VCE
User's Guide
and
Reference Manual
Version 6.0

Eildert Groeneveld, Milena Kovač and Norbert Mielenz

November 2008

Eildert Groeneveld
Institute of Farm Animal Genetics
Friedrich Loeffler Institute (FLI)
Mariensee
Höltystraße 10
D-31535 Neustadt, Germany
eildert.groeneveld@fli.bund.de

Milena Kovač
University of Ljubljana
Biotechnical Faculty
Department of Animal Science
Groblje 3, 1230 DOMŽALE, Slovenia
milena@mrcina.bfro.uni-lj.si

Norbert Mielenz
Institute of Agricultural and Nutritional Sciences
Martin Luther University
Halle-Wittenberg
D-06099 Halle, Germany
mielenz@landw.uni-halle.de

Contents

1 Introduction **13**
- 1.1 Availability and installation . . . 14
- 1.2 Test Data . . . 15
- 1.3 Documentation . . . 16
- 1.4 User Interface . . . 16
  - 1.4.1 Starting VCE . . . 16
  - 1.4.2 Interactive help. . . . 17
- 1.5 Manual . . . 21

2 Models and Methods **23**
- 2.1 General form of the models . . . 23
- 2.2 Examples . . . 24
  - 2.2.1 Heterogeneous covariances . . . 25
  - 2.2.2 Random regression models . . . 25
  - 2.2.3 Non-additive genetic effects . . . 25
- 2.3 Gibbs Sampling . . . 25
  - 2.3.1 Burn-in period . . . 26
  - 2.3.2 Plotting . . . 27
  - 2.3.3 In Gibbs: restricted choice of models . . . 27

3 Parameter file **29**
- 3.1 Definitions . . . 29
  - 3.1.1 Section . . . 29
  - 3.1.2 Statement . . . 30
  - 3.1.3 Keyword . . . 30
  - 3.1.4 Option . . . 30
  - 3.1.5 Variable . . . 30
  - 3.1.6 Delimiter . . . 30
  - 3.1.7 Expression . . . 31
  - 3.1.8 Operators . . . 31
  - 3.1.9 Functions . . . 31
  - 3.1.10 Constants . . . 31
- 3.2 An example parameter file . . . 31
- 3.3 COMMENT section . . . 31
- 3.4 DATA section . . . 33
  - 3.4.1 Data sets . . . 33

3.4.2 Keywords in DATA section . . . 35
3.5 MODEL section . . . 39
3.5.1 Model statements . . . 39
3.5.2 Transformation statements . . . 44
3.5.3 Restrictions . . . 47
3.6 COVARIANCE section . . . 47
3.6.1 Description of random effects . . . 48
3.6.2 Starting values for covariance components . . . 49
3.6.3 Some examples of COVARIANCE section . . . 51
3.6.4 Residual covariance matrices . . . 53
3.6.5 Covariance matrix for "trivial" random effects . . . 55
3.6.6 Additive genetic covariance matrix . . . 56
3.6.7 Dominance covariance matrix . . . 57
3.7 SYSTEM section . . . 57
3.7.1 Keywords driving iteration procedures . . . 58
3.7.2 Keywords concerning additive genetic relationship . . . 59
3.7.3 Keywords connected with data manipulation . . . 60
3.7.4 Other keywords in SYSTEM section . . . 60
3.8 File naming conventions . . . 61
3.9 OUTPUT section . . . 61
3.9.1 Keywords in OUTPUT section . . . 61
3.9.2 Options with keywords in OUTPUT section . . . 65
3.10 END section . . . 65

4 Examples and Use Cases **67**
4.1 Coding the input data . . . 67
4.2 One Random Effect . . . 70
4.2.1 One random effect, single trait . . . 70
4.2.2 One random effect, multiple traits . . . 72
4.3 More than one random effect and adding a fixed. . . . 75
4.4 The Litter Effect . . . 76
4.5 The Sire Model . . . 76
4.6 Univariate Animal Model . . . 76
4.6.1 Data preparation . . . 77
4.7 The Sire Model with Relationship . . . 78
4.8 Multivariate Animal Model . . . 79
4.9 Models with maternal additive genetic effect . . . 82
4.10 Example with fixed regression nested . . . 84
4.11 Setting up Random Regression Models . . . 84
4.11.1 Body mass in Golden Hamsters . . . 84
4.11.2 Daily Gain in Bulls . . . 94
4.11.3 Feed Intake in Beef . . . 97
4.11.4 Coding requirements . . . 98
4.11.5 Running the job . . . 99

4.12 Model with dominance genetic effect . . . 100
4.12.1 Dominance effect in pure breed populations . . . 100
4.12.2 An example: Egg production in laying hens . . . 102
4.12.3 Computational aspects in VCE . . . 105
4.13 Models with disconnected residual covariance structure . . . 106
4.13.1 Data Preparation . . . 106
4.13.2 One data file . . . 106
4.13.3 Two files . . . 107

5 FAQ **111**
5.1 I am getting status 3, what now? . . . 111
5.2 The degree of fill is above 80% and things are getting slow . . . 111
5.3 VCE says the computer does not have enough memory or SEGMENTATION violation . . . 112
5.4 Can I do a 20 trait model? . . . 112
5.5 Is there a 64bit version of VCE? . . . 113
5.6 Can I run a 32 bit version on a 64 bit computer? . . . 113
5.7 What is estimated: covariance components or ratios? . . . 113
5.8 Can I do a likelihood ratio test? . . . 114
5.9 BLUP vs VCE: different models? . . . 114
5.10 For which platforms is VCE available? . . . 114
5.11 Where do I click to start VCE? . . . 114
5.12 Does VCE produces standard errors? . . . 114
5.13 My VCE job has been running for a week, can I see the current estimates? . . . 115
5.14 Can I get the covariance matrix of the estimates? . . . 116
5.15 Can VCE assist me in passing the English test? . . . 116
5.16 Constraints and Restrictions . . . 116

6 Changes for Version 6.0 **119**

List of Tables

3.1 Keywords in DATA section 35
3.2 Description of columns in pedigree file 37
3.3 List of functions in **VCE** 43
3.4 Scaling options 45
3.5 Keywords in **MODEL** section 47
3.6 Statements in **COVARIANCE** section 48
3.7 Keywords in **COVARIANCE** section 52
3.8 Keywords driving iteration procedures 59
3.9 Keywords connected with data manipulation 60
3.10 Other keywords in SYSTEM section 60
3.11 Methods and solver 61
3.12 default file name for parameter file np01 62
3.13 Keywords in **OUTPUT** section 63
3.14 file related defaults in **OUTPUT** section 65

4.1 Sheep data 'mrode.data' 100

List of Sections and Parameter Files

3.1 General structure of a section 29
3.2 Example parameter file 32
3.3 COMMENT section 33
3.4 DATA section 34
3.5 A generic **MODEL** section 40
3.6 Some examples of model statements 41
3.7 Reduced Animal Model 42
3.8 A generic **COVARIANCE** section 48
3.9 Description of covariance matrices in ***'start_asc'*** 51
3.10 Only additive genetic effect 51
3.11 Nested covariance matrices for additive genetic effect and residual 53
3.12 Starting values for example 2 53
3.13 **System** section 57
3.14 **OUTPUT** section 62

4.1 Effect breed as main effect and nesting effect for linear regression 85
4.2 Single trait analysis with family subclass effect 101

1 Introduction

VCE is a program package to estimate dispersion parameters under a general linear model. The statistical models cover a variety of possibilities like heterogeneous covariance components for residuals as well as other random effects, models with longitudinal data, random regression, multi-environment analyzes. Additive and dominance relationships are implemented. Dispersion parameters are obtained by restricted maximum likelihood (REML) using analytical gradients, as well as Gibbs sampling.

As regards the methods: analytical gradients is the workhorse and should be used wherever possible. Gibbs sampling will also work for standard animal models but probably not for all model possible with analytical gradients; but it is really slow. So you would probably not want to use it, unless the memory constraints are much more severe on your machine than the CPU speed, or if you are interested in the posterior distributions.

Acknowledgments Many people have contributed to the development of **VCE**. Some of them have been involved personally while others have contributed via their code which they made publicly available. The following is a (probably not complete) list of persons and their engagement:

Arnold Neumaier[7]	did the math for the analytical gradients and standard errors
Didier Boichard	joint implementation of inbreeding; did most of the stuff that gives us approximations of standard errors
S. Gay & N. Nash	UNCMIN - unconstrained BFGS quasi-Newton optimizer
Ignacij Misztal & Miguel Enciso-Perez	Sparse inverse and factorization
Alberto Garcia-Cortez	joint implementation of Monte-Carlo EM and Gibbs sampling
Marcos Rico	involved on many fronts: coupling method, testing
Luis Varona	initial version of Gibbs sampling
Helmut Lichtenberg	did the Makefiles that help create the binaries
Spela Malovrh	prepared test data sets and run tests with VCE
Norbert Mielenz	finally fixed the standard errors
all those nice people	who allowed us to use their machines for testing and creating binaries and share data sets to demonstrate properties of VCE

We acknowledge financial support from DFG, the German Research Foundation, which facilitated the initial version 5 of VCE to a large degree.

Referencing VCE Publications presenting results that were generated through VCE should reference this Manual and the initial Neumaier&Groeneveld publication[7].

1.1 Availability and installation

VCE is free of charge for non commercial use but please acknowledge its use.

VCE is available from our anonymous ftp server: ftp.zgr.fal.de There are three types of files available: binaries, test data, and documentation .

VCE comes as a binary i.e. an executable program. This reduces the installation procedure to copying the binary to a directory that is in the search path of the prospective users. Binaries are available in the bin directory which currently contains the following entries (at the same time you can improve your German language skills):

If you want to install **VCE** on your machine, you simply need to select the corresponding binary that runs on your platform. Files ready for transfer are in compressed form. Be sure, you always pick the latest version, i.e. the one with the highest release number! Under UNIX for example, you need first to uncompressed and then to untar the files. You should move the file to a position in the file system where every user has read access and that is also in the search path. If you do not have root access, ask your system administrator.

You can test immediately if the binaries run on your machine by starting them:

```
eg@eno:~/newvce/release/6.0.2$ ./vce-Linux-x86_64-gfortran-6.0.2
  ************************************************
  *                      VCE-6                   *
  *                  version 6.0.2               *
  *             05-Nov-2008 @ 09:16:31           *
  *              Linux-x86_64-gfortran           *
  *                    written by                *
  *       Milena Kovac, Eildert Groeneveld       *
  *            and Alberto Garcia-Cortez         *
  ************************************************
VCE [help] pfile funk
eg>
```

Then you should be in business.

1.2 Test Data

Next, you may want to obtain data and parameter files for testing and demonstration purposes. On our ftp server, you need to go the directory *examples*. There you should see something like:

You should create a directory in your own space say vce6 and then unpack the test-data there:

```
mkdir vce6
cd vce6
gzip -d vce-examples.tgz
tar -xf vce-examples.tar
226 Transfer complete.
```

This will result in a directory test that has the following sub-directories:

```
`-----test
    | `-----data
    | `-----master_pfile
    | `-----verified
    | `-----verified/long
    | `-----temp
```

Have a look at them. The parameter files are stored in pfile. If you want to run a job, go to test/temp and run it there:

```
eg(forssa,~/newvce): cd test/temp
eg(forssa,~/newvce/test/temp): vce ../master_pfile/np01
Record : 1 from file :../data/diet2.d
   1 rasse     1
   1 sex       2
   1 betr     11
   1 tier      1
   1 rsp       3.20
   2 drip      8.10
Record : 2 from file :../data/diet2.d
   1 rasse     8
   1 sex       1
   1 betr      2
..
..
```

The verified results are stored in directory 'verified'.

1.3 Documentation

This reference manual documents the features of **VCE**. Statistical issues of the models implemented are more or less avoided, some are illustrated only to explain the examples given. Examples are not always meaningful from an animal breeder's point of view, as they are primarily used to demonstrate features of **VCE** rather than useful statistical models.

Documentations tends to be the less developed part of a package. Therefore, your comments are very much appreciated and can help us to make these pages useful.

You find documentation on our ftp server in the directory *doc* available as PDF, HTML, and plain ASCII text versions. Have a look every now and then to catch the updates.

1.4 User Interface

1.4.1 Starting VCE

VCE is controlled entirely via a parameter file, input data have to be available in one or more files for measurements, one file for the common pedigree, and one or more files describing

heterogeneity of trivial random effects. The process of estimating covariance matrices is then started by typing the program name (*vce*) at the system prompt:

```
>vce
```

If *VCE* is started without an argument, you will see the following:

```
  ********************************************
  *                    VCE                   *
  *                version 5.1.2             *
  *           04-Dez-2003 @ 16:32:46         *
  *           Linux 2.6.0-test11 i686        *
  *                written by                *
  *       Milena Kovac, Eildert Groeneveld   *
  *           and Alberto Garcia-Cortez      *
  ********************************************
VCE [help] pfile
```

From this message, you can determine date of compilation, platform and the version of **VCE** that you are using. This is important if you have bugs to report. If you want to start evaluation, you should type the name of parameter file. At this stage you can enter the name of the parameter file with a path that is appropriate to the current position in the file system.

1.4.2 Interactive help.

The program has a simple interactive help. If instead of the name of parameter file you type the reserved word 'help', you will enter the help mode. A help session is shown in listing 1.1. After choosing the section name (or "all") the user is prompted for the parameter file. If given, the content of the parameter file is listed together with all options for the chosen section. If no parameter file is typed but instead <return> is hit right away, we get the output given in Listing 1.1. As can be seen in lines 17–28 from Listing 1.1 all options for the chosen section name (model) are given.

Listing 1.1: VCE session for help

```
eg@eno:~/newvce/test/temp$ vce6
  ********************************************
  *                    VCE                   *
  *                 version 5.2              *
  *           05-Dez-2007 @ 08:45:27         *
  *         Linux 2.6.22-14-generic i686     *
  *                 written by               *
  *       Milena Kovac, Eildert Groeneveld   *
  *           and Alberto Garcia-Cortez      *
  ********************************************
VCE [help] pfile funk
eg> help
Section (comment, data, model, covariance, system, output, all) :
eg> model
Parameter file:
eg>
 ================================================================================
  Section        Keyword - full    short    Typ  Default values              Format
  -----------    ----------------  ------   ---  -------------------------- ------
 55 model         set               set          C
 56 model         scale             scale        C   non
 57 model         scaley            scaley       C   non
 58 model         scalex            scalex       C   non
 59 model         equate            equate       C
 60 model         multi             multi        C
 61 model         by                by           C
 62 model         /cf               /cf          L    F
 ================================================================================
```

The program asks you for a section name you would like to get help on. Answer with one of section names or type 'all'. Furthermore, you will be asked to type a name of parameter file. If you just want to get the standard output, press return. You will get a list of legal keywords within each section with complete and short name, type of variable and default values. The list of sections and keywords is given in Listing 1.2 and 1.3.

Listing 1.2: Keywords in VCE in sections COMMENT and SYSTEM

	Section	Keyword – full	short	Typ	Default values	Format
1	commen	job	job	C	np01	
2	system	burn_first	burn_first	I	1	
3	system	burn_max	burn_max	I	10000	
4	system	burn_next	burn_next	I	10	
5	system	burn_stop	burn_stop	D	0.100000000E-02	
6	system	iod_first	iod_first	I	1	
7	system	iod_max	iod_max	I	1000	
8	system	iod_next	iod_next	I	1	
9	system	iod_stop	iod_stop	D	0.100000000E-03	
10	system	mark_first	mark_first	I	500	
11	system	mark_max	mark_max	I	10000	
12	system	mark_next	mark_next	I	10	
13	system	mark_stop	mark_stop	D	0.100000000E-02	
14	system	pev	pev	L	F	
15	system	rao_black	rao_black	I	0	
16	system	post_detail	post_detai	I	1000	
17	system	next_post	next_post	I	1	
18	system	inbreeding	inbreeding	L	T	
19	system	mc_seed	mc_seed	D	0.551121231E+10	
20	system	method	method	C	AG	
21	system	missing_value	missing_va	R	0.111110000E+08	
22	system	non_zero	non_zero	I	999999999	
23	system	proportion	proportion	R	0.000000000E+00	
24	system	reparameterize	reparamete	L	T	
25	system	skip_value	skip_value	R	0.111120000E+08	
26	system	solve	solve	C	ioc	
27	system	total	total	I	4000000	
28	system	tolerance	tolerance	D	0.100000000E-06	
29	system	debug_stop	debug_stop	L	F	

Listing 1.3: Keywords in VCE in sections COVARIANCES, DATA, MODEL and OUTPUT

	Section	Keyword - full	short	Typ	Default values	Format
30	covari	start_bin	start_bin	Z		
31	covari	dump_bin	dump_bin	Z		
32	covari	start_asc	start_asc	Z		
33	covari	dump_bin	dump_bin	Z	np01.cov-bin	
34	covari	dump_asc	dump_asc	Z		
35	covari	cov_zero	cov_zero	I	0	
36	covari	datfile	datfile	L	F	
37	covari	form	form	C		
38	covari	format	format	C		
39	covari	level	level	I	0	
40	covari	link	link	C		
41	covari	values	values	D	0.000000000E+00	
42	data	crossbreeding	crossbreed	L	F	
43	data	datfile	datfile	C	../data/meat2.dat	
44	data	dep	dep	C	rsp	
45	data	dependent	dependent	C		
46	data	dominance	dominance	C		
47	data	format	format	C	(2f12.0,10f8.0)	
48	data	group_by	group_by	C		
49	data	header	header	I	1	
50	data	indep	indep	C	rasse	
51	data	independent	independen	C		
52	data	link	link	C	tier	
53	data	pedfile	pedfile	C	../data/meat2.ped	
54	data	ranfile	ranfile	C		
55	model	set	set	C		
56	model	scale	scale	C	non	
57	model	scaley	scaley	C	non	
58	model	scalex	scalex	C	non	
59	model	equate	equate	C		
60	model	multi	multi	C		
61	model	by	by	C		
62	model	/cf	/cf	L	F	
63	output	covfile	covfile	Z		
64	output	debug	debug	L	F	
65	output	inbreeding	inbreeding	Z		
66	output	lhs	lhs	Z		
67	output	log_gibbs	log_gibbs	Z		
68	output	mem_map	mem_map	L	F	
69	output	seldif_file	seldif_fil	Z		
70	output	solutions	solutions	Z		
71	output	dominance	dominance	Z		
72	output	family	family	Z		
73	output	vcm	vcm	Z		
74	output	reprint	reprint	L	F	

Whenever you give the name of your parameter files, default values of keywords which are always described by only one value will be updated from it. This way you can see, if **VCE** understands your parameter file. As an example, keyword 53 of the Listing 1.3 deals with the pedigree file. Apparently, VCE has picked up the file name '../data/meat2.ped' from the

parameter file.

So if you have the problem of VCE not doing what you thing it should be doing, try this and see, if VCE reads your parameter file the way you intend it. Possibly, you have used a wrong keyword, then VCE will not detect it and consequently not show the intended keyword value in this list output.

1.5 Manual

This manual is split into one part that serves as a reference manual with an exhaustive description of the parameter file syntax. The second part deals with examples and use cases. The prospective user may want to have a look at this and locate a problem, that is close to her own. This would give a good start for one's own parameter file.

Finally, we have added the ever popular FAQ section with serious and not so serious questions.

If you have any better, more reliable examples and you are ready to share them with others, we would be glad to incorporate them.

For the display of a number of sections from the parameter file the LaTeX class "Algorithm" is used as it is well suited for the required formatting; we know that the naming is not really appropriate, but bear with us.

2 Models and Methods

In this chapter we describe what kind of models are handled by VCE. Furthermore, the solving strategies for obtaining covariance component estimates are outlined.

2.1 General form of the models

The statistical models used in VCE have the generalized form shown in equations 2.1 to 2.5. The observations y_{ijt} are explained by effects w_i and v_j and some interactions wv_{ij} among them. Vectors ω, λ, ψ, δ present expected values and matrices Ω, Λ, Ψ, Σ contain dispersion parameters for w_i, v_j, ψ, and residual e_{ijt}, respectively. All effects in the model are treated as random. Then fixed effects are just special cases which have zero rows (and columns) in the covariance matrices. Observations in a time/space sequence are assumed to have correlated residuals in vector e_{ij}, thus the matrix Σ is assumed to be full unless the user sets some correlations to zero in a set of starting values.

$$y_{ijt} = [1,\ x'_{it},\ u'_i]w_i + [1,\ s'_{it},\ q'_j]v_j + [1,\ z'_{it}]wv_{ij} + ... + e_{ijt} \tag{2.1}$$

$$w_i = rand(\omega,\ \Omega) \tag{2.2}$$

$$v_j = rand(\lambda,\ \Lambda) \tag{2.3}$$

$$wv_{ij} = \psi_{ij} = rand(0,\ \Psi) \tag{2.4}$$

$$e_{ij} \ := \ e^T_{i,1:T} \ = \ rand(\delta,\ \Sigma) \tag{2.5}$$

where means:

y_{ijt}	-	observations
w_i, v_j	-	effects
wv_{ij}	-	interaction between w_i and v_j
1	-	constant, common characteristics
u'_i, q'_j	-	vectors of time/space independent characteristics
x'_{it}, s'_{it}	-	vectors of time/space dependent characteristics
e_{ijt}	-	residual
ω, λ, ψ, δ	-	expected values for w_i, v_j, ψ_{ij}, e_{ij}, respectively
Ω, Λ, Ψ, Σ	-	covariance matrices for w_i, v_j, ψ_{ij}, e_{ij}, respectively

Each effect is expressed in a different form of regression equations with coefficients describing:

- ▷ **common characteristics** like constant 1 which are characteristics common to all units (individuals, animals) which belong to a certain level
- ▷ **time/space independent characteristics** in vectors u'_i, q'_j which are special individual characteristics existing over the whole lifetime or over the whole space.
- ▷ **time/space dependent characteristics** in vectors x'_{it}, s'_{jt}, z'_{it} which are special individual characteristics changing over time or space.

2.2 Examples

Let's look at breed effect (B_i) on trait LMP in growing animals. Assume that the adequate expression in the model looks like the equation 2.6. The coefficient 1 stands for the usual breed effect, the second coefficient w_{ijt} is used to explain changes of LMP over growth interval in form of linear regression, the third coefficient s_{ij} adjusts LMP differences caused by uneven starting weights also as linear regression. The expression 2.6 can be extended to the equivalent term 2.7. It is clear that we expect the effects of weight (w_{ijt}) and starting weight (s_{ij}) to be specific for each breed. In other words, regressions are nested within breeds.

$$... + [1,\ w_{ijt},\ s_{ij}]B_i + ... \tag{2.6}$$

$$... + B_i + b_{wj}w_{ijt} + b_{si}s_{ij} + ... \tag{2.7}$$

Let's assume, the growth on the interval observed is better described by third order polynomial. Thus, we need to add additional elements to the model as shown in equation 2.8. The expression can be written in shorter forms like in and 2.10 and2.9. The characteristics within square brackets turned into coefficients of incidence matrices, the symbol B_i is a vector all parameters describing breed effect. For example, there are three parameters for each breed in equation 2.6 while equations 2.9 or 2.10 expressed breed effect using five parameters.

$$... + B_i + b_{1wj}w_{ijt} + b_{2wj}w^2_{ijt} + b_{3wj}w^3_{ijt} + b_{si}s_{ij} + ... \tag{2.8}$$

$$... + [1, w_{ijt}, w_{ijt}^2, w_{ijt}^3, s_{ij}]B_i + ... \tag{2.9}$$

$$... + [1, p3(w_{ijt}), s_{ij}]B_i + ... \tag{2.10}$$

2.2.1 Heterogeneous covariances

Models can incorporate heterogeneous covariances for residuals as well as random effects. The effect causing heterogeneity must be associated with a random effect in a separate data file. For genetic effects, it can be added into pedigree file. It is assumed that subpopulations with different covariance matrices for genetic effects do not have genetic ties and have the same source of heterogeneity. A special case is heterogeneity in static two way crossbreeding schemes with the possibility of adding more complex crossbreeding schemes.

2.2.2 Random regression models

A trait may be described by a function or more combined functions. Functions create coefficients of incidence matrices, which we also call characteristics. Characteristics are of three types. Common characteristics are the same for all units (individuals, animals) which belong to a certain level. Time/space independent characteristics are special individual characteristics which holds the whole lifetime (age at first delivery, birth weight) or over the whole space. The third type are special individual characteristics which are changing over time or space (like test day). Covariance functions for residual term are not supported yet but prepared to be added.

2.2.3 Non-additive genetic effects

VCE can handle models with dominance effect for direct, maternal, and/or paternal effects. The dominance relationship matrix is created from the pedigree file. If desired, users may obtain solutions of mixed model equations. Combination with heterogeneous covariances and random regression models is possible.

