Note: The following paper was written in 1989. Only small corrections or editorial changes were made in 2018. It provides a negative result in the prior attempt to apply the QR algorithm to the derivative-free REML calculations that are typical of animal breeding, and for that reason alone it could not be published. However, it is very helpful for those that seek to find an improved QR algorithm that does work well, because it strongly hints at what might work by first trying things that don't work well. What possibly works well comes with the attached discussion written entirely in 2018. All footnotes written for the 1989 paper were made in 2018.

Running Head: QR Algorithm and Absorption

Efficient Implementation of Gaussian Elimination in Derivative-Free REML, or How not to Apply the QR Algorithm

by

S.P. Smith, K. Meyer and B. Tier

The Animal Genetic and Breeding Unit
University of New England
Armidale, NSW 2351, Australia

SUMMARY

A QR algorithm was designed using sparse matrix techniques for likelihood evaluation in REML. The efficiency of the algorithm depends on how the order of columns in the mixed model array are arranged. Three heuristic orderings were considered. The QR algorithm was tested successfully in likelihood evaluation, but vector processing was needed to finish the procedure because of excess fill-ins. The improvements made for the QR algorithm also applied to the competing absorption approach, and hence absorption was found to be more competitive than the QR algorithm in terms of computing time and memory requirements. Absorption was made 52 times faster than a first generation absorption algorithm.

Key Words: Absorption, Derivative-free, Gaussian elimination, QR algorithm, REML, Sparse matrices, Vector Processing.

I. INTRODUCTION

Derivative-free REML has proven very useful in estimating variance components in simple additive genetic models (GRASER, SMITH & TIER 1987). This methodology is still developing, and recently extensions have been made for more complicated models

(MEYER 1989; THOMPSON & JUGA 1989). MEYER (1989) recognized that derivative-free REML is still computationally demanding for some models, and states that there is considerable scope for improvement. For example, extending POWEL's (1964) derivative- free search to (co)variance estimation may prove very useful. Alternative methods to calculate the log-likelihood should also be considered, as we do in this paper.

HUDSON (1986) described the QR algorithm and its application to an animal model. Given realistic data structures, the QR algorithm would need to be implemented in a sparse matrix mode. HUDSON notes that the QR algorithm can be organized in a way that operations are performed only on non-zero elements. More detailed accounts can be found in BJÖRCK (1976) and GILL & MURRAY (1976).

HUDSON concentrated on the prediction of breeding values and the calculation of prediction error variance. SMITH (1987) noted that the QR algorithm can also be used to compute the log-likelihood in REML. In fact, applying the QR algorithm in likelihood evaluation is an easier exercise than HUDSON's implementation; solutions to the mixed model equations are not needed.

The purpose of this paper is to test the QR algorithm with actual data and to determine its suitability in likelihood evaluation. Comparisons are made with linked list absorption (TIER & SMITH 1989). Improvements to both approaches are identified.

II. THEORY

A. Model and Log-Likelihood

The general mixed model is given as

$$\mathbf{y}=\mathbf{X}\beta+\mathbf{Z}\mathbf{u}+\mathbf{e}$$

where: $\mathbf{y}$, β, $\mathbf{u}$ and $\mathbf{e}$ are vectors of observations, fixed effects, random effects and random residuals; $\mathbf{X}$ and $\mathbf{Z}$ are incidence matrices; and

$$E\{\mathbf{u}\}=\mathbf{0},\ E\{\mathbf{e}\}=\mathbf{0},$$
$$Var\{\mathbf{u}\}=\mathbf{G},\ Var\{\mathbf{e}\}=\mathbf{R},\ Cov\{\mathbf{u},\mathbf{e}^T\}=\mathbf{0}.$$

The array $\mathbf{0}$ is understood to be a null vector or matrix of appropriate order.

The log-likelihood that is to be maximized in REML is

$$L= -½\{\log|\mathbf{R}|+\log|\mathbf{G}|+\log|\mathbf{C}|+ \mathbf{y}^T\mathbf{P}\mathbf{y}\},$$

where $|\mathbf{C}|$ is the determinant of one the largest non-singular submatrices of

$$\mathbf{C} = \begin{bmatrix} \mathbf{X}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}^T\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix}$$

and $\mathbf{P}=\mathbf{R}^{-1}-\mathbf{R}^{-1}\mathbf{X}(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-}\mathbf{X}^T\mathbf{R}^{-1}$ where $\mathbf{V}=\mathbf{Z}\mathbf{G}\mathbf{Z}^T+\mathbf{R}$. As noted by TIER & SMITH (1989), $|\mathbf{C}|$ and $\mathbf{y}^T\mathbf{P}\mathbf{y}$ can be computed as by-products of linked list absorption, i.e., by Gaussian elimination. Terms $|\mathbf{R}|$ and $|\mathbf{G}|$ are easily calculated as well (MEYER 1989).

B. Generalized Determinants

To evaluate L we calculate determinants. Quadratic $\mathbf{y}^T\mathbf{P}\mathbf{y}$ is a function of determinants, $|\mathbf{M}|/|\mathbf{C}|$ where

$$\mathbf{M} = \begin{bmatrix} \mathbf{X}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}^T\mathbf{R}^{-1}\mathbf{R} & \mathbf{X}^T\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} & \mathbf{Z}^T\mathbf{R}^{-1}\mathbf{Y} \\ \mathbf{y}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{y}^T\mathbf{R}^{-1}\mathbf{Z} & \mathbf{y}^T\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

