MTDFREML Workshop Notes: January, 2002

(Original from Vicente Vega, 20 Aug 1998 – revised by Alex Olvido, 11 Jan 2003)

1. A little log-likelihood (logL) and mixed model equations (MME)
2. Programs, data, answer files
3. Convergence
4. Models
5. Compiling and recompiling "Param.dat"
6. Example of MTDFREML

Keith Boldman
Lisa Kriese
Curt Van Tassell
Steve Kachman
Joerg Dodenhoff

MTDFREML Package:	Estimate variances and covariances
	Predicts breeding values and SE of the predictions
	Estimates fixed effects and SE
	Find expected values of solutions
	Handle messy, unbalanced data (main advantage over least-squares methods)
	Analyze more than one trait (not same model for all traits)
	Good for GxE analysis (2 sexes)

Theory: Derivative-free REML
REML = Restricted (or residual) Maximum Likelihood

Likelihood divided in two parts: (1) fixed effects, e.g. age and sex, and (2) random effects, i.e. variances and covariances.

Unlike random effects, the likelihood part for fixed effects is not maximized. Hence, likelihood function cannot be used with LRT for fixed effects. Would need to do ML or hold VC constant. At convergence for VC, can do t-tests, F-tests.

Goal is to find variance-covariance estimates that maximize the likelihood, given the data. It is easier, instead, to maximize the log of the likelihood, or logL.

In practice, however, we minimize the quantity –2logL; we multiplied by –2 to get rid of the ½ of the likelihood function (see Boldman et al. 1995, p. 54).

-2logL = F value

“Simplex algorithm” looks for a minimum ==> method is a minimizer

“REML is REML”

Usual REML methods are iterative: REML is itself not a method, but a goal. The goal is to maximize the likelihood any way you can. Hence, many algorithms for REML:

(1) Expectation maximization (EM) ==> sum of squares (S.S.) and E(S.S.)
   - 1st derivative
   - never outside parameter space

(2) Derivative-free (DF)

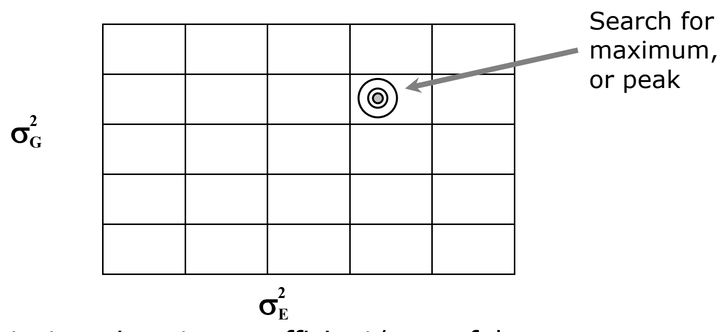
(3) Newton-Raphson
   - 1st and 2nd derivatives

(4) Average information (AI)

Except for DF, all of these methods need the inverse of the coefficient matrix ($\mathbf{C}^{-1}$). This requirement imposes a limit on what the program can do because required amount of time (T) scales to the cube of the number of elements (N) in $\mathbf{C}^{-1}$: $T = N^3$

Unlike other REML methods, derivative-free REML doesn’t need or use derivatives (hence, the name)

Instead, does something like a “grid search” (e.g. DF-REML) to find a maximum value for the likelihood function:



Unfortunately, this grid search strategy is not very efficient/powerful.

On the other hand, simplex algorithm (Nelder and Mead, 1965; see Boldman et al. [1995, Ch. 7]) makes DFREML more efficient:

- no derivatives
- no sum of squares (S.S.)
- no expected values of the sum of squares, E(S.S.)
- does <u>not</u> need inverse of the coefficient matrix

Simplex algorithm relies on a useful form of –2logL

General mixed model (MM):

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e} \quad ,$$

where **Y** is vector of (animal) records, **X** is association matrix for fixed effects, **ß** is vector of fixed effects, **Z** is association matrix for random effects, **u** is vector of random effects (e.g. genetic values), and **e** is vector of residuals, or unexplained variance.

