambda*inv(alpha*G+beta*A22)-omega*inv(A22)
!SINGLESTEP                      # add A-inverse during solving
!METHOD VanRaden2                # optional; default VanRaden2
# !ALFREQ ExampleAlFreq.txt        # optional; default calculated from data
# !CROSSBRED 3 ExampleBreeds.txt   # optional; default single breed
# !BREEDS_UNRELATED   # optional; default all genomic relationships are considered
# !ALLELES 1                       # optional; default genotypes
# !DENSE                           # optional; default markers in free format
# !NMARK 1000                      # optional; default is all markers
!MAF 0.01                        # optional; default is 0.005
!NUMPROC 12                      # optional; default is 1
!LAMBDA 0.85                     # optional; default 1.0
!ALPHA 0.95                      # optional; default is 1.0
!BETA 0.05                       # optional; default is 1.0 - ALPHA
!OMEGA 0.90                      # optional; default is LAMBDA

PEDFILE ExamplePedGG.txt !CALCINBR
   animal A
   sire   A
   dam    A
   blkped I
!GROUPS 1.0                      # affects both the PEDFILE and the ERMFILE sections

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```





## Example 5.9.3 New APY inverse genomic relationship matrix using a random core

```
TITLE New APY inverse genomic relationship matrix using a random core
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
#  APY inverse of G and avoiding explicit inverse of A22 matrix
ERMFILE ExampleGeno.txt
   animal  A
!CONSTRUCT SSMAT !APY            # lambda*APYinverse(alpha*G + beta*A22)
!SINGLESTEP                      # add A-inverse during solving
!APYCORERAN 2435                 # choose 2435 animals for core at random
# !METHOD VanRaden2                # optional; default VanRaden2
# !ALFREQ ExampleAlFreq.txt        # optional; default calculated from data
# !CROSSBRED 3 ExampleBreeds.txt   # optional; default single breed
# !BREEDS_UNRELATED   # optional; default all genomic relationships are considered
# !ALLELES 1                       # optional; default genotypes
# !DENSE                           # optional; default markers in free format
# !NMARK 1000                      # optional; default is all markers
# !MAF 0.01                        # optional; default is 0.005
!NUMPROC 12                      # optional; default is 1
!LAMBDA 0.85                     # optional; default 1.0
!ALPHA 0.95                      # optional; default is 1.0
!BETA 0.05                       # optional; default is 1.0 - ALPHA
!OMEGA 0.90                      # optional; default is LAMBDA

PEDFILE ExamplePed.txt !CALCINBR
   animal A
   sire   A
   dam    A
   blkped I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```

## Example 5.9.4 New APY inverse genomic relationship matrix using a predefined core

```
TITLE New APY inverse genomic relationship matrix using a predefined core
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
#  APY inverse of G and avoiding explicit inverse of A22 matrix
ERMFILE ExampleGeno.txt
    animal  A
!CONSTRUCT SSMAT !APY             # lambda*APYinverse(alpha*G+beta*A22)
!SINGLESTEP                       # add A-inverse during solving
!APYCORELIS ExampleCore.txt       # read animals for core from file
# !METHOD VanRaden2                 # optional; default VanRaden2
# !ALFREQ ExampleAlFreq.txt         # optional; default calculated from data
# !CROSSBRED 3 ExampleBreeds.txt    # optional; default single breed
# !BREEDS_UNRELATED   # optional; default all genomic relationships are considered
# !ALLELES 1                        # optional; default genotypes
# !DENSE                            # optional; default markers in free format
# !NMARK 1000                       # optional; default is all markers
# !MAF 0.01                         # optional; default is 0.005
!NUMPROC 12                       # optional; default is 1
!LAMBDA 0.85                      # optional; default 1.0
!ALPHA 0.95                       # optional; default is 1.0
!BETA 0.05                        # optional; default is 1.0 - ALPHA
!OMEGA 0.90                       # optional; default is LAMBDA

PEDFILE ExamplePed.txt !CALCINBR
    animal A
    sire   A
    dam    A
    blkped I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