2.3 Gibbs Sampling

The Gibbs method can be invoked in the system section by setting the '**method**' keyword as GI. This keyword will be in most cases enough to perform the Bayesian inference on the variance components, i.e., keeping the rest of the keywords as default values. Inferences are carried out by setting flat priors on fixed effects and variances, while priors on additive, random and maternal effects are assumed as multivariate normal distributions, as described in [8]. Although the use of flat priors may be not suitable for some user requirements, in some circumstances, prior distributions can be replaced by custom ones during the post-gibbs analysis from the VCE

Gibbs output. This feature is not actually implemented in VCE, but it can be done easily by advanced users.

The calculation of the burn-in period has been implemented as described in [1]. VCE includes determination of the burn-in period and discard these cycles from the calculation of the marginals. The procedure is as follows: VCE starts from two sets of initial variance matrices, and computes two chains with the same random deviates until they converge to the same values. When both chains produce exactly the same numbers (for a given tolerance), the burn-in period is assumed to be finished. The behavior of the convergence can be seen on the screen output and it is easy to follow the logic involved on it.

Defaults for these keywords should be right, but users can use them to customize the Gibbs output. For instance, to have just the chains and to do the post Gibbs analysis after VCE will be finished, just set *burn_max=0* to skip the coupled chains stuff and set gibbs_log='filename' in the output section. To set a fixed length for burn-in (10000 for instance), you can set *burn_max=10000* and *burn_stop=0*, etc.

Generally speaking, the choice of *burn_next* depends on the accuracy required in the determination of the burn-in period and the CPU time. For instance, if *burn_next=10* and the burn-in finish at iteration 2000, it means that the burn-in was reached between 1990 and 2000. *burn_next* should be set to one only when the calculation of the burn-in will be discussed as a part of the research.

After the burn-in period, only one of the chains is computed (really, there are no differences greater than *burn_stop* between them) and the Monte Carlo variance of the estimates is obtained from the single chain effective length size as described in [2].

2.3.1 Burn-in period

VCE computes the burn-in period on the fly by using the coupling method [4, 1], only two chains are computed. Starting covariances for both chains are hard coded and the default random number seed (0.551121231D+10) can be customized via the *mc_seed* keyword. After the burn-in period will be finished, only the first chain will continue running.

Keywords related with burn-in are: *burn_first*, the number of cycles until the first check for burn-in will be computed; *burn_next,* the number of cycles between burn-in checks; *burn_max*, the maximum number of cycles to reach the end of the burn-in period; and *burn_stop*, the stopping criteria for burn-in. The average of the differences in ratios will be compared against *burn_stop* every *burn_next-th* cycle. In general, default values should be right for most cases.

Advanced users will probably prefer a raw single-chain Gibbs sampler in order to do the post-analysis themselves. In this case, we recommend to set *burn_max=0* in the system section *and log_gibbs='your_filename'* in the output section. In this case, marginal posterior moments provided by VCE will be useless because of including the whole chain. Some comments about the chain length were included in the next section (*convergence*). Each cycle stores the covariances in *'your_filename'*, being instantaneously available.

2.3.2 Plotting

It is well-known that the analytical gradients estimator (AG) run several orders of magnitude faster than a Gibbs sampler. Nevertheless, there are some entertainments available during the the boring burn-in periods you will spend at the console. Setting the keyword *log_gibbs*, a small gnuplot[1] script called *'pfile.gnuplot'* is generated as shown in Listing 2.1. The script includes the name of your current *'log_file'* and the columns corresponding to the first component in both chains. An example of this feature is in algorithm 2.1, note that we set *log_gibbs='chain.np04'* in the output section of the parameter file. The np04 test example has 9 components, then the 3^{rd} column[2] and the 12^{th} column are displayed on a gnuplot window during the fly, other displays have to be set manually. The script also writes an exportable *burn_in.eps* file. Both the cycles to be written in *log_file* and the detail of the plots depend mainly on the *burn_next* keyword. Figure 2.1 shows a gnuplot-x11 output. Note that the second chain has a different color, allowing the determination of the burnin period around the cycle 950 (exact value will be in the output file). Both sides of the figure differ also in granularity, because only one every next-burn-th cycles are stored in the log during the burn-in period.

Listing 2.1: *Burn-in in VCE can be monitored on the fly by using gnuplot*

```
>$ cat vce_GI.plot
set data style lines
set nokey
set terminal x11
plot 'chain.np04' using 1:3
replot 'chain.np04' using 1:12
set terminal postscript eps
set output 'burn_in.eps'
plot 'chain.np04' using 1:3
replot 'chain.np04' using 1:12
>$gnuplot
(...)gnuplot welcome message
gnuplot> load 'vce_GI.plot'
gnuplot> load 'vce_GI.plot'
gnuplot> load 'vce_GI.plot'
gnuplot> (...)
```

2.3.3 In Gibbs: restricted choice of models

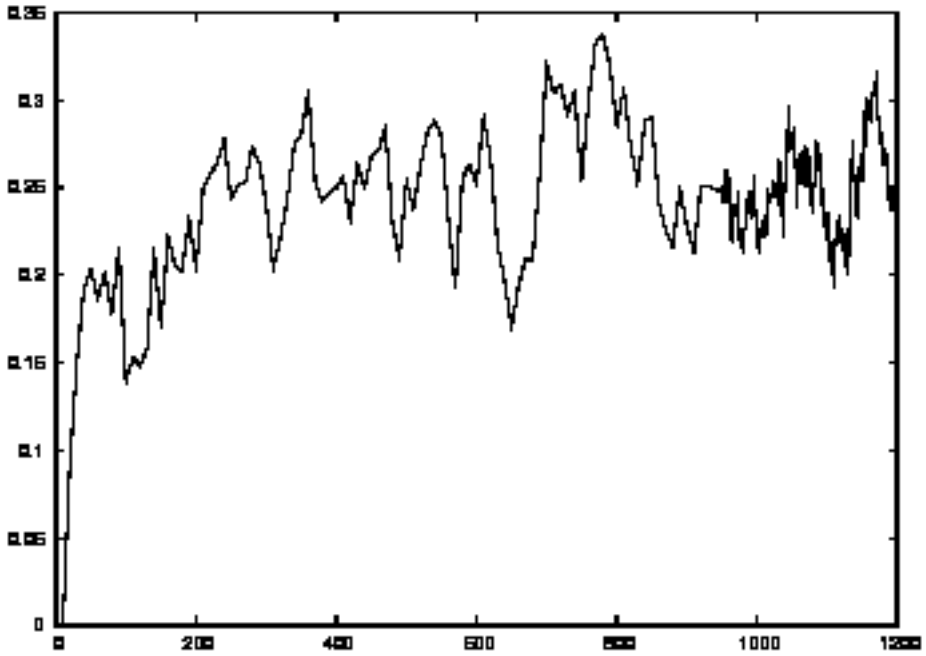
Not all models that are supported with analytical gradients in REML can also be done with Gibbs Sampling. The standard animal models in their univariate and multivariate rendition should work. In the end you need to see for yourself and just try it out. Switching from Gibbs sampling to REML is easy: just replace in the solver section the keyword method='GI' bay method='AG' and run again. This switch is also instructive as the runtime differences become very obvious:

[1] Gnuplot is free software, included in the Linux distribution packages, or down-loadable from mirrors of http://www.gnuplot.info/

[2] 1^{st} column includes cycle number, and 2^{nd} includes 'burn' or 'conv'

while for GI you may be waiting hours for convergence AG may produce results minutes if not seconds. Just give it a shot.

Figure 2.1: *Gnuplot output example for burn-in*



3 Parameter file

All actions of VCE are driven by a parameter file. It contains information about the data input and output files as well as information about the model definition. As a general principle, defaults are taken when specifics are not given in the parameter file.

3.1 Definitions

Instructions for **VCE** are written in parameter file. The name of a parameter file can be any legal file name under your operating system except the reserved word 'help'.

The structure of the parameter file is illustrated in the following. In order, to be more clear, the section names in "Algorithms" are written in red and small caps. Keywords are written in blue and italic, options are also italic but in cyan. In addition to space, there are other delimiters and operators, some of them are required and the others are optional. They will appear like in magenta. Words in curly brackets "{}" are optional and can be omitted.

Section names, keywords, and options are reserved words. The general structure is given in 3.1.

Algorithm 3.1 General structure of a section

```
SECTION_NAME
   keyword_1 = value11 {value12 ...}
   keyword_1 = value11 {value12 ...}
   ;
   keyword_2 = value21 /option = expression
   ;
```

3.1.1 Section

A section begins with section name which must be written at the beginning of the line, i.e. begin in column 1. Each section appears only once. The name must be the complete section name or contain at least 6 characters from the beginning of the word.

A parameter file may contain seven sections: **Comment**, **Data**, **Model**, **Covariance**, **System**, **Output** and **End**. It consists of one or more statements. The deliminator semicolon ';' must appear after each statements in **Data**, **Model**, and **Covariance** section. These three sections are also mandatory. A complete generic parameter file is shown in Algorithm 3.2 and detailed in later chapters.

3.1.2 Statement

Statements in the parameter file consist of one or more keywords separated by semicolon. They can be written in more than one line without any continuation sign. Indent a statement within a section by at least one space in order to avoid problems if it starts with a special character for comment line (see also section 3.3) .

Statements in sections **COMMENT**, **SYSTEM**, and **OUTPUT** contain only simple keywords with one value assigned and thus, semicolon may be omitted.

3.1.3 Keyword

Keywords are reserved words listed in algorithms 1.2 and 1.3. Keywords are followed by variable(s), delimiters, numeric or logical value(s), options, and expression. The name must be the complete keyword name or contains at least 6 characters from the beginning of the word.

VCE prints out keywords and their default values whenever you type '**help**' instead of name for parameter file. "Reserved" means that it must not be used in any user specified parts of the parameter file like for effect and trait names.

3.1.4 Option

Options are appended to the end of statements and are separated from the main expression by a slash. They start with a reserved word followed by an equal sign and expression. If there is more than one option in one statements, they are separated by blanks.

$$..../cf = CLASS(time);$$

3.1.5 Variable

Variable names consist of letters, numbers, and underscore(s). Other characters in the name may cause some problems. The names must not be repeated. Start the name with a letter!

Names are limited to 30 characters, but shorter names may be preferred. It is suggested, that names are chosen such that the variables are recognized by the first 10 characters. **VCE** will distinguish them by their full names, however, they will appear only with short names (up to 10 characters) on outputs. For example, the two variables $dependent11$ and $dependent12$ will be used in **VCE** as two different traits. On output, you will see only $dependent1$ for both variables.

3.1.6 Delimiter

Legal delimiters are blanks, tabulators, semicolons, commas, parentheses, square brackets, slashes and single quotes. VCE replaces a tabulator by a single blank. While only one blank is required as a delimiter more than one can be added.

3.1.7 Expression

Expressions are used in **Model** and **Covariance** sections to compute new variables, describe statistical model and covariance structure. Expression consists of variables, operators, and functions.

3.1.8 Operators

Valid operators are +, -, *, and / . Some of them are limited only to some keywords. Check later sections for limitations.

3.1.9 Functions

A function is introduced by the function name following by variable or list of variables enclosed in parentheses. For example, the square root of variable x can be written as *sqrt*(*x*) and the Wilmink equation for lactation curve as *lw*(*days_in_milk*).

Most of standard FORTRAN functions are implemented. In addition, polynomials and lactation curves are also covered. See Table 3.3 on page 43for more details.

3.1.10 Constants

Constants may have numeric, character, or logical values. Numeric constants are only real values used as coefficients in model section or starting values for variance components. Character constants are enclosed in ordinary quotes. They are used to describe file names (like 'data.txt')and formats. Logical constants are .true. or .false. .

3.2 An example parameter file

An example of a parameter file is given in Algorithm 3.2. It is a rather complicated statistical model that uses a number of features that VCE presents. In the following the section structures of the parameter file is described in some detail.

3.3 Comment section

The **Comment** section (Algorithm 3.3) is used to describe a job. Each job has a short name the length of which is at most 10 alpha-numeric characters and will appear on every page of output. The default value is taken from parameter file name: the first 10 characters from the beginning or after the last slash. Job name can be specified also in comment section using keyword "**job**".

Algorithm 3.2 Example parameter file

```
COMMENT job  = jobname
   This is generic pfile for random regression and other
   models used in VCE generation 5.
DATA
   datfile     ='diet1.dat'
      format   ='(3f12.0,10f8.0)'
      dep      = rsp dgain bfat
      indep    = rasse year sex betr animal wt100 age
      header   = 0 ;
   datfile     ='diet2.dat'
      format ='(2f12.0,10f8.0)'
      dep      = rsp drip weight gain
      indep    = season rasse sex time_var age betr individual maternal
      group_by = animal
;
   pedfile   = 'diet2.ped'
      header = 0
      format = '(6i10)'
      link   = individual {maternal, paternal}
      dominance = Dind {Dmaternal, Dpaternal}
      {indep = animal sire dam {nesting_additive} {nesting_dominance}{group}}
      {parent = 1, 2}{F1 = 3....};
   ranfile   = 'diet1.ped' format = '(36x, 2i2)' link = herd
      indep  = herd nesting_effect ;
MODEL
 rsp dgain bfat = rasse + p2(wt100) + [1,p3(age)] betr + [p5(age)]individual;
                    /  cf = time_var ;
  drip          = rasse  +  sex     + [1,p3(age)] betr + individual;
  gain          = rasse  +  sex     + [1,p3(age)] betr + [lw(age)]individual;
                   /  cf = time_var;
  set dgain = gain;
  multi = dgain bfat by = breed ;
COVARIANCE
   eff%name(cov%nesting):  ....+ [...f_i ...]trait_j + ....
      cov_zero = trait_i trait_ii
      cov_zero = [...f_i ...]trait_j [...f_l ...]trait_k
      cf = f_k(z_k);
   residual (datfile, eff%name()) : cf = function(time_var);
   start_asc = './mycov.ascii'
   dump_bin = './mycovdump.bin'
SYSTEM
   skip_value = value
   missing_value = value
OUTPUT
   log_gibbs ='file_name'
END
```

Algorithm 3.3 COMMENT section

```
COMMENT job  = jobname
   This is generic pfile for random regression and other
   models used in VCE generation 6.
```

Comments may be written also within other sections. In such cases, comment lines must start by one of characters: *c*, *C*, *t*, *T*, %, #, ∗and !. Such a special character must be in the first column and followed by at least one blank. Comment lines are completely ignored by VCE. The comment lines do not appear on output by error messages or by *help* contrary to the content of the COMMENT section which is printed in the output log.

In the COMMENT section some restrictions apply:

- ▷ Sometimes, if you do not increment keywords or not use blank after special characters, the statements can be interpreted wrong.
- ▷ **Do not use reserved section names as first words in comments!**

3.4 DATA section

In this section, input data to VCE is defined. This comprises one or more data files containing measurement as well as auxiliary information and pedigree data.

3.4.1 Data sets

The DATA section (3.4) starts with section name "**Data**". It contains description of all data-sets. The three types of data-sets are:

- ▷ "**datfile**": describes input files with measurements. It contains file name, lists of dependent (traits, keyword "**dep**") and independent (effects, source of heterogeneity for residuals, keyword "**indep**") variables, format (keyword "**format**"), and grouping factor (keyword "**group_by**"). Multiple datfiles may be specified.
- ▷ "**pedfile**": statement contains file name, format, the name or list of additive genetic effect (keyword "**link**"), the name or list of non-additive genetic effects (keyword "**dominance**"), and optionally, a list of columns introduced by keyword "**indep**". **Only one pedfile statement is allowed.**
- ▷ "**ranfile**": statement contains file name, name of random effect (keyword "**link**"), format (keyword "**format**"), and the list of variables (keyword "**indep**"). There are only two variables expected: one for levels of random effect and the other for levels of heterogeneity. If it is missing, a homogeneous covariance matrix is assumed.

The description of each file may be written in more than one row and must be terminated by semicolon. All valid keywords in the data section are listed in Table 3.1.

Algorithm 3.4 DATA section

```
DATA
   datfile      = 'file_name'
      format    = 'format_specification'
      dep       = variables
      indep     = variables
      group_by  = variable
      header    = integer
      ;
   pedfile      = 'file_name'
      format    = 'format_specification'
      link      = additive genetic effects
      dominance = dominance genetic effects
      indep     = variables
      header    = integer
      ;
   ranfile      = 'file_name'
      format    = 'format_specification'
      link      = variable
      indep     = variables
      header    = integer
      ;
```

3.4.2 Keywords in DATA section

File name. A file name follows the keyword which specifies the type of the data set. File names must appear within single quotes and can be any legal file name including the directory path. Notice, that file names may be case sensitive, depending on the operating system. Further, observe that the file name must be specified such, that the file can be reached from the point of invocation of VCE in the directory hierarchy the the file name string specified.

Examples. The data file *measure.dat* can be found in directory *../test/data*:

$$datfile\ '../test/data/measure.dat'$$

.

The pedigree file *measure.pedig* is located in the current directory:

$$pedfile\ 'measure.pedig'$$

Table 3.1: Keywords in DATA section

KeyWord	Defaults	Alternatives	Short description
datfile	' '	any legal file name	introduce data file
pedfile	' '	any legal file name	introduce pedigree file
ranfile	' '	any legal file name	file describing heterogeneity for trivial random effects
link		effects in the model	list of random effect(s) data in the file is to be used
dominance		effects in the model	list of genetic effects with dominance relationship
group_by		variable name	time or space dependent variable with residual correlated
format	free format (*)	any legal FORTRAN format	format for input data
header	1	0, 1, 2	number of rows to be skipped before reading data
indep			list of independent variables
dep			list of dependent variables

Format. The keyword "**format**" is followed by legal f90 formats and behave the same as in FORTRAN. The default is the FORTRAN free format and effective when the keyword "**format**" is missing. Formats must be written between ordinary quotes and parentheses. In data files, use only the F specifier for all variables including integers like effect codes. Dependent variables must be read first, followed by independent variables. On the other hand, only integer specifiers are expected for files describing random effects. T and X specifiers can be used for positioning.

Examples. The first four values have length 12 with 4 decimal digits. The next seven values have 8 digits and have no decimal points:

$$'(4f12.4,\ 7f8.0)'$$

Let's read the same data, but variables are in different order. First, we need to jump to the column 64 and read three variables of length 12 with 4 decimal digits. The last 12-digit value starts in the first column. Then, we need to skip 3 digits and six variables with length 6 and no decimal digits are read. The last variable starts in column 105 of the data file, has length 6 and no decimal digits.

$$'(t64,\ 3f12.4,\ t1,\ f12.4,\ 3x,\ 6f8.0,\ t105,\ f6.0)'$$

Header. Keyword "**header**" is used to skip the first few lines in data files if they contain header records. It must be placed within area for a data-set and is valid only for that data-set. The default value is "1" which is the correct setting is PEST is used for data coding.

Examples. Skip the first n-lines:

$$header = n$$

Do not skip any line:

$$header = 0$$

Dependent variables . Keyword "**dep**" or "**dependent**" introduces a list of dependent variables (traits). There may be more variables in the data section than needed in model section. In the list dependent variables can be separated by blanks, tabulator or comma.

Examples.

$$independent = daily\_gain,\ fce\ backfat$$

Independent variables. The keyword "**indep**" or "**independent**" introduces a list of independent variables: fixed and random effects, sources of heterogeneity, time or space variables as well as nesting or grouping variables. All variables needed for any purpose must be read at least from one data file. An exception is variable defined by keyword "**set**". A variable name may be repeated in data sets if it has exactly the same meaning.

In "**pedfile**", the list of independent variable is optional. The default columns in pedigrees file are animal, sire, and dam. If there is another type of pedigree information, for example

parent with grandparent, or effect causing heterogeneity for genetic effects, the specification of columns must follow the keyword "**indep**" or "**independent**". Independent variables can be separated by blanks, tabulator or comma. The names for the columns are reserved words from the Algorithm 3.2 and must not be changed. They define a position of variable in pedigree or level of heterogeneity for additive or non-additive genetic effects.

Table 3.2: Description of columns in pedigree file

KeyWord	Defaults	Alternatives	Short description
animal	animal	-	Animal; it does not need to be used for direct additive genetic effect.
sire	sire	-	Sire of the animal; it does not need to be used for paternal effect.
dam	dam	-	Dam of the animal; it does not need to be used for specification for maternal effect.
SS	SS	-	Sire of the sire; can not be present if sire exists.
DS	DS	-	Dam of the sire; cannot be present if sire exists.
SD	SD	-	Sire of the dam; cannot be present if dam exists.
DD	DD	-	Dam of the dam; cannot be present if dam exists.
group	group	-	Needed for selecting animals to be reported dominance effect for individuals.
any string	-	nesting effect for animal	The name of effect causing heterogeneity in additive genetic effects.
any string	-	nesting effect for dominance	The name of effect causing heterogeneity in dominance(s).

Examples. An ordinary list of independent variables in a data file is:

$$independent = breed\ season\ weight,\ time,\ litter,\ animal,\ fanimal$$

In this case, the model contains dominance effect $fanimal$. Remember, that effect *animal* is enough to compute dominance, but it must be read twice: once as additive (*animal*) and second as non-additive ($fanimal$) genetic effect. The effect names may differ, for example we want to use German names:

$$independent = rasse\ season\ gewicht,\ zeit,\ wurf,\ tier,\ dtier$$

Link. Keyword "**link**" connects file information with random effect it is used for. The effect name must be the same as in MODEL and COVARIANCE sections. Effects are not expected to be renamed. If the keyword "**link**" is followed by a list of effects, the effects are combined into one. The combined effects like animal and maternal are always treated as correlated (animal nd maternal should always be in this order: animal first; for mor information see later text).

Examples. The file is to be used for random effect litter:

$$link = litter$$

$$link = animal \;\; maternal$$

In the second example, the pedigree file is to be used for direct (animal) as well as maternal additive genetic effects. The two genetic effects are assumed to be correlated. The names specified behind LINK have to be from the list read from the data file. Each entry from this data file has to have an entry also in the pedigree file.

Paternal additive genetic effect may appear as the second or even as the third genetic effect on the list.

Dominance. Keyword "**dominance**" connects pedigree file information with the effects representing the dominance effect. As in the additive genetic effect, the dominance effect may also have maternal and paternal component. **The names given to non-additive and additive genetic components must not be the same.** The dominance effect must be listed as effect in **MODEL**and **COVARIANCE**section.

The dominance effect is automatically created from animal (default), maternal or paternal effect and is translated to a family effect. To ensure that the appropriate effect is assigned to a dominance component in the data files, we can read the correct level from the data file by listing effect names as independent variables. The second possibility exists in **MODEL**section with keyword '***set***'. Do not use both options at the same time! **VCE** may not understand you:-)

If you want to have also solutions for individual dominance effect, you must put in a request by using the keyword '***dominance***' into **OUTPUT**section.

Example 1. Use dominance relationship for effect $fanimal$:

$$dominance = fanimal$$

Let's try to evaluate individual ($dtier$) and maternal ($dsau$) dominance effect:

$$dominance = dtier \; dsau$$

There is a possibility to model dominance for paternal effect if you find it useful.

Example 2. Specify individual and maternal dominance effect. We write in **DATA** section for pedigree file:

$$dominance = fanimal\ fmaternal$$

To assign *animal* to $fanimal$ effect, one can read dominance *animal* and *maternal* effects twice as shown below. Be sure to use appropriate format.

$$independent = ... animal\ fanimal\ maternal\ fmaternal .....$$

The alternative is to write '***set***' statements under **MODEL** section.

$$set\ fanimal\ =\ animal;\ set\ fmaternal\ =\ maternal;$$

Group_by. The keyword "**group_by**" is needed to ensure correct grouping of correlated measurements. The keyword is followed by the grouping effect. The default value is 'record' or 'none'. In both cases, the observations are grouped as they appear in the input records. The grouping effect must be coded sequentially without missing values. For example, if the grouping effect is animal, animals with records must not be mixed with animals from pedigree, otherwise animals with data must also be coded twice: once with pedigree and second without pedigree.

3.5 MODEL section

MODEL section (Algorithm 3.5) consists of statements separated by semicolon. Statements describe statistical models, define transformations (keywords '**set**' , '**scale**', '**scaley**', '**scalex**',), or redefine traits (keywords '**equate**' and '**multi**' together with keyword '**by**').

3.5.1 Model statements

Model statements consists of three parts. On the left-hand-side, we always have a trait (dependent variable) name or list of traits separated by space or tabulator. A set of traits with the same model ends by equal sign (=). On the right-hand-side, the model statement contains explanatory effects, i.e. regression functions, constants, and independent variables - covariates. The last part of the model appears only in random regression models and is separated from the previous two parts by slash '/' and keyword '**/cf**'.