1st moment ==> Expected value of **Y**, or E(**Y**), equals **Xß**

2nd moments ==> Variance of **u** , or Var(**u**), equals **G** Variance of **e** , or Var(**e**), equals **R**

In a very simple animal model,

$\mathbf{G} = \mathbf{A}\sigma^2_G$ ==> **A** is the relationship matrix for individuals in your dataset

$\mathbf{R} = \mathbf{I}\sigma^2_E$ ==> **I** is an identity matrix, i.e.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Henderson’s MME (almost completely general; see Boldman et al. [1995, Ch. 4]):

estimates fixed effects

$$\begin{bmatrix} \mathbf{X'R^{-1}X} & \mathbf{X'R^{-1}Z} \\ \mathbf{Z'R^{-1}X} & \mathbf{Z'R^{-1}Z+G^{-1}} \end{bmatrix} \begin{bmatrix} \hat{\beta} \\ \hat{u} \end{bmatrix} = \begin{bmatrix} \mathbf{X'R^{-1}Y} \\ \mathbf{Z'R^{-1}Y} \end{bmatrix}$$

predicts random effects

coefficient matrix, **C**

solution vector, **s**

r, or vector of right-hand sides

Harville (1977) and Searle (1979):

==> results

$$\underset{\substack{\uparrow \\ \text{likelihood,L, given the data, Y; often presented simply as }-2\text{logL; but it is} \\ \text{important to remember that likelihood estimates are only as good as your} \\ \text{data ("garbage in, garbage out")}}}{-2\log(L|Y)} = \text{some constant} + \underset{\substack{\uparrow \\ \text{read as "log of the} \\ \text{determinant of the} \\ \textbf{R}\text{ matrix"}}}{\log|\mathbf{R}|} + \log|\mathbf{G}| + \log|\mathbf{C}| + \underset{\substack{\uparrow \\ \text{residual sum of squares} \\ = (\mathbf{Y'Y} - \mathbf{s'r}) \\ \downarrow \\ \mathbf{Y'R^{-1}Y}\text{, in general}}}{\mathbf{Y'PY}}$$

i.e. everything is a function of the MME

Search to find **R** and **G** that minimizes –2logL

C changes when you change **R** and/or **G**; **X**, **Y**, and **Z** stay the same

(Recall that **Y** is vector of animal records, and **X** and **Z** are association matrices for fixed and random effects, respectively.)

In our simple animal model, we can find log determinants for **R** and **G**:

For log|**R**|: let $\mathbf{R} = \mathbf{I}\sigma^2_E$ so that $\log|\mathbf{I}\sigma^2_E| = \log \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \sigma^2_E = \log \begin{bmatrix} \sigma^2_E & 0 & 0 & 0 \\ 0 & \sigma^2_E & 0 & 0 \\ 0 & 0 & \sigma^2_E & 0 \\ 0 & 0 & 0 & \sigma^2_E \end{bmatrix}$

$= \log(\sigma^2_E)^n = n\log(\sigma^2_E)$

where n is simply the number of elements (or number of records) along the diagonal of the **R** matrix.

And for log|**G**|: let $\mathbf{G} = \mathbf{A}\sigma^2_G$ so that $\log|\mathbf{A}\sigma^2_G| = \log|\mathbf{AI}\sigma^2_G|$

$= \log|\mathbf{A}| + \log|\mathbf{I}\sigma^2_G| = \log|\mathbf{A}| + q\log(\sigma^2_G)$

where q is the order of the numerator relationship matrix, **A** (Boldman et al. 1995, p. 56). Note that log|**A**| is a constant because **A** is the (predefined) numerator relationship matrix and could be ignored.

Of course, the coefficient matrix, **C**, for the MME (see Henderson's MME, above), changes with different combinations of **R** and **G**. Note that to calculate log|**C**|, **C** has to be full rank, meaning that the dimensions of **C** equals the number of linearly independent rows and columns in that matrix, thus resulting in a non-zero determinant (Mrode 1996, p. 161). Because **C** is usually not of full rank, a constrained **C** (i.e. **C***) is used.

Lastly, the generalized residual sum of squares, **Y′PY**, in the simplest case is

$$\frac{1}{\sigma_E^2}\,\Sigma y_i^2 - \mathbf{s'r}$$

A brief history of MTDFREML (see also Boldman et al. [1995, pp. 55-58])

1. Smith and Graser (1986, J. Dairy Sci.)
   = idea for a derivative-free system
   = use Gaussian elimination to find solutions for log|**C**| and **Y′PY** (in Harville-Searle MME, above)
   = in Gaussian elimination, you go from this ==>
   $$\begin{array}{c|c} \mathbf{C} & \mathbf{R} \\ \hline \mathbf{R'} & \mathbf{Y'R^{-1}Y} \end{array}$$
   ... to this ... ==> $\mathbf{Y'PY} = \mathbf{Y'R^{-1}Y} - \mathbf{s'r}$
   ... by eliminating one equation at a time.
   = At the end of Gaussian elimination, you get the determinants that you need.
2. Graser et al. (1987, J. Anim. Sci.)
3. Meyer (1988, 1989, 1991)
   = developed DFREML, the first program to carry out Gaussian elimination, but very slowly
   = a short time later, Keith Boldman messed around with subroutines contained in SPARSPAK and got 100-600X the speed of Meyer's original DFREML program

The basic strategy behind Smith and Graser's DFREML (and, thus, Meyer's DFREML programs) is to try out different combinations of R_0 and G_0 until you get the minimum value for –2logL. In the simplest case, let $G_0 = \sigma_G^2$ and $R_0 = \sigma_E^2$.

