UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS AGRÁRIAS E VETERINÁRIAS
CÂMPUS DE JABOTICABAL

# USO DAS ROTAÇÕES DE GIVENS MODIFICADAS COMO UM MÉTODO DIRETO PARA OBTENÇÃO E ATUALIZAÇÃO DAS SOLUÇÕES EM SISTEMAS COM ACUMULAÇÃO SEQÜENCIAL DE DADOS

**Eduardo da Cruz Gouveia Pimentel**

**Orientadores: Profa. Dra. Sandra Aidar de Queiroz**
**Dr. Luiz Alberto Fries**
**Prof. Dr. Flávio Schramm Schenkel**

Tese apresentada à Faculdade de Ciências Agrárias e Veterinárias – Unesp, Câmpus de Jaboticabal, como parte das exigências para a obtenção do título de Doutor em Zootecnia (Produção Animal).

JABOTICABAL – SÃO PAULO – BRASIL
Dezembro de 2007

Pimentel, Eduardo da Cruz Gouveia
P644u Uso das rotações de Givens modificadas como um método direto para obtenção e atualização das soluções em sistemas com acumulação seqüencial de dados / Eduardo da Cruz Gouveia Pimentel. – – Jaboticabal, 2007
v, 102 f.; 28 cm

Tese (doutorado) - Universidade Estadual Paulista, Faculdade de Ciências Agrárias e Veterinárias, 2007
Orientadora: Sandra Aidar de Queiroz
Banca examinadora: João Meidanis, Ricardo da Fonseca, Roberto Carvalheiro, Adhemar Sanches
Bibliografia

1. Modelo misto. 2. Rotações de Givens. 3. Valor genético. I. Título. II. Jaboticabal - Faculdade de Ciências Agrárias e Veterinárias.

CDU 636.082:636.2

Ficha catalográfica elaborada pela Seção Técnica de Aquisição e Tratamento da Informação – Serviço Técnico de Biblioteca e Documentação - UNESP, Câmpus de Jaboticabal.

## DADOS CURRICULARES DO AUTOR

**EDUARDO DA CRUZ GOUVEIA PIMENTEL –** nasceu aos 17 de janeiro de 1975, na cidade de Recife – PE, onde fez os cursos do ensino básico, do primário ao 2º grau. Em 1993, iniciou o curso de Zootecnia na Universidade Federal Rural de Pernambuco, também em Recife, tendo colado grau em 1999. Iniciou o curso de mestrado em Genética e Melhoramento Animal, na FCAV/Unesp – Jaboticabal, em março de 2002 e concluiu em fevereiro de 2004. Em março de 2004, iniciou o curso de doutorado em Zootecnia, área de Produção Animal, também na FCAV/Unesp. Durante o primeiro semestre de 2005, cumpriu parte dos créditos em disciplinas do doutorado na Universidade Estadual de Campinas – Unicamp. No início de 2006 foi contemplado com bolsa *sandwich* da CAPES para, de maio de 2006 a abril de 2007, desenvolver parte de sua pesquisa na Universidade de Guelph, Canadá, sob orientação do prof. Dr. Flávio Schenkel. Em dezembro de 2007, defendeu sua tese de doutorado na FCAV/Unesp.

*Ao mestre*
*e amigo*
*Luiz Fries*

*Dedico*

## AGRADECIMENTOS

A Deus.

A minha família: meu pai Clóvis, meus irmãos Marcelo e Ana Lúcia, minha cunhada Verônica e meus sobrinhos Laélia e Marcelinho.

A minha saudosa mãe, Laélia, que sempre fará parte de tudo em minha vida.

A Luiz Fries. A alegria do convívio infelizmente se foi, mas os sentimentos de gratidão, admiração e amizade irão me acompanhar por toda a vida.

À família Schenkel: Flávio, Sandra, Mariana e Daniel. Foi uma bênçao ter tido a oportunidade de conhecer e desfrutar de seu convívio e amizade.

A Sandra Queiroz pela orientação, amizade e sempre agradável convívio.

A meus colegas de sala Beto e Vânia, que sempre me ajudaram muito e se tornaram amigos muito queridos.

A Mehdi Sargolzaei, com quem muito aprendi nas proveitosas e agradáveis conversas e discussões em sua sala.

A todos os colegas e amigos que tive a oportunidade de conhecer em Jaboticabal, em Campinas e em Guelph.

Ao programa de melhoramento Conexão Delta G e à empresa GenSys - Consultores Associados S/S Ltda., pelo fornecimento dos dados usados neste trabalho.

Aos membros das bancas examinadoras de qualificação e defesa, prof. Euclides Malheiros, prof. João Ademir de Oliveira, prof. Maurício Alencar, prof. Danísio Munari, prof. João Meidanis, prof. Ricardo da Fonseca, prof. Adhemar Sanches e Dr. Roberto Carvalheiro pelas discussões, correções e sugestões.

A todos os professores do departamento de Zootecnia e ao programa de Pós-Graduação em Zootecnia pela oportunidade de cursar o doutorado nesta instituição.

À CAPES, pelo apoio financeiro.

# SUMÁRIO

Página

## LISTA DE ABREVIATURAS

cM – Centimorgan
CPU – Unidade central de processamento
EBV – Valor genético predito, com informação fenotípica e de genealogia
Gb – Gigabyte
GC – Gradiente Conjugado
GEBV – Valor genético predito, com informação molecular
GG – Gentleman-Givens
Mb – Megabyte
MME – Equações de modelos mistos de Henderson
PEV – Variância do erro de predição
QTL – Locos de característica quantitativa
RAM – Modelo animal reduzido
SAM – Modelo alternativo de Schaeffer
SNP – Polimorfismo de nucleotídeo único

NOTA: muitas das abreviações incluídas aqui mantêm as iniciais das palavras na língua inglesa. A opção de manter essas iniciais foi feita devido ao amplo, comum e bem conhecido uso de tais abreviações em trabalhos de melhoramento animal.

**USO DAS ROTAÇÕES DE GIVENS MODIFICADAS COMO UM MÉTODO DIRETO PARA OBTENÇÃO E ATUALIZAÇÃO DAS SOLUÇÕES EM SISTEMAS COM ACUMULAÇÃO SEQÜENCIAL DE DADOS**

**RESUMO –** O objetivo da pesquisa descrita nesta tese foi estudar possíveis aplicações do método das rotações modificadas de Givens na solução de sistemas de equações lineares tipicamente observados em problemas de melhoramento animal. Duas aplicações foram consideradas: a predição de valores genéticos com base em informação fenotípica e genealógica, por meio da metodologia dos modelos mistos; e a predição de valores genéticos com base em informação molecular, obtida pela genotipagem de painéis densos de SNPs. Na primeira aplicação, delineou-se o emprego de um modelo animal reduzido, combinado a uma ordenação do sistema que permitiu uma abordagem multi-frontal de decomposição. As matrizes frontais foram definidas como sendo as partes da triangular superior pertinentes a cada rebanho. Com isso, o problema pôde ser desmembrado em *n* subproblemas em que *n* é o número de rebanhos. Um conjunto de programas foi desenvolvido de modo a decompor as matrizes de dados de cada rebanho independentemente, e depois combinar as informações de todos eles na solução do sistema triangular geral, por retro-substituição. Concluiu-se que o método pode ser empregado em um sistema para atualização de predições de valor genético sob modelo animal reduzido, em que se aninham os efeitos de vacas dentro de rebanhos. Na segunda aplicação, comparou-se o emprego das rotações de Givens com o método do Gradiente Conjugado, na solução de sistemas lineares envolvidos na estimação de efeitos de SNPs em valores genéticos. O método das rotações demandou menos tempo de processamento e mais memória. Concluiu-se que, dado o crescente avanço em capacidade computacional, o método das rotações pode ser um método numérico viável e apresenta a vantagem de permitir o cálculo dos erros-padrão das estimativas.