The absolute value of determinants, denoted $||\cdot||$, have interesting geometrical interpretations. Let $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ be three vectors in R^3, or 3 dimensional Euclidean space. Then $||\mathbf{W}||$, where $\mathbf{W}=\{w_1 , w_2 , w_3\}$, is the volume of the parallelepiped (or trapezoid) defined by $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ (STRANG 1976). The edges of this trapezoid are (**a**, **b**), (**b**, **c**) (**c**, **d**) where **a** is the orgin, $\mathbf{b}=\mathbf{w}_i$, $\mathbf{c}= \mathbf{w}_i+\mathbf{w}_j$, $\mathbf{d} = \mathbf{w}_i+\mathbf{w}_j+\mathbf{w}_k$ and (i,j,k) assume any of the 6 permutations of (1,2,3) (see Figure 1). Hence $||\mathbf{W}||$ is the volume of a three dimension trapezoid in three dimensional space. This interpretation automatically extends to n dimensional space. More importantly, this view extends to rectangular matrices to give the root of Gram determinant: the volume of an m dimensional parallelepiped imbedded in an n dimensional Euclidean space, denoted as $G(\mathbf{W})^{½}$ where the columns of $\mathbf{W}_{n\times m}$, $m \leq n$, define the parallelepiped. When **W** is square

$$G(\mathbf{W})^{½} = ||\mathbf{W}||,$$

but in general

$$G(\mathbf{W})^{½} = |\mathbf{W}^T\mathbf{W}|^{½}, \text{ or just } G(\mathbf{W}) = |\mathbf{W}^T\mathbf{W}|.$$

When **W** is a column vector the parallelepiped is a line segment and $G(\mathbf{W})^{½}$ is the length of the segment, i.e. the Euclidean measure of distance. When **W** has 2 columns $G(\mathbf{W})^{½}$ is the area of a two dimensional surface.

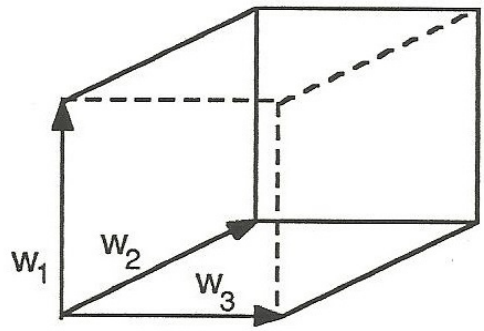

Figure 1. A Three dimensional Trapezoid in a three dimensional space. Vectors $\mathbf{w}_1$, $\mathbf{w}_2$, and $\mathbf{w}_3$ project out from the origin and define the edges of the trapezoid.

Following HUDSON[1], define the rectangular matrix **H** such that

$$\mathbf{H} = \begin{bmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{TZ} & \mathbf{TX} \end{bmatrix} \qquad (1)$$

where $\mathbf{L}^T\mathbf{L}=\mathbf{G}^{-1}$ and $\mathbf{T}^T\mathbf{T}=\mathbf{R}^{-1}$. By construction $\mathbf{H}^T\mathbf{H}=\mathbf{C}$. Defining $\hat{H}=\{\mathbf{H}, \hat{y}\}$ where $\hat{y}^T=\{\mathbf{0}^T, \mathbf{y}^T\mathbf{T}^T\}$ gives $\hat{H}^T\hat{H} =\mathbf{M}$. We have:

$$½\log|\mathbf{C}| = ½\log G(\mathbf{H})$$

$$\mathbf{y}^T\mathbf{P}\mathbf{y} = G(\hat{w}) = G(\hat{H}) /G(\mathbf{H})$$

where $\hat{w}$ is the projection of $\hat{y}$ in the null space of $\mathbf{H}^T$; i.e., $\hat{w}=\mathbf{M}_H\hat{y}$ where $\mathbf{M}_H=\mathbf{I}-\mathbf{H}(\mathbf{H}^T\mathbf{H})^-\mathbf{H}^T$.

The QR algorithm will be adapted for the computation of G(**H**) and $G(\hat{w})$.

C. The QR Algorithm

It is well known that any matrix **H** can be represented by the product $\mathbf{Q}\hat{\mathbf{U}}$ where **Q** is

[1] Hudson uses different notation.

orthogonal (i.e. $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$) and $\hat{\mathbf{U}}$ has the form

$$\hat{\mathbf{U}} = \begin{bmatrix} \mathbf{U} \\ \mathbf{0} \end{bmatrix} \quad ,$$

such that **U** is upper triangular and square (Press et al 1986). It follows immediately that

$$G(\mathbf{H})^{½} = G(\hat{\mathbf{U}})^{½} = ||\mathbf{U}||,$$

and $|\mathbf{U}|$ is simply a product of diagonal elements of **U**: i.e., $\prod_i u_{ii}$. If **H** does not have full column rank, **U** is singular and at least one of the diagonal elements will be zero. For derivative-free REML, we calculate the determinant of one of the largest non-singular submatrices of $\mathbf{C}=\mathbf{H}^T\mathbf{H}$. To find this submatrix we omit appropriate columns of **H**. Operationally, zero diagonals are skipped and the associated column of **H** is implicitly deleted. The log-determinant is calculated as $\Sigma_i \log|u_{ii}|$ where u_{ii} is never zero by implicit construction. To avoid overflow we never calculate the product, $\prod_i u_{ii}$, directly.

It is possible to evaluate $\mathbf{y}^T\mathbf{P}\mathbf{y}$ by computing $G(\mathbf{H})$ and $G(\hat{\mathbf{H}})$ and forming the ratio. However, for any square matrix **U** we have the following identity

$$G\left[\begin{pmatrix} \mathbf{U} & \mathbf{A} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}\right]^{\frac{1}{2}} = \|\mathbf{U}\| \times G(\mathbf{B})^{\frac{1}{2}}, \qquad (2)$$

the proof of which has two parts. First if **U** is singular, both sides of (2) are zero. For non-singular **U**,

$$\begin{aligned} G\left[\begin{pmatrix} \mathbf{U} & \mathbf{A} \\ \mathbf{0} & \mathbf{B} \end{pmatrix}\right]^{\frac{1}{2}} = \begin{vmatrix} \mathbf{U}^T\mathbf{U} & \mathbf{U}^T\mathbf{A} \\ \mathbf{A}^T\mathbf{U} & \mathbf{A}^T\mathbf{A} \end{vmatrix}^{\frac{1}{2}} &= \left|\mathbf{U}^T\mathbf{U}\right| \times \left|\mathbf{B}^T\mathbf{B} + \mathbf{A}^T\mathbf{A} - \mathbf{A}^T\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{A}\right|^{\frac{1}{2}} \\ &= \left|\mathbf{U}^T\mathbf{U}\right|^{\frac{1}{2}} \times \left|\mathbf{B}^T\mathbf{B}\right|^{\frac{1}{2}} \\ &= \|\mathbf{U}\| \times G(\mathbf{B})^{\frac{1}{2}} \end{aligned}$$