## Example 5.9.5 New APY inverse genomic relationship matrix using a random core determined by PCA

```
TITLE New APY inverse genomic relationship matrix using a random core set by PCA
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
#  APY inverse of G and avoiding explicit inverse of A22 matrix
ERMFILE ExampleGeno.txt
    animal  A
!CONSTRUCT SSMAT !APY             # lambda*APYinverse(alpha*G + beta*A22)
!SINGLESTEP                       # add A-inverse during solving
!APYPCA 98.0                      # use PCA to determine number of animals in core
# !METHOD VanRaden2                 # optional; default VanRaden2
# !ALFREQ ExampleAlFreq.txt         # optional; default calculated from data
# !CROSSBRED 3 ExampleBreeds.txt    # optional; default single breed
# !BREEDS_UNRELATED   # optional; default all genomic relationships are considered
# !ALLELES 1                        # optional; default genotypes
# !DENSE                            # optional; default markers in free format
# !NMARK 1000                       # optional; default is all markers
# !MAF 0.01                         # optional; default is 0.005
!NUMPROC 12                       # optional; default is 1
!LAMBDA 0.85                      # optional; default 1.0
!ALPHA 0.95                       # optional; default is 1.0
!BETA 0.05                        # optional; default is 1.0 - ALPHA
!OMEGA 0.90                       # optional; default is LAMBDA

PEDFILE ExamplePed.txt !CALCINBR
    animal A
    sire   A
    dam    A
    blkped I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

## Example 5.9.6 Existing APY inverse genomic relationship matrix

```
TITLE New APY inverse genomic relationship matrix using a random core set by PCA
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
#  APY inverse of G and avoiding explicit inverse of A22 matrix
ERMFILE ExtRelMatOrig.txt        # calculated using !CONSTRUCT SSMAT !APY !SINGLESTEP
   animal  A
!APY                             # lambda*APYinverse(alpha*G + beta*A22)
!SINGLESTEP                      # add A-inverse during solving

PEDFILE ExamplePed.txt !CALCINBR
   animal A
   sire   A
   dam    A
   blkped I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```





## Example 5.9.7 New APY inverse genomic relationship matrix using a random core and an explicit inverse of $A_{22}$

```
TITLE New APY inverse genomic relationship matrix using a random core and inverse A22
#     ssGBLUP: only part of animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
#  APY inverse of G and avoiding explicit inverse of A22 matrix
ERMFILE ExampleGeno.txt
   animal  A
!CONSTRUCT SSMAT !APY_A22        # lambda*APYinverse(alpha*G + beta*A22) minus
                                 #  omega*inverseA22
!SINGLESTEP                      # add A-inverse during solving
!APYCORERAN 2435                 # choose 2435 animals for core at random
# !METHOD VanRaden2                # optional; default VanRaden2
# !ALFREQ ExampleAlFreq.txt        # optional; default calculated from data
# !CROSSBRED 3 ExampleBreeds.txt   # optional; default single breed
# !BREEDS_UNRELATED   # optional; default all genomic relationships are considered
# !ALLELES 1                       # optional; default genotypes
# !DENSE                           # optional; default markers in free format
# !NMARK 1000                      # optional; default is all markers
# !MAF 0.01                        # optional; default is 0.005
!NUMPROC 12                      # optional; default is 1
!LAMBDA 0.85                     # optional; default 1.0
!ALPHA 0.95                      # optional; default is 1.0
!BETA 0.05                       # optional; default is 1.0 - ALPHA
!OMEGA 0.90                      # optional; default is LAMBDA

PEDFILE ExamplePed.txt !CALCINBR
   animal A
   sire   A
   dam    A
   blkped I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```

## Example 5.10 Regression on SNP covariates (RR-BLUP)

```
TITLE Regression on SNP covariates
#     RR-BLUP: all animals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
#  SNP covariates
SNPFILE !CALCVARSNP !PREDICT
    animal  A
SNP01 ExampleGeno.txt !REGTYPE R

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran SNP(1)
    phen2   ~ fix2 cov !RANDOM ran SNP(1)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





## Example 5.11 Regression on SNP covariates with non-genotyped individuals (ssRR-BLUP)

```
TITLE Regression on SNP covariates
#     ssRR-BLUP: only part of the individuals with phenotypes have been genotyped

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99      # created with PrepFern program
    animal  A
    NonG    R
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
#  SNP covariates
SNPFILE !CALCVARSNP !PREDICT !NOPRUNE !NOCHECK !INCLNONGENO
    animal  A
SNP01 ExampleGenoImp.txt !REGTYPE R       # created with PrepFern program

ERMFILE NonG_RelMat.txt                   # created with FDGRelMat program
    animal A

IMPFILE ExamplePedImp.txt                 # created with PrepFern program
!IMPIDCOL 1  # optional; default is 1
!IMPCOL 4    # optional; default is 4

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran SNP(1) G(animal*NonG)
    phen2   ~ fix2 cov !RANDOM ran SNP(1) G(animal*NonG)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

## Example 7.1 Statistical model with single direct genetic effect