Algorithm 3.5 A generic **MODEL** section

```
MODEL
y_1 ... y_i = INT + eff_1
          + [1, f_k1(x_l)...)]eff_j
          + [f_k2(x_l, x_l1) ...)]eff_j1 ...
                  /  cf = f_cf(x_t) ;
y_i1  = eff_1 + [f_k1(x_l), f_k2(x_l, x_l1)]eff_2
 set   variable = expression ;
 scale variable_i ... variable_j option ;
 scaley option ;
 scalex option ;
 equate y_i = y_i1 ;
 multi = y_i by = eff_j ;
```

Traits (dependent variables). Traits (*y_1*, ..., *y_i*, *y_i1*) are written on the left-hand-side of the model. If model is the same for more than one trait, traits are listed in the same model statement and separated by space or tabulator.

Let's suppose, you want to apply different models to the same trait measured in different environment (time, space, etc.). Measurements are given different names like *y_i* and *y_i1* in generic algorithm (3.5) and described each with appropriate statistical model. Up to here, they are treated as different traits. Use the keyword '**equate**' in order to insure that all measurements are treated as the same trait.

Effects (independent variables). An effect (in algorithm 3.5) is fully described by the name (*effect_name*) and coefficients listed in square brackets in front of the name.

$$[1,\ f\_k1(x\_l),\ f\_k2(x\_l1,\ x\_l3]effect\_name$$

A set of coefficients for the model equations or design matrix is represented by constant values (1) and/or functions (*f_k1*, *f_k2*) separated by commas. Blanks after commas are allowed but they are not required. The most often used constant is 1 and is treated as default. Therefore, if the list of coefficients is missing, a coefficient with value 1 is assumed for each level presented in the data. All functions available are explained in Table 3.3.

Categorical or class effects are identified simply by their name (e.g. *rasse*, *betr*, and *animal* in Algorithm 3.6), while regression - if not nested - will be indicated by the first covariate (p1): the effect *of wt*100 as a linear regression would be written as: *p*1(*wt*100). Nested regression is indicated by the nesting effect (see effects *betr* and *animal*).

If you would like to have an effect in the model as categorical effect and additionally as nesting effect for a regression as shown in 3.1, the effect can be presented in form 3.2. The example is taken from the first model in 3.6. There are three traits (*rsp*, *dgain*, and *bfat*) with the same

model. The model contains four effects: *rasse*, *wt*100, *betr* (shown as B_j in 3.1), and *animal*. The function *p*3 indicates a three order polynomial without intercept. To add the intercept (B_j) in the model for each polynomial, we put constant 1 in front of the polynomial function. Constants must be separated from the function by a comma.

$$B_j \; + \; b_{Ij} \left(x_{ijk} - \overline{x} \right) \; + \; b_{IIj} \left(x_{ijk} - \overline{x} \right)^2 + \; b_{IIIj} \left(x_{ijk} - \overline{x} \right)^3 \tag{3.1}$$

$$[1, \; p3\,(age)]\, betr \tag{3.2}$$

Algorithm 3.6 Some examples of model statements

```
MODEL
 rsp bfat = rasse  +  p2(wt100) + [1,p3(age)] betr + [p5(age)]animal;
 dgain    = rasse  +  p2(wt100) + [1,p3(age)] betr + [p5(age)]animal
                   /  cf = CLASS(test_day) ;
 drip     = rasse  +  sex       + [1,p3(age)] betr + animal;
 gain     = rasse  +  sex       + [1,p3(age)] betr + [lw(age)]animal;
```

Intercept. In some models, it may be useful to include intercept in the model. This can be done by effect name '**int**' as shown in 3.3 and 3.4. Both expressions can be used and will do the same. The first one is preferred if variable *x* is used more than once.

$$[1, \; p3\,(x)]\, INT \tag{3.3}$$

$$INT \; + \; p3\,(x) \tag{3.4}$$

The keyword INT is also used as a function which changes a real value into an integer. In such cases, it is written in front of variable enclosed in brackets like in 3.5.

$$int(variable) \tag{3.5}$$

Constant coefficients Constant coefficients with alternative values 1 and 0 are default for categorical effects and are, thus, usually omitted. If you want to apply different one (a), use function $const(a)$.

The use of constant coefficients is described in the following example. In order to set up Reduced Animal Model, you must read effect sire and dam from the data and treat them as the same effect which will be named after the first effect in the model. In our case, the common effect is named sire. Renaming of the effect to a new name (to animal for example) may not work. You should use it in **DATA** section linking pedigree file as well as **COVARIANCE** section introducing it as random effect. When setting up the equation system from the model, the constant 0.5 is applied.

Algorithm 3.7 Reduced Animal Model

MODEL

```
bfat = rasse  +  [const(0.5)]sire + [const(0.5)]dam;
equate sire dam ;
```

Regression functions Some functions (Table 3.3) are given default names, the covariates must be listed in appropriate order. The other functions may be combined with "FORTRAN" functions, however, check the list before use. If a function or set of functions is nested within some other effect, it is enclosed in square brackets [] ahead of nesting effect (3.6).

There are two types of polynomial regression. The functions *pn* perform scaling of independent variable before computing coefficients of polynomials while the functions *pan* scale coefficients after they are computed. If no scaling is required the *pn* and *pan* polynomials are the same. The coefficients from Table 3.3 assume that scaling is involved with option *all* .

Table 3.3: List of functions in **VCE**

Name	Short description	No. of parameters	Coefficients
$const(a)$	take the constant value a	1	real value
$reg(x)$	linear regression	1	x
$p1$	linear regression	1	x
$p2$	quadratic regression	2	$x,\ x^2$
pn	polynomials where n = 2, 3, 4,...9	n	$\left(\frac{x-avg(x)}{std(x)}\right)^m$; m =1,...n
pan	polynomials where n = 1, 2, 3, ... 9	n	$\frac{x^m-avg(x^m)}{std(x^m)}$; m=1,.. n
$plgn$	Legendre polynomials where n = 0, 1, ...99	n+1	
$sqrt$	reg with covariate x transformed	1	$sqrt\,(x)$
sin	reg with covariate x transformed	1	$sin\,(x)$
tan	reg with covariate x transformed	1	$tan\,(x)$
log	reg with covariate x transformed	1	$ln\,(x)$
$log10$	reg with covariate x transformed	1	$log_{10}\,(x)$
$lw(x,m)$	lactation curve - Wilmink	3	$b_0 + b_1x + b_2e^{mx}$
$l1(x)$	lactation curve - Log model	3	$b_0 + b_1x + b_2e^z/x$; $z = \left(\frac{-0.5(log10(x)-1)}{0.6}\right)^2$
$l2(x)$	lactation curve - Mixed Log model I	3	$b_0 + b_1\sqrt{x} + b_2ln(x)$
$l3(x)$	lactation curve - Mixed Log Model III	4	$b_0 + b_1\sqrt{x} + b_2ln(x) + b_3x^4$
$l4(x,m)$	lactation curve - Swalve	5	$b_0 + b_1x + b_2x^2z + b_3x^3z + b_4e^{mx}$; $z = sin(\frac{t}{100})$
$l5(x)$	lactation curve - modified Khanderkar I	5	$b_0 + b_1x + b_2x^2 + b_3x^3 + b_4ln(x)$
$ls(x)$	lactation curve - Ali & Schaeffer 1997	5	$b_0 + b_1z + b_2z^2 + b_3u + b_4u^2$; $z = \frac{x}{305}$ and $u = log(\frac{305}{x})$
sr	lactation curve - ls for small ruminants	5	$b_0 + b_1z + b_2z^2 + b_3u + b_4u^2$; $z = \frac{x}{150}$ and $u = log(\frac{150}{x})$

Covariates (independent variables) Independent variables are enclosed in brackets () after a function name (Table 3.3). The covariables must be prepared in data files or derived from data as described in paragraph 3.5.2.

Covariance functions Covariance functions are used to describe longitudinal changes of (co)variances over time. It (3.6) consists of keyword "**/cf**", function name, and time variable. The equal sign may be omitted. A time-variable usually appears in parentheses after a function name (3.6).

$$/CF \;=\; CLASS\ (variable_t) \tag{3.6}$$

The keyword "**class**" can be omitted (3.7), the time variable may appear without brackets. Class is defined as integer part of a time-variable and is calculated as shown in 3.7. Be sure, that you create a reasonable number of classes!

$$class_t \;=\; INT(variable_t) \tag{3.7}$$

The time-variable must be listed in the effects for each data-set with a trait measured repeatedly over time. It is assumed to be the same for all traits listed on the left-hand-side of the model. If you need to have different time-dependent variable for some traits, write separate models. The models are appropriate also whenever measurements are repeated along a spatial cline. Then, the space variable will be used instead of time variable.

3.5.2 Transformation statements

Statement SET Transformation statements start with the keyword '**set**' followed by an equation with the general form (3.8). Transformation may be imposed on traits and covariates .

$$SET\ \ variable_{new} \;=\; function(variable) \tag{3.8}$$

$$SET\ \ variable_{new} \;=\; variable_1\ operator\ variable_2 \tag{3.9}$$

The first type of statements '**set**' (3.8) can be used with functions *log*, *log*10, *sqrt*, *sin*, *cos*, *tan*, *abs*, *int*, while the second type (3.9) is appropriate for arithmetic operations like +, −, ∗, / , and ∗∗.

Statement SCALE Scaling of dependent (traits) as well as independent (covariates) variables are useful to speed up the convergence and to avoid numerical problems. If required by functions like in Legendre polynomials or in lactation curves, it is done by default and the user does not have to do anything. Scaling is invoked by keyword '**scale**' followed by variable name and option(3.10) or keyword '**scaley**' (3.11) for all traits or '**scalex**' (3.12) for all covariates.

$$SCALE\ variable\ option \tag{3.10}$$

$$SCALEY\ option \tag{3.11}$$

$$SCALEX\ option \tag{3.12}$$

Scaling is performed only for those covariates where the function applied allows scaling (for example, in polynomials). All results on output are shown on the original scale if covariates are scaled also by standard deviation (options *all* and *std*). Generally, scaling is performed just after reading the data and checking for values to be treated as missing or required to be skipped. Variables which are transformed by functions listed in Table 3.3 like $SQRT$, LOG, SIN, etc. before using in linear regression, are scaled (if required) after transformation as shown in example 2 below.

The options can be chosen from the list in Table 3.4. The option *avg* is default for covariates, *all* for traits, and *s*11 for covariate in Legendre polynomials.

Table 3.4: Scaling options

Option	Meaning	Distribution or range of transformed variable
non	no change	between minimum and maximum value
avg	v - avg(v)	N(0, std(v))
std	v / std (v)	N(avg(v)/std(v), 1)
all	(v - avg(v)) / std(v)	N(0, 1)
*s*01	(v - min(v)) / (max(v)-min(v))	between 0 and 1
*s*11	2*(v - min(v)) / (max(v)-min(v)) - 1	between -1 and +1

Example 1. The model contains among others polynomial regression with covariable x (3.13). Subtraction of the average is required for covariable x (3.14). There is no problem to use the model shown in equation 3.15.

$$... + p3(x) + ... \tag{3.13}$$

$$scale\ x\ avg \tag{3.14}$$

$$... + b_1(x - \overline{x}) + b_2(x - \overline{x})^2 + b_3(x - \overline{x})^3 + ... \tag{3.15}$$

Example 2. The model contains among others polynomial regression with covariable x (3.16). The difference from the previous example is in scaling. Subtraction of the average is required for covariable x for each power (3.17).

$$... + pa3(x) + ... \tag{3.16}$$

$$... + b_1(x - \overline{x}) + b_2(x^2 - \overline{x^2}) + b_3(x^3 - \overline{x^3}) + ... \tag{3.17}$$

Example 2. Let's assume the model requires a linear regression applied to log-transformed variable (expression 3.18). The user specifies:

$$... + log10(x) + ... \tag{3.18}$$

$$scale\ x\ all \tag{3.19}$$

VCE understands instructions this way:

$$... + b[\frac{log_{10}(x) - \overline{log_{10}(x)}}{\sigma_{log_{10}(x)}}] + ... \tag{3.20}$$

Transformation of variable x would lead to wrong results. Therefore, it is not performed, no matter what is specified in the parameter file! If you request scaling, it would be done on the log-transformed variable.

Example 3. Let us suppose that we are working with lactation curve *ls*. It is enough to write the model with expression 3.21.

$$... + ls(days\_in\_milk) + ... \tag{3.21}$$

This function is performing its own transformations of variable *days_in_milk*. If the **Model** section of your parameter file contained for example statement *scale days_in_milk all*, scaling would not be used for variable *days_in_milk*, but only for others where restrictions do not apply.

Example 4. We are trying to use Legendre polynomial of order 9 with independent variable *test_day* (3.22). The variable *test_day* is always transformed by option *s*11.

$$... + plg9(test\_day) + ... \tag{3.22}$$

Statement EQUATE The second form of the expression contains the keyword '**equate**' which can be used to equate traits with different models (3.23) or effects (3.24). Effects must be presented with the same function, however, they may have different constant.

$$EQUATE\ trait_i\ trait_j\ ... \tag{3.23}$$

$$EQUATE\ effect_i\ effect_j\ ... \tag{3.24}$$

Statement MULTI

$$MULTI\ =\ trait_i\ BY\ =\ breed$$

3.5.3 Restrictions

- ▷ Effect animal should not be the first effect in the model.
- ▷ Effect animal should be written in the model before maternal or paternal effects.
- ▷ Use effect name only once in the model. Try to combine as in 3.2 or rename them to avoid conflicts.
- ▷ **In MULTI statements defined sub-traits are assumed to have the same coefficients (limitation in DefineCoeff)**
- ▷ **Effects in EQUATE statements must be described with the same function. They can differ only in a constant coefficient.**

Table 3.5: Keywords in **MODEL** section

KeyWord	Defaults	Alternatives	Short description
set	-		allows transformation of dependent variables and covariates
equate	-	list of traits or effects	allows different models for the same trait or put some effects on the same starting position
multi ... by...	-		treats traits as different traits within effect named after *by*
/cf	' '	function(variable), variable	covariance function for residuals
INT	INT	-	intercept

3.6 COVARIANCE section

The **COVARIANCE** section describes the structure of covariance matrices. The section contains statements which must end with a semicolon. Statements may be extended to more than one

line. There are two types of statements (Algorithm 3.8): the first describes covariance matrices (*eff_name*, ***residual***) and the second one explains if covariance matrices are specified by the user (**start_asc, start_bin, dump_asc, dump_bin**). If residuals are treated homogeneous, the keyword '***residual***' can be omitted.

Algorithm 3.8 A generic **COVARIANCE** section

```
COVARIANCE
   eff_name(eff_nesting):  ....+ [...f_i ...]trait_j + ...;
   residual (datfile, eff_nesting);
   start_asc = './mycov.ascii'
   dump_bin = './mycovdump.bin'
```

3.6.1 Description of random effects

The specification of a covariance matrix has two parts. The first introduces a name of the random effect and the second lists traits where the effect is treated as random. The delimiter between the two parts is a colon. Whenever the effect is treated random for all traits, the second part and colon can be omitted.

Table 3.6: Statements in **COVARIANCE** section

Part	I	II
Statement	Effect_name(nesting):	List of traits with functions;
Example 1	animal	;
Example 2	animal(breed):	dailygain + backfat;
Example 3	residual(datfile)	;
Example 4	herd:	[lpg3]dailygain + backfat

Random effects. Each statement starts with a name for random effect. The name is obligatory for all random effects in the model except residual.

By default, all covariances are assumed to be homogeneous, also among data-sets. To make them heterogeneous, the effect causing heterogeneity has to be mentioned in brackets. Use keyword '**datfile**' to insure the heterogeneity among multiple data-sets. Besides the data-sets, there may be other nesting effects. The nesting effect does not need to be in the model, however, it must be defined in input file for each record.

Heterogeneity is assumed to be caused by the same source for all additive genetic effects. The same or the other source can be specified for dominance effects. The pedigree file must contain the levels for all sources of heterogeneity.

For trivial random effects, the source of heterogeneity is specified in a separate data-set in **DATA** section introduced by keyword '**ranfile**' (see 3.4.1). A record usually contains only the level code for random effect and level of heterogeneity.

Traits. The second part of statement contains the list of traits or functions within trait where the effect is treated as random. The list may be omitted, if the effect is treated as random for all traits. The effect can be mixed: some parts may be treated as random, some as fixed.

- ▷ If only the effect is mentioned, the random effect has expected value 0, it is treated as random for all traits; the rank of the covariance matrix is the same as its order.
- ▷ If only a trait is mentioned, an effect is random for all functions describing this trait.
- ▷ The function name is sufficient; the function listed is treated as random, the function skipped is fixed. All terms within function are treated the same.
- ▷ Constant coefficients like '1' are treated the same way as functions and has to be mentioned to be treated random.
- ▷ The statements "**residual**" has to appear whenever there are heterogeneous covariances either among data-sets or other source (effect). A list of traits is not expected.

Examples A few examples as given in Table 3.6 will be discussed here.

Example 1. The first example in Table 3.6 is the simplest case: it requests the effect *animal* to be random for all traits. If you want to add relationship, see **DATA** section (3.4.1).

Example 2. The second example in Table 3.6 requests heterogeneous covariance matrices for the effect *animal*. The heterogeneity is caused by effect *breed*. Effect *animal* is treated random for traits *dailygain* and *backfat* but not for others. For other traits, if there are any, the effect is treated as fixed.

Example 3. The third example in Table 3.6 shows how to consider heterogeneous covariances among data sets.

Example 4. The last example in Table 3.6 requires effect *herd* to be treated random for trait *backfat* and partially for trait *dailygain*. The covariance components for *dailygain* are estimated only for those coefficients produced by Legendre polynomial of order 3. Elsewhere, the effect is treated as fixed.

3.6.2 Starting values for covariance components

The default procedure for estimating covariance component uses starting values in the middle of the parameter space with covariances close to zero and variances evenly splitting the total phenotypic variation. However, starting values for optimization can be

- ▷ read from file produced by previous evaluation run of VCE
- ▷ read from file written by the user

The first and second option are activated by the keywords '***start_bin***' and '***start_asc'*** in **COVARIANCE** section following the file name.

Default starting values. It is the best if starting values are taken from the middle of parameter space. Thus, variances are equally distributed among components, covariances are assumed to be close to zero.

Starting values from previous optimization. Starting values may be taken from previous optimization. This is very useful if convergence is not reached because of low limits on iterations allowed ('***maxiter***'), if we get additional data, or if old job has been killed unexpectedly. Notice, that the binary dump file is always written even if you do not specify dump_bin in the covariance section. As stated above, it defaults to **'jobname.cov-bin**'

The file can be used if specified in **COVARIANCE** section by keyword '***start_bin'*** followed by any file name. If the file name is omitted, the default name applies which is the job name and extension '.cov-bin'. NOTICE THAT - DUE TO SOME SHORTCOMINGS IN THE PARSER THE EFFECTS MUST COME FIRST IN THE COVARIANCE SECTION!

The file contains covariance estimates and only those information that can not be extracted from parameter file. The file will be overwritten when you restart the job. If you want to keep it, make a copy which you can also use for reading in the values for restart. In order to continue, the parameter file must not change in **MODEL** and **COVARIANCE** section and the same method has to be chosen. It can be used only with the same covariance structure as the old optimization. In case of nested random effects or residual, the nesting effect should have the same number of levels. If you want to start (slightly) different job with the covariance components from old job, it is better to provide user defined starting values as described below.

User defined starting values for covariances. User defined starting values for covariance matrices are placed in a separate file. The file must be specified in covariance section by keyword '***start_asc***'

The file must contain the basic information about the matrices on input. Each matrix is first described by a heading followed by a matrix. The heading of a matrix must be written in one line only.

The heading must contain information to which covariance matrix the following matrix will be assigned. The name of covariance matrix follows the keyword ***'link'*** and must be the same as in **COVARIANCE** section. There is no default value! If no other information is present, the covariance components are expected to be in lower triangular form and will be read in free format and interpreted as ratios. The same starting values will be used for all levels of heterogeneity.

Starting values may be specified as covariance matrices ('nat') or ratios ('rat') following keyword '***type***'. If the keyword is omitted, the default value is 'rat'.

Matrices may be inserted in four different forms (keyword '***mform***'): 'full', 'upper', 'lower', and 'diag'. The default is the lower triangular matrix.

Algorithm 3.9 Description of covariance matrices in ***'start_asc'***

```
link = animal level = level_code
mform ='lower' type ='rat' format =any_fortran_format
 .30
 .20   .35
-.10   .01   .20
link = herd form = 'full'
 0    0     0
 0   .35   .01
 0   .01   .20
link = residual level = level_code datfile =any_legal_file_name
mform ='upper' type ='nat' format =any_fortran_format
 317   -10.17   56.01   .01
 2.45  -12.12    .25
 1756     .02
 0.35
```

Matrices must have the appropriate size (order) which depend on the number of traits and coefficients per trait. The covariances must be typed in the same order as defined by the model. Coefficients listed by random effect are nested within trait. You can check the number of coefficients for functions applied in table 3.3. If records contain longitudinal data, traits in residual covariances are nested within class (see 3.6.4).

If some coefficients (rows/columns) are to be treated as fixed, the covariance components are set to zero. If a component specified can not be estimated from the data, the component will be deleted.

If some levels for heterogeneous covariance matrices are missing, the corresponding matrices are filled with starting values for 'level' 1.

Notice, that you always also need a "residual" covariance matrix.

3.6.3 Some examples of COVARIANCE section

Example 1. The only random effect in the model is *animal*. Residual covariance is assumed to be the same for all data-sets.

Algorithm 3.10 Only additive genetic effect

```
COVARIANCE
   animal;
```

Example 2. Random effect *animal* has heterogeneous covariances among *breeds*. Herd is the second random effect for traits listed and fixed for all other possible traits with *herd* effect in

Table 3.7: Keywords in **COVARIANCE** section

KeyWord	Defaults	Alternatives	Short description
residual			introduce residual covariance matrix
datfile	.true.		causes residual covariances to be heterogeneous among data sets
start_asc	jobname.cov-asc	any file name	introduces file with user defined starting covariance matrices
start_bin	jobname.cov-bin	any file name	has to be a file from a previous run
dump_bin	jobname.cov-bin	any file name	file contains the best results from previous optimization
dump_asc	jobname.cov-asc	any file name	file contains the best results from previous optimization (mainly useful for the automatic testing done by the VCE developers)
link	' '	method//job//'.cov' covariance name	in case Keyword is present without a value covariance name must match with the name in covariance section
level	all levels	positive integer	code for nesting effect
mform	'lower'	'upper', 'full', 'diag'	how the matrix is written
type	'rat'	'nat'	do matrices contain ratios ('rat') or covariances ('nat')
format	'(*)'	Fortran format	format for the longest row

the model. Residual is heterogeneous among data-sets. Starting values will be read from file './myfile.cov.'.

Algorithm 3.11 Nested covariance matrices for additive genetic effect and residual

```
COVARIANCE
   animal (breed);
   herd                : bfft adgft adgst;
   residual (datfile);
   start_asc ='./myfile.cov'
```

Example 3. Starting values for covariance components are found in a separate file.

There are two residual matrices, one for each data-set. There are assumed to be diagonal matrix. Diagonal elements may be written in one or more rows. Residual covariances are assumed to be zero and will not be computed.

Covariance matrix for *herd* effect is complete. Make sure that matrix is symmetric! The last row and column are zero, because the *herd* effect is treated as fixed for the last trait.

Covariance matrix for the animal effect is given in upper form. Only covariances among neighboring elements are supposed to be non-zero. The three covariances where starting values are zero will not be optimized for.