**Palavras-chave:** decomposição de matriz, método numérico, modelo misto, rotações de Givens, valor genético

**USE OF MODIFIED GIVENS ROTATIONS AS A DIRECT METHOD FOR SOLVING AND UPDATING SOLUTIONS FROM SYSTEMS WITH SEQUENTIAL ACCUMULATION OF DATA**

**SUMMARY –** The aim of this study was to investigate possible applications of the modified Givens rotations on the solution of linear systems that typically arise in animal breeding problems. Two applications were considered: prediction of breeding values based on phenotypes and relationships, using mixed model methods; and prediction of breeding values based on molecular information, using genotypes from high density SNP chips. In the first application, the use of a reduced animal model, combined with a specific ordering of the system, made it possible to apply a multi-frontal decomposition approach. The frontal matrices were defined as the parts of the upper triangular corresponding to each herd. In this way, the problem could be partitioned into *n* sub-problems, where *n* is the number of herds. A set of programs was developed in order to factorize the data matrix of each herd independently, and then combine the information from all of them while solving the overall triangular system, by back-substitution. The conclusion was that Givens rotations can be used as a numerical method for updating predicted breeding values under a reduced animal model, if dam effects are nested within herds. In the second application, the modified Givens rotations were compared to the Conjugate Gradient method for solving linear systems that arise in the estimation of SNP effects on breeding values. Givens rotations required less processing time but a greater amount of high speed memory. The conclusion was that, given the increasing rate of advance in computer power, Givens rotations can be regarded as a feasible numerical method which presents the advantage that it allows for the calculation of standard errors of estimates.

**Keywords:** breeding value, Givens rotations, matrix decomposition, mixed model, numerical method

# CAPÍTULO 1 – CONSIDERAÇÕES GERAIS

O tema apresentado e discutido nesta tese envolve possíveis aplicações de um método numérico (Rotações de Gentleman-Givens) em problemas de melhoramento animal. O texto está dividido em quatro capítulos, brevemente descritos a seguir.

Este capítulo 1 inicia-se com uma introdução ao tema e a justificativa para a condução do estudo, seguida de uma breve revisão de literatura e da exposição dos objetivos do trabalho. Ainda neste capítulo são reportadas algumas experiências e informações julgadas interessantes de serem relatadas.

Nos capítulos 2 e 3 são apresentados estudos em que se investigou a viabilidade de emprego do método numérico em duas aplicações. No capítulo 4 são feitas algumas considerações finais sobre o trabalho, apresentado pontos que podem ser melhorados, implicações e sugestões de futuros estudos sobre o assunto.

Os códigos fonte, em linguagem Fortran 90, dos programas desenvolvidos neste trabalho são apresentados em apêndices, após o último capítulo da tese.

## 1.1 Introdução e justificativa

A prática da seleção, feita com base em predições de valores genéticos dos animais a serem usados na reprodução, constitui um dos mais eficientes métodos de melhoramento genético animal. O cálculo de valores genéticos preditos envolve a solução de sistemas de equações lineares. Na maioria dos casos, montam-se equações de modelos mistos.

De maneira geral, os métodos numéricos para solução de sistemas lineares podem ser classificados em duas categorias: diretos ou iterativos. Supondo-se um sistema de equações lineares $\mathbf{Ax} = \mathbf{b}$, em que $\mathbf{A}$ é uma matriz de coeficientes, $\mathbf{b}$ é um vetor correspondente à "mão direita" das equações e $\mathbf{x}$ é o vetor de parâmetros a serem estimados, um método iterativo envolve a escolha inicial de $\mathbf{x}^{(1)}$, uma aproximação da solução $\mathbf{x}$, e a determinação de uma seqüência $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$,... tal que: $\lim_{i\to\infty} \mathbf{x}^{(i)} = \mathbf{x}$.

Teoricamente, quando se emprega um método iterativo, um número infinito de operações aritméticas é necessário para a obtenção da solução **x**. Na prática, as iterações são cessadas quando se aceita que a aproximação obtida é satisfatória, de acordo com algum critério previamente estabelecido.

Por outro lado, métodos diretos fornecem uma solução exata (na ausência de erros de arredondamento) após um número finito de operações aritméticas. O método direto mais simples é a inversão da matriz **A** (admitindo que **A** é não-singular), em que a solução do sistema é calculada por $\mathbf{x} = (\mathbf{A}^{-1})\mathbf{b}$. Computacionalmente, a inversão de uma matriz é uma operação que demanda bastante tempo de processamento e memória para armazenamento de dados. Em razão disso, a maior parte dos métodos diretos se baseia no método de Eliminação de Gauss, que consiste em aplicar operações elementares em linhas e/ou colunas, para transformar o sistema de equações original em um sistema triangular, que pode então ser facilmente solucionado por substituição.

A questão acerca do emprego de um método direto ou de um método iterativo para solução de um sistema de equações não é de fácil resposta. Segundo GEORGE & LIU (1981), a questão de que categoria de método usar deve ser posta num contexto restrito e bem definido, tendo-se como referência o tipo de problema a ser resolvido. Na literatura, é comum a afirmação de que, para sistemas grandes e esparsos (com muitos elementos iguais a zero), métodos iterativos demandam menos recursos computacionais e tempo de processamento do que métodos diretos.

Nas aplicações práticas em melhoramento animal, a ordem das equações de modelos mistos está na casa dos milhares e verifica-se um grande número de zeros na matriz de coeficientes. Tal fato tem levado os pesquisadores dessa área a implementarem algoritmos iterativos nos programas computacionais usados para avaliação genética (SCHAEFFER & KENNEDY, 1986; MISZTAL & GIANOLA, 1987).

Se a matriz de coeficientes do sistema de equações é simétrica e positiva definida (o que se observa em problemas de melhoramento), então há garantia de que as iterações irão convergir para a solução. Contudo, o número de iterações necessárias para atingir a convergência pode ser muito grande. Além disso, o tipo de algoritmo

usado em um determinado processo iterativo pode ter influência sobre o número de iterações requerido para alcançar a solução. Uma das desvantagens no uso de métodos iterativos é que não se consegue obter os elementos da inversa da matriz de coeficientes, e estes são necessários para o cálculo dos erros padrão das predições. Com isso, os erros padrão das predições são sempre valores aproximados (SCHAEFFER, 1993).

Existem diversos tipos de algoritmos que empregam métodos iterativos, sejam as iterações feitas nos dados ou nas MME. Os métodos de iterações nas MME mais comuns na área de melhoramento são os métodos de Gauss-Seidel e de Jacobi, embora outros métodos mais sofisticados (LIDAUER et al., 1999; TSURUTA et al., 2001; STRANDÉN et al., 2002) venham sendo desenvolvidos no intuito de minimizar o número de iterações necessárias para convergência e o tempo de processamento. O método de iteração nos dados foi proposto por SCHAEFFER & KENNEDY (1986) e é o mais (em alguns casos, talvez o único) viável quando se faz avaliação genética usando grande volume de dados.

Segundo MOORE (1965), a capacidade dos computadores dobra a cada 18 meses. Esse incremento resulta do aumento em velocidade dos processadores e da crescente capacidade de disco e memória, mais especificamente do aumento no número de transistores por circuito integrado. Um Pentium com 1024 Mb de memória custava em torno de $5000 em 1998. Um IBM 3090 com configuração similar em 1988 podia custar 1000 vezes mais. O crescente aumento em capacidade dos computadores pode ser usado para eliminar otimizações que tornam os programas cada vez mais complicados (MISZTAL, 1999).

Nesse contexto, é possível hipotetizar que, no futuro, resgatem-se muitos métodos numéricos com propriedades interessantes (e.g., métodos de decomposição), mas que até então não podiam ser cogitados por limitações em disponibilidade de memória e velocidade de processador.

Entre os benefícios que se pode ter pelo emprego de métodos de decomposição, merece destaque o fato de que: i) a decomposição da matriz, embora custosa em termos de processamento, pode ser aproveitada para resolver novos problemas

envolvendo a matriz original e ii) muitas decomposições de matrizes podem ser atualizadas, o que pode representar grandes economias em computação (STEWART, 2000). Como a acumulação seqüencial de dados é rotina em programas de melhoramento, o custo de um método direto passa a ser essencialmente o da atualização e solução do sistema triangular dada a decomposição inicial, visto que o custo da decomposição inicial amortizado contra todas as soluções é negligenciável. Além disso, com o emprego de um método de decomposição, torna-se possível a obtenção dos elementos da inversa da matriz de coeficientes.