Hence, if $\mathbf{H}=\mathbf{Q}\hat{\mathbf{U}}$, then

$$\mathbf{Q}^T\hat{\mathbf{H}} = \begin{pmatrix} \mathbf{U} & \mathbf{y}_1 \\ \mathbf{0} & \mathbf{y}_2 \end{pmatrix} \quad \text{where} \quad \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{pmatrix} = \mathbf{Q}^T\hat{\mathbf{y}},$$

implying by (2) that $G(\hat{H}) = ||\mathbf{U}||^2 \times G(\mathbf{y}_2)$. The ratio is just $G(\mathbf{y}_2)=u_{nn}^2$. Note that $G(\hat{w})=G(\mathbf{y}_2)$, but that $\hat{w} \neq \mathbf{y}_2$ as the vectors $\hat{w}$ and $\mathbf{y}_2$ have different dimensions.

To calculate $\frac{1}{2}\log|\mathbf{C}|$ and $\mathbf{y}^T\mathbf{P}\mathbf{y}$, the QR algorithm is applied only to $\hat{H}$. If n is the number of columns of $\hat{H}$ after redundant columns have been deleted, then $\log|u_{ii}|$, i=1,2, ··· n-1, contribute additively to $\frac{1}{2}\log|\mathbf{C}|$ and $u_{nn}^2 = \mathbf{y}^T\mathbf{P}\mathbf{y}$.

D. Implementing the QR algorithm

Matrix **Q** can be a product of Householder transformations or Givens rotations. GILL & MURRAY (1976) describe sparse matrix implementation of the QR algorithm. For the present study we programed a version based on HUDSON's presentation of the Householder method.[2] A linked list similar to that of TIER & SMITH (1989) was used. Our strategy is outlined below.

Given a rectangular matrix $\mathbf{H}_1 = \hat{H}$, a Householder matrix, $\mathbf{Q}_1$ is found such that $\mathbf{Q}_1^T\mathbf{H}_1$ has only one non-zero element in the lead column. We can assume that the non-zero element is in the first row because $G(\cdot)$ is invariant to row permutations. Thus, the first row of $\mathbf{Q}_1^T\mathbf{H}_1$ equals the first row of $\hat{\mathbf{U}}$. The non-zero element being on the diagonal of **U** contributes to $||\mathbf{U}||$. By (2), the first row and column of $\mathbf{Q}_1^T\mathbf{H}_1$ contribute nothing further to the determinant. Hence the first row and column are removed and placed in the garbage collector, and $\mathbf{H}_2$ is formed. Matrix $\mathbf{H}_2$ has one less row and column than $\mathbf{H}_1$. Next a new Householder transformation $\mathbf{Q}_2$ is found such that $\mathbf{Q}_2^T\mathbf{H}_2$ has only one non-zero element in the lead column. With row permutations the non-zero element is moved to the first row. Thus, the first row of $\mathbf{Q}_2^T\mathbf{H}_2$ represents the second row of $\hat{\mathbf{U}}$. A matrix $\mathbf{H}_3$ is formed by removing the first row and column of $\mathbf{Q}_2^T\mathbf{H}_2$. This pattern is iterated until we have a vector $\mathbf{H}_n = \mathbf{y}_2$ (see Figure 2), and $\mathbf{Q}_n^T\mathbf{H}_n$ is zero for all its entries except for the first element that is positive.

Our procedure is designed for the calculation of the likelihood and it is not surprising that it differs with the method given in GILL & MURRAY. Their method is a general purpose algorithm designed for solving systems of linear equations. It requires saving the Householder transformations: components of matrices $\mathbf{Q}_i$. Columns of **U** are dumped to an exterior file as they are calculated and they are not saved in core. For our case, fill-ins are needed as we progress down the sequence $\mathbf{H}_i$, i=1, 2, ···, n. Moreover, we do not save $\mathbf{Q}_i$ and **U** as they are no longer required.

---

[2] Had more attention been given to Givens rotations the results found would have very likely been more positive.

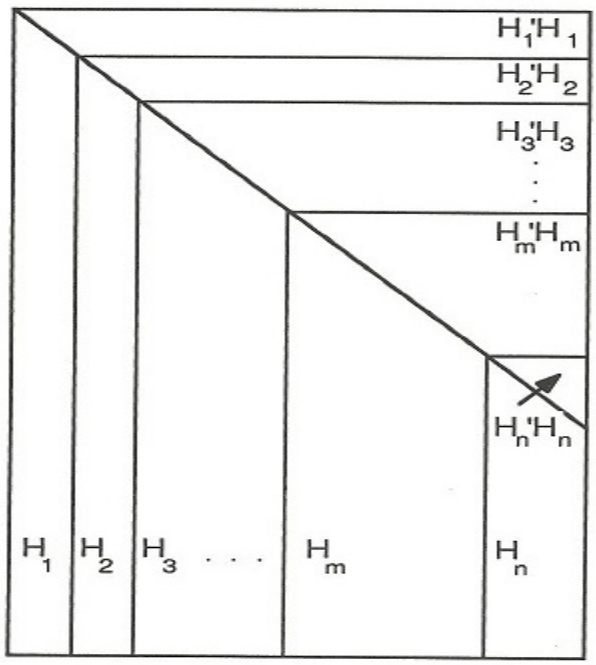

Figure 2. Diagrammatic representation of the QR algorithm showing a sequence of shrinking rectangles depicted by the arrays $\mathbf{H}_1$, $\mathbf{H}_2$, ···, $\mathbf{H}_n$. The sequence of triangles depicts half-stored matrices $\mathbf{H}_1'\mathbf{H}_1$, $\mathbf{H}_2'\mathbf{H}_2$, ···, $\mathbf{H}_n'\mathbf{H}_n$ which are the intermediate arrays in absorption.