Algorithm 3.12 Starting values for example 2

```
link = residual mform ='diag' datfile ='./data1'
  317.  2.45  1756.  0.35
link = residual mform ='diag' datfile ='./data2'
  217.  1.45  1456.  0.45
link = herd mform ='upper'
 145.   .01    .01   0
  .01  1.37    .01   0
  .01   .01   826.   0
   0     0      0    0
link = animal mform ='upper'
 145.   .01
       1.37   .01
              826.   .01
                     0.18
```

3.6.4 Residual covariance matrices

Simple models. Residual covariance matrix R (3.25) is a direct sum of residual matrices R_{ij} (3.26) which differs due to missing values presented by selector matrix P_i or heterogeneity

caused by an effect (level j). The transformation matrix P_i is derived from identity matrix by setting diagonal element that correspond to missing observation to zero. R_{0j} (3.27) is matrix of order trait by trait which contains variances on diagonal and covariances on off-diagonal elements for residuals of all traits in a multi-trait model.

$$R = \sum \bigoplus R_{ij} \tag{3.25}$$

$$R_{ij} = P_i * R_{0j} * P_i^T \tag{3.26}$$

$$R_{0j} = \begin{bmatrix} \sigma^2_{je1} & \sigma_{je12} & \sigma_{je13} \\ & \sigma^2_{je2} & \sigma_{je23} \\ sym. & & \sigma^2_{je3} \end{bmatrix} \tag{3.27}$$

Random regression models. Measurements repeated over time have different variance, the changes may be described by a covariance function or "within time-classes". The residuals are correlated within group-classes. The residual matrix R has also block-diagonal structure as in 3.25. The covariance matrix in each block R_{ij} may differ because of missing values or heterogeneity. Assuming no heterogeneity (subscript j is omitted), the block-diagonal matrices R_i are derived from R_0 in 3.28. R_0 is suitable for models with three traits (dependent variables) which are measured sequentially m-times.

$$R_0 = \begin{bmatrix} \sigma^2_{e11} & \sigma_{e11e21} & \sigma_{e11e31} & \cdots & \sigma_{e11e1t} & \sigma_{e11e2t} & \sigma_{e11e3t} & \cdots & \sigma_{e11e1m} & \sigma_{e11e2m} & \sigma_{e11e3m} \\ & \sigma^2_{e21} & & \cdots & \sigma_{e21e1t} & \sigma_{e21e2t} & \sigma_{e21e3t} & \cdots & \sigma_{e21e1m} & \sigma_{e21e2m} & \sigma_{e21e3m} \\ & & \sigma^2_{e31} & \cdots & \sigma_{e31e1t} & \sigma_{e31e2t} & \sigma_{e31e3t} & \cdots & \sigma_{e31e1m} & \sigma_{e31e2m} & \sigma_{e31e3m} \\ & & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ & & & & \sigma^2_{e1t} & \sigma_{e1te2t} & \sigma_{e1te3t} & \cdots & \sigma_{e1t1m} & \sigma_{e1t2m} & \sigma_{e1t3m} \\ & & & & & \sigma^2_{e2t} & \sigma_{e2t3t} & \cdots & \sigma_{e2t1m} & \sigma_{e2t2m} & \sigma_{e2t3m} \\ & & & & & & \sigma^2_{e3t} & \cdots & \sigma_{e3t1m} & \sigma_{e3t2m} & \sigma_{e3t3m} \\ & & & & & & & \ddots & \vdots & \vdots & \vdots \\ & & & & & & & & \sigma^2_{e1m} & \sigma_{e1me2m} & \sigma_{e1m3m} \\ & sym. & & & & & & & & \sigma^2_{e2m} & \sigma_{e2m3m} \\ & & & & & & & & & & \sigma^2_{e3m} \end{bmatrix} \tag{3.28}$$

Random regression models with repeated records. The models are the same as above. In addition, some measurements may be repeated at the same time.

Let's assume that in the last measurement (trait 3 in time m) is repeated three times. The matrix R_{03m} presents all the elements of matrix R_0 (3.28) except the last row and column. The last column without the last element σ^2_{e3m} is assigned to vector R_{03m}. Then, the extended residual

matrix R_0^* looks like (3.29). Element σ_{e3m} presents covariance among measurements of trait 3 in time m.

$$R_0^* = \begin{bmatrix} R_{03m} & R_{3m} & R_{3m} & R_{3m} \\ & \sigma^2_{e3m} & \sigma_{e3m} & \sigma_{e3m} \\ & & \sigma^2_{e3m} & \sigma_{e3m} \\ sym. & & & \sigma^2_{e3m} \end{bmatrix} \tag{3.29}$$

3.6.5 Covariance matrix for "trivial" random effects

Simple models. Matrix is block-diagonal (3.30) with block (3.31) size equal to the number of traits.

$$G_H = I \otimes G_h \tag{3.30}$$

$$G_h = \begin{bmatrix} \sigma^2_{h1} & \sigma_{h12} & \sigma_{h13} \\ & \sigma^2_{h2} & \sigma_{h23} \\ sym. & & \sigma^2_{h3} \end{bmatrix} \tag{3.31}$$

Random regression models. A trait is described by a function (or more functions, see model section 3.5) with one or more coefficients. In our example (3.32), the first trait is described by a function with two, trait i with three, and trait n with q coefficients . The zero columns mean that these parts are treated as fixed.

$$G_h = \begin{bmatrix} \sigma^2_{h11} & \sigma_{h11h12} & \cdots & \sigma_{h11hi1} & \sigma_{h11hi2} & \sigma_{h11ai3} & \cdots & 0 & 0 & \sigma_{h11hnq} \\ & \sigma^2_{h12} & \cdots & \sigma_{h12hi1} & \sigma_{h12hi2} & \sigma_{h12hi3} & \cdots & 0 & 0 & \sigma_{h12hnq} \\ & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ & & & \sigma^2_{hi1} & \sigma_{hi1hi2} & \sigma_{hi1hi3} & \cdots & 0 & 0 & \sigma_{hi1hnq} \\ & & & & \sigma^2_{hi2} & \sigma_{hi2hi3} & \cdots & 0 & 0 & \sigma_{hi2hnq} \\ & & & & & \sigma^2_{hi3} & \cdots & 0 & 0 & \sigma_{hi3hnq} \\ & & & & & & \ddots & \vdots & \vdots & \vdots \\ & & & & & & & 0 & 0 & 0 \\ & sym. & & & & & & & 0 & 0 \\ & & & & & & & & & \sigma^2_{hnq} \end{bmatrix} \tag{3.32}$$

Random regression for random effect with heterogeneous covariances Covariance matrices vary among nesting effect. The covariance matrix G_H (3.33) is direct sum of blocks G_{hj}, where subscript j stays for nesting effect. Each G_{hj} has the same form as G_h in 3.32. Nesting effect must be specified explicitly in **Data** section under keyword '**ranfile**' (3.4.1)!

$$G_H = \sum \oplus G_{hj} \tag{3.33}$$

3.6.6 Additive genetic covariance matrix

Simple models. Additive genetic covariance matrix

$$G_A = A \otimes G_a \tag{3.34}$$

$$G_a = \begin{bmatrix} \sigma^2_{a1} & \sigma_{a12} & \sigma_{a13} \\ & \sigma^2_{a2} & \sigma_{a23} \\ sym. & & \sigma^2_{a3} \end{bmatrix} \tag{3.35}$$

Random regression for direct additive genetic effects.

$$G_a = \begin{bmatrix} \sigma^2_{a11} & \sigma_{a11a12} & \cdots & \sigma_{a11ai1} & \sigma_{a11ai2} & \sigma_{a11ai3} & \cdots & \sigma_{a11an1} & \cdots & \sigma_{a11anq} \\ & \sigma^2_{a12} & \cdots & \sigma_{a12ai1} & \sigma_{a12ai2} & \sigma_{a12ai3} & \cdots & \sigma_{a12an1} & \cdots & \sigma_{a12anq} \\ & & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & \sigma^2_{ai1} & \sigma_{ai1ai2} & \sigma_{ai1ai3} & \cdots & \sigma_{ai1an1} & \cdots & \sigma_{ai1anq} \\ & & & & \sigma^2_{ai2} & \sigma_{ai2ai3} & \cdots & \sigma_{ai2an1} & \cdots & \sigma_{ai2anq} \\ & & & & & \sigma^2_{ai3} & \cdots & \sigma_{ai3an1} & \cdots & \sigma_{ai3anq} \\ & & & & & & \ddots & \vdots & \vdots & \vdots \\ & sym. & & & & & & \sigma^2_{an1} & \cdots & \sigma_{an1anq} \\ & & & & & & & & \ddots & \vdots \\ & & & & & & & & & \sigma^2_{anq} \end{bmatrix} \tag{3.36}$$

Models with two or more additive genetic effects. Additive genetic covariance matrix is more complex if maternal (m) and/or paternal (p) effects are added:

$$G_A = A \otimes \begin{bmatrix} G_a & G_{am} & G_{ap} \\ & G_m & G_{mp} \\ sym. & & G_p \end{bmatrix} \tag{3.37}$$

Heterogeneity of additive genetic effects. It is possible to account for heterogeneity of additive genetic effects. In such case, each animal must be assigned to an effect causing heterogeneity. The effect must be listed as additional column in pedigree file.

3.6.7 Dominance covariance matrix

Simple models. Dominance covariance matrix

$$G_D = D \otimes D_a \tag{3.38}$$

$$D_a = 4D_f = \begin{bmatrix} \sigma^2_{d1} & \sigma_{d12} & \sigma_{d13} \\ & \sigma^2_{d2} & \sigma_{d23} \\ sym. & & \sigma^2_{d3} \end{bmatrix} \tag{3.39}$$

Models with two or more dominance genetic effects. Dominance genetic covariance matrices can be more complex if maternal (m) and/or paternal (p) effects are added:

$$G_D = D \otimes \begin{bmatrix} D_a & D_{am} & D_{ap} \\ & D_m & D_{mp} \\ sym. & & D_p \end{bmatrix} \tag{3.40}$$

Heterogeneity of dominance genetic effects. It is possible to account for heterogeneity of dominance genetic effects. In such case, each animal must be assigned to an effect causing heterogeneity. The effect must be listed as additional column in pedigree file. It is assumed that family and its progeny belong to the same level, while individual parent may belong to the other. level.

3.7 SYSTEM section

SYSTEM section holds a whole variety of keywords they can be used to modify default values.

Algorithm 3.13 System section

```
SYSTEM
   skip_value = value
   missing_value = value
c ---------------------------- for gibbs
   burn_stop = real_value
   mark_stop = .001
```

3.7.1 Keywords driving iteration procedures

The keywords in the following table are driving iteration procedures.

The Gibbs sampling method can be invoked in the system section by setting the '**method**' keyword as GI. This keyword will be in most cases enough to perform the Bayesian inference on the variance components, i.e., keeping the rest of the keywords as default values.

Keyword '**mc_seed**' is used as seed value for random function.

Keywords involved in the first burn-in step are the following.

- '**burn_first**': Cycle number where the convergence for burn-in is checked for the first time.
- '**burn_max**': Maximum number of cycles allowed for burn-in.
- '**burn_next**': After '**burn_first**' cycles, the convergence for burn-in will be checked every '**burn_nex**t' cycles. VCE computes '**burn_next**' cycles of the first chain and '**burn_next**' cycles of the second one, then computes maximum differences. That is what is printed on the screen.
- '**burn_stop**': Convergence criteria is the maximum difference on ratios between both chains.

VCE computes the effective length size every several cycles looking for a Monte Carlo error small enough. It can be customized also with the following keywords.

- '**mark_first**': Cycle number where the Monte Carlo error is checked for the first time. It has to be set big enough to break the autocorrelation of the chain. If '**mark_first**' was set small and VCE stopped just after a few cycles, try to increase it to be sure that the Monte Carlo error does not grow after some hundreds of cycles more. We recommend to use the default or even to increase it in cases of unexpected fast convergence.
- '**mark_max**': Maximum number of cycles allowed for convergence after burn-in. Of course, it depends on your processor and your patience.
- '**mark_next**': After '**mark_first**' cycles, the convergence for Monte Carlo error will be checked every '**mark_next**' cycles. Users have to take into account that the calculation of the effective length size is slow for long chains. For that reason it is not recommended to calculate it every cycle.
- '**mark_stop**': Maximum Monte Carlo error allowed. Note that the Monte Carlo error has an exponential decay. Then, the Gibbs sampler will run forever to achieve tiny Monte Carlo errors.

As in the case of the burn-in, defaults should be enough for a usual variance component estimation job. Researchers interested in the behavior of the algorithms, or just because they prefer to do the post Gibbs analysis themselves, can configure these keywords to run fixed length chains, infinite chains, etc., or they can use the '**mc_seed**' keyword to repeat simulation experiments.

In case of '**method**' *sol*, user can specify also options for iterative solvers applying the following keywords.

- ▷ '**iod_first**': Iteration number where the convergence for solution is checked for the first time. It is not very important. Default value is 2.
- ▷ '**iod_max**': Maximum number of cycles allowed for iterative solvers.
- ▷ '**iod_next**': After '**iod_first**' iterations, the convergence for solutions will be checked every '**iod_next**' iterations. Default value is 1.
- ▷ '**iod_stop**': Convergence criteria is the maximum difference between two successive solutions.

Table 3.8: Keywords driving iteration procedures

KeyWord	Defaults	Alternatives	Short description
mc_seed	5511212312.0d0	any large number	seed value
burn_stop	.001	real value	stopping criteria for burnin-loop (first)
burn_first	1	integer value	number of tries within a chain in burnin-loop
burn_next	10	integer value	
burn_max	10000	integer value	maximum number of steps in burnin-loop
mark_stop	.001	real value	stopping criteria for mark-loop (second)
mark_first	500	integer value	number of steps before first checking in mark-loop
mark_next	10	integer value	number of steps between checkings in mark-loop. It must be smaller than mark_first!
mark_max	10000	integer value	maximum number of steps in mark-loop
iod_stop	.0001	real value	stopping criteria in solver iod
iod_first	2	integer value	number of steps before first checking in iod solver
iod_next	1	integer value	number of steps between checkings in iod solver
iod_max	1000	integer value	maximum number of iterations in solver iod
restart	.false.	.true.	restart after burning

3.7.2 Keywords concerning additive genetic relationship

The model with genetic groups is used by default whenever genetic groups are present in the pedigree file. It is expected that the pedigree file is coded with codes for genetic groups appended to the list of animals. However, it is assumed that the genetic group does not have a record in pedigree file. The number of animals is equal to the number of records in pedigree. Further, no unknown animals are allowed in this case. For more discussion see section Data Preparation 4.6.1. If consideration of inbreeding is not desired (I would not know, why one would want to use an "incorrect" procedure other than having a look), then keyword "inbreeding = .false." can be used.

3.7.3 Keywords connected with data manipulation

The options specified under **SYSTEM** section are valid for all data sets. Keywords define values to be skipped (***'missing_value'***, ***'skip_value'***) or set to zero (***'tolerance'***).

Table 3.9: Keywords connected with data manipulation

KeyWord	Defaults	Alternative values	Short description
missing_value	11111000.		values treated as missing
skip_value	11111000.		additional values treated as missing, also for covariates
tolerance	epsilon(real value)*1000	any real value	any small real value to be considered as zero

Missing_value: The keyword***'missing_value'*** is used to define missing value for dependent variable (trait). Value is deleted when read from the file. It is not used in any evaluation. It is applied to all of them at the same time.

Skip_value: The keyword ***'skip_value'*** is used to define missing value for dependent variables (traits) as well as for independent variables (covariables or class effects). It is applied to all of them at the same time.

Tolerance: The keyword '***tolerance***' specified the smallest value to be treated different from zero. Default value depends on machine precision.

3.7.4 Other keywords in SYSTEM section

Table 3.10: Other keywords in SYSTEM section

KeyWord	Default values	Alternative values	Short description
method	AG	GI	which method is to be used
solve	see table 3.11	dir, ioc, iod, both, cg	which solver is to be used
non_zero	1000000		number of non-zero elements
total	4000000		total amount of memory required by solver
reparameterize	.true.	.false.	eliminate dependent rows and columns

Method In VCE6, two methods are available: AG and GI are used for estimation of dispersion parameters. They are introduced by keyword '***method'***.

The default method is a gradient method AG while the method GI implies Gibbs sampling.

Solve Keyword '***solve'*** is specified to choose the solver. The default value depends on the method (Table 3.11).

Table 3.11: Methods and solver

Solver	Description	AG	GI
dir	direct solution (SMP)	+	+
ioc	iteration on coefficients	+	+
default		dir	ioc

3.8 File naming conventions

VCE produces a number of files all related to one run. It makes sense to be able to handle all those together, either to copy, move or delete all of them through one command like “cp np01* np01_dir/”. As a default (i.e. if filenames are not specified by the user in the parameter file), all newly generated files start with the parameter file name. The default set of file names is derived from the name of the parameter file. All the default files start with the parameter file name and then get an extension that relates to the type of file. In this way, you can move/delete all files from one parameters files run with one command like : mv np01-AG* backupdir/, or : “ls np01-AG*” will list all files that belong to the run specified in parameter file np01-AG. However, you are not completely free in choosing the parameter filename if you want this feature to work: The parameter file should not contain a "." as this is used for the derived files. Thus, if you want to qualify a parameter file use a"-something" like in np01-AG because the “.AG” would get dropped for derived filenames if you choose the parameter file to be named “np01.AG”.

The list of default files for the parameter file “np01” is given in table 3.12.

If we want to change the base filename for all the new files created by VCE this is done by:

“COMMENT job=best_run”

Then all default output file names will look like: “best_run.lst”, “best_run.cov-bin” etc.

3.9 OUTPUT section

OUTPUT section is used to create desirable outputs. Each statement starts with a keyword listed in algorithm 3.14. The default output file name consists of the job name and gets three letters as extension separated by a dot.

3.9.1 Keywords in OUTPUT section

Coefficient matrix.

The keyword ‘**lhs**’ allows to print out part of the coefficient matrix as well as right-hand-side. If the matrix is larger, only 40 rows and columns are printed. The right-hand-side is printed in the first column. The default format is ‘(41f9.4)’ and can be changed by option ‘**format**’.

Table 3.12: default file name for parameter file np01

file name	description	can be changed by section:keyword
np01.lst	contains the complete run log with results, ready to be printed	
np01.cov-bin	is the binary results file; It is written in each iteration and contains the cov matrices and can be used to print the covs in user defined formats; This is done via the the keyword reprint	covariance:dump_bin
np01.cov-asc	the same as previous, but in ASCII formatted output	output:covfile
np01.dom	dominance	output:dominance
np01.fam	family information	output:family
np01.gib	the Gibbs log file	output:gibbs_log
np01.inb	inbreeding coefficients	output:inbreeding
np01.lhs	left hand side of the mixed mode equations - make only sense for really small examples	output:lhs

Algorithm 3.14 OUTPUT section

```
OUTPUT
   covfile     = 'file_name' format = '(format)' next = integer;
   inbreeding  = 'file_name' format = '(format)' next = integer;
   lhs         = 'file_name' format = '(format)' next = integer;
   dominance   = 'file_name' format = '(format)' next = integer;
   vcm         = 'file_name' format = '(format)' next = integer;
   family      = 'file_name';
   log_gibbs   = 'file_name';
```

Table 3.13: Keywords in **OUTPUT** section

KeyWord	Defaults	Alternatives	Short description
log_gibbs	-	any legal file name	write information about optimization also into a file
dominance	-	any legal file name	write solutions for individual dominance effect
family	-	any legal file name	write family subclasses into a file
covfile	-	any legal file name	Natural and ratio covariances are printed into file
inbreeding	-	any legal file name	Inbreeding coefficients is printed into file
lhs	-	any legal file name	LHS is printed out into file
vcm	-	any legal file name	the variance covariance matrix of all components is written to the file
format	-	any legal FORTRAN format	
mform	-	FULL|UPPER|LOWER	specifies the form of the covariance matrices
reprint			uses the binary log file to reformat the cov matrices. It can be used as the vcestat in vce4. Together with the covfile and its format and form keyword, the matrices can be printed as full, upper or lower.

The option '**next**' determines the iteration when the coefficient matrix is to be printed into the file.

Dominance. The user can print out predictors for individual dominance effect by writing the keyword '**dominance**' under output section. They will be printed in a separate file.

Inbreeding. The keyword '**inbreeding**' prints out the inbreeding coefficients for the animals.

Gibbs_log. Keyword '**gibbs_log**' makes VCE write the Gibbs samples to the corresponding file. See 2.3.2 on page 27for details and use.

Reprint. Sometime, the format of the covariance matrices need to be changed from the default output that VCE generates. This can be done by specifying the covfile keyword in the OUTPUT section. However, this would require rerunning the job (which you may not want to do if you have already waited for two weeks before). As the covariance matrices are stored in full floating point accuracy in the binary log file (defaults to jobname.cov-bin) it is sufficient to use this. REPRINT does just that: if the parameter file is run again with REPRINT added as a keyword in the OUTPUT section, it reads the binary log file (either the default or the one specified by DUMP_BIN='filename' in the COVARIANCE section) and prints it in the format and form specified in the COVFILE . If you want to use the covariance matrices directly in some other package like PEST, you will need it in full format and not in the default upper triangle. Also, a matrix may not be positive definite in its default printed format. Then something like this may help:

```
OUTPUT
  REPRINT;
  COVFILE='MYFULLMATRIX.TXT' FORMAT='(12f14.7)' MFORM='FULL';
```

Notice: if you insert the above REPRINT lines into your original parameter file and run it, the run log file will get overwritten. If you do not want that, rename the parameter file to something else.

Mem_map. The keyword '**mem_map**' is intended for debugging purposes.

VCM. The keyword **VCM** triggers the printing of the covariance matrices of the component estimates. Thus, if you have a simple univariate model with one random and one animal component, VCM would result in the printing of the variance covariance matrix of these three variances: residual, simple random and animal component. This would then be a matrix of order 3.

3.9.2 Options with keywords in OUTPUT section

Options in **OUTPUT** section are available only for keywords listed in Table 3.14 and they apply only if keyword is used.

Table 3.14: file related defaults in **OUTPUT** section

Keyword	Default file name	Default **format**	Default **next**
log_gibbs	job//'.gib'	/	/
dominance	job//'.dom'		
family	job//'.fam'	/	/
covfile	job//'.cov-fmt'	'(e10.4)'	1
inbreeding	job//'.inb'	'(i10,f8.5)'	0
lhs	job//'.lhs'	'(41f9.4)'	1
vcm	job//'.vcm'	'(10e18.10)'	

3.10 END section

Section END is optional. It can be written at the end of parameter file or where we want the instruction to end. The text below is ignored. Therefore, you can put the last results or longer comments at the end of parameter file.

4 Examples and Use Cases

In the this chapter we shall present a number of typical models that users can take as guidelines for their own problems. The examples include parameter files and discusses issues that seem worth mentioning. We present typical models used in animal breeding. This is not intended to be an exhaustive treatment of the issue but rather presents a mixed bag of models that a prospective user can look at to find one that is closest to his own problem. Then this particular one could be used as a starting point.

Before we can deal with any model we need to explain the general problem of coding input data.

4.1 Coding the input data

To be able to run VCE all classes effects need to be coded consecutively starting at 1 and not having any "holes". Let us assume that we have four animal names in the dataset (used to be the case in historical times), so I pick the names of some of our cows that I grew up with): Edda, Sirene, Fokka, Olga. After coding the column that contains the animal names (i.e. their ID) would contain the number 1 to 4. The procedure first sorts the animal names: Edda, Fokka, Olga and Sirene. Correspondingly, Edda becomes 1, Fokka 2, Olga 3, and Sirene 4. The same procedure is applied to all class codes: FEMALE becomes 1, CASTRATE 2, and MALE 3. This can be done with any program. Often, people use PEST, which can be used conveniently for this purpose. A corresponding parameter file is given in listing 4.1. If PEST is being used a few comments are in order.

1. The objective of a PEST run in the context of variance component estimation as a preparatory step for running VCE is only the data coding. To this effect an OUTFILE keyword has to be specified in the RELATIONSHIP and DATA section as shown in lines 5 and 13. "[text]" has to be appended, to generate an ASCII file, else you will get a binary file that VCE cannot handle. Notice that the 'data/ex01.dat' will create the file ex01.dat in the directory 'data' from the point in the directory tree, where PEST is started.
2. under INPUT (line 14) all traits and effects are listed. For more information see the PEST User's Guide.
3. the MODEL section contains all those elements from the above INPUT part that you want included in the coded data file. This means, all those traits should get included here, that you want to used in the VCE run. More precisely, the VCE run can use a subset but obviously not more than specified here. In this example we use the five traits backfat, loin, drip, redfibr and feed plus the effects age, sdate, sex, herd, litter and animal.

4. As we are not interested in the computation of BLUPs and BLUEs, the composition of the covariance matrices are irrelevant as long as they are positive definite.

5. The same rational applies to the SOLVER section: no solutions are required, only the data coding part needs to be done, and then we want PEST to exit gracefully as soon as possible. Thus, we limit the number of iterations as in line 51 and 52. Also, we do not want to use a lot of memory, that's why we put the effects into IOD and IOD_GS and specified in the same lines. Any other combination would also do.

6. Finally, in the original dataset 999 where used to indicate missing values. Thus, we put in a TRANSFORMATION section (lines 53–59), which translates the 99 into the string 11111111. used as a default in VCE to indicate zero.

The first 4 lines from the data file 'scr-msee.dat' are given in Listing 4.2. This is the output from a PEST run that was used for recoding the data (Listing 4.1 on the facing page).

