A Parallel Solver for Multi-Trait Animal Models

Per Madsen[1] and Martin Larsen [2]

[1]*Danish Institute of Agricultural Sciences, Research Centre Foulum, P. O. Box 50, DK-8830 Tjele, Denmark*
[2]*UNIC Danish Computing Centre for Research and Education, DTU, Bld. 304, DK-2800 Lyngby, Denmark*

Introduction

The use of Animal Models for prediction of breeding values, have led to the need for solving large systems of simultaneous equations. This is especially true in multi-variate models, where the number of equations to be solved is proportional to the number of animals times the number of traits. Even though the Mixed Model Equation (MME) for an Animal Model is very sparse, it is to large to build and solve directly even in sparse format. Therefore an iterative solver has to be used. Schaeffer and Kennedy (1986) described an iterative procedure for solving MME without constructing the equations explicitly. The procedure is known as AIteration on DataA.

Changing from single- to multi-trait evaluations and to more complex operational models increases both the size and the complexity of the MME. At the same time more frequent evaluations are wanted. One way to overcome this computational challenge is to distribute the computations on several computers i. e. parallel computations.

This paper describe a parallel implementation of AIteration on DataA for a distributed memory architecture computer. The development of the parallel solver started as a joint project between DIAS and UNIC in 1995, and is now supported by EU as a High Performance Computer Network (HPCN) Technology Transfer Node (TTN) project, under the title Continuos Estimation of Breeding values Using Supercomputers (CEBUS).

Model

Let the general multivariate linear mixed model be:

$$y = X_1\beta_1 + X_2\beta_2 + \sum_{i=1}^{r} Z_i u_i + Z_a a + e$$

where

y is a vector of observations on t traits,
β_1 and β_2 are vectors of fixed effects
u_i , i=1, 2,Y,r are vectors of random effects
a is a vector of random additive genetic effects and
e is a vector of random residuals.
$\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{Z}_i$,i=1, 2,Y, r, and $\mathbf{Z}_a$ are known incidence matrices.

Assumptions on (co)variances are:

V[u_i]	= $\mathbf{G}_i$	= $\mathbf{G}_{0i}$ q **I**, i = 1, 2,Y,r,
V[a]	= $\mathbf{G}_a$	= $\mathbf{G}_{0a}$ q **A,**
V[e]	= **R**	= $\mathbf{R}_0$ q **I,**
Cov[u_i, u_j]		= 0 , if i ¥ j,
Cov[u_i, a]		= 0 , ¼ i,
Cov[u_i, e]		= 0 , ¼ i and
Cov[a, e]		= 0 , ¼ i

where

$\mathbf{G}_{0i,}$ i=1, 2,Y,r are (co)variance matrices for the traits influenced by the i-th of the r random effects other than animal
$\mathbf{G}_{0a}$ is the additive genetic (co)variance matrix for the t traits
$\mathbf{R}_0$ is the residual (co)variance matrix for the t traits and
A is the additive relationship matrix.

Let: $\mathbf{G} = \sum_{i=1}^{r} + G_{i'}$

$\mathbf{Z} = [\mathbf{Z_1}\ \mathbf{M}\ \ \mathbf{Z_2}\ \mathbf{M}\ \mathbf{Y}\ \mathbf{M}\ \ \mathbf{Z_r}]$,

$u = [u_1\ \mathbf{M}\ u_2\ \mathbf{M}\ \mathbf{Y}\ \mathbf{M}\ \ u_r]$

Then the MME is:

$$\begin{pmatrix} \mathbf{X}'\mathbf{N}_1\mathbf{R}^{-1}\mathbf{X}_1 & \mathbf{X}'\mathbf{N}_1\mathbf{R}^{-1}\mathbf{X}_2 & \mathbf{X}_1'\mathbf{N}\mathbf{R}^{-1}\mathbf{Z} & \mathbf{X}'\mathbf{N}_1\mathbf{R}^{-1}\mathbf{Z}_a \\ \mathbf{X}'\mathbf{N}_2\mathbf{R}^{-1}\mathbf{X}_1 & \mathbf{X}'\mathbf{N}_2\mathbf{R}^{-1}\mathbf{X}_2 & \mathbf{X}_2'\mathbf{N}\mathbf{R}^{-1}\mathbf{Z} & \mathbf{X}'\mathbf{N}_2\mathbf{R}^{-1}\mathbf{Z}_a \\ \mathbf{Z}'\mathbf{N}\mathbf{R}^{-1}\mathbf{X}_1 & \mathbf{Z}'\mathbf{N}\mathbf{R}^{-1}\mathbf{X}_2 & \mathbf{Z}'\mathbf{N}\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} & \mathbf{Z}'\mathbf{N}\mathbf{R}^{-1}\mathbf{Z}_a \\ \mathbf{Z}_a'\mathbf{N}\mathbf{R}^{-1}\mathbf{X}_1 & \mathbf{Z}_a'\mathbf{N}\mathbf{R}^{-1}\mathbf{X}_2 & \mathbf{Z}_a'\mathbf{N}\mathbf{R}^{-1}\mathbf{Z} & \mathbf{Z}_a'\mathbf{N}\mathbf{R}^{-1}\mathbf{Z}_a + \mathbf{G}_a^{-1} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ u \\ a \end{pmatrix} = \begin{pmatrix} \mathbf{X}'\mathbf{N}_1\mathbf{R}^{-1}y \\ \mathbf{X}'\mathbf{N}_2\mathbf{R}^{-1}y \\ \mathbf{Z}'\mathbf{N}\mathbf{R}^{-1}y \\ \mathbf{Z}_a'\mathbf{N}\mathbf{R}^{-1}y \end{pmatrix} \quad (1)$$

Solving strategies

The MME in (1) can be solved by Gauss-Seidel (GS), Jacobi or a combination of the two methods. The differences between GS and Jacobi iteration can be illustrated as follows: Rewrite (1) as: $\mathbf{C}x = b$. Decompose $\mathbf{C}$ as $\mathbf{C} = \mathbf{L} + \mathbf{D} + \mathbf{U}$, where $\mathbf{L}$ is lower and $\mathbf{U}$ is upper triangular and $\mathbf{D}$ is diagonal. The Gauss-Seidel iteration is:

$$\mathbf{D}x^{k+1} = (-\mathbf{L}x^{k+1} - \mathbf{U}x^{k} + b) = crhs$$

$$\Upsilon\, x^{k+1} = \mathbf{D}^{-1}\, crhs \quad (2)$$

where

x^{k+1} denotes the $(k+1)^{th}$ iterate to the solution vector, and
crhs is corrected right hand side vector

Then the first order Jacobi method is:

$$\mathbf{D}x^{k+1} = -(\mathbf{L} + \mathbf{U})x^{k} + b = crhs$$

$$\Upsilon\, x^{k+1} = \mathbf{D}^{-1}\, crhs \quad (3)$$

In Jacobi iteration, only solutions from the previous round of iteration are used, while the GS method always use the most recent updates of the solutions. Jacobi iteration is known to have poor convergence rate on equation systems like the MME. Extending the method to second-order Jacobi improves rate of convergence (Misztal & Gianola, 1987).

The extension of (3) to second-order Jacobi iteration is:

$$x^{k+1} = \mathbf{D}^{-1}\, crhs + \alpha(x^{k} - x^{k-1}) \quad (4)$$

where α is a relaxation factor.

Both GS and Jacobi iteration can be used in a blocked form. $\mathbf{D}$ then contains squared blocks, and elements of x and *crhs* are grouped in subspaces of dimensions corresponding to the blocks in $\mathbf{D}$. Solving (1) by a combination of blocked GS and Jacobi iteration goes as follows:

The effects in β_1 is chosen as the one with the largest number of levels (typically herd-year-season or management group effect) and solutions are obtained by the GS method while the effects in β_2, u and a are solved by the second- order Jacobi method. The blocks used in the iterative solver are defined by level codes for β_1, u and a. For example, all equations for additive genetic effects for one animal are treated as a block and solved simultaneously. Equations corresponding to β_2 are treated as a single block. To save computations during the iteration, the diagonal blocks can be accumulated during an initial pass through data, inverted and stored in core or on disk.