## 1.2 Revisão de literatura

### *1.2.1 Métodos de decomposição de matrizes*

Resumidamente, uma decomposição é a fatoração de uma matriz em fatores mais simples. Historicamente, o método de Eliminação de Gauss e suas variantes (incluindo a Decomposição de Cholesky) têm sido usados para a solução de sistemas lineares do tipo $\mathbf{Ax} = \mathbf{b}$, por meio de sua redução para sistemas triangulares equivalentes.

#### *1.2.1.1 Decomposição de Cholesky*

Suponha-se um sistema de equações lineares $\mathbf{Ax} = \mathbf{b}$, em que $\mathbf{A}$ é uma matriz de coeficientes esparsa, simétrica e positiva definida, $\mathbf{b}$ é um vetor correspondente à "mão direita" das equações e $\mathbf{x}$ é o vetor de parâmetros a serem estimados. A aplicação da Decomposição de Cholesky em $\mathbf{A}$ produz a fatoração triangular $\mathbf{A} = \mathbf{LL}^T$, em que $\mathbf{L}$ é triangular inferior com elementos positivos na diagonal. Pode-se provar que esse tipo de fatoração sempre pode ser feito quando $\mathbf{A}$ é simétrica e positiva definida (MORRIS, 1983). O sistema pode agora ser representado por $\mathbf{LL}^T\mathbf{x} = \mathbf{b}$.

Substituindo-se $\mathbf{L}^T\mathbf{x}$ por $\mathbf{y}$, tem-se o sistema $\mathbf{Ly} = \mathbf{b}$, do qual pode-se obter o vetor $\mathbf{y}$ por substituição (lembrando que $\mathbf{L}$ é triangular inferior). Em seguida, o vetor $\mathbf{x}$ pode ser facilmente obtido do sistema $\mathbf{L}^T\mathbf{x} = \mathbf{y}$ por retro-substituição ($\mathbf{L}^T$ é triangular superior).

Um dos aspectos mais importantes relacionados à aplicação da Decomposição de Cholesky em matrizes esparsas é que, durante a fatoração, a matriz **A** geralmente sofre preenchimentos, ou seja, a matriz **L** apresenta elementos não-nulos em posições que eram ocupadas por zeros na parte triangular inferior de **A** (GEORGE & LIU, 1981).

Segundo BOLDMAN & VAN VLECK (1991), o número de não-nulos na matriz **L** é altamente dependente da ordenação do sistema. Pacotes para solução de sistemas esparsos, que aplicam Decomposição de Cholesky ou outro tipo de fatoração, podem reduzir dramaticamente o tempo de processamento por meio da reordenação do sistema original, de modo que a esparsidade seja preservada durante a fatoração.

O número de operações elementares necessárias para a obtenção da solução varia em torno do cubo da ordem da matriz, representado por $O(n^3)$. O grande número de multiplicações executadas durante o cálculo pode tornar o acúmulo de erros de arredondamento um sério problema (GIVENS, 1954).

*1.2.1.2 Decomposição QR*

Particularizando a solução do sistema linear em um problema cujo objetivo consiste na obtenção de estimativas de quadrados mínimos, a partir das equações normais, pode-se representar o sistema $\mathbf{Ax} = \mathbf{b}$ como $\mathbf{X^T X\beta = X^T y}$. Essa mudança de notação se mostrará conveniente na demonstração a seguir.

Diversos métodos podem ser empregados para formar uma triangular superior **R**, tal que $\mathbf{R^T R} = \mathbf{X^T X}$, pela aplicação direta de transformações ortogonais às colunas de **X**. Esses métodos são facilmente adaptados para lidar com adições ou exclusões de dados (MAINDONALD, 1984), e usam o fato de que $|| \mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{y} || = || \mathbf{Q^T X}\hat{\boldsymbol{\beta}} - \mathbf{Q^T y} ||$, em que || denota norma euclideana e $\mathbf{Q^T}$ é uma matriz ortogonal, e por meio de uma série de pré-multiplicações (cujo produto denota-se por **Q**) reduz $\mathbf{X}_{n\times p}$ a: $\mathbf{QX} = \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}$

em que **R** é uma matriz triangular superior de ordem $p \times p$. A esse processo dá-se o nome de Decomposição **QR** de **X**. Uma vez que a redução para a forma triangular

superior se completa, $\hat{\boldsymbol{\beta}}$ pode ser obtido por retro-substituição resolvendo-se $\mathbf{R}\hat{\boldsymbol{\beta}} = \mathbf{c}$, em que: $\mathbf{Qy} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}$ (EĞECIOĞLU & SRINIVASAN, 1995).

De acordo com HOUSEHOLDER (1958), como $\mathbf{X^TQ^TQX} = \mathbf{X^TX} = \mathbf{R^TR}$, a matriz **R** corresponde precisamente à matriz triangular que seria alcançada pela aplicação da decomposição de Cholesky sobre a matriz $\mathbf{X^TX}$ (ou da matriz **A**, conforme demonstrado na seção anterior). Um aspecto vantajoso apresentado por este método, apontado por SMITH (1995), reside no fato de que as operações para obtenção da triangular superior são feitas sobre a matriz de dados **X**, dispensando a multiplicação necessária para montar a matriz de coeficientes. MATSTOMS (1997) destacou dois potenciais problemas que podem advir da montagem de $\mathbf{X^TX}$: perda em precisão aritmética e ocorrência de preenchimentos.

Segundo BAI et al. (2001), algumas vantagens apresentadas por métodos de fatoração do tipo **QR** são: (a) elas nunca falham e sempre produzem uma matriz ortogonal **Q** e uma matriz triangular superior **R**; (b) fatorações ortogonais são bastante robustas e numericamente estáveis; e (c) a ortogonalidade da matriz **Q** torna a solução do sistema de equações original fácil de ser obtida pela solução do sistema triangular superior com o fator triangular **R**.

Como incorporações de novos registros aos arquivos de dados, analisados a cada safra, são rotinas freqüentes em programas de melhoramento animal, a maior vantagem no emprego de uma decomposição **QR** pode ser considerada a facilidade em atualizar a decomposição.

Suponha-se que uma matriz **X** de ordem $n \times p$ tenha sido decomposta em $\mathbf{X} = \mathbf{QR}$, em que **Q** é ortogonal de ordem $n \times p$ e **R** é triangular superior de ordem $p \times p$. Faça-se com que $\overline{\mathbf{X}}$ seja uma matriz de ordem $r \times s$ obtida de **X** pela inserção ou deleção de uma linha ou coluna. A decomposição **QR** de $\overline{\mathbf{X}}$, $\overline{\mathbf{X}} = \overline{\mathbf{Q}}\overline{\mathbf{R}}$, em que $\overline{\mathbf{Q}}$ é ortogonal de ordem $r \times s$ e $\overline{\mathbf{R}}$ é triangular superior de ordem $s \times s$, pode ser computada em $O(np)$ operações aritméticas pela atualização de **Q** e **R**. O número de operações necessárias para atualizar **Q** e **R** é favorável quando comparado com $O(np^2)$ operações

necessárias para decompor a matriz inicial de ordem $n \times p$ (REICHEL & GRAGG, 1990).

De acordo com COLEMAN et al. (1986), numericamente, **Q** e **R** podem ser computadas por reflexões de Householder ou rotações de Givens. Métodos de decomposição ortogonal baseados em transformações de Givens apresentam duas grandes vantagens: zeros na matriz de dados representam ganhos substanciais na solução do problema, e a matriz de dados pode ser naturalmente processada por linhas (GENTLEMAN, 1985). O procedimento, contudo, é custoso pois a inclusão de cada nova linha de dados custa em torno de $2p^2$ multiplicações e $p$+1 radiciações. Segundo GENTLEMAN (1973), o possível custo total para a solução de um problema de quadrados mínimos ficaria por volta de $2np^2$ multiplicações e np radiciações.

FRIES (1984) comparou o método da decomposição de Cholesky com o das rotações de Givens quanto ao número de operações necessárias para obtenção das soluções, simulando dez anos de avaliações em um programa de melhoramento. Comparando os resultados obtidos para o primeiro ano, o autor concluiu que o método de Cholesky é muito mais eficiente que o de Givens, quando a informação é usada apenas uma vez. Quando há um acúmulo seqüencial de informações, ano após ano, as rotações de Givens se tornam mais eficientes que a decomposição de Cholesky.