For the simple example,

$$\mathbf{X'R^{-1}X} = \mathbf{X'I}\frac{1}{\sigma_E^2}\mathbf{X}$$

The coefficient matrix, **C**, would be

$$\begin{bmatrix} \mathbf{X'I}\frac{1}{\sigma_E^2}\mathbf{X} & \mathbf{X'I}\frac{1}{\sigma_E^2}\mathbf{X} \\ \mathbf{Z'I}\frac{1}{\sigma_E^2}\mathbf{X} & \mathbf{Z'I}\frac{1}{\sigma_E^2}\mathbf{Z} + \mathbf{A}^{-1}\frac{1}{\sigma_G^2} \end{bmatrix}$$

And the vector of right-hand sides, **r**, would be

$$\begin{bmatrix} \mathbf{X'I}\frac{1}{\sigma_E^2}\mathbf{Y} \\ \mathbf{Z'I}\frac{1}{\sigma_E^2}\mathbf{Y} \end{bmatrix}$$

In practice, MTDFREML uses the simplex algorithm as a search strategy

ADVANTAGES of MTDFREML:

1. relatively simple procedure

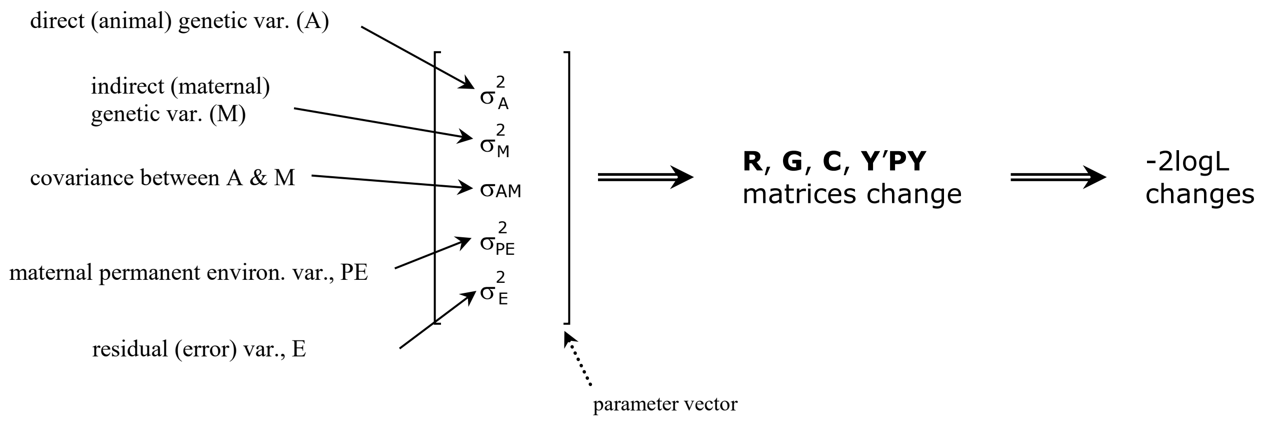
2. does not need sum of squares (S.S.)
3. does not need analysis of variance (ANOVA)
4. can use <u>sparse</u> matrix (i.e. non-zero elements of coefficient matrix) techniques; which is important for practical reasons:
   Suppose we have 1000 MME
   Then, # of coefficients = (1000 MME)^2 = 1,000,000 coefficients (in our square **C** matrix)
   Each coefficient takes up 8 bytes (double precision) of computer memory
   ==> 8,000,000 bytes = 8 megabytes (8 Mb) required computer memory

DISADVANTAGES of MTDFREML:

1. too easy; hence, no compelling need to find a better way
2. analyses with $\geq$2 traits take too long to reach convergence point
   = time to converge (Misztal) goes up by the 5th power of the # of traits, i.e. (# of traits)^5
3. doesn't give standard errors (SE) of variance-component point estimates
4. doesn't always give SE for h^2 and r_G estimates, especially in multiple-trait analyses
5. computations are just "black boxes"
   a. simplex algorithm
   b. SPARSPAK

About the (downhill) simplex algorithm (Nelder and Mead [1965])
= also known as the "polytope" method [Boldman et al. 1995, Ch. 7]) and "amoeba"[1] routine
= Numerical Recipes to search for minimum value of −2logL

Vector of parameters
= e.g. in a model with "maternal effect" variances:


direct (animal) genetic var. (A)
indirect (maternal) genetic var. (M)
covariance between A & M
maternal permanent environ. var., PE
residual (error) var., E
σ^2_A
σ^2_M
σ_{AM}
σ^2_{PE}
σ^2_E
R, **G**, **C**, **Y′PY** matrices change
-2logL changes
parameter vector

The simplex algorithm...