PEST inserts a first line containing the names of the covariables, the traits and finally the class effects. The format of the continuous variables may look somewhat strange, but it is the general floating point representation from the programming language FORTRAN, the language that VCE is written in. From the PEST coding traits are always written first to the file. In this case they are columns 1 through 5. This is followed by one covariable and then followed by the class effects.

Listing 4.1: Pest parameter file coding

```
comment ex01
  some meat quality data
relationship
  infile ='data/scr.msee.ped'
  outfile='data/ex01.ped' [text]
  rel_for animal
  input
    animal 1,4
    m_p    5,4
    f_p    9,4
data
 infile ='data/scr.msee.dat'
 outfile='data/ex01.dat' [text]
 input
    age              0    96 3 0
    sdate          999    82 3
    Rasse          999    92 1
    sex              9    93 1
    herd           999    94 2
    litter        2222    99 6
    animal        3333     1 4
    sire          2100     5 4
    dam           2100     9 4
    backfat          0  18 2 1
    loin             0  20 3 1
    Drip             0  33 3 1
    Flfa             0  36 3 1
    ph1              0  39 4 2
    redfibr          0  43 4 1
    feed             0  71 3 2
model
  backfat     = age+sdate+sex+herd+litter+animal
  loin        = age+sdate+sex+herd+litter+animal
  Drip        = age+sdate+sex+herd+litter+animal
  redfibr     = age+sdate+sex+herd+litter+animal
  feed        = age+sdate+sex+herd+litter+animal
VE
  1 .1 .1 .1 .1
  .1 1 .1 .1 .1
  .1 .1 1 .1 .1
  .1 .1 .1 1 .1
  .1 .1 .1 .1 1
VG
  vg_for animal
  1 .1 .1 .1 .1
  .1 1 .1 .1 .1
  .1 .1 1 .1 .1
  .1 .1 .1 1 .1
  .1 .1 .1 .1 1
solver
 iod sdate,sex,herd,litter    [max_iter = 3
 iod_gs animal                [max_iter = 3
transformation
   treated_as_missing
    backfat       none     99     none
    loin          none     9.99   none
    Drip          none     99.9   none
    redfibr       none     999.9  none
    feed          none     9.99   none
```

This coded data file can be used for many different models in VCE, ranging from single trait to 5 trait models and those that contain any of the factors/effects that were used in the initial PEST run used for coding.

4.2 One Random Effect

This is the simplest model in variance component estimation. Let us assume that in our dataset we know that belonging to a certain litter cause variation among litters and that this is supposed to be random. The dataset contains animal records from a growing and slaughter experiment in swine.

4.2.1 One random effect, single trait

The parameter file is given in listing 4.3. For clarity, the section names are capitalized. A number of things need to be noted:

COMMENT section use the COMMENT section for just that: comments giving some details on the run.

DATA section

- ▷ in the DATA section the input file needs to be specified. The name of the file given between the hyphens must lead to the actual file from the position where VCE is started. Remember, that VCE is started from the command line (old school, know what you are doing :)). For the given parameter file to work, i.e. to find the file 'data/scr-msee.dat' you need to go to the directory that contains this file (by the 'cd' command which is the same for all real operating systems like Unix, Linux, MacOS, plus even Windows. Then, also the output files will be written by the current run into this directory.
- ▷ the 'depend' keyword specifies the continuous variables that can either be used as traits or right hand side or as covariables, then part of the left hand side. The order needs to be the one from the data file. Also, all columns must be listed here, as the file is read from left to right reading as many floating point variables as given in the 'depend' statement.
- ▷ reading of the record is continued with the number of variables listed in the 'indep' line. Also here, no columns should be skipped.
- ▷ if you want to skip some columns, a 'FORMAT' can be specified in the 'datfile' line.
- ▷ THE FIRST LINES READ FROM THE FILE ARE PRINTED BY VCE, CHECK IF YOU GOT THE CORRECT COLUMNS! There is nothing more embarrassing then reporting a heritability for daily gain but due to a mistake you read the backfat column.

MODEL section

- ▷ our trait of interest is BACKFAT, the model is dead simple: and intercept is fitted through INT (which is the only element in the model equation that does not have to be specified in the DATA section). This is followed by the the LITTER effect. It is worth remembering, that each model must have a fixed effect, which in normal life is always the case. Should you not have a real one, use INT.
- ▷ effects listed in the COVARIANCE section become random, i.e. for these effects covariance components get estimated. Again, here it is the LITTER component.
- ▷ in the SYSTEM section the expected number of nonzero elements together with a total needs to be specified. If you specify a wildly large number, your computer may not be big enough for VCE to allocate this amount of memory. It is good policy to start with a few hundred thousands and then increase it, when VCE stops telling you to increase it. While NON_ZERO has to reflect the number of non zero coefficients of the mixed model equations is TOTAL required to for the additional storage requirements over and above that for the non zero elements which is used for storing the Cholesky factors and the inverse elements of the mixed model equations and other auxiliary data. As a rule of thumb you may want to start TOTAL with a figure that is about 2 – 4 times the value of NON_ZERO. If VCE does not have enough space it will reallocate more and try again, repeating the process until enough space is available. However, it may be faster to abort this process (i.e. VCE), increase TOTAL in the parameter file and start VCE again.

Listing 4.2: Input Data

```
1  BACKFAT     LOIN      DRIP      REDFIBR     FEED      AGE       SDATE  SEX  HERD  LITTER  ANIMAL
2  0.320E+01 0.388E+02 0.810E+01 0.525E+02 0.349E+01 0.182E+03    10     2    11     2       1       1
3  0.170E+01 0.355E+02 0.220E+01 0.652E+02 0.364E+01 0.185E+03    10     1     2     4       2       1
4  0.330E+01 0.358E+02 0.350E+01 0.602E+02 0.340E+01 0.182E+03    10     1     1     5       3       1
```

Listing 4.3: One Random Effect

```
1  COMMENT EXAMPLE_1
2      input file:
3      BACKFAT LOIN DRIP REDFIBR FEED AGE SDATE SEX HERD LITTER ANIMAL
4      Simple one trait random model
5  DATA
6      datfile    = 'ex01.dat'
7      dependant = BACKFAT LOIN DRIP REDFIBR FEED
8      independ   = AGE SDATE SEX HERD LITTER ANIMAL;
9  MODEL
10     BACKFAT = int litter;
11 COVARIANCE     litter;
12 SYSTEM
13     non_zero=200000
14     total    =300000
15 end
```

The results for a single trait one random effect model are actually only two numbers: the residual variance and the variance attributed to the random component in the model, as shown in line 12 and 18 in listing 4.4.

Listing 4.4: One Random Effect, output in file ex01.vce.list

```
1  17.12.2007  13:19:01                                                    page 4
2  ********************************************************************************
3  *                E S T I M A T E S   I N F O R M A T I O N                     *
4  ********************************************************************************
5
6  Mon Dec 17 13:19:01 2007          ex01.vce            CPU time used:     0:00:00
7
8  AG Log likelihood :    1979.5324  status :  1   at iteration:      19 /    19
9
10 ------------------------------ Matrices: NATURAL --------------------------------
11  Type: R Level:   1  litter                        No.:       1112 Pattern: T
12    0.033955
13
14  Type: E Level:   1  residual                      No.:       1997 Pattern: T
15     0.20171
16
17 ---------------------------- Matrices: Phenotypic --------------------------------
18     0.23567
19
20 ------------------------------ Matrices: RATIOS --------------------------------
21  Type: R Level:   1  litter
22     0.14408
23
24  Type: E Level:   1  residual
25     0.85592
26
27 ---------------------- Matrices: STD_ERR of components -------------------------
28  Type: R Level:   1  litter
29    0.7334E-02
30
31  Type: E Level:   1  residual
32    0.8871E-02
33
34 ------------------------ Matrices: STD_ERR of ratios --------------------------
35  Type: R Level:   1  litter
36    0.030117
37
38  Type: E Level:   1  residual
39    0.030117
```

It is important to check for the status line as printed in the run log (Listing on the previous page, line 8): the status itself should always be 1, then safe convergence is guaranteed. Sometime, with bad data structure relative to the model you may get 2 or even 3. This does not necessarily mean that the estimates are useless. For further information see the discussion status.

4.2.2 One random effect, multiple traits

A two trait model is easily implemented as shown in listing 4.5, line 10 by simply adding the desired trait - here it is DRIP - to the list.

Listing 4.5: One Random Effect

```
COMMENT EXAMPLE_2
    input file:
    BACKFAT LOIN DRIP REDFIBR FEED AGE SDATE SEX HERD LITTER ANIMAL
    Simple one trait random model
DATA
    datfile    = 'ex01.dat'
    dependant = BACKFAT LOIN DRIP REDFIBR FEED
    independ   = AGE SDATE SEX HERD LITTER ANIMAL;
MODEL
  BACKFAT DRIP = int litter;
COVARIANCE     litter;
SYSTEM
    non_zero=200000
    total    =300000
end
```

With two traits we get one residual covariance matrix of order 2 by 2 and one for the random litter effect. These are in the Listing 4.6. As can be seen, the estimates of the variance for the trait BACKFAT are slightly different now. The estimate for BACKFAT for the LITTER component changed from .033955 to .03401 for the bivariate model and went from .20167 to .20171 for the residual variance.

Experience tells us, that massive changes do not happen when going to higher dimensional models. Should this be the case, do look again at the status of the runs involved. Also, when doing more traits, it may be useful to quickly do a number of univariate runs to get an idea about the heritabilities. In higher dimensional models these will be different but not by a large margin.

Listing 4.6: One Random Effect, output in file ex02.vce.list

```
1  ****************************************************************************
2  *                  E S T I M A T E S   I N F O R M A T I O N                 *
3  ****************************************************************************
4
5  Tue Dec 18 11:18:06 2007          ex02.vce            CPU time used:     0:00:00
6
7  AG Log likelihood :    3920.5394  status :  1   at iteration:      27 /    27
8
9
10 ------------------------------ Matrices: NATURAL --------------------------------
11  Type: R Level:   1  litter                          No.:       1112 Pattern: T T
12     0.03401      -0.01982
13                   2.16280
14
15  Type: E Level:   1  residual                        No.:       1997 Pattern: T T
16     0.20167       0.00755
17                   6.39363
18
19 --------------------------- Matrices: Phenotypic ---------------------------------
20     0.23567      -0.01228
21                   8.55643
22
23 ------------------------------ Matrices: RATIOS --------------------------------
24  Type: R Level:   1  litter
25     0.14430      -0.07309
26                   0.25277
27
28  Type: E Level:   1  residual
29     0.85570       0.00665
30                   0.74723
31
32 ---------------------- Matrices: STD_ERR of components -------------------------
33  Type: R Level:   1  litter
34     0.00645       0.03190
35                   0.27586
36
37  Type: E Level:   1  residual
38     0.00823       0.03409
39                   0.28400
40
41 ------------------------- Matrices: STD_ERR of ratios --------------------------
42  Type: R Level:   1  litter
43     0.02664       0.11796
44                   0.02930
45
46  Type: E Level:   1  residual
47    0.026641      0.030014
48                  0.029302
```

In multiple trait model so called “missing values” are an important issue. This means that for one animal not both or all traits may have measurement. Here it is important that the user checks if missing values are correctly identified. The default is taken from the PEST coding. Missing values are indicated by 111111000 (which in exponential notation is .11111E+08) as given in Listing 4.7.

Listing 4.7: Missing Data indicated by .11111E+08

```
1  0.11111E+08 0.14500E+03 0.11111E+08 0.11111E+08    1562      1
2  0.11111E+08 0.17000E+03 0.11111E+08 0.11111E+08     433      2
3  0.11111E+08 0.20000E+03 0.21500E+03 0.30500E+03    1741      2
4  0.11111E+08 0.14800E+03 0.17500E+03 0.20500E+03    1741      2
5  0.11111E+08 0.16500E+03 0.18500E+03 0.28500E+03    1741      2
6  0.11111E+08 0.13000E+03 0.11111E+08 0.11111E+08    1741      2
```

4.3 More than one random effect and adding a fixed.

This actually presents nothing much new. Just add another entry from the INDEP list in the MODEL and also under the covariances section (Listing on this page). Then VCE will treat this as a random effect and try to estimate the corresponding covariances.

However, let us add a word of warning. There has to be a good reason for an effect to be considered random. Sex as an example is certainly not a suitable candidate: it has only two or three classes and we cannot assume that their occurrence can be considered samples from an underlying population. This is different with head/year/season (HYS) classes (which are often also considered fixed). Often, there is a large number of HYS in a data set. Further, it can be argued that they can be considered samples form an underlying "population" of HYS. In short: know what you are doing!

Listing 4.8: One Random Effect

```
1  COMMENT EXAMPLE_3
2      input file:
3      BACKFAT LOIN DRIP REDFIBR FEED AGE SDATE SEX HERD LITTER ANIMAL
4      Two trait random model with fixed effect
5  DATA
6      datfile    = 'ex01.dat'
7      dependant = BACKFAT LOIN DRIP REDFIBR FEED
8      independ   = AGE SDATE SEX HERD LITTER ANIMAL;
9  MODEL
10     BACKFAT DRIP = int age herd litter;
11 COVARIANCE
12     litter;
13 SYSTEM
14     non_zero=200000
15     total    =300000
16 end
```

Fixed effects are likewise simple to add: they also have to occur in the INDEP list of the DATA section (except for INT, the intercept). Also, the user must have a good reason for adding the effect. With fixed effects you need to know how many observations you have in the cells: estimating a fixed effect on the basis of 3 observations is nonsense. However, you will never find this out, unless you preprocess the data prior to the VCE run and generate these statistics. Again: know what you are doing!

4.4 The Litter Effect

In pig breeding the common litter effect is usually considered an important source of variation. It is the non genetic effect that being part of a common litter has on the animal's performance. This is relevant for the coding the data. How can we identify a litter in the total dataset? Clearly, the dam identification is required. If this is used on its own, then all offspring from one dam will get the same ID, i.e. also those from different litter of the same dam. Thus, a code for the litter effect can be defined by a concatenation dam ID and its litter number. For instance, for dam 4711 with three litters the offspring could have the litter codes: 4711-1, 4711-2, 4711-3. Alternatively, a concatenation of dam ID and the birth date of its offspring should also work. That may be 4711-2006-01-13, 4711-2006-06-01, 4711-2007-03. After coding either of them we shall get 1,2,3.

Sometimes the concatenation of ID and birth date may be too long for PEST to allow coding which has a maximum of 15 characters. Then the birth year and month will be sufficient: no sow has two litters in the same month.

Thus, for coding the litter effect the sow ID and the birth date should be placed next to each other in the data file. Then the litter effect can be coded by including the sow ID and birth year and month in one code.

4.5 The Sire Model

BLUP started with sire models. The model is simple in that the effect to be placed in the model is the sire of the recorded animals. This results in a small system of normal equations as we may have 100000 animals but originating from only 500 sires. Accordingly, the order of the mixed model equations which has to be set up and solved in each iteration is only of dimension 500 (ignoring any other effects). As a result, the estimation of the sire covariance structure will be much faster. To obtain genetic parameters, the sire variance component has to be interpreted in additive genetic terms. As the sire component catches 1/4 of the additive genetic variance, the results from VCE for the sire component needs to be multiplied by 4 to obtain the additive genetic variance.

4.6 Univariate Animal Model

Animal models have become the work horses in genetic evaluation and selection in animal populations following the sire models. The animal component - in conjunction with the additive genetic relationship matrix which is derived from the pedigree data estimates the additive genetic variance. As far as the model section in VCE goes, the animal effect is determined by the coded animal IDs. However, animal models need the numerator relationship matrix to make sense. Thus, we need to consider the inclusion of the pedigree of the animals analyzed.

4.6.1 Data preparation

Again, data need to be coded for all inputs to VCE. Apart from the data file now also the pedigree file needs to be coded. This means, that an animal with ID DE21009_HJ0090 must have the same integer number in both the pedigree and the data file. Again, coding can be conveniently done with PEST or any other way that you can master.

Here, we are giving a PEST example in Listing 4.9. The file 'midped.cod' and 'mid800.cod' will contain the coded data as shown in Listing 4.10. The first block gives the first few lines from the coded pedigree file. It starts with a header line "animal ancestor_1 ancestor_2 type b_order" which stands for Animal, Sire, Dam in this example, followed by two columns that are only relevant to PEST. Type is 1 if all two ancestors are known, 2 if the sire is known but not the dam, 3 if the dam is known but not the sire, while 4 means that no parent is known. Also, it can be noticed, that each animal, including all ancestors must have a record in the pedigree file, the base parents then have 0 for the unknown parents (and type = 4).

Listing 4.9: Coding of pedigree data with PEST

```
1  RELATIONSHIP
2  rel_for animal
3  infile = '../data/mid.ped'
4  outfile = 'midped.cod' [text]
5  input
6  animal 1 6
7  m_p    8 6
8  f_p   15 6
9  ..
10 data
11     infile = '../data/mid800.dat'
12     outfile= 'mid800.cod' [text]
13      input
14        animal   3000
15        testdate  200
16        line       10
17        herd       10
18        sex         2
19        weight      0
20        bf          0
21        dg_test     0
22        fce         0
23        dg_farm     0
24 model
25        bf       =  int weight(line)      line           animal
26        dg_test  =  int                   line   testdate animal
27        fce      =  int                   line   testdate animal
28        dg_farm  =  int   herd   sex      line   testdate animal
```

PEST codes animals always first with the base animals (i.e. those that do not have further ancestors) last. The same holds for coding genetic groups. Assume that you have 1000 animals and in all 10 genetic groups. Then, PEST would code the animals from 1 – 1000 and from 1001 – 1010 the genetic groups.

Listing 4.10: Coded pedigree and data

```
1  midped.cod:
2  animal     ancestor_1    ancestor_2    type     b_order
3        1680         486          2564          1          0
4        1681        2567          2571          1          0
5        1682         273          2571          1          0
6        1683         273          2571          1          0
7        1684         273          2571          1          0
8        1685         479          2625          1          0
9        1686        1902           643          1          0
10       1687        1902           643          1          0
11 ...
12 mid800.dat:
13   BF            DG_TEST       FCE           DG_FARM       WEIGHT      INT  ANIMAL  TESTDATE  LINE  HERD  SEX
14  0.11000E+02 0.11111E+08 0.11111E+08 0.61300E+03-0.50000E+01  1       1        18     1     2     2     1
15  0.10700E+02 0.11111E+08 0.11111E+08 0.43100E+03-0.15000E+02  1       2        19     1     2     2     1
16  0.13700E+02 0.11111E+08 0.11111E+08 0.65800E+03 0.20000E+01  1      10        20     2     2     1     1
17  0.95000E+01 0.11111E+08 0.11111E+08 0.56100E+03-0.13000E+02  1      11        20     2     2     1     1
18  0.13300E+02 0.11111E+08 0.11111E+08 0.58100E+03-0.10000E+02  1      13        20     2     2     2     1
19  0.14700E+02 0.11020E+04 0.23300E+01 0.11111E+08-0.40000E+01  1      12        19     2     2     1     2
20  0.11111E+08 0.11111E+08 0.23400E+01 0.11111E+08 0.00000E+00  1    2113         1     5     1     1     3
```

When running genetic groups model, notice that PEST does not allow unknown parents: each animal has to be assigned to either a a sire or a dam or a genetic group. The same applies to the coding for VCE. Thus, with genetic groups all animals must either have a parent or a genetic group as the final record in the pedigree recursion. If this is done with PEST and its GROUPS keyword in the RELATIONSHIP section, then all base animals will in fact be treated as genetic groups. If on the other hand the groups keyword is dropped in the pest run, then a longer pedigree file will be created which also has for all base animals one record each, containing the animal ID and a 0 for unknown sire and dam.

Thus, for coding outside of PEST (actually, the same applies to using PEST for the coding purpose), you have two situations.

1. no genetic groups: create a pedigree file that has for each animal ID a record independent if that animal has a known parent or not. Unknown parents are identified through '0'
2. with genetic groups: use actual parents for each animal where this is known. Assign those animal for which the parents are unknown to a genetic group. If you run this through PEST use the keyword GROUPS in the RELATIONSHIP section. In a genetic groups model no unknown parents must be used in the pedigree file.

4.7 The Sire Model with Relationship

In a sire model without additive genetic relationships it is assumed that all the sires are unrelated. Often this is not the case. Then the pedigree of the sires should be used in the model. The definition in the VCE parameter file is identical to that of an animal model as described above. However, there is one issue worth considering. All animals in the data file must have a representation in the pedigree file. This is a requirement of VCE. Should this condition not be met, VCE will stop. However, the pedigree file itself may contain many more entries also from animals that are not at all related to those with records in the data file. While this will not produce wrong results, it will unduly increase the size of the set of mixed model equations

and therefore increase computing requirements. Thus, the pedigree file should be constructed such, that it contains only records that are tied to the performance records in the data file. The procedure to follow should be:

1. for each record in the the data file
2. identify the animals parent
3. for each parent identify its own parents - perform this step recursively until no more parents are known
4. all animals touched in step 3 will make up the pedigree file

There are a number of way to perform this checking. There is a Fortran90 program by Eildert Groeneveld (genped.f90) which will do just that. It can also be used to perform some other filtering of the source data and will create a consistent data/pedigree set ready for coding. Then there are other ways to do this. Some use SAS while similar Perl procedures are part of the database framework APIIS(CreatePediStack.pl).

One further concern is the depth of the pedigree. Imagine a data set of size 5000 animals with records. In an animal model the size of the mixed model equations will be minimally of order 5000 (other factors need to be added). If we now add the pedigree derived from the performance recorded animals, this number will increase. With 2 generations of ancestors it may go up to 9000 (or whatever). If the pedigrees are very long, i.e. the pedigree data go back many generations the system size may increase very much. So at this point it may be worth putting up an upper limit on the number of generations to include.

4.8 Multivariate Animal Model

Above the term “work horse” has been used in the context of Animal Model. Actually, its multiple trait rendition is the actual work horse in genetic evaluation in animal breeding. Accordingly, variance component estimation is of prime importance. “real life” model are always multivariate often with traits from different environments, implying that usually not not all traits are measured on all animals. The following gives an example from pig breeding which is reasonably close to real life.

Listing 4.11: Multi Trait Animal Model

```
DATA
c ....................  field traits
  datfile     = '../data/field.dat'
  format      = '(2f12.0,36x,f12.0,12x,f8.0,f8.0,f8.0,4f8.0)'
  dep         = bfft adgft
  indep       = wofft animal sex litter hysft ;

c ....................  station traits
  datfile     = '../data/station.dat'
  format      = '(24x,3f12.0,12x,f12.0,3f8.0,8x,f8.0)'
  dep         = adgst vc bfst
  indep       = hms animal sex litter stys ;

c ....................  pedigree file
  pedfile     = '../data/np37.ped' format= '(4I10)' link=animal;

MODEL
  bfft  = reg(wofft)          animal sex litter hysft;
  adgft =                     animal sex litter hysft;
  adgst =                     animal sex litter       stys;
  vc    =                     animal sex litter       stys;
  bfst  =           reg(hms)  animal sex litter       stys;

COVARIANCE
  animal ; litter ; hysft  ; stys   ;
  residual(datfile);
SYSTEM
  non_zero=963000
  total= 11000000
end
```

As can be seen, there are two groups of traits: the first is measured in the field on young boars and gilts, here we have ultrasonic backfat and daily gain. On a second set of animals we have station test traits, also daily gain but here through the testing period, then valuable cuts after slaughter of the animals followed by another backfat measurement. Notice that these two sets of traits are mutually exclusive: an animal is either in the field or at a test station. Further, each set has its own set of factors influencing the traits, but both are of course tied together through the animal identification with a joint pedigree, on the basis of which additive genetic covariances are estimated.

The setup in Listing 4.11 read data from two files: one for the station test data and the other for the field test, each with its own format. Once we get to the model definition all traits are treated independent of their source file. However, for each trait the appropriate model is specified. Here, the first two traits are the field test traits while the last originated at the test station. The COVARIANCE now determines the covariance matrices to be estimated: the animal component estimates the additive genetic variance, then we have a common litter component, also for each trait in the model, and then one herd x year x season covariance matrix for the field test traits and a station x year x season for the other three station test traits. Finally, the residuals are estimated within each data file, with the dimension 2 x 2 for the field test and 3 x 3 for the station test traits.