Tal argumentação pode ser reforçada pelo estudo de uma situação hipotética de um programa de melhoramento em que avaliações genéticas são feitas durante vinte anos, sucessivamente. Para tanto, assumem-se os seguintes parâmetros: (1) no rebanho em questão, 10000 bezerros são desmamados a cada ano; (2) o rebanho de vacas é de 12000 vacas com reposição anual de 2000 vacas; (3) número constante de touros, igual a 400, com a reposição de 100 touros por ano; (4) 200 grupos de contemporâneos por ano. Calculando-se o número de operações aritméticas executadas em cada avaliação, pelo método da decomposição de Cholesky e pelo das rotações de Givens ($p^3/6$ e $2np^2$ operações, respectivamente, conforme LAWSON & HANSON (1974)), podem-se construir os gráficos apresentados na Figura 1.

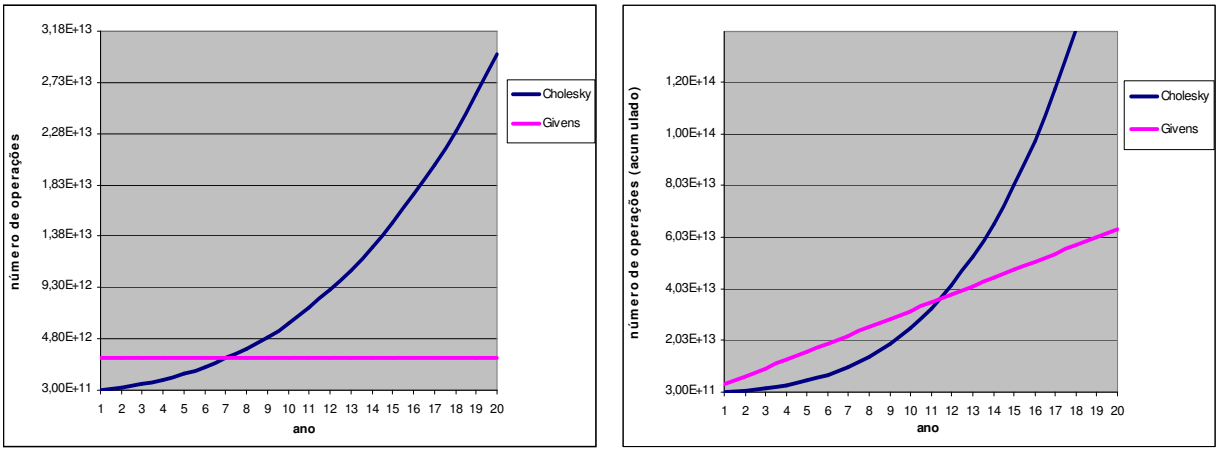


Figura 1. Número de operações aritméticas executadas em cada ano de avaliação genética, e número acumulado de operações em todos os anos de avaliação pelos métodos de decomposição de Cholesky e de Givens.

Percebe-se que a partir do 8º ano de avaliação o número de operações executadas pelo uso do método Givens já é menor que o necessário para implementação de Cholesky. Quanto ao número de operações acumuladas ao longo dos anos, a vantagem do método Givens se torna evidente a partir do 12º ano de avaliação. Ademais, em programas de melhoramento, mesmo dentro de cada ano de avaliação, são freqüentes as ocorrências de pequenas correções nas análises, feitas por meio de adições ou retiradas de linhas de dados. Se esta particularidade for levada em consideração, a vantagem do método Givens pode ser verificada em bem menos tempo.

A construção da matriz de transformações de Givens requer pelo menos quatro operações de multiplicação, uma adição, uma divisão e uma radiciação. A aplicação da transformação a cada vetor de ordem 2 requer quatro multiplicações e duas adições (HANSON & HOPKINS, 2004).

GENTLEMAN (1973) propôs uma modificação do método original das rotações de Givens no intuito de diminuir o número de operações aritméticas requeridas pelo procedimento, eliminando as radiciações. Além disso, segundo HANSON & WISNIEVSKY (1979), há uma redução de quatro operações de multiplicação e duas de

adição para duas de multiplicação e duas de adição em cada aplicação da transformação em um vetor de ordem 2.

LAWSON et al. (1979) compararam a eficiência da modificação de Gentleman em relação ao método original das transformações de Givens. Os autores concluíram que, no contexto da triangularização de matrizes, o método modificado é mais eficiente em tempo de processamento, por fatores que variam entre 1,4 e 1,6.

## 1.3 Objetivos

O objetivo deste trabalho foi estudar possíveis aplicações do método das rotações modificadas de Givens na solução de sistemas de equações lineares tipicamente observados em problemas de melhoramento animal. Mais especificamente, buscou-se estudar a viabilidade do método em duas aplicações: na formulação de um procedimento para atualização de predições de valor genético (EBVs); e em análises de estimação de efeitos de marcadores do tipo SNP (*Single Nucleotide Polymorphism*) com aplicações em seleção genômica.

## 1.4 Algumas considerações sobre predição de EBVs

### *1.4.1 Exploração de esparsidade*

Como a predição de EBVs geralmente envolve a solução de sistemas lineares (equações de modelos mistos – MME) bastante grandes, tratar o problema como denso e armazenar as matrizes inteiras não são uma opção viável. Felizmente, além de grandes, as MME são geralmente bastante esparsas. Seguindo a definição de WILKINSON (1965), uma matriz pode ser considerada esparsa quando o número de zeros é tão grande que ela “paga” para que se explore essa esparsidade. O ponto que define se determinado método numérico é viável ou não reside na sua capacidade de explorar esparsidade para economizar memória.

Uma das estratégias mais usadas para explorar esparsidade de sistemas lineares visando o emprego de um método numérico direto é descrita em DUFF et al.

(1986). O procedimento consiste em resolver o problema em três passos: *analisar*, *fatorar* e *resolver*. No passo *analisar*, inicialmente faz-se uma fatoração simbólica do sistema para que se identifique o padrão de esparsidade na matriz triangular, dada a ordenação original de linhas e colunas no sistema. Depois implementa-se um algoritmo para reordenar o sistema de tal modo que o preenchimento na triangular seja minimizado. No passo *fatorar* se procede a fatoração numérica do sistema reordenado e no passo *resolver* soluciona-se o sistema triangular por substituição.

Como a idéia proposta é usar um método direto com o objetivo principal de atualizar as soluções, essa estratégia apresenta uma limitação forte: a minimização de preenchimento feita em uma primeira avaliação pode ser perdida ao se incorporarem ao sistema novas linhas de dados. Ou seja, é de se esperar que a otimização inicial leve a um mínimo local. Após algumas atualizações, o sistema pode se tornar denso, isto é, "não suficientemente esparso", seguindo definição de Wilkinson (1965).

Uma particularidade que se observa em programas de melhoramento, em que se fazem avaliações genéticas sucessivas, é que a cada ano ou safra, novos animais são incorporados aos bancos de dados. Essa característica introduz um outro elemento complicador no processo de atualização de EBVs: o sistema não cresce apenas em número de linhas, mas também em número de colunas, conforme ilustrado na Figura 2.