1. changes the values of the elements in the parameter vector according to certain rules
2. keeps parameters within "parameter space," e.g. $\sigma_{AM} \leq \sqrt{(\sigma^2_A \cdot \sigma^2_M)}$

[1] A type of microscopic, one-celled organism that continually changes its shape during locomotion and food acquisition.

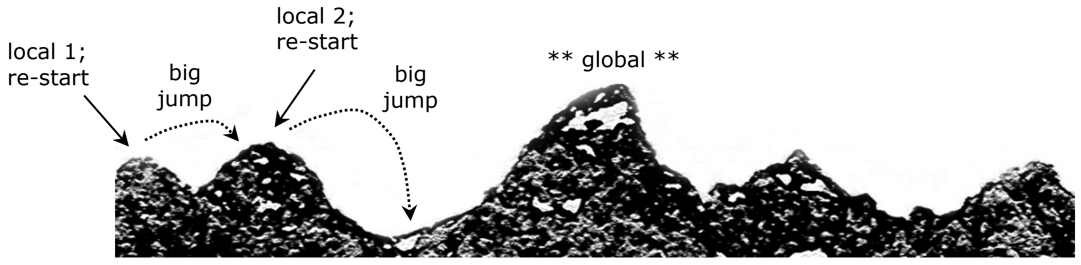
3. saves (p+1) sets of parameters (i.e. those in the parameter vector) and the value of –2logL in each run
4. always saves a set of (p+1) parameters including the best, where p is number of parameters (5 in the above maternal effects model)

Variance of the simplex, Var(-2logL), is used as the stopping criterion

When Var(-2logL) is small ==> convergence

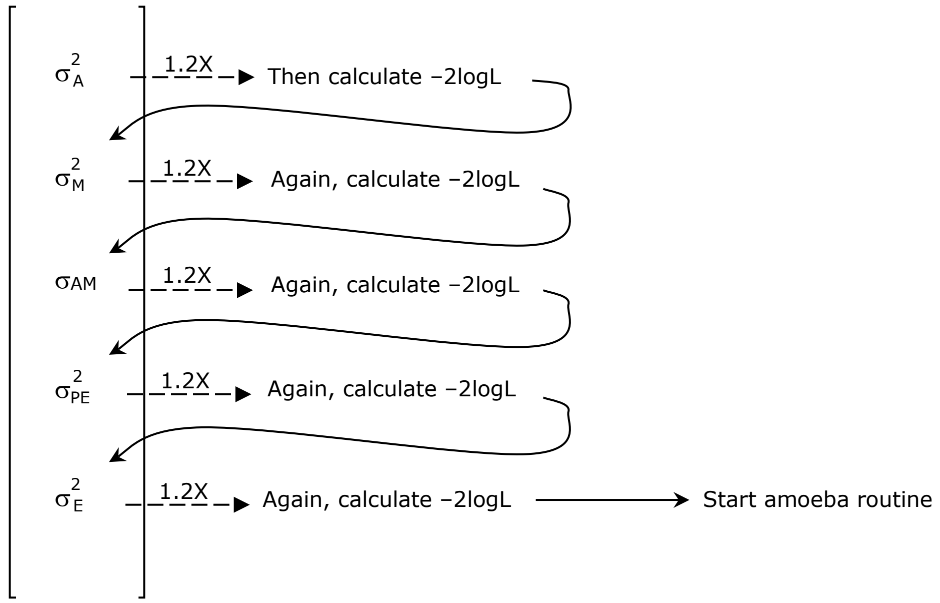
What is small? Typically, start with $1x10^{-6}$ as the initial convergence criterion; later, use $1x10^{-9}$

Local maximum versus global maximum: A mountain peak analogy



As the simplex moves towards a peak, successive steps get smaller. This is a problem if the peak is local, since our goal is to reach a global peak. Hence, we re-start the “amoeba” routine once the simplex has stopped at a local peak in order to make the simplex take “big jumps” that move it away from that local peak and (hopefully) towards the global peak—there is, by definition, only one global peak.