The explicit construction of the MME can be avoided by processing the data in level code sequence of the effect in β_1. When all data belonging to one level code of β_1 has been processed, there are no additional contributions to the *crhs* elements for that group of equations, and updated solutions can be obtained by expression (2). Contributions to *crhs* for the remaining part of the MME can then be calculated using the updated solutions for β_1 and the values for β_2, u and a from the previous round of iteration. When all data has been processed the complete β_1 vector has been updated, and all contributions from data to the elements in *crhs* corresponding to equations for β_2, u and a have been calculated. Correction due to the relationship matrix for the part of *crhs* corresponding to animal equations (a) are then calculated. Finally updated solutions for β_2, u and a are obtained by expression (4).

The solving strategy described above has been implemented in the DMU5 module of the DMU package of Jensen and Madsen (1994), and the program layout is illustrated in the left part of Figure 1.

Parallelization

The iterative algorithm outlined in the previous section is B at least in principle- very well suited for parallelization on a distributed memory architecture computer.

A prototype of a parallel solver based on the Parallel Virtual Machine (PVM) system and a master-slave concept has been developed. In an initialisation round, the master reads the data file, and distributes it among all the processors. In order to simplify matters and minimise communication, all data from one level of β_1 must reside on one and only one processor. In this initial round, diagonal blocks needed on each processor are also calculated, inverted and stored.

The iterative solver is then as follows: Each processor carries out (for any k until a stopping criteria becomes true), the $(k+1)^{th}$ round of the Gauss-Seidel iteration on its part of the data i. e. solve for its part of β_1. At the same time it builds up contributions to *crhs* from its part of data in a local vector. When all processors have processed their data, the master processor calculate contributions to *crhs* for animal equations (*a*) due to the relationship matrix. Contributions to *crhs* for β_2, *u* and *a* are then summed over all processors in a global reduce operation. Each processor has now its own copy of global *crhs* for β_2, *u* and *a*. The master processor solves for the $(k+1)^{th}$ iterate of β_2 and the slaves solve for the $(k+1)^{th}$ iterate for their part of the vectors *u* and *a*. Each processor then sends its part of the solution vector to all the other processors and receives solutions calculated on the other processors.

The described parallel implementation is outlined in the right part of Figure 1.

Test runs and comments about parallelization from test runs and comments about parallelization

The prototype of the parallel program has been tested on UNIC-s IBM SP computer. The data used consists of protein yield for all recorded dairy cows in Denmark. The operational model used was the Danish Multi-Breed Animal Model (Madsen *et al.,* 1997). The size of the MME was app. 11.000.000, and the speedup in wall clock time when running on 16 processors each with 256 MB memory was app. 2 compared to a single processor run on a processor with 1 GB memory. The main reasons for the low speedup was:

1. The working vectors for current and previous solutions and *crhs* could not be in memory on the processors in the parallel run, which lead to paging.
2. The global reduce operation used to sum contributions to *crhs* across processors was implemented as a recursive doubling. When executing on 16 processors this means, that each processor sends and receives a vector of length corresponding to size of the equation system solved by the Jacobi method four times in each round of iteration. In the test run this lead to a total amount of communication between the processors of 5.3 GB in each round of iteration.

In the last months of 1997 work has been done on reducing the memory requirement on each processor by only storing the part of the solution vector from previous round, that is needed on the individual processor. At the moment work on reducing the communication in the global reduce of *crhs* is going on. One possible way of doing that is to let each slave processor send its non-zero contributions to *crhs* to the master. The master receive contributions, calculate the global sum and sends to each slave processor the part of *crhs* needed on the individual slave processor.

References

Jensen, J. & Madsen, P. 1994. DMU: A package for the analysis of multivariate mixed models. *Proc. of 5th World Congress on Genetics Applied to Livestock Production 22*, 45-46.

Madsen, P., Jensen, J. & Nielsen, U.S. 1997. Effect of Heterosis and Imported Germ Plasm on Production Traits Estimated in the Danish Multi-breed Animal Model. *Proc. 1997 INTERBULL Meeting Vienna, 16*, 85-88.

Misztal, I. & Gianola, D. 1987. Indirect Solution of Mixed Model Equations. *J. Dairy Sci. 70*, 716-723.