E. Improvements

The efficiency of the QR algorithm is a function of the order columns are transformed. A desirable ordering is one where the sparsity is maintained. Initially our construction used an ordering determined as good prior to the first Householder transformation: we ordered the columns by the number of non-zero elements in each column. We were optimistic as the original algorithm was applied to a matrix with over 160000 columns. The procedure was very fast initially but slowed down after the first 80000 columns and became unusable. The QR algorithm was not working and modifications were clearly needed. This same problem was also observed for matrices of order 6000; the first 50 per cent of the columns were processed quickly but again the fill-ins became too numerous.

Our first observation was that the criterion for ordering was not unique and that

alternative methods could be used. Secondly, a desirable ordering can change after several columns have been processed. It is neccessary to find an ordering dynamically. Finding the best ordering is NP-complete (TARJAN, 1976) and hence we can only hope for an approximate solution that has been arrived at heuristically.

We studied three criteria for ordering columns dynamically. As indicated above, the first procedure is to select columns with least non-zero elements. This is referred to as the column-fill heuristic. The second procedure is to order columns on least fill-ins. The fill-ins resulting from processing a single column can be calculated and updated dynamically. Our third method is to order columns on least commonality: the commonality of column A is the number of columns that are affected when applying the QR algorithm to process column A. The column-fill, fill-ins and commonality need only be calculated implicitly: at any one time we need only know those coefficients smaller than some limit where the limit grows as the QR algorithm progresses. A special program was developed to find the heuristic orderings: simply the QR algorithm with added overheads for switching columns and calculating heuristics. Orderings are found only once; subsequent likelihood evaluations can use the same ordering.

The QR algorithm can be used to implement absorption. For example, $\mathbf{B}^T\mathbf{B}=\mathbf{H}_2^T(\mathbf{I}- \mathbf{H}_1(\mathbf{H}_1^T\mathbf{H}_1)^- \mathbf{H}_1)\mathbf{H}_2$ is the array obtained from processing columns of $\mathbf{H}_1$ where

$$\mathbf{Q}^T\begin{pmatrix}\mathbf{H}_1 & \mathbf{H}_2\end{pmatrix}=\begin{pmatrix}\mathbf{U} & \mathbf{A}\\ \mathbf{0} & \mathbf{B}\end{pmatrix} \qquad (3)$$

Half-stored arrays, $\mathbf{B}^T\mathbf{B}$, are represented diagrammatically as triangles in Figure 2. Therefore, it is not surprising that the ordering based on least commonality corresponds to the ordering of pivots in absorption (TIER & SMITH, 1989) where rows with least elements are absorbed first: the commonality of columns of $\mathbf{H}$ equals the number of non-zero elements in rows (or columns) of $\mathbf{H}^T\mathbf{H}$. The advantage of ordering columns by the least commonality heuristic is shared by both the QR algorithm and absorption.[3] Alternatively, the criteria based on least column-fill and least fill-ins do not seem to have analogs in absorption. There are some matrices where $\mathbf{H}$ is sparse and $\mathbf{H}^T\mathbf{H}$ is non-sparse (BJÖRCK 1976). For these cases, heuristic orderings for the QR algorithm are of little value in absorption. There are situations where the QR algorithm should be used because absorption is infeasible.

Heuristic orderings found dynamically worked well: we were now able to process 130000 columns of the total of 160000. Unfortunately, we were unable to finish many

[3] What is described as the least commonality is the same as the minium degree, and produces the minium degree ordering.

large or moderately sized implementations. Additional improvements were needed and we abandoned our attempt to apply the QR algorithm on huge arrays. After applying the QR algorithm to obtain **B** in (3), where the number of columns in **B** was greatly reduced, we switched over from sparse matrix manipulation to vector processing. There was a choice of two avenues. The first involves applying the QR algorithm and the second entails forming $\mathbf{B}^T\mathbf{B}$ and applying absorption to a half-stored array: all operations can be vectorized. Because with non-sparse matrices the number of operations and storage requirements are greater for the QR algorithm we decided on the latter.

Note that improvements made for the QR algorithm apply also to absorption. Linked list absorption was designed to take advantage of the commonality heuristic and vector processing. Vector processing was implemented when the number of non-zero elements (in the row being absorbed) became greater than 14 per cent. The theoretical optimum for switching to vector processing is $N^{-½}$ per 1 fill where N=32 represents the number of processors in the GOULD NP1 computer being used. Because there is some overhead with linked list we decided on 14 per cent.

III. APPLICATIONS TO REAL DATA

The QR algorithm and the modified linked list absorption was tested in two large data structures: an animal model with direct and maternal additive genetic effects; and an animal model with dominance effects.

A. A direct and maternal effects model

The data was collected for a selection experiment conducted at the Agricultural Research Center at Trangie (Peter Parnell, 1989, personal communication). The model was:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Z}_d\mathbf{a}_d + \mathbf{Z}_m\mathbf{a}_m + \mathbf{e}$$

where **y** is a vector of birth weight recorded on 1814 bull calves, **b** a vector of 18 management group effects, $\mathbf{a}_d$ and $\mathbf{a}_m$ are additive genetic effects for the direct and maternal influences, and **e** is a vector of random residuals. All known genetic relationships were used to define the covariance among $\mathbf{a}_d$ and $\mathbf{a}_m$. The order of **M** was 6066.[4] The log-likelihood is a function of 4 genetic parameters: two genetic variances, one genetic covariance, and one residual variance.

HUDSON (1986) noted that computing the **L** matrix in (1), say $\mathbf{L}_h$, for an animal model is easier to build than even the numerator relationship matrix **A**. For the present case

[4] Referring to **M** in Section IIA.

$$\mathbf{G}^{-1} = \mathbf{M}^{-1} \otimes \mathbf{A}^{-1} = \mathbf{S}^{T}\mathbf{S} \otimes \mathbf{L}_h^{T}\mathbf{L}_h = (\mathbf{S} \otimes \mathbf{L}_h)^{T}(\mathbf{S} \otimes \mathbf{L}_h),$$

where $\otimes$ is the Kronecker product, $\mathbf{S}^{T}\mathbf{S}=\mathbf{M}^{-1}$ is the inverse of a 2 by 2 (co)variance matrix for direct and maternal effects and $\mathbf{L}_h^{T}\mathbf{L}_h = \mathbf{A}^{-1}$. Hence, $\mathbf{L}=\mathbf{S} \otimes \mathbf{L}_h$ and it too is easier to build than **G**.