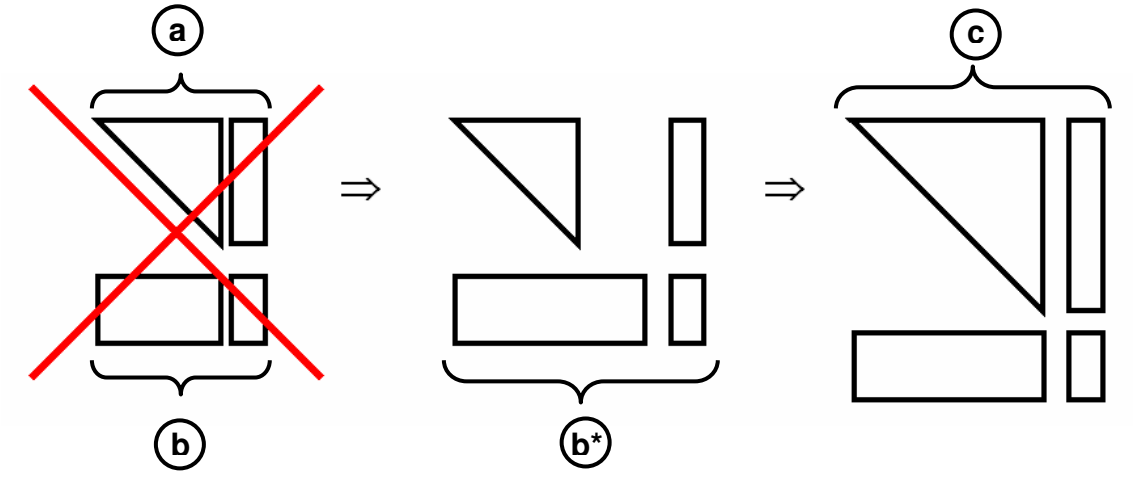


Figura 2. Atualização da triangular superior, com aumento de parâmetros. a = sistema triangular inicial; b = novas observações; c = sistema triangular redimensionado

Suponha-se que, em uma primeira avaliação, um número *n* de observações em um número *p* de parâmetros sejam rotadas com uma matriz triangular superior **R** de ordem *p* x *p* e uma mão direita **z** de ordem *p* x 1, representadas na Figura 2 pelo sistema triangular '**a**'. A idéia é armazenar esse sistema em arquivo e, à medida que um conjunto '**b**' de novas observações nos *p* parâmetros se tornem disponíveis, rotá-las com o sistema armazenado '**a**' e obter, por retro-substituição, as soluções do sistema atualizado. Na prática, contudo, o que se observa é que os novos dados a serem incorporados ao sistema se constituem em conjuntos de observações '**b***' em um número de parâmetros maior que *p*. Torna-se necessário redimensionar o sistema '**a**' em uma nova estrutura '**c**' de forma a acomodar as colunas referentes aos novos parâmetros. Se o sistema triangular estiver sendo armazenado em estrutura esparsa, é preciso que esses redimensionamentos não provoquem excessivo preenchimento e, ainda, que mantenha previsível o padrão de esparsidade.

A estratégia adotada no estudo foi então a busca por uma forma de predizer o padrão de preenchimento inicial em **R** e os preenchimentos adicionais que decorressem das atualizações sucessivas no conjunto de dados. Para tanto, optou-se por definir um modelo estatístico de trabalho e definir *a priori* uma ordenação de colunas que incorresse em menor preenchimento. Tal abordagem é possível porque em dados usados em avaliações genéticas tanto as posições dos não-nulos na matriz de dados quanto nas MME são conhecidos sem a necessidade de uma fatoração simbólica e o padrão de preenchimento em **R** pode ser determinado seguindo uma regra bem simples. A chamada *regra givens local* se define como segue: "suponha que uma linha i seja usada como pivô para zerar o não-nulo da coluna c na linha j. Após a rotação, o elemento [j,c] é zerado, o [i,c] permanece não-nulo e nas outras colunas tomamos a união do padrão de preenchimento nas duas linhas" (COLEMAN et al., 1986).

Definido o padrão de preenchimento na triangular superior, pode-se então empregar uma estrutura estática de armazenamento de matriz esparsa. Uma das estruturas de armazenamento de matriz esparsa mais populares é a lista ligada (KNUTH, 1975). O seguinte exemplo, adaptado de DUFF et al. (1986), ilustra como são organizados os elementos de uma lista ligada. Tome-se a seguinte matriz **A**:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 \\ 2 & 0 & -2 & 0 & 3 \\ 0 & -3 & 0 & 0 & 0 \\ 0 & 4 & 0 & -4 & 0 \\ 5 & 0 & -5 & 0 & 6 \end{pmatrix}$$

A representação da matriz **A** em uma lista ligada por linhas é apresentada na Tabela 1. São necessários três vetores de ordem igual ao número de não-nulos, sendo dois de tipo inteiro e um de tipo real, além de um vetor de tipo inteiro de ordem igual ao número de linhas na matriz.

Tabela 1. Representação de matriz esparsa em uma lista ligada por linhas

| índice | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| IROWST | 4 | 3 | 6 | 10 | 11 | | | | | | |
| JCN | 4 | 5 | 1 | 1 | 5 | 2 | 4 | 3 | 3 | 2 | 1 |
| VAL | -1 | 3 | 2 | 1 | 6 | -3 | -4 | -5 | -2 | 4 | 5 |
| LINK | 0 | 0 | 9 | 1 | 0 | 0 | 0 | 5 | 2 | 7 | 8 |

Para ilustrar o uso dessa estrutura, suponha-se que se tenha interesse em acessar os elementos da linha 4. Temos que IROWST(4)=10, JCN(10)=2 e VAL(10)=4, portanto o primeiro elemento não-nulo na linha 4 está na posição [4,2] e tem valor 4,0. Temos ainda que LINK(10)=7, JCN(7)=4 e VAL(7)=–4, portanto o não nulo seguinte está na posição [4,4] e tem valor –4,0. Como LINK(7)=0 não há mais elementos não nulos na linha 4.

*1.4.2 Modelo*

Inicialmente, considerou-se a aplicação do método proposto a um sistema de atualização de EBVs sob modelo animal. Seja a equação do modelo: **y** = **Xb** + **Zu** + **e**.

Temos que as equações de modelos mistos (MME) são:

$$\begin{bmatrix} \mathbf{X'X} & \mathbf{X'Z} \\ \mathbf{Z'X} & \mathbf{Z'Z + A^{-1}\lambda} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{X'y} \\ \mathbf{Z'y} \end{bmatrix}$$

Utilizando-se um algoritmo para decomposição **QR** da matriz de dados para a solução do sistema, não é necessário montar explicitamente as equações normais ou, no caso de modelo misto, as MME. Portanto, é preciso definir uma matriz de pseudo-observações a ser incorporada à matriz de dados de forma que tal incorporação seja equivalente à adição da matriz $\mathbf{A^{-1}}\lambda$ à parte das equações correspondente ao efeito aleatório.

Chame-se a matriz de dados de $\mathbf{D} = \begin{bmatrix} \mathbf{X} & \mathbf{Z} \end{bmatrix}$

Então: $\begin{bmatrix} \mathbf{X'X} & \mathbf{X'Z} \\ \mathbf{Z'X} & \mathbf{Z'Z + A^{-1}\lambda} \end{bmatrix} = \mathbf{D'D + E'E}$

em que: $\mathbf{E'E} = \begin{bmatrix} \tilde{\mathbf{0}} & \tilde{\mathbf{0}} \\ \tilde{\mathbf{0}} & \mathbf{A^{-1}\lambda} \end{bmatrix}$

Portanto, precisa-se adicionar à matriz de dados uma matriz $\mathbf{E}$: $\begin{bmatrix} \mathbf{D} \\ \mathbf{E} \end{bmatrix}$

Tal que: $\mathbf{E} = \begin{bmatrix} \tilde{\mathbf{0}} & \mathbf{P}\sqrt{\lambda} \end{bmatrix}$

Com $\mathbf{P'P} = \mathbf{A^{-1}}$

Temos, da decomposição descrita em HENDERSON (1976), que: $\mathbf{A = TBT'}$

Logo $\mathbf{P} = (\mathbf{B}^{0,5})^{-1}\mathbf{T}^{-1}$

Tal matriz **P** tem a seguinte forma:

$$\mathbf{P} = \begin{bmatrix} \sqrt{\frac{1}{b_{ii}}} & & & & & & \\ & \ddots & & & \tilde{0} & & \\ & & \sqrt{\frac{1}{b_{ii}}} & & & & \\ & \tilde{0} & & \ddots & & & \\ & & & & \sqrt{\frac{1}{b_{ii}}} & & \\ -0.5p_{ii} & -0.5p_{ii} & & & & \sqrt{\frac{1}{b_{ii}}} & \\ & & -0.5p_{ii} & -0.5p_{ii} & & & \ddots \\ & & & & -0.5p_{ii} & -0.5p_{ii} & \sqrt{\frac{1}{b_{ii}}} \end{bmatrix}$$

Percebe-se que se trata de uma matriz triangular inferior, com cada elemento da diagonal principal igual à raiz quadrada do recíproco da variância da segregação dentro de família, denotada $b_{ii}$. Os elementos fora da diagonal são iguais a zero nas linhas correspondentes aos animais base. Para os animais com informação de genealogia, tem-se o elemento da respectiva diagonal multiplicado por –0,5 nas colunas correspondentes a pai e mãe do animal.