Schaeffer, L.R. & Kennedy, B.W. 1986. Computing Solutions to Mixed Model Equations. *Proc. of 3rd World Congress on Genetics Applied to Livestock Production 12*, 382-393.

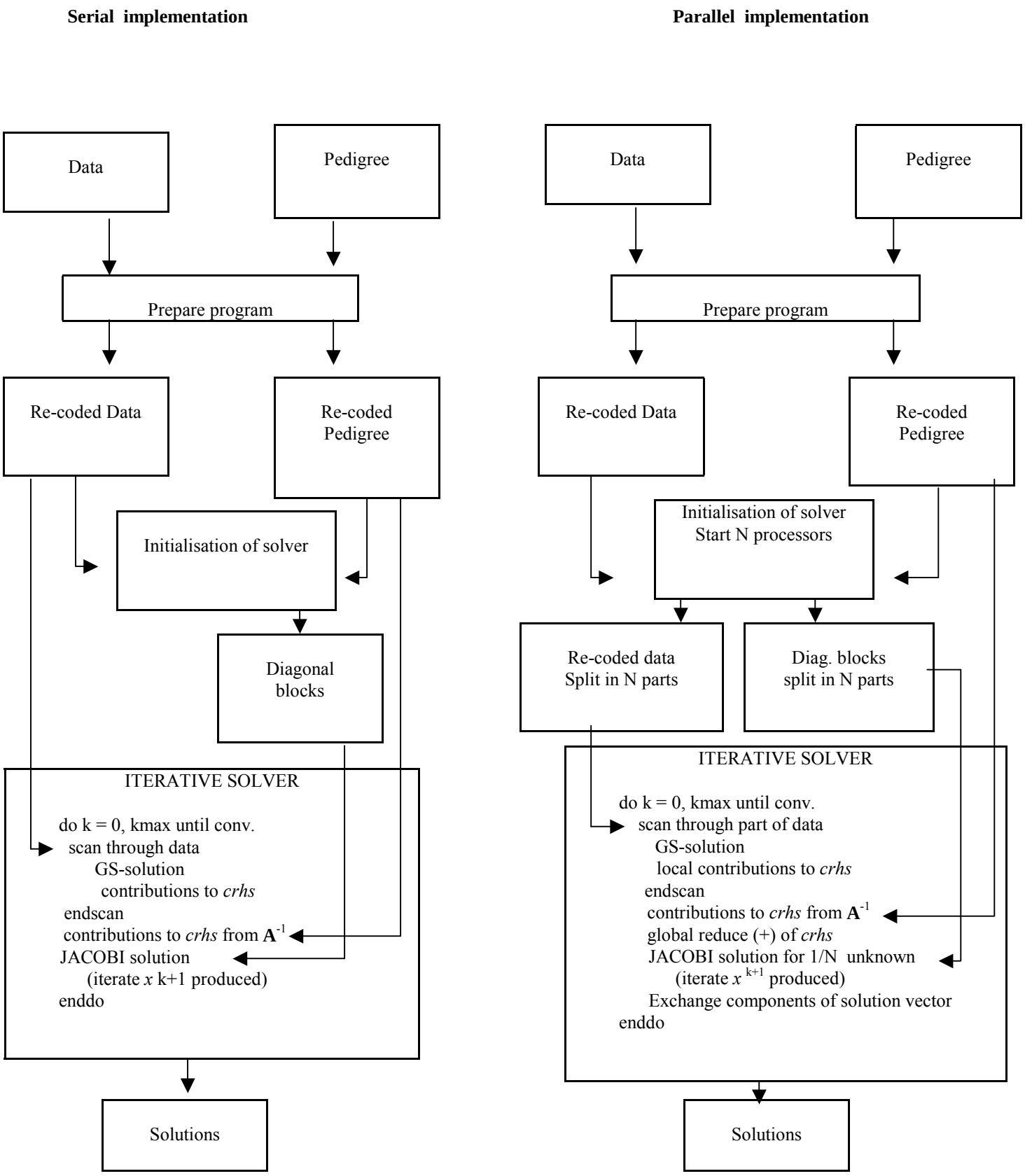

Figure 1. Program layout for a serial (left) and a parallel (right) implementation of iteration on data.