Listing 4.12: Estimates for Multi Trait Animal Model

```
1
2  ****************************************************************************
3  *                  E S T I M A T E S   I N F O R M A T I O N                 *
4  ****************************************************************************
5  ------------------------------ Matrices: NATURAL --------------------------------
6   Type: A Level:   1  animal                        No.:      10410 Pattern: T T T T T
7         0.01         0.18        -0.81        -0.11         0.01
8                    410.04         527.51        -1.56         0.08
9                                  2021.97         7.67         0.59
10                                                 2.93        -0.15
11                                                              0.03
12
13  Type: R Level:   1  litter                        No.:       4659 Pattern: T T T T T
14        0.00         0.29         1.33         0.02         0.00
15                   828.51      -286.37         4.28         0.56
16                               3222.37         8.18        -0.65
17                                                 0.08         0.01
18                                                              0.00
19
20  Type: R Level:   1  hysft                         No.:        661 Pattern: T T
21       0.058         2.193
22                   910.369
23
24  Type: R Level:   1  stys                          No.:        117 Pattern: T T T
25     1306.55        -3.67        -0.37
26                     0.98        -0.04
27                                  0.02
28
29  Type: E Level:   1  residual/field.dat             No.:
    4563 Pattern: T T F F F
30        0.015        0.693          ---          ---          ---
31                   513.194          ---          ---          ---
32                                    ---          ---          ---
33                                                 ---          ---
34                                                              ---
35
36  Type: E Level:   1  residual/station.dat           No.:
    3231 Pattern: F F T T T
37          ---          ---          ---          ---          ---
38                       ---          ---          ---          ---
39                                 4217.91       -38.99         4.29
40                                                 2.81        -0.16
41                                                              0.06
```

The resulting estimate are given in Listing 4.12. It is these components that get estimated by VCE, all the following matrices like phenotypic variances and ratios are computed on the basis of these natural variance covariance components. As can be seen, for each random component we have one matrix, starting with the animal component which in genetic terms can be interpreted as additive genetic covariance matrix. The “No.” gives the number of levels involved in each of the matrices. Thus, for the additive genetic component we have 10410 animals with 4659 entries for the litter matrix. Notice that for the factor “stys” we have only 117 levels, which is not really a lot.

4.9 Models with maternal additive genetic effect

Maternal effects models are a special case of the animal models. The statistical model behind it assumes that there is an additive genetic component of the individual for its records or measurement and likewise another effect of mother on that same trait. Now, the mother and the individual are not genetically independent but rather related through the numerator relationship matrix. All of this results for one trait in having to estimate a 2 x 2 matrix, containing the additive genetic effect of the animal (also called direct) on the trait plus the additive genetic effect of the dam of the animal on that same trait plus a covariance between the two.

A model file for maternal effect models is described in 4.13.

Listing 4.13: Model with maternal effect

```
1  Comment np02
2    here we have a univariate animal model with maternal and additive
3    direct genetic effects. This is done by specifying to link addresses
4    (dam animal).
5  Data
6    datfile = '../test/data/a1testm.dat' format='(2f12.0,5f8.0,8x, f8.0)'
7        dep    = birth
8        indep  = damage  sex hys dam animal;
9    pedfile = '../test/data/a12.ped' format='(3I10)' link = animal dam;
10 Model
11   birth = p1(damage)  sex  hys animal dam;
12 Covariance
13   animal;
14   hys;
15 System
16   non_zero= 200000
17   total   = 300000
18 End
```

Beware, there is a bug in the parsing of the parameter file: in maternal effects models you MUST always have ANIMAL DAM/ last in the model, else VCE may stop.

Listing 4.14: Model with maternal effect - results

```
1  ***************************************************************************
2  *                  E S T I M A T E S   I N F O R M A T I O N              *
3  ***************************************************************************
4
5  Wed Oct 22 11:11:19 2008                              CPU time used:      0:00:00
6
7  AG Log likelihood :     1845.8601   status :  1   at iteration:      27 /     27
8
9
10 ------------------------------ Matrices: NATURAL --------------------------------
11  Type: R Level:    1  hys                              No.:          39 Pattern: T
12     1.69934
13
14  Type: A Level:    1  animal|dam                       No.:        1428 Pattern: T T
15     8.40442      -0.58809
16                   0.46936
17
18  Type: E Level:    1  residual                         No.:         543 Pattern: T
19     4.22685
```

The coding as a preparation for VCE is different with maternal effects models. Remember, that there are the two effects animal and dam for any one record sitting in two columns in the data file. Ordinary coding would sort all entries in the animal column and sort them from 1 to n. Then, and independently, the dam column would also be sorted from 1 to total number of dams. If we have a record for animal CLARA and CLARA has a few years later an offspring with a record, then CLARA will show up once in the animal column and once in the dam column of its offspring. Standard i.e. independent coding of the two columns will most likely result in different codes for CLARA as an animal and CLARA as a dam. This is wrong and needs to be prevented. Thus, if you use your own coding software, you need to ensure that you put both columns into one “pot” and then sort and code. In PEST you can use a coding feature from sire/dam models. The corresponding parameter file is given in Listing 4.15. As can be seen from line 2 in Listing 4.15 both effects (here sire and dam) originating in two columns are coded jointly. This is exactly what we need to do for maternal effects models.

Listing 4.15: Coding for maternal effect

```
RELATIONSHIP
      rel_for sire dam
      infile = '../data/de_mpa.ped'
      input
            sire
            m_p
            f_p
DATA
      INFILE  =   '../data/de_mpa.dat'
      INPUT  [  VAR_NAME     MAXLEVEL START_COLUMN VAR_LENGTH DECIMAL ]
                  sl_wt          0
                  month        140
                  herds        150
                  sire        1200
                  dam         1200
                  litter       400
                  daily_gain     0
MODEL

     daily_gain  =  sire dam  herds month litter
```

The standard procedure is to have the same traits in both the direct and maternal additive genetic effect. This restriction does not apply to VCE: the user may specify different traits in maternal and direct (and, hopefully, be able to justify this).

4.10 Example with fixed regression nested

Regressions can easily be nested in effects as specified in Algorithm 4.1. The linear regression on wt100 for bf100 is nested with in breeds. As a result one regression curve will be fitted to each breed.

4.11 Setting up Random Regression Models

In this section we shall deal with the class of Random Regression Model. In particular the interpretation of the parameter estimates are not as straight forward as in the other models.

4.11.1 Body mass in Golden Hamsters

Introduction into the theory demonstrated for this example In this paragraph we shall deal with theoretical aspects of Random Regression Model explained in the context of a small dataset which has been generated from a small lab population of Golden Hamsters (Krause, 2008). There are body weight records available on 314 animals from 57 litters taken at 7 different dates. While the amount of data is certainly not sufficient for proper genetic evaluation, it serves well for demonstration purposes.

Algorithm 4.1 Effect breed as main effect and nesting effect for linear regression

```
COMMENT Boars Slovenia - Podgrad
DATA
   datfile   = '../test/data/b100po.dat'
      format = '(7f12.5,5f8.0)'
      dep    = tp_r30 tp_3060 tp_60100 fce_3060 fce_60100 bf_100
      indep  =  wt100 seas breed litter animal;
   pedfile   = '../test/data/b100po.ped'
      format = '(4i10)'
      link   = animal;
COVARIANCE
  animal;
  litter;
MODEL
   bf_100              = seas+litter+animal+[1, p1(wt100)]breed;
   tp_r30 tp_3060 tp_60100 fce_3060 = seas+ litter+animal+breed;
SYSTEM
   non_zero   = 1000000
   total      = 3000000
END
```

The lab animals were produced in seven consecutive batches (matings) over a period of 2 years, which we shall refer to as "batches". Body weight was recorded weekly from day 28 through 70 in the live of the hamsters. If the average weight is plotted over weeks and linear or quadratic growth can be expected. Likely fixed effects are the batches, the litter size as a continuous covariable and a sex effect. Furthermore, we need to consider the fact that all offspring from one dam were raised in the same litter environment, and that we have repeated measurements for each animal. As a result, the random component Dam and Animal have to be part of the model.

Let $y_{ij}(t)$ be the performance record of animal j with dam i at time t. Further, we assume a linear growth during the measurement phase. If all random components are taken relative to the population mean we get the following model equation:

$$y_{ij}(t) = (\beta_0 + d_{0i} + a_{0ij})x_0 + (\beta_1 + d_{1i} + a_{1it})x_1(t) + e_{ij}(t) \tag{4.1}$$

with $x_0 = 1$ and $x_1(t) = t$. Further, β_0 and β_1 are fixed and population specific, while d_{0i} and d_{1i} are random maternal and a_{0ij} and a_{1ij} random animal specific regression coefficients (each with an expected value of zero). If the model (4.1) only contains the random intercepts $d_{0i}$and a_{0ij}, then the litter common environmental variance and the variance of the animal effects need to be considered age independent. The random slopes $d_{1i}$and a_{1ij}result in the corresponding variance components of the dam and animal effects showing an age dependence.

(It should be noted that the dam effects will contain genetic components apart from the common environmental effects while the animal effects are not solely made up of additive genetic effects if no pedigree information is used.)

Nesting the fixed regression coefficients within sex the VCE representation of equation 4.1 becomes:

$$y = [1, p1(t)]sex + [1, p1(t)]dam + [1, p1(t)]anim \tag{4.2}$$

Model equation (4.1) is an appropriate description of repeated body measurements if growth is only linear. Usually, higher order polynomials are used to describe this process, with a minimum of three. In this case, four covariables would be required: $x_0 = 1; x_1 = t; x_2 = t^2$and $x_4 = t^3$.

Let $x(t) = (x_0(t), x_1(t), \cdots, x_n(t))'$ be a vector of $(n + 1)$ covariables (as an example: $x_r = t^r$ or in a standardized version: $x_r = (t/t_{max})^r$ with $r = 0, \cdots, n$). With pedigree information animal specific effects can be partitioned into additive genetic and permanent environmental variance. Let $\beta = (\beta_0, \cdots, \beta_n)'$ be the vector of fixed regression coefficients and $d_i = (d_{0i}, \cdots, d_{ni})'$ the vector of regression coefficients of mother i. Further, let $a_{ij} = (a_{0ij}, \cdots, a_{nij})'$ and $p_{ij} = (p_{0ij}, \cdots, p_{nij})'$ den be the vector of additive genetic and permanent environmental regression coefficients, resp., of animal j within dam i. The covariance matrices of these vectors would then be:

$$Var(d_i) = K_d;\ Var(a_{ij}) = K_a \text{ und } Var(p_{ij}) = K_p \tag{4.3}$$

Only, the additive genetic regression coefficients of different animals are considered correlated. The dam component is not getting further decomposed into additive genetic maternal and common environmental effects. Look at coefficients a, d and p across animals the following applies:

$$Var(d) = I_d \otimes K_d;\ Var(a) = A \otimes K_a \text{ und } Var(p) = I_p \oplus K_p \tag{4.4}$$

In formula (4.4) $I_d$and I_a are identity matrices while A is the numerator relationship matrix.

Using the above expressions the model (4.1) for repeated measurements can be expanded in the following way:

$$y_{ij}(t) = x'(t)\beta + x'(t)d_i + x'(t)a_{ij} + x'(t)p_{ij} + e_{ij}(t) \tag{4.5}$$

Following our basic model in quantitative genetics for partitioning the variance of an observation into an additive genetic and environmental component $y_{ij} = g_{ij} + u_{ij}$ we get:

$$\begin{aligned} g_{ij}(t) &= x'(t)a_{ij} \\ u_{ij}(t) &= x'(t)d_i + x'(t)p_{ij} + e_{ij}(t) \end{aligned} \tag{4.6}$$

Thus, the additive genetic variance can be computed as:

$$\begin{aligned} \sigma^2_g(t) &= Var(x'(t)a_{ij}) \\ &= x'(t)Var(a_{ij})x(t) \\ &= x'(t)K_a x(t) \\ &= \sum x_i(t) \cdot x_j(t) \cdot K_a\{i, j\} \end{aligned} \tag{4.7}$$

The Variance function at time t and the covariance function at t_1 and t_2 is then:

$$\begin{aligned} Var(y(t)) &= x'(t)K_d x(t) + x'(t)K_a x(t) + x'(t)K_p x(t) + \sigma^2_e \\ cov(y(t_1), y(t_2)) &= x'(t_1)K_d x(t_2) + x'(t_1)K_a x(t_2) + x'(t_1)K_p x(t_2) \end{aligned} \tag{4.8}$$

Based on the vectors of covariables $x(t)$ and matrices K_d, K_a and K_p heritability-curves and genetic correlations can be easily determined. For instance:

$$h^2(t) = \frac{\sigma_g^2(t)}{\sigma_y^2(t)} \quad \text{mit} \quad \sigma_y^2(t) = Var(y(t)) \tag{4.9}$$

With higher order polynomials orthogonality improves convergence. For instance the Legendre polynomials are orthogonal over the interval $[-1, 1]$. As a consequence, the independent variable – let's say "age" as from the dataset – has to be transformed as:

$$t = \frac{2.0 \cdot (age - t_{min})}{(t_{max} - t_{min})} - 1.0; \tag{4.10}$$

The Legendre polynomials as used in VCE have the form (up to order 4):

$$\begin{aligned} x_0(t) &= \sqrt{\tfrac{1}{2}} \\ x_1(t) &= \sqrt{\tfrac{3}{2}} \cdot t \\ x_2(t) &= \sqrt{\tfrac{5}{2}} \cdot \tfrac{1}{2} \cdot (3 \cdot t^2 - 1) \\ x_3(t) &= \sqrt{\tfrac{7}{2}} \cdot \tfrac{1}{2} \cdot (5 \cdot t^3 - 3 \cdot t) \\ x_4(t) &= \sqrt{\tfrac{9}{2}} \cdot \tfrac{1}{8} \cdot (35 \cdot t^4 - 30 \cdot t^2 + 3) \end{aligned} \tag{4.11}$$

The necessary transformations are done by VCE. Then:

$$\begin{aligned} plg1(t) &= (x_0, x_1)' \\ plgn(t) &= (x_0, x_1, \cdots, x_n)' \end{aligned} \tag{4.12}$$

Thus, in the VCE parameter file model (4.1) expanded by a permanent environmental effect and using Legendre polynomial can be written as:

$$y = [plg1(t)]sex + [plg1(t)]dam + [plg1(t)]anim + [plg1(t)]perm \tag{4.13}$$

Coding of data Before input data can be used in VCE all fixed and random effects need to be recoded. This means, that all class effects need to start with a numerical 1 and code the levels in an ascending order with no "holes". This can be conveniently done by PEST. The corresponding PEST parameter file is given in 4.17.

Listing 4.16: Input Data (lab_dat.txt)

```
0090    w  2  W03/312  0090      41.0    8.0   28.0    28
0091    m  2  W03/222  0091      52.0    8.0   28.0    28
0092    w  2  W03/362  0092      57.0    9.0   28.0    28
0093    m  2  W03/212  0093      53.0    7.0   28.0    28
0096    m  2  W03/262  0096      51.0   11.0   28.0    28
0098    m  2  W03/362  0098      57.0    9.0   28.0    28
```

Listing 4.17: Pest parameter file coding (p_rrm01.job)

```
comment
 hamster data
relationship
 rel_for anim
 infile  = 'lab_ped.txt'
 outfile  = 'ped.cod' [text]
c .........'12345678'
 undefined ='0        '
 input
    anim      1,8
    m_p       9,8
    f_p      18,8
data
 infile  = 'lab_dat.txt'
 outfile = 'dat.cod' [text]
 INPUT   [  VAR_NAME     MAXLEVEL   START_COLUMN    VAR_LENGHT    DECIMAL]
              anim          1000          1             8
               sex            10          9             3
             serie            10         12             3
               dam            60         15             8
              perm          1000         23             8
                km             0         31             6
                wg             0         37             6
               age             0         43             6
              week            10         49             6
c.. give statistical model
model
 km = wg age sex serie week dam perm anim
transformation
 treated_as_missing
 km none 0.0 none
ve
 1.0
vg
 vg_for anim
 1.0
 vg_for dam
 1.0
 vg_for perm
 1.0
c.. do not need converged results:
solver
 max_iter=2
 ioc [stop=.01]
system_size
 non_zero=1000000
printout
  outfile ='p_rrm01.list'
```

Legendre polynomials and the computation of variance functions In the following we shall continue to use the model defined above 4.1. Further fixed effects are added to model (4.13). Growth is modeled with in sex and batches by a second order polynomial. The effect of litter size is included through a covariable (linear regression) nested within weeks. For convergence reasons only a zero order Legendre is fitted to the maternal component as describe in the VCE parameter file4.19.

The variance functions are computed according to formula (4.8) using the SAS IML matrix package. The simple program code is given in Listing 4.20.

Listing 4.18: Input Data (dat.cod)

```
1
2  KM            WG            AGE           ANIM   SEX   SERIE   DAM   PERM   WEEK
3  0.41000E+02  0.80000E+01  0.28000E+02      6      2      1      14     1      1
4  0.52000E+02  0.80000E+01  0.28000E+02      7      1      1       5     2      1
5  0.57000E+02  0.90000E+01  0.28000E+02      8      2      1      20     3      1
6  0.53000E+02  0.70000E+01  0.28000E+02      9      1      1       4     4      1
```

Listing 4.19: VCE parameter file (rrm01.job)

```
1  comment Labortiere Hamsterdaten
2
3  DATA
4     datfile='dat.cod' format='(3f12.0,6f8.0)'
5     dep= km
6     indep= wg age anim sex serie dam perm week;
7     pedfile='ped.cod' format='(4i10)' link=anim;
8
9  COVARIANCE
10    anim;
11    perm;
12    dam;
13 c   start_ascii = 'cov_start.txt';
14
15 MODEL
16 km = [plg2(age)]sex [plg2(age)]serie [1,p1(wg)]week
17       [plg1(age)]anim [plg1(age)]perm [plg0(age)]dam;
18 c      scalex non;
19 c      scaley non;
20
21 SYSTEM
22     non_zero = 15000
23     total = 1100000;
24
25 OUTPUT
26   debug = .true.
27   covfile='cov_rrm01.txt' format='(10f12.5)' form='full';
28   solutions = 'sol_rrm01.txt' format='(10f12.5)'
29 END
```

Listing 4.20: SAS-File to compute the variance function for RRM01

```
/* calculation of variance functions */
Proc iml;
start main;
filename out 'rrm03_var.txt';
file out;
dim=2;x=shape(0,dim,1);
create hamster var{age,t,vp,vd,va,ve,vy};
tmin=28;
tmax=70;
/*****************************/
/* Resultate aus VCE */
Kd={42.3793};
Ka={18.6858        6.9194,
     6.9194        4.7530};
Kp={59.0451       14.2472,
    14.2472        6.5623};
ve=6.60635;
/*****************************/
na=nrow(Ka);nd=nrow(Kd);np=nrow(Kp);
do age=tmin to tmax;
 t = 2.0*(age-tmin)/(tmax-tmin)-1.0;
 x[1] = sqrt(0.5);
 x[2] = sqrt(3/2)*t;
 va=0.0;vp=0.0;vd=0.0;
 do i=1 to na;
  do j=1 to na;
   va=va+Ka[i,j]*x[i]*x[j];
  end;
 end;
 do i=1 to np;
  do j=1 to np;
   vp=vp+Kp[i,j]*x[i]*x[j];
  end;
 end;
 do i=1 to nd;
  do j=1 to nd;
   vd=vd+Kd[i,j]*x[i]*x[j];
  end;
 end;
 vy=vd+va+vp+ve;
 put (age) 4.0 (va) 12.5 (vp) 12.5 (vd) 12.5 (ve) 12.5 (vy) 12.5;
 append;
end;
closefile out;
finish;
run;
proc gplot data=hamster;
 plot vd*age va*age vp*age ve*age vy*age / overlay;
run;
```

Listing 4.21: Results from VCE (p_rrm01.job)

```
1  ************************************************************************
2  *                E S T I M A T E S   I N F O R M A T I O N             *
3  ************************************************************************
4  Thu Aug 07 10:48:22 2008                              CPU time used:     0:00:26
5  AG Log likelihood :   -2958.5107  status : 1   at iteration:     90 /    90
6  ------------------------------ Matrices: NATURAL --------------------------------
7   Type: A Level:   1  anim                          No.:       479 Pattern: T T
8      18.6858        6.9194
9       6.9194        4.7530
10  Type: R Level:   1  perm                          No.:       314 Pattern: T T
11     59.0451       14.2472
12     14.2472        6.5623
13  Type: R Level:   1  dam                           No.:        57 Pattern: T
14     42.3793
15  Type: E Level:   1  residual                      No.:      2198 Pattern: T
16     6.60635
17 ---------------------------- Matrices: Phenotypic --------------------------------
18     180.365          ---
19         ---          ---
20 ------------------------------ Matrices: RATIOS ----------------------------------
21  Type: R Level:   1  dam
22     0.23496
23  Type: R Level:   1  perm
24     0.32736       0.72379
25     0.72379       0.03638
26  Type: A Level:   1  anim
27     0.10360       0.73422
28     0.73422       0.02635
29  Type: E Level:   1  residual
30    0.036628
31 ---------------------- Matrices: STD_ERR of components -------------------------
32  Type: A Level:   1  anim
33     35.4774        7.0991
34      7.0991        2.3134
35  Type: R Level:   1  perm
36     14.7511        3.4160
37      3.4160        1.2898
38  Type: R Level:   1  dam
39     13.2896
40  Type: E Level:   1  residual
41     0.22849
42 ------------------------ Matrices: STD_ERR of ratios ---------------------------
43  Type: R Level:   1  dam
44    0.075797
45  Type: R Level:   1  perm
46     0.10673       0.11529
47     0.11529       0.00726
48  Type: A Level:   1  anim
49     0.18827       0.70696
50     0.70696       0.01264
51  Type: E Level:   1  residual
52    0.4186E-02
```

Scaling and computation of standard errors The VCE run from jobfile RRM01 in Listing 4.19 implies scaling of traits and covariables. As VCE is a package for computing covariance

components, these are scaled back on output. If, however, the regression coefficients are to be plotted for each sex, those coefficients have to be on the original scale. This can be done by starting VCE again without scaling and using starting values that are close to the final results.

The standard errors produced by VCE are based on the observed Hessian matrix. This means, that the approximation is done through the 2nd derivatives of the likelihood with respect to the parameters. However, only the first derivatives are computed analytically, while the 2nd derivatives are approximated during the iteration by the difference quotient of the first derivatives.

As a result, the predefined starting values have to be substantially different from the optimal solution (under scaling). This can be done by increaing the values on the diagonal of the covariance matrices and reducing the offdiagonals or to use results from a VCE with less fixed factors. Uncomment the appropriate lines in the Model and Covariance section of Jobfile RRM01 (Listing 4.19) to get a run without scaling. The starting values from Listing 4.22 results in the output in Listing 4.23.

Remark 1: parameter estimates in tables 4.21 and 4.23 belong to different starting values. As indicated by the status 1 for both runs, convergence was reached and the two sets are practically identical. In contrast, the estimated standard errors are in part quite different. This is the result from different path ways that the two runs followed during iteration, a result that will be particularly pronounced in small samples. If the status is reported as 1, one can at least be sure that a local maximum has been reached. Theoretically, repeated runs with different starting values reaching the same optimum would ensure, that the optimum is indeed a global one. However, at this point we are not aware that ever two different sets of solutions were found with status 1. The local maximum reported by Kovac and Groeneveld [5] were based on the simplex algorithm which is much less efficient than the BFGS algorithm based on first derivatives as used in VCE.

Remark 2: Models with polynomials of different order cannot be compared on the basis of the likelihood ratio test (LRT). When computing the likelihood values for optimization VCE in the case of a RRM drops the term $rank(K_a) \cdot det(A)$ because it is not required for optimization and difficult to compute. The rank of matrix K_a equals the order of the polynomial plus 1, and A is the numerator relationship matrix. Accordingly, the difference between 2 likelihood values resulting from polynomials of order $n_1$and n_2 are also dependent on the expression $(n_1 - n_2) \cdot det(A)$. As a result, a LRT can only be done for models where $n_1 = n_2$or A equals the identity matrix.