Os elementos $b_{ii}$ podem ser calculados por meio das fórmulas descritas em QUAAS (1976):

- $b_{ii} = [0.5 - 0.25(F_S + F_D)]$ quando os dois pais são conhecidos;
- $b_{ii} = (0.75 - 0.25F_P)$ quando se tem apenas um pai conhecido;
- $b_{ii} = 1$ quando não se tem nenhuma informação de genalogia.

Neste ponto, fica claro que trabalhar com a matriz de dados, em vez das MME, traz a vantagem de que não é preciso montar a matriz de parentesco aditivo, nem sua inversa. Toda a informação referente às relações de parentesco são incorporadas à

análise por meio da adição da matriz **P** ao conjunto de dados. Para tanto, faz-se necessário o cálculo dos coeficientes de endogamia (F). No programa aqui desenvolvido, o cálculo dos coeficientes de endogamia foi feito usando-se o algoritmo descrito em SARGOLZAEI et al. (2005).

Esse procedimento em que se trabalha com a matriz de dados e se adiciona uma matriz **E** para incorporar a informação de parentesco se mostrou viável no caso de um esquema convencional de avaliação genética, i.e., considerando o conjunto de dados inteiro em cada análise. Nesse caso, um algoritmo do tipo 'analisa → fatora → resolve' pode ser empregado de maneira eficiente. Porém, em um esquema de atualizações sucessivas usando rotações GG, a formulação apresenta duas fortes limitações: quantidade excessiva de preenchimento e dificuldade na predição do padrão de esparsidade.

Note-se que a matriz de incidência **Z** é meramente uma matriz identidade (exceto para animais base), o que tornaria bastante fácil tanto a predição de estrutura em **R** quanto a manutenção da esparsidade. Os dois problemas mencionados surgem quando rotam-se as linhas da matriz de pseudo-observações **P**. Por conta dos elementos fora da diagonal, nas posições de touros e vacas, a definição *a priori* de uma melhor ordem para as colunas do sistema se torna praticamente impossível. Isso porque não se pode prever se determinado pai/mãe vai gerar progênie em safras futuras, tampouco se determinado bezerro se tornará pai/mãe ou apresentará registro apenas como progênie. Como não se pode otimizar a ordenação das colunas, as linhas de **P** provocam, nas atualizações, excessivo preenchimento em **R**. Segundo GENTLEMAN (comunicação pessoal), quando tal preenchimento excessivo é detectado, todo o trabalho que se teve até então é desperdiçado. Seria possível retroceder a análise a um estágio anterior se valores intermediários fossem guardados, mas esse não é o caso.

Decidiu-se então empregar um modelo alternativo (SAM) descrito por Schaeffer (2002). O modelo consiste em uma redefinição do modelo animal em que se representa o efeito genético aditivo aleatório de cada animal como ½ do valor genético aditivo do pai e ½ do valor genético aditivo da mãe, considerados efeitos fixos para minimizar efeito de seleção, somados a um efeito aleatório da segregação mendeliana:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{Z}_s\mathbf{s} + \mathbf{Z}_d\mathbf{d} + \mathbf{Zm} + \mathbf{e}$$

em que:

**y** é o vetor de valores observados para a característica;

**b** é um vetor de efeitos fixos de grupos de contemporâneos;

**s** é um vetor de efeitos fixos de capacidade de transmissão dos pais;

**d** é um vetor de efeitos fixos de capacidade de transmissão das mães;

**m** é um vetor de efeitos aleatórios de segregação mendeliana;

**e** é um vetor de efeitos residuais aleatórios;

**X**, $\mathbf{Z_s}$, $\mathbf{Z_d}$, e **Z** são matrizes de incidência.

Com: $$\mathbf{Var}\begin{pmatrix}\mathbf{m}\\ \mathbf{e}\end{pmatrix} = \begin{pmatrix}\mathbf{B}\sigma_a^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\sigma_e^2\end{pmatrix}$$

em que **B** vem de: $\mathbf{A} = \mathbf{TBT}^T$

Como se vê acima, o modelo pressupõe que a estrutura de (co)variância do efeito de segregação mendeliana é dada pela matriz diagonal $\mathbf{B}\sigma_a^2$. A matriz de pseudo-observações a ser adicionada à matriz de dados é então uma matriz diagonal, provocando bem menos preenchimento e tornando mais factível a predição do padrão de esparsidade em **R**.

Ordenando-se as colunas do sistema da forma $\mathbf{Z} \,||\, \mathbf{Z_d} \,||\, \mathbf{X} \,||\, \mathbf{Z_s}$ e predizendo-se o preenchimento conforme a *regra givens local*, verifica-se que a triangular superior **R** pode ser armazenada em estrutura estática. O pequeno exemplo a seguir ilustra a maneira como as posições de memória podem ser reservadas, dada a estrutura do conjunto de dados. Suponha-se o conjunto de dados apresentados na Tabela 2.

Tabela 2. Conjunto de dados ilustrativo da análise com o modelo alternativo de Schaeffer (SAM), contemplando touro, vaca e segregação mendeliana

| animal | pai | mãe | grupo |
|---|---|---|---|
| 7 | 1 | 4 | 1 |
| 8 | 2 | 5 | 1 |
| 9 | 1 | 6 | 1 |
| 10 | 3 | 4 | 2 |
| 11 | 2 | 6 | 2 |
| 12 | 7 | 5 | 2 |
| 13 | 3 | 5 | 3 |
| 14 | 2 | 8 | 3 |
| 15 | 7 | 4 | 3 |
| 16 | 3 | 9 | 3 |

Na Figura 3 são ilustradas as matrizes de incidência dos dados e de pseudo-observações a serem rotadas com a triangular superior **R**, dada a formulação do modelo SAM e a ordenação de colunas descrita acima. Na Figura 4 é mostrado o padrão de preenchimento em **R**, predito conforme a *regra givens local*, dadas as matrizes de dados e de pseudo-observações.