At start of each MTDFREML run, everything moves 1.2X (or 20% greater than) each element in the parameter vector:



SPARSPAK

= originally licensed by University of Waterloo; the version now used was purchased by USDA and is now license-free for MTDFREML

= FSPAK ==> not entirely public domain, but close

= stores sparse (non-zero) coefficients, i.e. the coefficient matrix, **C**, for the MME

= reorders the MME to speed up calculations, and remembers the reordering

= based on Choleski factorization (see Boldman et al. [1995, pp. 59-61])

$$\mathbf{C} = \mathbf{LL}' \qquad \mathbf{L} = \begin{bmatrix} \ell_{11} & 0 & 0 \\ \ell_{21} & \ell_{22} & 0 \\ \ell_{31} & \ell_{32} & \ell_{33} \end{bmatrix} \qquad \mathbf{L}' = \begin{bmatrix} \ell_{11} & \ell_{12} & \ell_{13} \\ 0 & \ell_{22} & \ell_{23} \\ 0 & 0 & \ell_{33} \end{bmatrix}$$

= from Choleski factorization, can get log|**C**| very easily...

$$\log|\mathbf{C}| = 2\log|\mathbf{LL}'| = \log|\mathbf{L}| + \log|\mathbf{L}'| = 2\Sigma\log(\ell_{ii})$$

= ...and **s** from **r** and **L**

$\mathbf{LL's} = \mathbf{r}$

let $\mathbf{u} = \mathbf{L's}$;

thus, $\mathbf{Lu} = \mathbf{r}$ ==> down-solve for **u**

and $\mathbf{L's} = \mathbf{u}$ ==> up-solve for **s**

= ...which provides complete solutions to MME of the form $\mathbf{Cs} = \mathbf{r}$

= factorization (**LL′**) is slow; but once **L** is found, this gives basis for finding

(1) SE of fixed effects ==> fast

(2) SEP of random effects ==> fast

(3) expectation of solutions ==> slow, because need to multiply by **C**

Implementation of MTDFREML: 3 programs (coded in Fortran77)

(1) MTDFNRM.FOR

= numerator relationship (=genetic association) matrix, **A**

= calculates elements of $\mathbf{A}^{-1}$ from a list of animals, sires, and dams via Henderson-Quaas rules

= needs a pedigree list

(2) MTDFPREP.FOR

= data-preparation program

= takes history file and converts it to a useful, more condensed file

= re-orders levels of factors to equation numbers

= handles missing traits in the $\mathbf{R}^{-1}$ matrix

(3) MTDFRUN.FOR

= major computing program for the simplex; uses several other subroutines:

a. MTDFLIK.FOR ==> subroutine to calculate the log-likelihood and other things

b. SPARSPAK.FOR ==> subroutine for sparse storage, reordering, solves

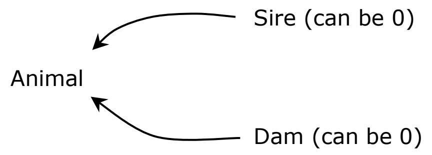
c. MTDFSUB.FOR ==> subroutines, e.g. to calculate eigenvalues and generalized inverse

d. MSTIME.FOR ==> timer subroutine

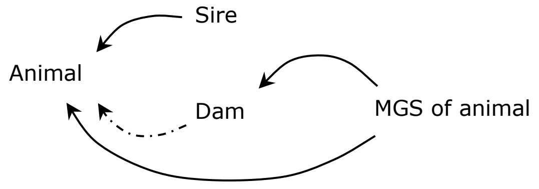
All three programs expect pedigree information (but can be unknown for all animals)

Henderson-Quaas ($\mathbf{A}^{-1}$) rules:

(1) Animal-sire-dam model



(2) Animal-sire-maternal grandsire (MGS)



(3) No relationship ($\mathbf{A}^{-1} = \mathbf{I}$)

set up your pedigree this way ==>	Animal	Sire	Dam
	69xxx	0	0

MTDFREML handles two types of data fields:

(1) fields of "integers," containing only integers

= integers in free (= ASCII) format (space or comma between the fields)

= no decimals, no alpha-numeric

= integer must be less than 2^{31}-1 (size of 4 byte integer)

(2) field of "reals," which can accomodate both integers and decimals

= traits and covariates

= missing trait value indicator ==> "0" or "-9"

= no missing value indicator for covariates (must check data and discard?)

MTDFNRM, the first program in MTDFREML, accepts only integer fields (i.e. animal, sire, and dam ID numbers); the next two MTDFREML program, MTDFPREP and MTDFRUN, both handle "reals."

A closer look at MTDFNRM

= converts original animal ID's to sequential ID's, e.g. 69xxx --> 1, 69xxx --> 2, etc.

= requires that parental ID numbers be less than animal (i.e. progeny) ID numbers

= produces output in several files:

(1) MTDF56 (ASCII file) ==> log file; shows no. of animals (unique ID numbers)

(2) MTDF44 (binary file) ==> elements of $\mathbf{A}^{-1}$ (not summed); used in MTDFRUN program

(3) MTDF11 (ASCII file) ==> matches original and reordered ID's; used in MTDFPREP program

(4) MTDF13 (ASCII file) ==> outputs inbreeding coefficients for animals, sires, and dams; optional

recoded ID			*original ID*			*F**		
animal	*sire*	*dam*	*animal*	*sire*	*dam*	*animal*	*sire*	*dam*

* merge with your data set to include inbreeding in the analysis;

= also, with Westell groups:

for crossbreeding experiments, put in ==> <u>animal</u> <u>sire breed</u> <u>dam breed</u>

keeps track of fraction by breed --> ** **

= input file could be a separate pedigree file or the actual data file.