Listing 4.22: Starting values (rrm02.job)

```
1  link=anim form='full' type='nat'
2                   83.0      3.0
3                    3.0      5.0
4  link=perm form='full' type='nat'
5                   42.0     12.0
6                   12.0      7.0
7  link=dam form='full' type='nat'
8                   33.0
9  link=residual form='full' type='nat'
10                   7.0
```

Listing 4.23: Results from VCE (rrm02.job)

```
1
2   ************************************************************************
3   *                E S T I M A T E S   I N F O R M A T I O N             *
4   ************************************************************************
5   Thu Aug 07 10:52:46 2008                          CPU time used:     0:00:27
6   AG Log likelihood :    8953.8733  status :  1   at iteration:     92 /    92
7   ------------------------------ Matrices: NATURAL --------------------------------
8    Type: A Level:   1  anim                          No.:       479 Pattern: T T
9       18.6881        6.9088
10       6.9088        4.7498
11   Type: R Level:   1  perm                          No.:       314 Pattern: T T
12      59.0433       14.2520
13      14.2520        6.5630
14   Type: R Level:   1  dam                           No.:        57 Pattern: T
15      42.3739
16   Type: E Level:   1  residual                      No.:      2198 Pattern: T
17      6.60637
18  ---------------------------- Matrices: Phenotypic --------------------------------
19      180.346          ---
20          ---          ---
21  ------------------------------ Matrices: RATIOS ----------------------------------
22   Type: R Level:   1  dam
23      0.23496
24   Type: R Level:   1  perm
25      0.32739       0.72400
26      0.72400       0.03639
27   Type: A Level:   1  anim
28      0.10362       0.73330
29      0.73330       0.02634
30   Type: E Level:   1  residual
31     0.036632
32  ---------------------- Matrices: STD_ERR of components ------------------------
33   Type: A Level:   1  anim
34      39.4948        6.1771
35       6.1771        2.1753
36   Type: R Level:   1  perm
37      16.1943        3.1503
38       3.1503        1.2882
39   Type: R Level:   1  dam
40      12.9375
41   Type: E Level:   1  residual
42      0.22232
43  ------------------------- Matrices: STD_ERR of ratios --------------------------
44   Type: R Level:   1  dam
45     0.077798
46   Type: R Level:   1  perm
47      0.11755       0.10028
48      0.10028       0.00723
49   Type: A Level:   1  anim
50      0.20945       0.67359
51      0.67359       0.01189
52   Type: E Level:   1  residual
53     0.4416E-02
```

4.11.2 Daily Gain in Bulls

Model definition with Legendre polynomials Body weights on more than 6000 bulls from the Czech Fleckvieh were available for this investigation with an average of 12 measurements per bull between 12 and 420 days of age. Data collection was carried out over a period of 20 years at 7 test stations. daily gain was computed on the basis of three successive weighings. The total test period from day 12 – 420 was subdivided into 8 equidistant blocks of 50 days, to be able to a multi trait analysis and also use a random regression model (see KREJČOVÁ, u.a., 2006). That daily gain within each of the 8 blocks was chosen, which was closest to the middle of the time span. After editing each bull had between 4 and 8 repeated measurements in daily gain.

HYS classes were used to capture the effects of station, year and season, the latter consisting of three months starting with December. Test month instead of test day was used as an environmental effect as daily gain was computed based on three consecutive measurements. A graphical analysis revealed a quadratic trend on age. Furthermore, as the growth curves differed among HYS classes, they were nested within these HYS groups.

Listing 4.24: VCE parameter file RRM03

```
1  COMMENT daily gain of bulls
2
3  DATA
4      datfile = 'dat03.txt'
5      format='(2F12.0,3F8.0)'
6      dep  = gain
7      indep = age anim hys3 perm;
8      pedfile ='ped03.txt' format='(4I10)' link = anim;
9
10 MODEL
11     gain  =  [plg2(age)]hys3 + [plg2(age)]anim + [plg2(age)]perm;
12
13 COVARIANCE
14     anim;
15     perm;
16
17 SYSTEM
18     non_zero = 1500000
19     total = 11000000;
20
21 OUTPUT
22     covfile='cov_rrm03.txt' format='(10f12.5)' form='full';
23     solutions = 'sol_rrm03.txt' format='(10f12.5)'
```

Listing 4.25: Results from VCE (rrm03.job)

```
1  ************************************************************************
2  *                E S T I M A T E S   I N F O R M A T I O N             *
3  ************************************************************************
4  Wed Aug 06 12:14:27 2008                              CPU time used:     0:03:15
5  AG Log likelihood :   22419.2665  status :  1   at iteration:     70 /    70
6  ------------------------------ Matrices: NATURAL --------------------------------
7   Type: A Level:   1  anim                          No.:       7029 Pattern: T T T
8      5931.40       341.91      -1460.85
9       341.91      2417.14       -797.11
10    -1460.85      -797.11       2400.37
11  Type: R Level:   1  perm                          No.:       6374 Pattern: T T T
12     2326.94      -342.17       1214.44
13     -342.17      4887.83       1739.99
14     1214.44      1739.99       1394.73
15  Type: E Level:   1  residual                      No.:      32273 Pattern: T
16     30287.8
17 ------------------------------ Matrices: Phenotypic -------------------------------
18     51038.6          ---           ---
19         ---          ---           ---
20         ---          ---           ---
21 ------------------------------ Matrices: RATIOS -----------------------------------
22  Type: R Level:   1  perm
23     0.04559     -0.10146      0.67412
24    -0.10146      0.09577      0.66641
25     0.67412      0.66641      0.02733
26  Type: A Level:   1  anim
27     0.11621      0.09030     -0.38716
28     0.09030      0.04736     -0.33093
29    -0.38716     -0.33093      0.04703
30  Type: E Level:   1  residual
31     0.59343
32 ---------------------- Matrices: STD_ERR of components -------------------------
33  Type: A Level:   1  anim
34     841.877      269.016      470.207
35     269.016      640.895      375.295
36     470.207      375.295      505.526
37  Type: R Level:   1  perm
38     776.808      232.613      444.509
39     232.613      706.888      375.116
40     444.509      375.116      490.210
41  Type: E Level:   1  residual
42     313.088
43 ------------------------- Matrices: STD_ERR of ratios ---------------------------
44  Type: R Level:   1  perm
45     0.01523      0.07428      0.16989
46     0.07428      0.01346      0.14907
47     0.16989      0.14907      0.00952
48  Type: A Level:   1  anim
49     0.01611      0.07148      0.15012
50     0.07148      0.01247      0.16230
51     0.15012      0.16230      0.00973
52  Type: E Level:   1  residual
53    0.5585E-02
```

Computation of the h^2- function Listing 4.26 demonstrate how the SAS procedure IML can be used to compute the h^2- function. Only simple DO loops are required which should make

translation into other programming languages straight forward.

Listing 4.26: SAS-File for coumputing the h^2 - function for RRM03

```
Proc iml;
start main;
filename out 'rrm03_heri.txt';
file out;
dim=3;tmin=49;tmax=393;
x=shape(0,dim,1);
create gain var{age,h2,vg,vp,ve,vy};
/* results from VCE (rrm03.job) */
ge={ 5931.43      341.94     -1460.82,
      341.94     2417.16      -797.16,
    -1460.82     -797.16      2400.39};
ng=nrow(ge);
pe={ 2326.90     -342.19      1214.42,
     -342.19     4887.83      1740.02,
     1214.42     1740.02      1394.76};
np=nrow(pe);
ve=30287.8;
do age=tmin to tmax;
/* transformation */
 t = 2.0*(age-tmin)/(tmax-tmin)-1.0;
 x[1]=sqrt(0.5);
 x[2]=sqrt(3/2)*t;
 x[3]=sqrt(5/2)*(3/2*t**2-0.5);
 vg=0.0;
 do i=1 to ng;
  do j=1 to ng;
   vg=vg+ge[i,j]*x[i]*x[j];
  end;
 end;
 vp=0.0;
 do i=1 to np;
  do j=1 to np;
   vp=vp+pe[i,j]*x[i]*x[j];
  end;
 end;
 vy=vg+vp+ve;
 h2=vg/vy;
 put (age) 4.0 (vg) 12.5 (vp) 12.5 (ve) 12.5 (h2) 9.5;
 append;
end;
closefile out;finish;
run;
proc gplot data=gain;
 axis1 order=(49 to 393 by 20);
 plot h2*age;
run;
```

4.11.3 Feed Intake in Beef

In this example we analyze beef data from a trial with individual feed intakes measured. The test starts right after weaning after around 7 months plus a 4 week adaptation period. The feed intake and the body weight in kg of the animal are recorded each week. The total test lasts 72 weeks. The first few lines of the data file are given in table 4.27.

Listing 4.27: data file for feed intake

	FI	AGE	DAMAGE	LENG	CCG	SEQ	ANIMAL	
1	0.50000E+02	0.19800E+03	0.10600E+03	1	139	1	6921	1
2	0.74000E+02	0.21900E+03	0.10600E+03	2	139	1	6921	1
3	0.75000E+02	0.24000E+03	0.10700E+03	3	139	1	6921	1
4	0.78000E+02	0.25400E+03	0.10700E+03	4	139	1	6921	1
5	0.64000E+02	0.27500E+03	0.10800E+03	5	139	1	6921	1
6	0.50000E+02	0.25400E+03	0.10600E+03	1	81	2	6923	1
7	0.58000E+02	0.27500E+03	0.10700E+03	2	81	2	6923	1
8	0.75000E+02	0.29600E+03	0.10800E+03	3	81	2	6923	1
9	0.78000E+02	0.31000E+03	0.10800E+03	4	81	2	6923	1

What do we want to do?

1. we want to get BLUP for each animal for every possible week (or point in time during the test)
2. we assume that the residual variances change over the course of the test

While a function can be fit in VCE for a random component (i.e. getting a BLUP as a function of time), this cannot be done for residuals at this point. Instead, for the residual (co)variances the measurements of independent variable have to be grouped in classes where the time span within a class is sufficiently small that a constancy of variance can be assumed while on the other side not too many classes are generated (the impact of this is discussed later).

To approach a complete random regression analysis we shall first start with a simple model which uses – for demonstration purposes only – linear regressions for the random components.

Genetic analysis for early growth (first 6 weeks) In our sample dataset we want to do a genetic evaluation for the first 6 months only. If we plot the growth over the first 6 weeks we get a straight line. Thus, clearly the relationship between feed intake and age is sufficiently described by a linear function. Using the VCE syntax for this the animal effect in the model would look like:

```
feed_intake = .... [p1(age)]animal
```

This implies a linear regression of feed intake on age nested within each animal, i.e. we get a regression coefficient for each animal. Now what does that mean? The animal component is interpreted (on the basis of our model definition) as the additive genetic effect. What these values mean we shall discuss later.

Listing 4.28: VCE Parameter file (linear RR)

```
comment rr-test1
   feed intake in beef cattle with weekly weighings
data
   datfile = 'fi_ma10000.dat'
   depend  = feed_intake
   indep   = age damage leng animal pe    group_by pe ;
   pedfile = 'fi_ma10000.ped' format='(4I10)' link = animal;

model
   feed_intake = int p2(age)  [1, p1(age)]pe [1, p1(age)]animal / cf = CLASS(leng) ;

covariance
   animal;
   pe;
system
   non_zero =  281490
   total    = 7525354
Output
  covfile;
End
```

Listing 4.29: VCE Parameter file (plg2)

```
comment rr-test3
   feed intake in beef cattle with weekly weighings
   Legendere

data
   datfile = 'fi_ma10000.dat'
   depend  = fi
   indep   = age damage leng animal pe    group_by pe ;
   pedfile = 'fi_ma10000.ped' format='(4I10)' link = animal;
model
   fi = int p2(age) plg2(age)]pe [plg2(age)]animal / cf = CLASS(leng) ;

covariance
   animal;
   pe;
system
   non_zero =  500000
   total    = 7525354

Output
   covfile;
End
```

4.11.4 Coding requirements

In random regression models data need to be coded in a different manner. Relevant parts in this context are the statements *group_by* and */cd=class(leng).*

The latter statement puts the measurements along the time trajectory (here we have AGE) into separate residual variances. This means, that the age 198 (first record in the data file in Listing

4.27) into the class 1, i.e. the first residual variance as indicated by the 1 in *LENG*. Likewise, the fourth record places the age record of 254 days into the 4th variance as given by the 4 under *LENG*.

How are these columns set up? Clearly, the *AGE* column is obvious: it is the age of the animal at which the weight was recorded. To place a record into the correct position in the residual covariance matrix the column *LENG* (or whatever you may call this) needs to indicate the class that this age measurement is placed into. In our sample dataset we have data from the first 5 weeks only. Accordingly. we have measurements taken every three weeks (with a little variation). What is important to notice, is that there must not be more than one measurement per animal in one such *LENG* class. If you do have this, VCE will stop with an appropriate message. The LENG class is coded just like any other class effect and needs therefore be coded as 1,2,3...n.

4.11.5 Running the job

If we run parameter file in table 4.28 we get the results given in Listing 4.30. According to the model statement in Listing 4.28, we get three covariance matrices: a 2 x 2 matrix for the permanent environmental effect (pe) and the animal effect (which is considered additive genetic) and a 5x5 for the residual covariance matrix.

Listing 4.30: Results from rr-test1

```
MFLOPs during factorization        :        150.41

*****************************************************************************
*                  E S T I M A T E S   I N F O R M A T I O N                *
*****************************************************************************

Thu Oct  2 11:03:30 2003                                CPU time used:      0:12:06

AG Log likelihood : ------------  status :  1   at iteration:     111 /    111


------------------------------ Matrices: NATURAL --------------------------------
             3            1            1 R  pe
     2000 T T
          40.81590   -0.01614
                      0.00001

             4            2            1 A  animal
    11728 T T
          53.3030    -0.1833
                      0.0219

             5            3            1 E  residual
     2000 T T T T T
          88.697     -7.764     -0.692     -9.917     -6.307
                    101.432     11.380      9.548     -7.070
                               128.078      3.185     -5.049
                                          113.363     -0.015
                                                     105.870
```

What do these values mean? Let us start with the residual covariance matrix. As you remember, we are assuming heterogeneous variances along the time axis. That is why we grouped the time span of the feeding trial in these blocks of around a month, thereby allowing different residual variances for each month. This is what we have on the diagonal: as we can see: the residual variances increases from $88.7kg^2$to $101.4kg^2$to$128.1kg^2$and then goes down again to $113kg^2$ and finally $105.9kg^2$ for the last group. The covariances give a somewhat mixed picture. Actually, we would expect the residual covariance to diminish as the time increases between measurements. But that is not really shown here: the covariances in the first line go from -7.8 over -0.69 to -9.9 to -6.3, not a consistent pattern.

While the estimates from the residual covariance matrix are on the original scale of the measurements, this is not so for the random regression matrices. Here we are getting covariances for the regression coefficients of the function chosen. In the case for the linear polynomial the situation is slightly more complicated. The variance components are given for the coefficients of the function used. Thus, if we want to know the variance we need to specify a given time and compute them.

4.12 Model with dominance genetic effect

4.12.1 Dominance effect in pure breed populations

Let us assume:

Table 4.1: Sheep data 'mrode.data'

Sheep	Sire	Dam	Season	Weight	tr2	tr3	tr4
5	1	2	1	17.0	2.7	0.34	0.34
6	3	4	1	20.0	3.1	0.40	0.40
7	6	5	1	18.0	2.9	0.24	0.24
8	-	5	1	13.5	3.5	0.31	-
9	3	8	2	20.0	2.5	0.28	0.28
10	3	8	2	15.0	3.3	0.39	0.39
11	6	8	2	25.0	2.5	0.22	-
12	6	8	2	19.5	2.9	0.18	0.18

Single trait analysis using family subclass effect.

$$y = X\beta + Za + ZWf + e$$

$$y \sim N(X\beta,\ ZG_aZ' + ZWD_fW'Z' + R^{-1})$$

Algorithm 4.2 Single trait analysis with family subclass effect

```
COMMENT job = dom01
   the model is based on family effect
DATA
   datfile='../test/data/mrode.data'
      format='(t13,f6.0, t10, f3.0, t1,f3.0,t1,f3.0)'
      header=0
      dependent=weight
      independent=sex tier family;
   pedfile='../test/data/mrode.ped'
      format='(5i3)'
      header=0
      link=tier dominance = family;
COVARIANCE
   tier;
   family;
   start_asc ='../test/data/mrode.cov';
MODEL
   weight = sex + tier + family;
   scaley non;
SYSTEM
   tolerance = 1.0d-10
   method='SO'
OUTPUT
   dominance;
   lhs;
   debug = .true.
END
```

4.12.2 An example: Egg production in laying hens

Estimating dominance variance will be demonstrated using a real life dataset from laying hens.

Introduction, data, and model

Estimation of dominance variance in data from a line of laying chicken will be explained in this section. For statistical evaluation 8652 data records were available on surviving chicken from 3 generations.[6] Traits investigated were:

- ▷ number of eggs between 20 – 28 weeks (EN1-2), between 28 and 48 weeks (EN3-7) and between 20 and 48 weeks (EN1-7)
- ▷ egg weight in week 28 (EW1), 33 (EW2), 40 (EW3) and the average egg weight from the three measurements (EW)

The model used for estimating additive genetic and dominance variance is:

$$y_{ij} = x'_{ij}\beta + b \cdot \Delta_{ij} + a_{ij} + f_i + e_{ij}$$

with: y_{ij} =performance record of hen j from the full sib family i, x'_{ij}= design vector of the fixed effects, β =vector of fixed effects of house, hatch and floor, b =regression coefficient, $\triangle_{ij}$= inbreeding coefficient of hen j (as covariable), a_{ij} =additive genetic effect of hen j, f_i =random effect of the full sib family i and e_{ij}= random residual.

Let a be the vector of all additive genetic effects and f the vector of all family effects, which are required to describe the phenotypic performance records and for the computation of the inverses of the relationship matrices. From this follows:

$$Var(a) = A \cdot \sigma_a^2; \quad Var(f) = F \cdot \sigma_f^2 \quad \text{and } \sigma_f^2 = \frac{1}{4} \cdot \sigma_d^2$$

For direct and successive computation of F^{-1} additional (dam×sire)-subclasses are required[3]. The number of elements in vector f does not necessarily correspond to the number of full sib families in the data. With complex family structures the dimension of f may considerably exceed that of a. The vector size of the example are 10099 and 12789 elements for the vectors$a$ and f, respectively.

setting up the parameter file in VCE

The structure of the input file after coding (e.g. using PEST) and addition of the inbreeding coefficient is shown in Listing 4.31. In the DATA section the variable FAMILY has to be used to read the animal code as shown in Listing 4.32. Reading the same animal column into variables family and animal is done through the FORTRAN specific format "format='(4f12.0,2f8.0,t57,f8.0)"', where the "t57" directs the input to column 57 in the file to read the last variable (i.e. family): after reading the three traits, the covariable and the factors HSE and ANIM sequentially from left to right in the format, control then jumps back to column 57 as indicated by the t57 to pick up the variable FAMILY).

Comment 1: The inbreeding coefficient has been included in the statistical model of this example (see above). Also VCE , in a prerun, can be used to compute the inbreeding coefficient by defining in the OUTPUT section the command "inbreeding='file_name"'. This inbreeding coefficients written to file 'file_name' will then have to be moved into the original datafile by a tool of your choice. Notice, that this has nothing to do with estimating dominance variance but rather with the choice of the model.

Comment 2: VCE can also do multivariate dominance model. For this example this would be specified in the MODEL section by writing: EN1_2 EN3_7 EW = hse + p1(inbr) + anim + family;

Listing 4.31: Input Data (dat11.cod)

```
EN1_2  EN3_7  EW  INBR  HSE  ANIM
 0.39000E+02  0.12600E+03  0.66300E+02  0.00000E+00      1      3641      1
 0.26000E+02  0.13500E+03  0.61300E+02  0.00000E+00      1      3642      1
 0.49000E+02  0.13500E+03  0.61100E+02  0.00000E+00      1      3643      1
 0.37000E+02  0.13300E+03  0.61400E+02  0.00000E+00      1      3644      1
 0.32000E+02  0.13600E+03  0.54000E+02  0.00000E+00      1      3645      1
```

Listing 4.32: VCE parameter file (dom01.job)

```
comment Daten von Legehennen

DATA
   datfile='dat11.cod' format='(4f12.0,2f8.0,t57,f8.0)'
   dep=    EN1_2 EN3_7 EW
   indep=inbr hse anim family;
   pedfile='ped11.cod' format='(5i10)' link=anim dominance = family;

COVARIANCE
   anim;
   family;

MODEL
    EN3_7 = hse + p1(inbr) + anim + family;

SYSTEM
   total=19246617

OUTPUT
    solutions='vad_EN3_7.txt'
    inbreeding='inb_koef.txt';

END
```

The VCE output is given in table 4.33. The optimization converges after 31 iteration with the best status: 1. The dominance variance is estimated by multiplying the family variance by 4.

The additive and dominance variance accounts for 15% and 13% of the total variance for trait EN3-7, respectively.

Listing 4.33: Results from VCE (dom01.job)

```
1  ********************************************************************************
2  *                E S T I M A T E S   I N F O R M A T I O N                     *
3  ********************************************************************************
4  Tue Sep 16 13:34:58 2008                                CPU time used:     0:02:38
5  AG Log likelihood :   18976.4471  status :  1   at iteration:      31 /     31
6  ------------------------------ Matrices: NATURAL --------------------------------
7   Type: A Level:   1  anim                          No.:       10099 Pattern: T
8      8.31193
9   Type: D Level:   1  family                        No.:       12789 Pattern: T
10     1.73330
11  Type: E Level:   1  residual                      No.:        8625 Pattern: T
12     44.5001
13 ---------------------------- Dominance = 4 * family ------------------------------
14  Type: D Level:   1  family                        No.:       12789 Pattern: T
15     6.93319
16 ---------------------------- Matrices: Phenotypic --------------------------------
17     54.5453
18 ------------------------------ Matrices: RATIOS ----------------------------------
19  Type: D Level:   1  family
20     0.12711
21  Type: A Level:   1  anim
22     0.15239
23  Type: E Level:   1  residual
24     0.81584
25 ---------------------- Matrices: STD_ERR of components -------------------------
26  Type: A Level:   1  anim
27     1.39198
28  Type: D Level:   1  family
29     2.08659
30  Type: E Level:   1  residual
31     1.08729
32 ------------------------ Matrices: STD_ERR of ratios ---------------------------
33  Type: D Level:   1  family
34    0.9846E-02
35  Type: A Level:   1  anim
36    0.024484
37  Type: E Level:   1  residual
38    0.020870
39 ---------------------- Matrices: Phenotypic correlations -----------------------
40        ---
41 ********************************************************************************
42 *                  Optimization finished with status :   1                      *
43 ********************************************************************************
44  Terminated with gradient small, components are probably optimal.
45 ********************************************************************************
46 *                         Thank you, for choosing VCE!                         *
47 ********************************************************************************
```

4.12.3 Computational aspects in VCE

Dominance effects are created in VCE from a pedigree file in two steps. In the first step, known families are determined and used to set up the system.

Computation of F^{-1}

1. pedigree files are created as for standard animal models as for A^{-1}
2. VCE then creates mates (sire, dam) following [3]

Prediction of individual dominance effect

On output, one may request to print out individual dominance effects. This is done by calculating dominance due to Mendelian sampling (4.15) and adding family effect as shown in (4.16).

$$y = X\beta + Za + ZWf + Z\delta + \varepsilon \tag{4.14}$$

$$\widehat{e} = Z\widehat{\delta} + \widehat{\varepsilon} = y - X\beta - Za - ZWf$$

$$\widehat{\delta} = D_\delta Z' R^{-1} \widehat{e} \tag{4.15}$$

$$\widehat{d} = W\widehat{f} + \widehat{\delta} \tag{4.16}$$

4.13 Models with disconnected residual covariance structure

This model deals with a situation common in pig breeding with two testing environments in a joint genetic evaluation: a field test and a station test on other animals. This means that we have data on animals in the field and other tested at the test station and both groups of animals tied together via common ancestors.

4.13.1 Data Preparation

There are two ways of preparing the data: either all data can be in one file or we can have them in two files. The latter would be the natural thing to do, as station and field comprise different traits and effects. Also, the source is usually different. But one file may on the whole be easier to handle. But that depends on your coding setup.

4.13.2 One data file

Specifying input for one datafile is simple and given in table 4.35. Here, we assume the default format as it comes out of the coding process in PEST. Therefore, no format needs to be specified, you can also leave the format for the pedigree out.

The traits within the data file have been arranged such that the two from the field come first: bfft and adgft. They are followed by the three station test traits adgst, vc and bfst. Having prepared the data yourself you would know that there are no records that have all 5 traits because the animals were either on the farm or at the test station. This is given in the list output (see table 4.34). In this situation with absolutely no records that have all traits, VCE – in its wisdom – decides that there should be no full residual covariance matrix. Instead, it creates a block diagonal matrix with a 2x2 block in the upper left corner for the field test traits and a 3x3 in the lower right for the station traits.

Listing 4.34: pattern of traits for Np08

```
1 Pattern of traits
2
3     Count          bfft     adgft     adgst          vc       bfst
4
5         0          x        x         x         x          x
6      4563          x        x         -         -          -
7      3231          -        -         x         x          x
8      3231          -        -         x         x          x
9
```

This is shown in table 4.36 which gives the starting values.The prerequisite for having VCE reduce the residual covariance matrix to those two blocks is that the traits belonging together in

one block are indeed placed adjacent in the data file. Thus, if you are mixing the station and field test, then VCE will produce a full matrix.