```
TITLE breeding value estimation with single direct genetic effect

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





## Example 7.2 Statistical model with multiple records per individual

```
TITLE breeding value estimation with multiple records per individual
#     repeatability model

# Observations & systematic effects
DATAFILE ExampleDatPE.txt !MISSING -99
    animal  A
    perm    A               # content is identical to field animal
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM perm G(animal)
    phen2   ~ fix2 cov !RANDOM perm G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

## Example 7.3 Statistical model with direct and maternal genetic effect

```
TITLE breeding value estimation with direct and maternal genetic effect
#     maternal genetic model

# Observations & systematic effects
DATAFILE ExampleDatMatGen.txt !MISSING -99
    animal  A
    foster A              # foster dam; has to be present in pedigree
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal, foster)
    phen2   ~ fix2 cov !RANDOM ran G(animal, foster)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





## Example 7.4.2 Statistical model with direct and social genetic effects and equal group size

```
TITLE evaluation with direct and social genetic effects; equal groups of five
#     social genetic model

# Observations & systematic effects
DATAFILE ExampleDatSocGen.txt !MISSING -99
    animal A
    pmate1 A              # pen mate 1; has to be present in pedigree
    pmate2 A              # pen mate 2; has to be present in pedigree
    pmate3 A              # pen mate 3; has to be present in pedigree
    pmate4 A              # pen mate 4; has to be present in pedigree
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExampleParSocGen.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal, pmate1 AND pmate2 AND pmate3 AND pmate4)
    phen2   ~ fix2 cov !RANDOM ran G(animal, pmate1 AND pmate2 AND pmate3 AND pmate4)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

ExampleParSocGen.txt:

```
ran
phen1 0.1
phen2 0.0 0.1

G
phen1(animal) 0.3
phen2(animal) 0.0 0.3
phen1(pmate1) 0.0 0.0 0.1
phen2(pmate1) 0.0 0.0 0.0 0.1

Res
phen1 0.5
phen2 0.0 0.5
```

### Example 7.4.3 Statistical model with direct and social genetic effects and slightly varying group size

```
TITLE evaluation with direct and social genetic effects; group size may differ from 3
#     social genetic model

# Observations & systematic effects
DATAFILE ExampleDatSocGen.txt !MISSING -99
   animal  A
   pmate1 A               # pen mate 1; has to be present in pedigree
   pmate2 A               # pen mate 2; has to be present in pedigree
   pres1  R               # pen mate 1 is present (1) or not (0)
   pres2  R               # pen mate 2 is present (1) or not (0)
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
   animal  A
   sire    A
   dam     A
   blkped  I

# Components of variance and covariance among traits
PARFILE ExampleParSocGen.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal, pmate1*pres1 AND pmate2*pres2)
   phen2   ~ fix2 cov !RANDOM ran G(animal, pmate1*pres1 AND pmate2*pres2)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```

ExampleParSocGen.txt:

```
ran
phen1 0.1
phen2 0.0 0.1

G
phen1(animal) 0.3
phen2(animal) 0.0 0.3
phen1(pmate1*pres1) 0.0 0.0 0.1
phen2(pmate1*pres1) 0.0 0.0 0.0 0.1

Res
phen1 0.5
phen2 0.0 0.5
```





### Example 7.5.2 Statistical model with non-genetic random regression

```
TITLE evaluation with non-genetic random regression
#     random regression model; must be fitted as regression nested in class variable

# Observations & systematic effects
DATAFILE ExampleDatRanReg.txt !MISSING -99
   animal  A
   mean    I                  # class variable; may be a single class for all records
   fix1     A
   fix2     I
   cov      R
   ran      A
   phen1    T
   phen2    T
   blk      I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
   animal  A
   sire    A
   dam     A
   blkped  I

# Components of variance and covariance among traits
PARFILE ExampleParRanReg.dat

# Statistical models
MODEL
   phen1   ~ fix1 !RANDOM ran mean*cov G(animal)
   phen2   ~ fix2 !RANDOM ran mean*cov G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000
```

ExampleParRanReg.txt:

```
ran
phen1 0.1
phen2 0.0 0.1

mean
phen1(mean*cov) 0.2
phen2(mean*cov) 0.0 0.2

G
phen1(animal) 0.3
phen2(animal) 0.0 0.3

Res
phen1 0.5
phen2 0.0 0.5
```