= If using separate pedigree and data files, important that the ID numbers in both files are same; okay to have "extra" animals in the pedigree file

How MTDFNRM works:

Pass 1: stores all unique ID numbers; if animal (or sire or dam) ID repeated, stores only one set
sorts all unique ID numbers into an ordered vector of ID's

Pass 2: recodes animal, sire, and dam ID
stores all ID numbers into 3 separate vectors
then calculates $\mathbf{A}^{-1}$ elements and F; amount of time = (number of animals)^2

MTDFPREP in a nutshell

= prepares data and formulates model from history file

= reorders levels with 1, 2, 3,... to assign equation numbers

= matches animal original ID to recoded ID, e.g. 1, 2, 3,...

= deviates the observations and covariates from the raw trait means to reduce rounding errors in the computations

= two kinds of input files: (1) MTDF11 file from MTDFNRM and (2) data file

= output files:

(1) MTDF66 (ASCII file) ==> summary (or log) file; gives equation numbers

(2) MTDF50 (ASCII file) ==> set up equation numbers; used in MTDFRUN

(3) MTDF51 (binary file) ==> recoded data files; used in MTDFRUN

(4) MTDF52 (binary file) ==> recoded data files; used in MTDFRUN

(5) MTDF21 (ASCII file) ==> allows matching solutions with original levels of fixed effects; used in MTDFRUN

(6) MTDF22 (ASCII file) ==> optional; for matching original codes with solutions for uncorrelated effects; used in MTDFRUN

HINTS for MTDFPREP:

= MTDFPREP does not handle zero levels for fixed effects.

= we want sparseness; so if you can combine (fixed) factors, e.g. herd, year, sex, age of dam, etc., then do so beforehand; this way, you will maximize efficiency by creating a diagonal and sparse (**C**) matrix.

= also, don't use too many regression coefficients; sometimes better to break up the effect into separate classes.

How MTDFPREP works:

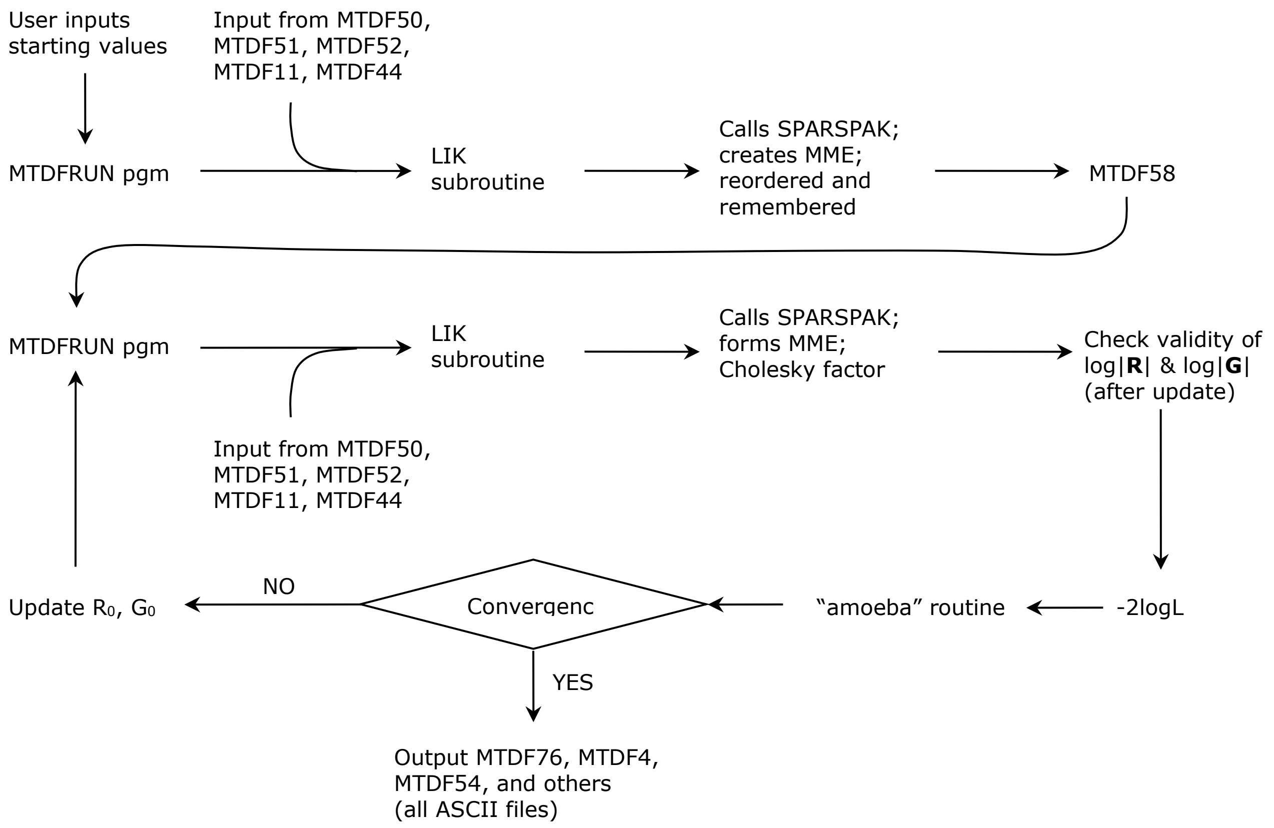
Pass 1: reads, then stores, levels for all factors