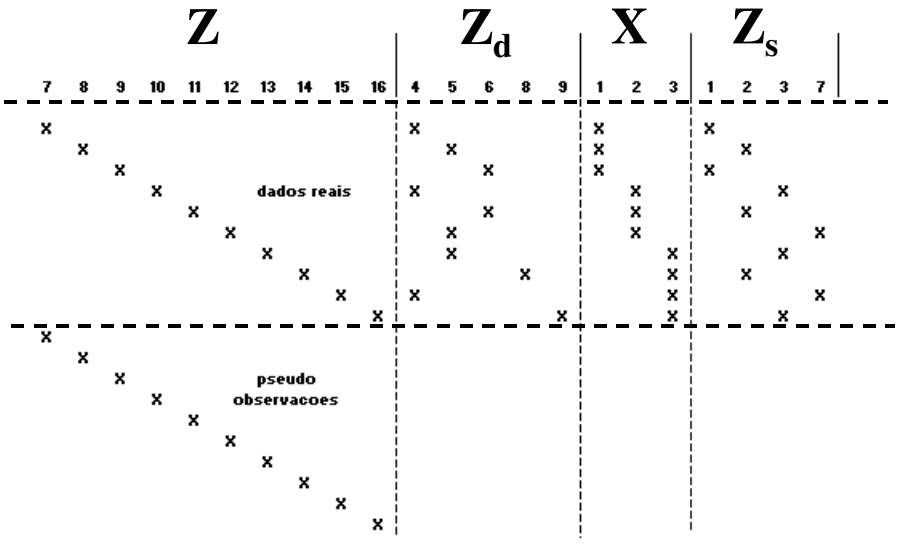


Figura 3. Matriz de dados reais e pseudo-observações, com modelo SAM

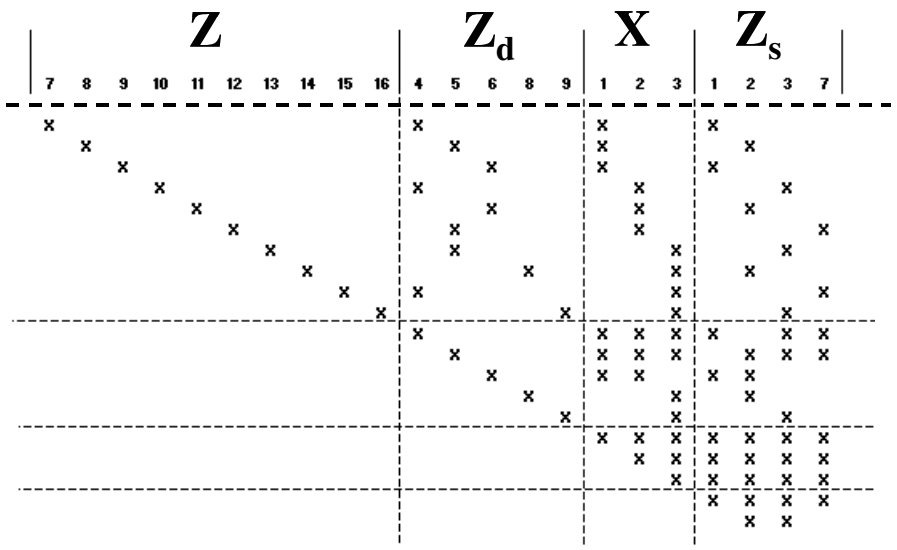


Figura 4. Padrão de preenchimento na matriz R, com modelo SAM

Como a matriz de pseudo-observações é uma matriz diagonal de mesma ordem que **Z**, verifica-se nas partes '**Z** x **Z**', '**Z** x $\mathbf{Z_d}$', '**Z** x **X**' e '**Z** x $\mathbf{Z_s}$' o mesmo padrão de preenchimento da matriz de dados. A parte '$\mathbf{Z_d}$ x $\mathbf{Z_d}$' é preenchida apenas na diagonal e as partes '$\mathbf{Z_d}$ x **X**' e '$\mathbf{Z_d}$ x $\mathbf{Z_s}$' são preenchidas nas caselas dos grupos em que as vacas produzem e dos touros com os quais elas foram acasaladas, respectivamente. As partes '**X** x **X**', '**X** x $\mathbf{Z_s}$' e '$\mathbf{Z_s}$ x $\mathbf{Z_s}$' sofrem preenchimento advindo tanto dos líderes não-nulos nas colunas de vacas quanto nas colunas de grupos, portanto são tratadas como densas. Um programa foi então desenvolvido de maneira a armazenar a triangular superior **R** em esquema esparso numa lista ligada por linhas.

Durante a fase de testes do programa, percebeu-se que a presença de dependências lineares no sistema, decorrentes da parametrização do efeito de segregação mendeliana, levava à necessidade de imposição de um número de restrições maior que o esperado. Como esse número extra de restrições implicava em maior preenchimento, optou-se por considerar o efeito da segregação nas análise por meio da adição de pseudo-observações, conforme descrito em FRIES (2000). Essa formulação, detalhada no capítulo 2, é equivalente à adoção de um modelo animal reduzido.

## 1.5 Referências

BAI, Z.Z.; DUFF, I.S.; WATHEN, A.J. A class of incomplete orthogonal factorization methods. I: methods and theories. **BIT Numerical Mathematics**, v.41, n.1, p.53-70. 2001.

BOLDMAN, K.G.; VAN VLECK, L.D. Derivative-Free Restricted Maximum Likelihood Estimation in Animal Models with a Sparse Matrix Solver. **Journal of Dairy Science**, v.74, p.4337-4343. 1991.

COLEMAN, T.F.; EDENBRANDT, A.; GILBERT, J.R. Predicting Fill for Sparse Orthogonal Factorization. **Journal of the Association for Computing Machinery**, v.33, n.3, p.517-532. 1986.

DUFF, I.S.; ERISMAN, A.M.; REID, J.K. **Direct methods for sparse matrices**. Oxford University Press. 1986.

EĞECIOĞLU, Ö.; SRINIVASAN, A. Givens and Householder Reductions for Linear Least Squares on a Cluster of Workstations. In: INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING (HiPC), 1995, New Delhi, **Proceedings...** p.734-739.

FRIES, L.A. **A study of weaning weights of Hereford cattle, in Rio Grande do Sul, Brazil**. Thesis (Ph.D. In Animal Breeding) Iowa State University of Science and Technology, Ames, Iowa, 1984.

FRIES, L.A. Aggregating ancestral, performance and progeny information into EBVs from a sire & dam model. **Asian-Australasian J. Anim. Sci.**, v.13, p.203-206. 2000.

GENTLEMAN, W.M. Least Squares Computations by Givens Transformations Without Square Roots. **Journal of the Institute of Mathematics and its Applications**, v.12, p.329-336. 1973.

GENTLEMAN, W.M. Basic Procedures for Large, Sparse or Weighted Linear Least Squares Problems. In: GRIFFITHS, P.; HILL, I.D. (Ed.). **Applied Statistics Algorithms**. Chichester: Ellis Horwood series in mathematics and its applications, 1985. p.130-144.

GEORGE, A.; LIU, J.W. **Computer Solution of Large Sparse Positive Definite Systems**. Englewood Cliffs: Prentice-Hall Series in Computational Mathematics, 1981. 324p.

GIVENS, W. Numerical Computation of the Characteristic Values of a Real Symmetric Matrix. **Oak Ridge National Laboratory Report**, Oak Ridge, ORNL – 1574, 1954.

GOLUB, G. Numerical Methods for Solving Linear Least Squares Problems. **Numerische Mathematik**, v.7, p.206-216. 1965.

HANSON, R.J.; WISNIEWSKI, J.A. A Mathematical Programming Updating Method Using Modified Givens Transformations and Applied to LP Problems. **Communications of the ACM**, v.22, n.4, p.245-251. 1979.

HANSON, R.J.; HOPKINS, T. Algorithm 830: Another Visit With Standard and Modified Givens Transformations and A Remark On Algorithm 539. **ACM Transactions on Mathematical Software**, v.30, n.1, p.86-94. 2004.

HENDERSON, C.R. A simple method for computing the inverse of a numerator relationship matrix used in prediction of breeding values. **Biometrics**, v.32, p.69-79. 1976.

HOUSEHOLDER, A.S. Unitary Triangularization of a Nonsymmetric Matrix. **Journal of the Association for Computing Machinery**, v.5, p.339-342. 1958.

KNUTH, D.E. **The art of computer programming, Vol 1: fundamental algorithms**, 2nd ed. Reading: Addison-Wesley, 1975.

LAWSON, C.L.; HANSON, R.J. **Solving Least Squares Problems**. New Jersey: Prentice-Hall series in automatic computation, 1974. 340p.

LAWSON, C.L.; HANSON, R.J.; KINCAID, D.R. *et al*. Basic Linear Algebra Subprograms for Fortran Usage. **ACM Transactions on Mathematical Software**, v.5, n.3, p.308-323. 1979.

LIDAUER, M.; STRANDÉN, I.; MÄNTYSAARI, E.A. *et al*. Solving Large Test-Day Models by Iteration on Data and Preconditioned Conjugate Gradient. **Journal of Dairy Science**, v.82, p.2788-2796. 1999.

MAINDONALD, J.H. **Statistical computation**. New York: Wiley series in probability and mathematical statistics, 1984. 370p.

MATSTOMS, P. Sparse Linear Least Squares Problems in Optimization. **Computational Optimization and Applications**, v.7, p.89-110. 1997.

MISZTAL, I. Complex Models, More Data: Simpler Programming? **Interbull**, Bulletin 20, p.33-42. 1999.

MISZTAL, I.; GIANOLA, D. Indirect solution of mixed model equations. **Journal of Dairy Science**, v.70, p.716-723. 1987.

MOORE, G.E. Cramming more components onto integrated circuits. **Electronics**, v.38, n.8, p.114-117. 1965.

MORRIS, J.L. **Computational Methods in Elementary Numerical Analysis**. New York: John Wiley & Sons, 1983. 410p.