It took 85, 122 and 128 seconds on a GOULD NP1 computer to determine the orderings based on heuristics of least commonality, least fill-ins, and least column-fill. For the least commonality and fill-in heuristics, the QR algorithm required 82 seconds to evaluate the log-likelihood. The QR algorithm needed 86 seconds for the least column-fill heuristic. The competing approach of absorption with the least commonality heuristic required only 69 seconds. A detailed account of the computing times and memory uses for the algorithms is presented in Table 1. The least commonality and least fill-in heuristics appeared marginally better than the column-fill heuristic. From Table 1, we find that the number of fill-ins for the column-fill heuristic was slightly greater during the sparse matrix implementation.

While the QR algorithm is usable to calculate log-likelihoods for the present data structure, it was slower than the absorption routine. Numerical stability is the only justification that can be given for use of the QR algorithm over absorption. However, in our example rounding errors did not seem to be a problem with absorption. Except for the last or the 15-th decimal place, $\mathbf{y}^{T}\mathbf{P}\mathbf{y}$ and $\log|\mathbf{C}|$ were calculated the same for all approaches. This comparison might be questionable because all procedures finish with absorption for the last 13 per cent of the columns.

From Table 1, we find that the QR algorithm was in fact faster than absorption during the sparse matrix implementation. The loss in time was due to calculating the product $\mathbf{B}^{T}\mathbf{B}$ which was by-passed in absorption.[5] The product was computed via vector processing but the procedure was nevertheless uncompetitive. Matrices $\mathbf{H}_i$ are generally sparser than $\mathbf{H}_i^{T}\mathbf{H}_i$ during the start of the QR algorithm. Unfortunately, from figure 2 we see that prior to vectorizing, $\mathbf{H}_i$ contained many more elements than $\mathbf{H}_i^{T}\mathbf{H}_i$. The QR algorithm applied to non-sparse **B** requires more work than forming $\mathbf{B}^{T}\mathbf{B}$ and applying absorption. This is the major problem of the QR algorithm. Perhaps it is advisable to form $\mathbf{B}^{T}\mathbf{B}$ when **B** is still sparse, then apply sparse matrix absorption, and end up in vector processing. However, forming $\mathbf{B}^{T}\mathbf{B}$ for sparse **B** can still be difficult because the work is proportional to $\frac{1}{2}\Sigma_i n_i^2$ where n_i is the number of non-zero elements in the i-th row of **B**; in absorption we initially set up $\mathbf{B}^{T}\mathbf{B}$ where $\mathbf{B}=\hat{\mathbf{H}}$ when all the n_i are still very small, and hence the overhead is minor.

B. Dominance Model

These data were collected for a second selection experiment, involving egg-laying hens,

[5] Referring to **B** in equation (3).

at the Department of Animal Breeding (Agricultural Research Centre, Jokioinen, Finland). A brief description of the data set can be found in SMITH & MÄKI-TANILA (1989), but summary statistics vary slightly because of editing. The model was:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Z}_a\mathbf{a} + \mathbf{Z}_d\mathbf{d} + \mathbf{e}$$

where **y** is a vector of 1858 female phenotypes, egg number; **b** is a vector of fixed effects including 3 management groups and a continuous covariate for inbreeding depression; **a** is a vector of additive gametic effects; **d** is a vector of dominance effects; and **e** is a residual.

There were 396 hens which were inbred. Because the population was partially inbred, the log-likelihood was a function of five genetic parameters (SMITH & MÄKI-TANILA, 1990). Further, the (co)variances involving **a** and **d** belong to a large matrix, **E**, of order 71440 and $\mathbf{E}^{-1}$ can be evaluated directly. The **L** matrix of (1) is defined such that $\mathbf{L}^T\mathbf{L}=\mathbf{E}^{-1}$. It is easier to calculate **L** than $\mathbf{E}^{-1}$ and further, **L** is sparser; 204583 elements compared to 313439. We thought that the QR algorithm should show some advantage because $\hat{\mathbf{H}}^T\hat{\mathbf{H}}$ has many more non-zero elements than $\hat{\mathbf{H}}$. The QR algorithm and absorption were tested on this large data structure, $\hat{\mathbf{H}}$, with 71445 columns. This example was studied precisely because $\hat{\mathbf{H}}$ was very large. Our purpose was not to estimate the five genetic parameters and indeed with only 396 inbred hens there was little information to estimate some of them.

Ordering columns took 3.0 and 4.3 hours for the least commonality and least fill-ins heuristics. The least column-fill heuristic was not studied as is explained below. An account of the computing times and memory uses required for likelihood evaluation is displayed in Table 2. The QR algorithm did not perform well. It took 1.1 and 2.1 hours to evaluate the likelihood using the commonality and fill-in heuristics. Absorption worked surprisingly well, requiring 22 minutes in total to evaluate the likelihood, which maybe longer than desirable for derivative-free REML.[6] Rounding errors were not a noticeable problem as each job gave nearly identical results. However, the QR algorithm with the least fill-in heuristic produced discrepancies at the tenth decimal place.

For absorption (QR algorithm), vectorizing started when a first row (column) was encountered with more than 14 per cent fill (commonality). Unlike the data structure with 6066 columns, there were considerable differences when vectorizing started. For absorption and the QR algorithm with the commonality heuristic, the half-stored array created for vectorizing was of order 3864. For the QR algorithm with the least fill-in

[6] As a comparison with computing in 2018, Smith and Mäki-Tanila (2018) used the very same data set, but enlarged to include all 2706 hens with records on egg number (rather than the culled number 1858), and performed a derivative-free REML. That calculation required 17 minutes of computing per likelihood evaluation using a home computer; and involved factorizing an indefinite matrix of order 332937, rather than the positive definite matrix **M** of order 71445.

heuristic the order was 6985. An attempt was made to determine an order for the column-fill heuristic, but after 5 hours of computing only the first 63000 columns of a total of 71445 had been processed. No attempt was made to study the column-fill heuristic further as it suffered from a high degree of fill-ins and was clearly inferior. We can infer that the order of the half-stored array for the column-fill heuristic would have been about 8445. The column-fill heuristic was of little value. From Table 2, the commonality heuristic out performed the fill-in heuristic. Choosing columns on least fill-ins is by definition short sighted. The commonality heuristic apparently had a superior long term behavior.