## Example 7.5.3 Statistical model with genetic random regression

```
TITLE evaluation with genetic random regression
#     random regression model; must be fitted as regression nested in class variable

# Observations & systematic effects
DATAFILE ExampleDatRanReg.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExampleParRanReg.dat

# Statistical models
MODEL
    phen1   ~ fix1 !RANDOM ran G(animal*cov)
    phen2   ~ fix2 !RANDOM ran G(animal*cov)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

ExampleParRanReg.txt:

```
ran
phen1 0.1
phen2 0.0 0.1

G
phen1(animal*cov) 0.3
phen2(animal*cov) 0.0 0.3

Res
phen1 0.5
phen2 0.0 0.5
```





## Example 7.5.4 Statistical model with polynomial random regression

```
TITLE evaluation with polynomial random regression
#     random regression model; must be fitted as regression nested in class variable

# Observations & systematic effects
DATAFILE ExampleDatCVR.txt !MISSING -99
    animal A
    perm    A
    fix1    A
    fix2    I
    cov     R
    ran     A
    day     I !CVRIND
    phen1   T
    phen2   T
    blk     I

CVRTABLE !CVRMAKE LEG !CVRNUM 8      # creates a Legendre polynomial of the 8th order

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExampleParCVR.dat

# Statistical models
MODEL
    phen1   ~ fix1 CVR(8) !RANDOM perm*CVR(3) G(animal*CVR(2))
    phen2   ~ fix2 CVR(8) !RANDOM perm*CVR(3) G(animal*CVR(2))

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

ExampleParCVR.txt (please note that cvr00 to cvr03 has to be lowercase):

```
perm
phen1(perm*cvr00) 0.1
phen2(perm*cvr00) 0.0 0.1
phen1(perm*cvr01) 0.0 0.0 0.05
phen2(perm*cvr01) 0.0 0.0 0.0 0.05
phen1(perm*cvr02) 0.0 0.0 0.0 0.0 0.01
phen2(perm*cvr02) 0.0 0.0 0.0 0.0 0.0 0.01
phen1(perm*cvr03) 0.0 0.0 0.0 0.0 0.0 0.0 0.005
phen2(perm*cvr03) 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.005

G
phen1(animal*cvr00) 0.3
phen2(animal*cvr00) 0.0 0.3
phen1(animal*cvr01) 0.0 0.0 0.08
phen2(animal*cvr01) 0.0 0.0 0.0 0.08
phen1(animal*cvr02) 0.0 0.0 0.0 0.0 0.02
phen2(animal*cvr02) 0.0 0.0 0.0 0.0 0.0 0.02

Res
phen1 0.5
phen2 0.0 0.5
```

## Example 7.6 Statistical model with weighted residual effects

```
TITLE breeding value estimation with weighted residuals

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    wtphen1 R
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   !WEIGHT wtphen1 ~ fix1 !RANDOM G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !MAXIT 1000
```





## Example 7.7 Statistical model with fixed effects combined across traits

```
TITLE breeding value estimation with fixed effects combined across traits

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1_1  T # phen1 for parity 1
    phen1_2  T # phen1 for parity 2
    phen1_3  T # phen1 for parity 3
    phen2    T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1_1 ~ fix1 cov !RANDOM G(animal)
    phen1_2 ~ fix1 cov !RANDOM G(animal)
    phen1_3 ~ fix1 cov !RANDOM G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

COMBINE
    fix1 ~ phen1_1 phen1_2 phen1_3 # effect is estimated across these traits
    cov  ~ phen1_1 phen1_2 phen1_3 #

# Control of analysis and output
SOLVING
    !MAXIT 1000
```

## Example 7.8 Statistical model with correction of heterogeneous residual variances

```
TITLE breeding value estimation with correction of heterogeneous residual variances

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   fix3    A
   cov     R
   ran     A
   phen1    T
   phen2      T
   blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
   animal  A
   sire    A
   dam     A
   blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL # model for unweighted and weighted analysis of traits
   phen1    ~ fix1 cov !RANDOM G(animal)
   phen2    ~ fix2 cov !RANDOM ran G(animal)

VARMODEL # model of the analysis of linearized squared residuals of traits
   LSRphen1 ~ fix3 !RANDOM G(animal)     !VARTRAIT phen1  # may be different from MODEL
   LSRphen2 ~ fix2 !RANDOM ran G(animal) !VARTRAIT phen2

# Control of analysis and output
SOLVING
   !MAXIT 1000
   !DHGLM       # prepare for multiple calls of the kernel
   !HETCOR      # create data files and instructions for heterogeneity correction
```