Listing 4.35: parameter file Np08 (one file)

```
DATA
c ....................  field traits
  datfile    = '../data/np37.dat'
  dep  = bfft adgft adgst vc bfst
  indep= wofft hms animal sex litter hysft stys ;
c ....................  pedigree file
  pedfile    = '../data/np37.ped' format= '(4I10)' link=animal;
MODEL
  bfft  = reg(wofft)          animal sex litter hysft;
  adgft =                     animal sex litter hysft;
  adgst =                     animal sex litter       stys;
  vc    =                     animal sex litter       stys;
  bfst  =           reg(hms)  animal sex litter       stys;
COVARIANCE
  animal ;
  litter ;
  hysft  ;
  stys   ;
SYSTEM
  non_zero=963000
  total = 10299579
end
```

Listing 4.36: default starting values with disconnected datasets

```

            8           5           1 E  residual
      0.25000   0.01789   0.00000   0.00000   0.00000
                0.24990   0.00000   0.00000   0.00000
                          0.24992   0.01789   0.01788
                                    0.24988   0.01820
                                              0.24985
```

4.13.3 Two files

Using the "natural" data representation, i.e. having data separately in two files, is the route given in table 4.37. As can be seen we have two blocks in the DATA section, one for field traits and another for station traits. At the same time there are also variables common to both: clearly the animal but also litter and sex. As we can see more than one file is used by simply adding another datfile block with its specifications of format dependent and independent. The problem with more than one file lies in the issue of coding. As you will know by now, input data to VCE must be coded 1,2,3 and so on. While this is no problem for effects that reside solely in one file, it may become problem with "animal". If we were coding only one file at a time, animals in the other file may not be included. This will not be the case if we would use the same pedigree file for the two coding runs for field and station. If coding is done with PEST then this setup will produce

correct results provided the pedigree file is correct for the sum of the field plus the station test file. This again is a question of how the pedigree file itself is generated. One usually starts with the animal records, i.e. the measurement. Then one should go recursively back through all the pedigree records one has and put only those in the pedigree file that have connections with the data. One way of doing this is to use our program gen_ped.f90 for flat files. Another way in the database context of APIIS is to use the PERL scripts that we developed there. Some are using SAS for this and I am sure that there are still other ways to generated pedigrees.

Going back to data input to VCE: one needs to be aware of the problem of coding. This is the objective of the paragraph above. As can be seen from the parameter file 4.37, the two files used here are actually only one. Further down we shall discuss the setup of specifying only one file. Here, it was pretend we have two file: the first contains field test traits ultrasonic backfat (bfft) and average daily gain (adgft), while the second contains the average daily gain on test, the valuable cuts and the measured backfat on the carcass. Because VCE is written in FORTRAN the format specifiers have to be those of FORTRAN. Have a look at a FORTRAN manual in case you do not know them by heart. Because we actually read from one file, we need to skip the station test columns in the field test read and conversely for the station test access.

The MODEL section is straight forward: we have a linear regression for some of the traits plus a number of random components. As regards coding: perhaps “litter” is worth a comment. Here all those records belonging to one litter have to have the same code. This is conveniently done by concatenating the sow identification and the birthdate of the litter.

The COVARIANCE section tells us which of the effects are random, here it is animal, litter, herd/year/season in the field and station/year/season at the test station. If the line “residual(datfile)” would not be in this section then we would always assume a full residual covariance matrix, i.e. a five by five matrix for this VCE6 run. This is different in VCE4: each input file created one residual covariance matrix. Thus, VCE4 would have created from this parameter file two residual covariance matrices: a 2x2 for the two field test traits and a 3x3 for the three station test traits. This (meaningful) setup is defined in VCE6 by adding the keyword “residual(datfile)” to the COVARIANCE section as done in parameter file as shown in table 4.37.

Listing 4.37: parameter file Np08

```
 DATA
c ....................  field traits
  datfile     = '../data/np37.dat'
  format      = '(2f12.0,36x,f12.0,12x,f8.0,f8.0,f8.0,4f8.0)'
  dep  = bfft adgft
  indep= wofft animal sex litter hysft ;

c ....................  station traits
  datfile     = '../data/np37.dat'
  format      = '(24x,3f12.0,12x,f12.0,3f8.0,8x,f8.0)'
  dep  = adgst vc bfst
  indep= hms animal sex litter stys ;
c ....................  pedigree file
  pedfile     = '../data/np37.ped' format= '(4I10)' link=animal;
MODEL
  bfft  = reg(wofft)          animal sex litter hysft;
  adgft =                     animal sex litter hysft;
  adgst =                     animal sex litter       stys;
  vc    =                     animal sex litter       stys;
  bfst  =           reg(hms)  animal sex litter       stys;
COVARIANCE
  residual(datfile);
  animal ;
  litter ;
  hysft  ;
  stys   ;
SYSTEM
  non_zero=963000
  tolerance=1.E-10
  total = 10299579
end
```

The output does look a little different from the VCE4 setup: here, the matrix printed is a 5x5, however the rows and columns not pertaining to the dataset are all zero 4.38. The more general setup behind this is that the datafiles are viewed a sources for heterogeneous variances.

Listing 4.38: residual covariance matrices (heterogeneous)

```
            8           5           1 E  residual/np37.dat
      4563 T T F F F
           0.015     0.693     0.000     0.000     0.000
                   513.171     0.000     0.000     0.000
                               0.000     0.000     0.000
                                         0.000     0.000
                                                   0.000
            9           6           1 E  residual/np37.dat
      3231 F F T T T
           0.00      0.00      0.00      0.00      0.00
                     0.00      0.00      0.00      0.00
                            4217.92    -38.98      4.29
                                       2.81     -0.16
                                                 0.06
```

Without the "residual(datfile)" we get a full residual covariance matrix for all 5 traits in the model as given in table 4.39. Firstly, there is only one matrix generated, which is to be expected. Then, the residual covariances among station and field test traits are not zero, however, the tend to be small. But one should be very clear about these covariances: there are NO data to estimate them, because there is not one record in the data set that has all five records measured. Therefore, this full covariance matrix is meaningless and should not be generated for the given dataset!

Listing 4.39: residual covariance matrices (full)

```
1                    8          5          1 E  residual
2          0 T T T T T
3              0.015      0.693      0.137      0.001      0.000
4                         513.198     30.981      0.128      0.089
5                                   4217.538    -38.982      4.287
6                                               2.811     -0.163
7                                                          0.058
```

5 FAQ

In this sections you can find answers to frequently asked questions. We would be pleased if you could supply us with more answers to questions not covered here. Send them and we shall incorporate the text.

5.1 I am getting status 3, what now?

Estimating covariance components with VCE is an iterative process. As such there has to be a stopping criterion which is used to indicate convergence. In VCE this is the first derivatives of the likelihood with respect to the parameters. If all goes well, i.e. if the first derivative is indeed zero, then a status 1 is issued. If you get this, you know that you have reached convergence and that you can trust the covariance components. Digital computer do have a limited accuracy. With not very well behaved systems (whatever that may be) numerical problems may arise because of limited accuracy. Then you may get a status = 2. And finally, you may get 3. Runs with status 2 are often OK, while the status 3 is often an indication that the model used does not really fit the data.

What can be done in this situation? A status > 1 arises usually in higher dimensional model, i.e. in multiple trait models: the more traits the more often this may happen. Also, with random regression model this may happen more often. Sometimes the results are obviously nonsensical, for instance if you get a heritability of .8 on some reproduction trait, you know that something is wrong. In this situation go back to the model and ask yourself if it is really appropriate. Levels of fixed effects with few observation may also be a cause. Then simplify the model, but always know your data structure.

If on the other hand the results look reasonable, they may indeed be just fine, even with a status 3. In such a case do a few univariate or bivariate runs. Their results will not differ a lot from the higher dimensional run, if this one is OK. Then you can be confident to use the results even from a run that finished with status 3.

5.2 The degree of fill is above 80% and things are getting slow

In the initial phase of a VCE run, the non zero coefficient of the mixed model equations are set up and stored in memory. This is done in sparse format using the IA, JA, A storage scheme. To locate coefficients in the IA buffer a hashing scheme is used. This means, that on the basis of the mixed model address (row, column) a hashing value is computed. This places all row and

column combination as evenly as possible over the complete vector. Such a newly computed hash value is used to access vector location in IA and JA. If this location is free, then the row and column numbers are stored in the respective location determined by the hash in IA and JA and all is well. If however this location already holds a value (from some other row/column pair), then the next sequential locations are checked until a free location is found. When the degree of fill of the IA/JA vectors goes higher than 70% it will happen increasingly often, that the pre-computed location is already occupied and that increasingly more further locations will have to be checked to find a free spot, which slows the initial process of storing the non zero coefficients down to a degree that it may not come to an end.

Thus, if the degree of fill goes beyond 70% and the hit rate starts creeping up from a normal 1.2 or so, then you should kill VCE (CNTR C), increase in SYSTEM the NON_ZERO elements and start again.

Once all coefficients are loaded in memory, the following computations are independent if the hit rate has gone up to 1.1 or 10.

5.3 VCE says the computer does not have enough memory or SEGMENTATION violation

You have a smallish kind of data set, and a two trait model. You are the proud owner of a computer with 1GB of RAM. Then you start VCE and it tells you SEGMENTATION violation or that you may not have enough memory. All very strange: big computer, small data set and model. The explanation may be easy: in the SYSTEM section the user needs to specify the number of NON_ZERO elements and the TOTAL. This is sometime done in the fashion of putting the finger down on the 9 and then keeping it there for a while, which may result in something like: non_zero = 999999999 . Lets look at this: we are anticipating one less than 1 billion non zero elements! So be a little more humble and start with less. If the number you give for TOTAL VCE will incrementally and automatically stop the current operation, and reallocate a bigger chunk and start again. But you can equally well stop the process and increase the number yourself.

The traps involved with determining the number of non zero elements have been described above in paragraph 5.2 on the previous page.

The program 'top' which is available on every real computer (don't know about windows boxes) should be used by you to see how much RAM your VCE process uses.

5.4 Can I do a 20 trait model?

That would be nice, but can likely not be done. There are a number of reasons for this.

Memory memory requirements increase quadratically with the number of traits. In a univariate model for each effect x effect combination we have 1 non zero element, with 2 traits that

would be 4, with 3 9 elements and with 20 it would be 20*20 = 400. This will easily blow the physical memory of your computer, even if you have GB instead of megabytes in your machine.

CPU-time per round Along with memory requirements increase the computations required also goes up, as they are a direct function of the number of non zero elements. Thus, even if you may be able to store a large system in memory, the computation may take longer than you have time to wait.

Number of iterations Furthermore, higher dimension systems often need more rounds iterations to converge. So this will further add to the computing time.

So what can be done safely? Well, it all depends on the data set and the model in your special situation. Five trait model are common, some have done ten trait runs. Actually, we should ask among the VCE users to find out.

5.5 Is there a 64bit version of VCE?

As we have seen above, VCE can suck up a lot of memory and thereby possibly hit the 2GB limit that 32 bit operating systems pose. In the last years 64 bit machines have become increasingly available, on which the addressable memory is 8TB which is about 8000GB. So for large systems 64 bit machines will be useful tools. While this will address the memory issue it will still leave us with the computing time issues outlined above.

But anyway, the question was: " are there 64bit VCE versions available". The answer is: "Yes", at least for some platforms. To date we can produce ourselves X86_64 versions for Linux which will run on Xeons and AMD. To check have a look at our ftp site `ftp.tzv.fal.de/pub/vce`.

5.6 Can I run a 32 bit version on a 64 bit computer?

Yes indeed! If you do not need the extended address space beyond 2GB then a 64bit version has no advantages. Contrary to popular belief, 64bit versions are not faster on the same problem than a 32 bit version.

5.7 What is estimated: covariance components or ratios?

There are never programs without errors. This certainly also applies to VCE. We have been careful testing and debugging. Also, we have put in lots of output that the user can and should be checking. And this is also what she should do with the final results. What is actually estimated by VCE is only the variance and covariance components for the residual and the random factors. Ratios and phenotypic are computed by VCE on the basis of these estimates as a service to the users. However, you yourself should verify if this is done correctly.

5.8 Can I do a likelihood ratio test?

The answer is here a clear NO. While this may be unfortunate, this is the situation. The likelihood value given by VCE is actually different from the real likelihood in that only that part required for optimization is computed. Further information can be found here 4.11.1.

5.9 BLUP vs VCE: different models?

Often the process of estimating covariance components is treated as something totally different from that of computing BLUP and BLUE in selection programs. The effects considered in the model may be very different and also the dimensionality in terms of number of traits involved. Sometimes genetic evaluation is done on, e.g. 5 traits, while covariance components are estimated on bivariate models with some averaging elements that came out of multiple runs. A statistical model is an attempt the account for the variances in data based on knowledge of the population structure and what influences records. Thus, there is no (good) reason to have different set of model for different purposes for the same datasets.

As a general rule: the models should be identical for BLUP and variance component estimation. Clearly, with iteration on data much larger systems can be solved for BLUP, while REML requires storage of coefficients in memory. Thus, one could consider using a smaller dataset for the latter. But changing the model does not sound like a good idea.

5.10 For which platforms is VCE available?

Currently, VCE is available for Linux both on the Intel i686 and x64 platform (the 64 bit Xeon and AMD processors. Also, binaries for Windows are available for 32 and (hopefully) 64 bit. A 32bit version is also there for the Intel based Mac OS. Furthermore, a binary is available for IBM AIX/RISC-6000 achines.

5.11 Where do I click to start VCE?

Nowhere! Actually, that is not quite correct. On Windows machines you need to click to start a console. From there on you only work in the console by using your wee little fingers to push the keys and not the mouse. This is Old School and usually way faster that pointing and clicking.

If you do not like it: too bad.

5.12 Does VCE produces standard errors?

Indeed, VCE produces estimates of standard errors of the components and ratios. However, a few constraints need to be noticed:

- ▷ standard errors are based on the second derivatives which are approximated through finite differences.
- ▷ as a result standard errors only make sense if full convergence (i.e. status 1) is reached.
- ▷ further, if VCE has been (re)started with starting values close to the solutions, resulting in fast convergence, VCE has not had enough time to build up the approximated second derivatives. As a result the standard errors will not be meaningful.

5.13 My VCE job has been running for a week, can I see the current estimates?

Yes, VCE writes during each iteration the outputs to a binary log file. If your parameter file is np01 look for file 'np01.cov-bin'. To get the current estimates you do the following:

1. cp np01 np01cur (Linux) - make a copy of the pfile
2. edit the pfile: in section COVARIANCES put the keyword DUMP_BIN='np01.cov-bin'
3. continue editing: in section OUTPUT put keyword REPRINT;

```
covariance
  litter; animal;
  dump_bin='np09.cov-bin';
output
  reprint;
end
```

produces:

```
---------------------------------- VCE 5.3.0  -----------------------------------
  09.09.2008  16:08:33              np09CURR                                page 1
 ----------------------------- Matrices: NATURAL  -------------------------------
  Type: A Level:   1  animal                          No.:          0 Pattern: T T T
     2612.74        -1.95         4.06
                     0.01         0.05
                                  2.60
  Type: R Level:   1  litter                          No.:          0 Pattern: T T T
     1197.14        -1.18         2.83
                     0.01         0.02
                                  1.03
  Type: E Level:   1  residual                        No.:          0 Pattern: T T T
     3227.13        -3.30        12.05
                     0.02         0.06
                                  3.16
```

If you want a full matrix with 6 decimal digits the output section becomes:

```
output
```

```
  reprint;
  mform='full' format='(F12.6)'
end
```

and the output is:

```
eg@eno:~/newvce/test/temp$ cat np09CURR.lst
 ---------------------------------- VCE 5.3.0  -----------------------------------
  09.09.2008  16:13:42              np09CURR                                page 1
 ----------------------------- Matrices: NATURAL  -------------------------------
  Type: A Level:   1  animal                          No.:          0 Pattern: T T T
  2612.742893     -1.952381      4.063978
    -1.952381      0.012723      0.050691
     4.063978      0.050691      2.601138
  Type: R Level:   1  litter                          No.:          0 Pattern: T T T
  1197.144739     -1.183935      2.832973
    -1.183935      0.005427      0.021632
     2.832973      0.021632      1.025040
  Type: E Level:   1  residual                        No.:          0 Pattern: T T T
  3227.127040     -3.299546     12.054797
    -3.299546      0.018211      0.061492
    12.054797      0.061492      3.158345
```

5.14 Can I get the covariance matrix of the estimates?

Yes, indeed. this is done by the VCM (standing for Variance Covariance Matrix). This is how it is done:

```
output
  vcm='varcov.vcm';
end
```

5.15 Can VCE assist me in passing the English test?

For once, the answer is a clear NO.

5.16 Constraints and Restrictions

- ▷ file names are limited to 132 character
- ▷ trait and effect names are limited to 30 characters
- ▷ the effect animal should not be the first effect in the model.
- ▷ the effect animal should be written in the model before maternal or paternal effects.

- ▷ Use effect name only once in the model. Try to combine as in 3.2 or rename them to avoid conflicts.
- ▷ **In MULTI statements defined sub-traits are assumed to have the same coefficients (limitation in DefineCoeff)**
- ▷ **Effects in EQUATE statements must be described with the same function. They can differ only in a constant coefficient.**

6 Changes for Version 6.0

While not many new features have been implemented, this version is still considered a major release that should replace all preceding version because of the large number of bugs fixed. This is indicated by moving from version 5 to version 6.

1. Now the form='full' needs to be replaced by mform='full' as the old version clashed with the format='(f12.5)' keyword.
2. In previous versions standard errors are incorrect and therefore older versions should get replaced.
3. Under certain circumstances maternal effects produced wrong results.
4. Now different number of traits can used for direct and maternal effects.
5. Also 64 bit binaries are available to access larger memory above 3 GB.

Bibliography

[1] L. A. García-Cortés, M. Rico, and E. Groeneveld. Using coupling with the Gibbs sampler to access convergence in animal models. *Journal of Anim. Sci.*, 76(2):441–447, February 1998.

[2] C. J. Geyer. Practical Markov chain Monte Carlo. *Statistical Science*, 4:473–483, 1992.

[3] I. Hoeschele and P. M. VanRaden. Rapid inversion of dominance relationship matrices for noninbred populations by including sire by dam subclass effects. *Journal of Dairy Science*, 74:557–569, 1991.

[4] V. E. Johnson. Studying convergence of Markov chain Monte Carlo algorithms using coupled sample paths. *J. Am. Stat. Assoc.*, 91:154–166, 1996.

[5] M. Kovač and E. Groeneveld. Local maxima in multiple trait variance component estimation. In *American Dairy Sci. Association and American Society of Animal Sci. Combined Annual Meeting, Teaming Up for Animal Agric., Abstracts, Lexington, Kentucky*, page 33, July 31 - August 4 1989.

[6] Norbert Mielenz, Milena Kovač, Eildert Groeneveld, Rudolf Preisinger, Mathias Schmutz, and Lutz Schüler. Genetische Parameter für Merkmale der Eiproduktion geschätzt mit additiven und Dominanzmodellen bei Legehennen. *Arch. Tierz., Dummerstorf*, 46(1):77–84, 2003. ISSN 0003-9438/46/1.

[7] A. Neumaier and E. Groeneveld. Restricted Maximum Likelihood Estimation of Covariances in Sparse Linear Models . *Genet. Sel. Evol.*, 1(30):3–26, 1998.

[8] C. S. Wang, J. J. Rutledge, and D. Gianola. Marginal inference about variance components in a mixed model using Gibbs sampling. *Gen. Sel. Evol.*, (25):41–62, 1993.

Index

/cf, 44
32 bit, 113
64 bit, 113

additive genetic effect, 56
address space, 113
AMD, 113
Animal Model, 41
animal model, 76
APIIS, 79, 108

beef, 97
binaries, 14
binary log file, 115
burn_first, 59
burnin, 26
burn_max, 59
burn_next, 59
burn_stop, 59

chicken, 102
coding, 76, 77, 87, 98, 106
coefficient, 40
coefficient matrix, 61
comment line, 33
comment section, 31
commercial use, 14
constant coefficients, 41
constants, 31
convergence, 45
coupling, 13
covariable, 60
COVARIANCE, 108
Covariance, 50
covariance function, 39, 44, 47
covariance section, 47
covariates, 42, 44
covfile output, 63

data coding, 36, 83, 87
data preparation, 77, 106
data section, 33, 35
datfile, 33, 48
default value, 18
degree o fill
    70%, 112
    IA, 112
degree of fill, 111, 112
    hit rate, 112
delimiter, 30
dependent variable, 36, 40, 44, 60
dependent variables, 49
DFG, 14
dir, 61
direct additive genetic effect, 38
documentation, 14, 16
Dominance, 64, 105
dominance, 33, 38, 63, 100, 102
dominance variance, 102, 103
dump_asc, 48
dump_bin, 48

effect, 40, 60
English test, 116
equate, 39, 40, 47
expression, 31

families, 105
FAMILY, 102
family, 63
family effect, 102, 105
FAQ, 21, 111
field test, 80, 106
file name, 35

form, 50
format, 33, 35, 63, 102
FORTRAN, 42, 102, 108
ftp server, 14
functions, 31

gen_ped.f90, 108
genetic groups, 59, 78
genped.f90, 79
Gibbs, 25
Gibbs sampling, 13
Gibbs_log, 64
gnuplot, 27
group_by, 39
GROUPS, 78

hamsters, 84
header, 36
help, 17, 30
hen, 102
heritability curve, 87
Hessian, 92
home page, 14
HYS, 75

IML, 89, 95
Inbreeding, 64
inbreeding, 13, 63
inbreeding coefficient, 103
indep, 36
independent, 36
independent variable, 33, 36, 37, 40, 44, 60
individual dominance, 105
INT, 41
Intel, 114
intercept, 41
ioc, 61
iod_first, 59
iod_max, 59
iod_next, 59
iod_stop, 59

jobname.cov-bin, 50

keyword, 30
keywords, 18

laying hens, 102
Legendre, 89, 94
LENG, 99
likelihood, 92
likelihood ratio test, 92, 114
link, 33, 38, 50
Linux, 113
litter, 108
Litter Effect, 76
litter effect, 76
log_gibbs, 63
log_gibbs, 26
LRT, 92

makefile, 13
manual, 21
mark_first, 59
mark_max, 59
mark_next, 59
mark_stop, 59
maternal effect, 83, 86
maternal effects, 38, 82
mates, 105
maxiter, 50
Mem_map, 64
memory, 112
Mendelian sampling, 105
method, 60
mform, 63
missing values, 74
missing_value, 60
MODEL, 108
model, 39
model section, 39
MonteCarlo EM, 13
multi, 39
multivariate, 79, 103

nesting, 84
non-additive, 25
non-zero elements, 60
numerical problems, 45

operators, 31
option, 30
orthogonal, 87

parameter file, 17, 29
paternal additive genetic effect, 38
pedfile, 33, 36
pedigree, 80
pedigree file, 78
PERL, 108
permanent environment, 86
PEST, 36, 64, 68, 77, 78, 83, 87, 107
platform
    IBM AIX, 114
    Intel i686, 114
    Intel x84_64, 114
    Mac OS, 114
platforms, 114
polynomial, 87, 89
polynomial regression, 45, 46
prediction, 105

RAM, 112
random effects, 48
random regression, 25, 39, 54, 84, 97
random slope, 85
ranfile, 33
reference manual, 16
regression, 42, 84
regression coefficient, 100
regression functions, 42
RELATIONSHIP, 78
REML, 13
reparameterize, 60
reprint, 63, 64
residual, 49
residual covariance matrix, 53
residual disconnected, 106
residual variance, 100
residual(datfile), 108
restart, 50, 59
restrictions, 116
results, 45

SAS, 89, 95, 108
scale, 39
scalex, 39
scaley, 39
scaling, 42, 45
section, 18, 29
set, 39, 44, 47
sire model, 76, 78
skip_value, 60
SMP, 61
solve, 60
solving strategies, 23
sparse inverse, 13
standard error, 91, 114
standard errors, 13, 92
start_asc, 48, 50
start_bin, 48, 50
starting VCE, 16
statement, 30
station test, 80, 106
status, 111
    random regression, 111
stopping criterion, 111
SYSTEM, 112
system section, 57

test data, 14, 15
tolerance, 60
total memory, 60
trait, 40, 60
type, 50

UNCMIN, 13
unpack, 15
use case, 21
user interface, 16
Users guide, 16

variable, 30, 36
variance functions, 89
VCM, 116
vcm, 64
verify, 16

Xeon, 113

ZERO, 112