IV. CONCLUSION

In terms of computing times and memory requirements the QR algorithm was found to be inferior to Gaussian elimination for the two data structures studied. Unlike absorption, the QR algorithm could not run from start to finish in a sparse matrix mode. It was necessary to interrupt the procedure and invoke vector processing. In our case we used absorption to finish what we were unable to do in a sparse matrix mode. It is important to point out that the QR algorithm was found to be non-competitive only as we have implemented the procedure: there may be better implementations.[7]

The QR algorithm is numerically stable and does not suffer from the rounding errors (BJÖRCK, 1976) that may occur when implementing absorption or matrix inversion. Further, we did not use the complete the "QR algorithm" because we turned to absorption in the vector processing stage. If absorption is to be used it should be implemented in double or higher precision. While rounding errors and collinearity may be problems, particularly with continuous covariates, it does not follow that these problems will automatically occur with all random effect models used in REML. This needs to be investigated. If a random effects model has continuous covariates, corresponding rows should be saved and not absorbed. It is possible to switch from absorption to a more numerically stable method to finish likelihood evaluation. As the array would be very small, computing time for the numerically stable procedure would be negligible.

Although our results are disappointing, it was surprising how well absorption worked. Improvements made for the QR algorithm apply directly to absorption. The enhancements resulted from the commonality heuristic, and vector processing. The first generation software of GRASER et al (1986) used a good ordering of columns which was determined prior to absorption. The software did not use vector processing. We tested our software again omitting vector processing and using the initial commonality ordering rather than one determined dynamically. For the direct and maternal effects model, likelihood evaluation required 59 minutes and 50 seconds. Hence, the combined effects of the dynamically determined ordering and vector processing was an increase in

[7] See the 2018 discussion.

computing speed by 52 times. We tested the programs twice again omitting in turn each one of the improvements. Vector processing by itself resulted in a 28 fold increase in speed, and the dynamic ordering by itself resulted in a 2.9 fold increase. It appears that the next generation of derivative-free software will be considerably faster than earlier generations. These improvements apply directly to the programs of MEYER (1988).

Scientists should not become complacent just because a very competitive REML algorithm appears to now be 52 times more competitive.[8] Faster algorithms allow the solving of larger problems and hence calculations are still likely to be demanding. Research in numerical methods should continue until REML becomes a computational triviality. We have studied three heuristic orderings designed for the QR algorithm. It was fortunate that the commonality heuristic corresponded to eliminating rows in absorption with least elements first. There are other heuristics that should be considered that are designed particularly for absorption. For example, a least fill-in heuristic for absorption should be considered, and it is not the same as the least fill-in heuristic for our QR algorithm.

REFERENCES

BJÖRCK, Å. 1976. Methods for sparse linear least squares problems. Sparse Matrix Computations, eds J.R. Bunch and D.J Rose. 453 pp. Academic Press Inc, New York.

GILL, P.E., and MURRAY, W. 1976. The orthogonal factorization of a large sparse matrix. Sparse Matrix Computations, eds J.R. Bunch and D.J Rose. 453 pp. Academic Press Inc, New York.

GRASER, H.-U., SMITH, S.P., and TIER, B. 1987. A derivative free approach for estimating variance components in animal model by REML. J. Anim. Sci. 64:1362-1370.

HUDSON, G.F.S. 1986. Computing genetic evaluations through application of generalized least squares to an animal model. Genet. Sel. Evol. 18:31-40.

MEYER, K. 1988. DFREML, programs to estimate variance components for individual animal models by restricted maximum likelihood

[8] Regarding absorption, or factorization of a matrix, there are three main ways to apply the Cholesky decomposition to arrive at factorization: the outer product form, the inner product form, and the bordering method. While the outer product form was good enough for typical animal breeding problems back in 1989, the inner product form and the bordering method are found to perform better on some sparse matrix structures (See Ng and Peyton 1993).

(REML). Institute of Animal Genetics, Edinburgh University, Scotland. Memo.

MEYER, K. 1989. Estimation of variance components for individual animal models I. Univariate analyses. Genet. Sel Evol (in press).

POWELL, M.J.D. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal, 7:155-162.

PRESS, W.H., FLANNERY, B.P., TEUKOLSKY, S.A., and VETTERLING, W.T. 1986. Numerical Recipes. 818 pp. Cambridge Unversity Press, Cambridge.

SMITH, S.P. 1987. Estimation of genetic parameters in non-linear models. International symposium, Advances in Statistical Methods for the Genetic Improvement of Livestock. Eds D. Gianola and K. Hammond. Armidale, Australia.

SMITH, S.P. and MÄKI-TANILA, A. 1990. Genotypic covariance matrices and their inverses for models allowing dominance and inbreeding. Submitted Genet. Sel. Evol, 22, 65-91.

SMITH, S.P. and MÄKI-TANILA, A. 1989. Inverting the extended genomic table: a case study. Memo, vixRa archived, Quantitative Biology, 2018.

STRANG, G. 1976. Linear algebra and its applications. Academic Press, Inc. New York, NY.

TARJAN, R.E. 1976. Graph theory and Gaussian elimination. Sparse Matrix Computations, eds J.R. Bunch and D.J Rose. 453 pp. Academic Press Inc, New Yorki, NY.

THOMPSON, R. and JUGA, J. 1989. A derivative-free approach for estimating variance and covariance components in bivariate animal models by restricted maximum likelihood. Submitted ?

TIER, B. and SMITH, S.P. 1989. Use of sparse matrix absorption in animal breeding. Submitted Genet. Sel Evol.

Table 1. Computing times[1] and memory[2] uses needed for absorption and the QR algorithm applied to a data structure with 6066 columns.