ExamplePar.dat:

```
ran
phen2    0.1
LSRphen2 0   0.007

G
phen1(animal) 0.3
phen2(animal) 0.02  0.3
LSRphen1(animal) 0     0    0.06
LSRphen2(animal) 0     0    0     0.06

Res
phen1    0.5
phen2    0.0 0.5
LSRphen1 0   0   0.05
LSRphen2 0   0   0.0 0.05
```





## Example 8.1 Control of the analysis

```
TITLE breeding value estimation using pedigree with single code for unknown parents

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
   animal  A
   sire    A
   dam     A
   blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
   phen1   ~ fix1 cov !RANDOM ran G(animal)
   phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !MAXIT 1000        # default is 5000
   !STOPCRIT 1.0E-05 # default is 1.0E-04
   !NOPEEK            # default is to produce PEEK files
   !PEEKFIRST 100     # default is 500
   !PEEKEVERY 50      # default is 500
   !PEEKKEEP          # keep all intermediate solutions; default is keep only last set
   !RESTART           # use existing set of (intermediate) solutions as starting values
```

## Example 8.2 Control of output

```
TITLE breeding value estimation using pedigree with single code for unknown parents

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I

# Genetic similarity among individuals
PEDFILE ExamplePed.txt
    animal  A
    sire    A
    dam     A
    blkped  I

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL
    phen1   ~ fix1 cov !RANDOM ran G(animal)
    phen2   ~ fix2 cov !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !BASEANIMALZERO # deduct the average solutions by trait for animals in
#                   BaseAnimals.dat from each solution
    !YHAT           # write out predicted phenotypes: Xb+Wp+Zu
    !EHAT           # write out residuals: Y-Xb-Wp-Zu
    !YIELDDEV       # write out adjusted phenotypes: Y-Xb-Wp
    !IDD            # write out individual daughter yield deviations
    !DYD            # write out daughter yield deviations by sire
    !KEEPTMP        # keep internal temporary files; default is to delete these files
```





## Example 9.4.1 Reliabilities for an analysis using pedigree only

```
TITLE reliabilities using pedigree only

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
    animal  A
    fix1    A
    fix2    I
    cov     R
    ran     A
    phen1   T
    phen2   T
    blk     I !BLOCK

# Genetic similarity among individuals
PEDFILE ExamplePed.txt  # !CALCINBR must not be used
    animal  A
    sire    A
    dam     A
    blkped I !BLOCK

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL  # for each trait just a single fixed effect taken within blocks: BL(...)
    phen1   ~ BL(fix1) !RANDOM ran G(animal)
    phen2   ~ BL(fix2) !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
    !RELIABILITY
    !MAXNONZERO 10000   # default is 90,000,000
```

## Example 9.4.2 Reliabilities for an analysis using pedigree and genomic information

```
TITLE reliabilities using pedigree and genomic information

# Observations & systematic effects
DATAFILE ExampleDat.txt !MISSING -99
   animal  A
   fix1    A
   fix2    I
   cov     R
   ran     A
   phen1   T
   phen2   T
   blk     I !BLOCK

# Genetic similarity among individuals
#   use ExtRelMatOrig.txt created with !CONSTRUCT SSMAT !SINGLESTEP for breeding
#   value estimation or re-calculate from genotypes
ERMFILE ExtRelMatOrig.txt
   animal  A
!SINGLESTEP                    # add A-inverse during solving
!LAMBDA 0.85                   # optional; default 1.0
!OMEGA 0.90                    # optional; default is LAMBDA

PEDFILE ExamplePed.txt  # !CALCINBR must not be used
   animal  A
   sire    A
   dam     A
   blkped I !BLOCK

# Components of variance and covariance among traits
PARFILE ExamplePar.dat

# Statistical models
MODEL  # for each trait just a single fixed effect taken within blocks: BL(...)
   phen1   ~ BL(fix1) !RANDOM ran G(animal)
   phen2   ~ BL(fix2) !RANDOM ran G(animal)

# Control of analysis and output
SOLVING
   !RELIABILITY
   !MAXNONZERO 10000   # default is 90,000,000
```

MiXBLUP