sorts levels from low to high

Pass 2: recodes levels

calculates deviations of observations and covariates from means to reduce rounding errors

MTDFRUN: This is where the heavy-duty number-crunching takes place


User inputs starting values
Input from MTDF50, MTDF51, MTDF52, MTDF11, MTDF44
MTDFRUN pgm
LIK subroutine
Calls SPARSPAK; creates MME; reordered and remembered
MTDF58
MTDFRUN pgm
LIK subroutine
Calls SPARSPAK; forms MME; Cholesky factor
Check validity of log|R| & log|G| (after update)
Input from MTDF50, MTDF51, MTDF52, MTDF11, MTDF44
Update R0, G0
NO
Convergenc
"amoeba" routine
-2logL
YES
Output MTDF76, MTDF4, MTDF54, and others (all ASCII files)

** VERY IMPORTANT**

= once you've reached initial convergence, re-start MTDFRUN with new solutions stored in MTDF4 (first, rename MTDF4 to "MTDF4.1;" later, you can use a DOS batch program to re-run MTDFRUN)

= compare values of –2logL from adjacent runs (-2logL = FVALUE); if no difference between current and previous run, then assume convergence at global maximum

= D.V.V. rule-of-thumb: (1) same –2logL value at 2nd decimal place AND (2) h^2 and r_G not changed at 2nd decimal place ==> probably reached global maximum

The "CONTINUE" option in MTDFRUN
= as if did not stop previous run
= uses MTDF54 (simplex takes smaller and smaller steps towards local or global maximum)
= also can be used for calculating SE, contrasts, PEV, or expected values, after convergence
 = use "continue" (not "start") option, i.e. OPTION 4, to automatically read MTDF54 file; then MTDFRUN will go to contrasts, etc; if run other program in between, then enter **G** and **R** priors by hand

Entering starting values in MTDFRUN
= program prompts you to enter priors, depending on which model you want to run
 (1) In model with 2 traits and maternal genetic component ==> 10 inputs:

$$\begin{array}{ccccc} & a1 & a2 & m1 & m2 \\ a1 & 1 & & & \\ a2 & 2 & 5 & & \\ m1 & 3 & 6 & 8 & \\ m2 & 4 & 7 & 9 & 10 \end{array}$$

 (2) In model with 2 traits only ==> 3 inputs:

$$\begin{array}{ccc} & a1 & a2 \\ a1 & 1 & \\ a2 & 2 & 3 \end{array}$$

 (3) In model with 1 trait and maternal genetic component ==> 3 inputs:

$$\begin{array}{ccc} & a1 & m1 \\ a1 & 1 & \\ m1 & 2 & 3 \end{array}$$

= cannot use "0" in variance position (numbers along the diagonal); zero okay in covariance (= off-diagonal) position, but will be constrained to zero.
= if re-entering incorrect values, just need to re-enter the ones that you want to correct
= other uncorrelated random effects:

field (dam)

$$\begin{array}{ccc} & T1C3 & T\boxed{2}C3 \\ T1C3 & 1 & \\ T2C3 & 2 & 3 \end{array}$$

	T1C3	T2C3	T1C4
T1C3	1		
T2C3	2	4	
T1C4	3	5	6

= in either case, covariances should be zero for row-column intersections with different field numbers (e.g. entries 3 and 5 in table above)

= convergence criterion ==> Var(-2logL)

= enter 1.D-6; works most of the time

= number of simplex rounds

= D.V.V. rule ==> enter "1" for the first run in order to get an idea of the program runtime

= when restart ==> about 2 likelihoods evaluated per round

= how many simplex rounds to reach convergence will depend on number of traits you analyze at once