	Absorption		QR Algorithm					
			LCH[3]		LFH[4]		LCF[5]	
	Space	Time	Space	Time	Space	Time	Space	Time
Column[6]:								
0	51101	.0	34325	.0	34325	.0	34325	.0
400	48715	.1	32723	.1	32710	.1	32936	.1
800	46320	.3	31131	.2	31110	.1	31553	.2
1200	43920	.5	29531	.3	29510	.2	30173	.3
1600	41838	.6	28020	.4	28008	.3	28791	.4
2000	40201	.9	26820	.5	26793	.4	27403	.5
2400	38163	1.1	25620	.6	25557	.5	26001	.6
2800	35958	1.4	24420	.6	24304	.6	24602	.7
3200	34179	1.6	23313	.7	23041	.7	23187	.9
3600	32995	2.0	22372	.8	21761	.8	21782	1.0
4000	32745	2.6	22417	1.1	22120	1.0	22191	1.2
4400	36005	3.7	26096	1.5	25724	1.5	26318	1.7
4800	46362	6.2	38585	2.5	37589	2.4	39489	2.5
5200	91685	17.5	115848	7.6	113339	7.5	118417	7.9
last[7]	113162	28.1	167704	12.1	158494	10.6	167874	11.5
Vectoring[8]:		1.0		30.4		30.0		31.5
Absorption[9]:		39.6		39.4		41.9		42.7
Total:		68.7		81.9		82.5		85.7

1 seconds on a GOULD NP1 computer
2 number of non-zero elements in associated array listed under space
3 least commonality heuristic
4 least fill-in heuristic
5 least column-fill heuristic
6 number of non-zero elements and cumulative elapsed times to process
7 last column was 5253 for absorption, LCH and LCF; 5249 for LFH
8 time required to construct half-stored array for vector processing. For absorption, LCH and LCF the order was 813. For LFH the order was 817.
9 vector processing

Table 2. Computing times[1] and memory requirements[2] needed for absorption and the QR algorithm applied to a data structure with 71445 columns.

	Absorption		QR Algorithm			
			LCH[3]		LFH[4]	
	Space	Time	Space	Time	Space	Time
Column[5]:						
0	335164	.0	215736	.0	215736	.0
4000	324653	.3	207733	.5	200736	1.5
8000	314303	.7	199733	1.1	192561	2.4
12000	304115	1.0	191733	1.7	184396	3.3
16000	294041	1.3	183733	2.2	176170	4.3
20000	284056	1.7	175733	2.7	168054	5.3
24000	274501	2.2	167903	3.3	159919	6.4
28000	262840	2.8	160317	4.1	151796	7.7
32000	257242	3.8	153951	5.0	143679	9.2
36000	247557	5.1	146167	6.2	135611	11.1
40000	235146	6.5	139202	7.8	127573	13.6
44000	224441	9.1	132990	9.3	119513	17.0
48000	203379	11.2	124406	11.2	110829	20.6
52000	187191	14.0	119390	14.0	108084	23.5
56000	169270	18.4	116696	17.4	119521	30.2
60000	152538	24.8	118564	23.5	190907	58.4
64000	161404	40.1	147269	39.1	948688	331.9
last[6]	1640075	636.8	2059264	731.4	1995167	867.2
Vectorizing[7]:		15.5		2430.8		4643.3
Absorption[8]:		663.5		689.8		3212.7
Total:		1315.8		3852.0		7723.2

1 seconds on a GOULD NP1 computer
2 number of non-zero elements in associated array listed under space
3 least commonality heuristic
4 least fill-in heuristic
5 number of non-zero elements and cumulative elapsed time to process
6 last column was 67581 for absorption and LCH; 64460 for LFH
7 time required to construct half-stored array for vector processing.
For absorption and LCH the order was 3864. For LCF the order was 6985.
8 vector processing

Discussion of Smith, Meyer and Tier's 1989 Paper

by S.P Smith

2018

Davis (2016) provides an extensive review of direct methods to solve linear equations, including use of the QR algorithm and its sparse-matrix variants. Björck (1996) provides a detail account of the QR algorithm in the context of linear least squares. Given the volume of research its difficult to make a singularly significant recommendation today, but in 1989 it was more difficult.

If the goal is to calculate the log-likelihood function using a method that is faster, and uses less memory, than absorption or Gaussian elimination, then the QR algorithm by way of Householder transformations offers a possible avenue to proceed. The series of rectangular matrices, $\mathbf{H}_1$, $\mathbf{H}_2$, ⋯, $\mathbf{H}_n$, shown in Figure 1 are ever shrinking and can be stored in sparse-matrix mode. Moreover, as only the diagonals of **U** are needed, the top rows of **U** can be released from memory in turn once they are completely computed. Unfortunately, Smith, Meyer and Tier demonstrated that the approach was uncompetitive because of excess fill-in occurring in the matrices, $\mathbf{H}_1$, $\mathbf{H}_2$, ⋯, $\mathbf{H}_n$. George and Liu (1987) indicate that this is a more general criticism of the sparse-matrix application of using the Householder transformations to perform the QR algorithm, and that the more popular way to perform the QR algorithm is to apply Givens rotations to build **U** completely (rather than releasing it) while limiting access to **H**, one row at a time (the Row-Givens). The fill-in that results in building **U** can be well tolerated, whereas that fill-in found in $\mathbf{H}_1$, $\mathbf{H}_2$, ⋯, $\mathbf{H}_n$ can be prohibitive.

George and Liu (1987) also provided a fix to the Householder approach making it as affective as the method that uses Givens rotations. Their fix uses *general row merge trees* that leads to a submatrix annihilation technique that utilizes Householder transformations. Kauffman (1987) generalized the Householder transformation using a blocking strategy, and it quite possible to use Kauffman's generalization with the submatrix annihilation technique.