= for σ_G^2 and σ_E^2 in single trait analysis ==> 30-40 rounds (maybe)

= for two-trait analyses with σ_M^2 and σ_{AM} ==> maybe something between 200 and 500 rounds

= starting values must be reasonable, e.g. phenotypic variance 2X as big as real value is unreasonable

= find reasonable variance by using MTDF66 output:

Raw total phenotypic variance, σ_Y^2 = $(SD)^2$

The real one will be smaller; therefore, try $0.6*(SD)^2$

For MTDFRUN input, force total of other variances to be $\sim 0.6*(SD)^2$

= find reasonable covariance from phenotypic SD's:

in theory, $$-1 \le r_{G1,G2} = \frac{\sigma_{G1,G2}}{\sigma_{G1} \cdot \sigma_{G2}} \le +1$$

so that $$\left|\sigma_{G1,G2}\right| \le \left|\sigma_{G1} \cdot \sigma_{G2}\right|$$

therefore, guess sign and (fractional) magnitude of r_G, e.g. 20% of $\sigma_{Y1,Y2}$, or $0.2*(SD_{Y1} \cdot SD_{Y2})$

= also, try checking for G_0 eigenvalues

If negative eigenvalues, then stops program ==> "Starting values out of bounds"

Models

= fixed factors – subclasses

= covariates (linear, quadratic, and/or cubic)

= covariates into classes

= random effects (= factors)

Animal models

(1) Animal genetic only (simplest): genetic random effects

= make sure **A** not equal to $\underbrace{\mathbf{I}}_{a\,+\,e}$

(2) Dairy example ($\mathbf{I}\sigma^2_{PE}$ and $\mathbf{I}\sigma^2_{TE}$ uncorrelated): $\underbrace{a}_{\sigma^2_G} + \underbrace{pe}_{\sigma^2_{PE}} + \underbrace{te}_{\sigma^2_{TE}}$ (repeated lactations)

= repeatability, r, is $(\sigma^2_G + \sigma^2_{PE})/\sigma^2_P$

(3) Beef example: $\underbrace{a_i + m_j}_{\text{genetic components, } \sigma^2_A,\ \sigma^2_M,\ \sigma_{A,M}} + pe_j + te_i$

= must have a dam ID; if not known, then assign a unique ID to each unknown dam

= genetic covariance includes a covariance between a_i and m_i, i.e. σ_{AM}

(4) Swine example: $\underbrace{a_i}_{\text{pig}} + \underbrace{m_j + pe_j}_{\text{sow}} + \underbrace{\text{litter}}_{\text{animals born at the same time}} + te_i$

Sire models

(1) Sire only model

= okay if **A** equals **I**

= could use **A** for all animals

= could use **A** from males only

(2) Sire and dam model

= sire as 1st genetic effect (a), dam as 2nd genetic (m), and maternal permanent environmental effect (pe)

= alternatively, sire as genetic effect (a), and dam as uncorrelated random effect (m+pe)

(3) Sire and grand-maternal sire (MGS) model

= sire as 1st genetic, MGS as 2nd genetic, and dam (with MGS records) as uncorrelated random effect

= could use **A** via males (sire-MGS rules)

No genetic effects

= MTDFREML program expects **A**, but could be **A** = **I**

= you should designate a random factor to be the animal, e.g. herd instead of animal

in the pedigree file ==> $\underbrace{\underline{\text{Animal}}}_{\text{herd number}}\quad \underbrace{\underline{\text{Sire}}\quad \underline{\text{Dam}}}_{\text{set to zero}}$

No random factors (i.e. only σ^2_E)

= you could designate one factor as random and make up a dummy pedigree

= then, in MTDFRUN, indicate "0.000001" as starting value (hold constant) for the variance of the "fixed factor" that was designated as the random factor (makes that factor almost fixed)

Genotype-by-environment interaction (GxE)

= e.g. interaction between lamb weight and age of dam (1, 2, 3) <== fixed effects for each age

= create a different field for each age

integer field — field of reals

AGE OF DAM	AGE1	AGE2	AGE3
1	40	0	0
2	0	42	0
3	0	0	39

********************* ADDITIONAL WORKSHOP NOTES **************************

Comparing different models: Likelihood Ratio Test (LRT)

$$LRT = 2\log L_{FULL} - 2\log L_{REDUCED}$$

= LRT approximately follows χ^2 distribution with degrees of freedom (df) = $\#VC_{FULL} - \#VC_{REDUCED}$

= applies only for models with the same fixed effects

= typical full model, for example, has 5 variance components (σ^2_A, σ^2_M, σ_{AM}, σ^2_{PE}, σ^2_E)

= a reduced version might have only 3 or 4 of those variance components, e.g. σ^2_A, σ^2_M, and σ^2_E