The Row-Givens method is simple and one of the better recommendations developed by George and Heath (1980). This method takes a row $\mathbf{h}_i^T$ from **H**, one at a time, forms a partition matrix with **U** on top, and then applies Givens rotations to annihilate the row representing $\mathbf{h}_i^T$ while updating **U**. This operation is outlined below for the i-th row of **H**

$$\mathbf{G}_i \times \begin{bmatrix} \mathbf{U}_{before} \\ \mathbf{h}_i^T \end{bmatrix} = \begin{bmatrix} \mathbf{U}_{after} \\ \mathbf{0}^T \end{bmatrix},$$

where $\mathbf{G}_i$ is the orthogonal matrix containing the Givens rotations. The upper triangular

matrix **U** is initialized to null, and when all the rows of **H** have been processed **U** returns as the transpose of the Cholesky decomposition of $\mathbf{H}^T\mathbf{H}$. The columns of **H** are pre-permuted using the minimum degree ordering (George and Heath 1980), or some other suitable heuristic (Duff 1974), which can be determined symbolically. This lets **U** share the same sparse structure as the Cholesky decomposition. The rows of **H** can also be permuted to minimize the operation count, particularly when combined with variable pivot strategies (Robey and Sulsky 1994, Pandian, Parthasarathy and Soman 1999). An interesting way to permute rows and columns of **H** follows, for examples found in animal breeding.

Why not initialize **U** to something that is already upper triangular coming from a column permutation and a row permutation of **H**? Then apply the Row-Givens only on those rows left out!

Consider again the direct and maternal effects model. Note that $\mathbf{S}^T\mathbf{S}=\mathbf{M}^{-1}$, the inverse of a 2 by 2 (co)variance matrix for direct and maternal effects described in Section III A. Because **S** is already upper triangular (i.e., its transpose is the Cholesky decomposition of $\mathbf{M}^{-1}$), the order of direct and maternal effects are left unchanged in our goal to reconstruct **U**. If that order were switched that would induce a general substitution of **S** with $\hat{\mathbf{S}}=\mathbf{P}_r\mathbf{S}\mathbf{P}_r$, where $\mathbf{P}_r$ is the permutation matrix that reverses the order making $\hat{\mathbf{S}}$ in this case lower triangular. The inverse additive genetic matrix is in a form where oldest animals are ordered first, implying that $\mathbf{L}_h^T\mathbf{L}_h=\mathbf{A}^{-1}$, where $\mathbf{L}_h$ is lower triangular. By reversing the order of animals from oldest first, to youngest first, then the permuted inverse matrix is in the form $\bar{\mathbf{L}}_h^T\bar{\mathbf{L}}_h=\mathbf{A}_p^{-1}$ where $\bar{\mathbf{L}}_h=\mathbf{P}_r\mathbf{L}_h\mathbf{P}_r$. Now **L** of (1) is in the form $\mathbf{L}=\mathbf{S}\otimes\bar{\mathbf{L}}_h$, and is upper triangular. However, its better to perform one more permutation to keep direct and maternal effects together for any animal in the model, turning **L** of (1) into the upper triangular matrix $\mathbf{U}=\bar{\mathbf{L}}_h\otimes\mathbf{S}$. The matrix $\hat{\mathbf{H}}$ that is subjected to the Row-Givens has been put into the following form.

$$\hat{\mathbf{H}}=\begin{bmatrix}\mathbf{U} & \mathbf{0} & \mathbf{0}\\ \mathbf{TZ} & \mathbf{TX} & \mathbf{Ty}\end{bmatrix}$$

The matrix **Z** has also been adjusted by the prior permutations. To apply the Row-Givens leave **U** intact, and skip the first 6047 rows and only work on the last 1814 rows. Even if $\mathbf{U}^T\mathbf{U}$ was subjected to the Cholesky decomposition in the order given, the factorization would not return $\mathbf{U}^T$ with its sparse structure preserved because Gaussian elimination is blind to mathematically defined zeros. Therefore, this approach might still perform well even if a good heuristic was not actually used dynamically to define column permutations. There are only 1814 rows to process before fill-in becomes a problem, and so much was skipped.

The last 1814 rows of $\hat{\mathbf{H}}$ can be arranged by sorting rows by the position of the first non-zero element in each row. Morever, the entries of each row of **TZ** are almost entirely

zero except for two non-zero elements that are positioned with the entry for the maternal effect to come first followed by the direct effect. Therefore, once the leading columns of $\hat{H}$ have been zeroed out below the diagonals of **U**, which is now designed to happen automatically with the Row-Givens because of prior sorting, the leading rows of **U** can be released from the computer's memory (as Smith, Meyer and Tier did when they used Householder transformations). The information can be saved to an exterior file. However, only the diagonals of **U** are needed for likelihood evaluation.

Another common data structure found in animal breeding has to do with repeat records, where rows that cut through **TZ** and **TX** have the same or nearly the same sparse structure, and are otherwise identical with the exception of the last elements representing **Ty**. Rows of $\hat{H}$ representing repeat records are better processed together using Householder transformations rather than Givens rotations.

Supplementary References

Björck, Å., 1996, *Numerical Methods for Least Squares Problems*, Philadelphia, SIAM.

Davis, T.A., 2016, A survey of direct methods for sparse linear systems, Technical Report, Department of Computer Science and Engineering, Texas A & M University.

Duff, I., 1974, Pivot selection and row ordering in Givens reduction on sparse matrices, *Computing*, 13, 239-248.

George, A., and M.T. Heath, 1980, Solution of Sparse Linear Least Squares Problems using Givens Rotations, *Linear Algebra and Its Applications*, 34, 69-83.

George, A., and J.W.H. Liu, 1987, Householder Reflections versus Givens Rotations in Sparse Orthogonal Decomposition, *Linear Algebra and Its Applications*, 88-89, 223-238.

Kauffman, L., 1987, The generalized Householder Transformation and sparse matrices, *Linear Algebra and Its Applications*, 90, 221-234.

Ng, E.G., and B.W. Peyton, 1993, Block sparse Cholesky algorithms on advanced uniprocessor computers, *SIAM Journal of Scientific Computing*, 14 (5), 1034-1056.

Pandian, A., K. Parthasarathy and S.A. Soman, 1999, Towards faster Givens rotations based power system estimator, *IEEE Transactions on Power Systems*, 14 (3), 837-843.

Robey, T.H., and D.L. Sulsky, 1994, Row ordering for a sparse QR decomposition, *SIAM Journal of Matrix Analysis and Applications*, 15 (4), 1208-1225.

Smith, S.P. and A. Mäki-Tanila, 2018, Estimating genetic parameters in a dominance model that includes inbreeding, vixRa archived, Quantitative Biology.