QUAAS, R.L. Computing the diagonal elements and inverse of a large numerator relationship matrix. **Biometrics**, v.32, p.949-953. 1976.

REICHEL, L.; GRAGG, W.B. Algorithm 686 – FORTRAN Subroutines for Updating the QR Decomposition. **ACM Transactions on Mathematical Software**, v.16, n.4, p.369-377. 1990.

SARGOLZAEI, M., IWAISAKI, H., COLLEAU, J.J. A fast algorithm for computing inbreeding coefficients in large populations. **J. Anim. Breed. Genet.**, v.122, p.325-331. 2005.

SCHAEFFER, L.R. Computing Strategies 4 – Iteration Techniques. In: UNIVERSITY OF GUELPH. **Linear Models and Computing Strategies in Animal Breeding**. Guelph, 1993. part.2, p.1-15.

SCHAEFFER, L.R. **ANSC637 Set 17 – Selection bias**. [online] Disponível na Internet via WWW. URL: http://www.aps.uoguelph.ca/~lrs/ANSC637/LRS17/. 2002.

SCHAEFFER, L.R.; KENNEDY, B.W. Computing strategies for solving mixed model equations. **Journal of Dairy Science**, v.69, p.575-579. 1986.

SMITH, D.M. Iterative refinement of parameter estimates and residuals from a regression model fitted using QR decomposition methods. **Computational Statistics & Data Analysis**, v.19, p.327-349. 1995.

STRANDÉN, I.; TSURUTA, S.; MISZTAL, I. Simple preconditioners for the conjugate gradient method: experience with test day models. **Journal of Animal Breeding and Genetics**, v.119, p.166-174. 2002.

TSURUTA, S.; MISZTAL, I.; STRANDÉN, I. Use of the preconditioned conjugate gradient algorithm as a generic solver for mixed-model equations in animal breeding applications. **Journal of Animal Science**, v.79, p.1166-1172. 2001.

WILKINSON, J.H. **The algebraic eigenvalue problem**. Oxford University Press, 1965.

## CAPÍTULO 2 – UM SISTEMA PARA ATUALIZAÇÃO DE PREDIÇÕES DE VALOR GENÉTICO SOB MODELO ANIMAL REDUZIDO

**RESUMO** – O objetivo do presente trabalho foi investigar a viabilidade do emprego das rotações de Givens como um método numérico direto para implementação de um sistema de atualização de EBVs sob modelo animal reduzido, com aplicações em avaliações genéticas em gado de corte. Uma abordagem multi-frontal de decomposição foi delineada, em que as matrizes frontais foram definidas como sendo as partes da triangular superior correspondentes às componentes trapezóides de cada rebanho. Com isso, o problema pôde ser desmembrado em *n* subproblemas em que *n* é o número de rebanhos. Um conjunto de programas foi desenvolvido de modo a decompor as matrizes de dados de cada rebanho independentemente, e depois combinar as informações de todos eles na solução do sistema triangular geral, por retro-substituição. Concluiu-se que o método pode ser empregado em um sistema para atualização de predições de valor genético sob modelo animal reduzido, em que se aninham os efeitos de vacas dentro de rebanhos. A abordagem multi-frontal usada para armazenar uma trapezóide para cada rebanho tornou o sistema ágil e fez com que os requerimentos de memória convencional fossem mantidos em níveis baixos. O sistema se mostra adequado ao desenvolvimento de uma estratégia de processamento paralelo que tire proveito da independência das trapezóides frontais, de modo a melhorar o desempenho. Recomenda-se um estudo mais profundo sobre a viabilidade de uma estratégia desse tipo.

**Palavras-Chave:** atualização, avaliação genética, fatoração QR multi-frontal, método direto, rotações de Givens

## Introdução

Bancos de dados analisados em programas de melhoramento são acrescidos de novas informações a cada safra. Essa natureza cumulativa com que os dados são gerados motivou alguns estudos com o intuito de desenvolver formas de atualização das predições de valor genético sem que se use o banco de dados inteiro no processo. CHRISTENSEN (1981) apresentou um método alternativo, baseado na teoria dos índices de seleção. JAMROZIK & SCHAEFFER (1991) discutiram algumas formas de utilizar subconjuntos de dados para montar equações de modelos mistos (MME), sob modelo animal.

GEORGE & HEATH (1980) demonstraram como um método numérico direto pode ser empregado na atualização de soluções de mínimos quadrados em sistemas esparsos, por meio da aplicação de rotações planares de Givens nas linhas da matriz de dados. O emprego desse método na implementação de um sistema para atualização de predições de valor genético (EBVs) pode ser viável, uma vez que as transformações são aplicadas a apenas uma linha em cada passo do processo. Com isso, pode-se delinear uma estratégia em que as informações já processadas são armazenadas em arquivo, que pode ser atualizado à medida que novas informações são disponibilizadas.

De maneira geral, o emprego de um método direto tende a demandar maior número de operações aritméticas e mais memória para armazenamento de dados do que a aplicação de um método iterativo. Contudo, devido à crescente taxa de avanço em capacidade computacional que se tem observado nas últimas décadas, é possível que restrições desse tipo se tornem cada vez menos limitantes.

Para que a implementação de um sistema de atualização desse tipo seja viável, é preciso que o sistema seja capaz de lidar com uma particularidade dos bancos de dados de programas de melhoramento que dificulta a exploração de esparsidade: às incorporações de novas observações, somam-se inclusões de novos parâmetros no sistema. Precisa-se então definir uma estrutura de armazenamento de matriz esparsa que possa acomodar essa adição de colunas ao sistema.

Um benefício que se pode ter pelo uso de um método direto é a possibilidade de cálculo dos elementos da inversa da matriz de coeficientes, necessários para calcular as acurácias das predições.

O objetivo do presente trabalho foi investigar a viabilidade do emprego das rotações de Givens como um método numérico direto para implementação de um sistema de atualização de EBVs sob modelo animal reduzido, com aplicações em avaliações genéticas em gado de corte.

## Material e métodos

*Método numérico*

Rotações planares de Givens (GIVENS, 1954) constituem-se num método estável e eficiente para decomposição de uma matriz, sendo amplamente usado em problemas de solução de sistemas lineares. O algoritmo é usado para aplicar rotações sucessivas nas linhas da matriz de dados, de modo que no final ela seja reduzida à forma triangular. Esse tipo de transformação é conhecido como decomposição **QR** de uma matriz de dados **A**, em que **Q** é ortogonal e **R** triangular superior. Seja **x** uma linha da matriz de dados **A**, a transformação consiste em tomar-se uma linha **r** da triangular **R**, inicialmente nula, como pivô para zerar o elemento $x_1$ da seguinte forma:

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} r_1 & \cdots & r_k \\ x_1 & \cdots & x_k \end{bmatrix}=\begin{bmatrix} r'_1 & \cdots & r'_k \\ 0 & \cdots & x'_k \end{bmatrix}$$

$$c=\frac{r_1}{\sqrt{r_1^2+x_1^2}} \qquad s=\frac{x_1}{\sqrt{r_1^2+x_1^2}}$$

$$r'_i = cr_i + sx_i$$

$$x'_i = -sr_i + cx_i$$

Rotando-se sucessivamente todas as linhas da matriz de dados, zeram-se todos os elementos abaixo da diagonal principal. Aplicando-se as mesmas transformações à

mão direita das equações, obtém-se um sistema triangular superior que pode ser facilmente resolvido por retro-substituição.

Um aspecto negativo do método é o elevado número de operações aritméticas a serem realizadas, sobretudo as de radiciação. GENTLEMAN (1973) propôs uma alteração no método com objetivo de evitar a operação de radiciação. No método modificado, a matriz **R** é decomposta em uma matriz diagonal **D** e uma matriz triangular superior unitária $\overline{\mathbf{R}}$ de modo que:

$$\mathbf{R} = \mathbf{D}^{0.5}\overline{\mathbf{R}}$$

O algoritmo modificado então rota uma linha do produto $\mathbf{D}^{0.5}\overline{\mathbf{R}}$ com uma linha escalonada de **X**:

$$\begin{array}{cccccccc} \mathbf{0} & \cdots & \mathbf{0} & \sqrt{\mathbf{d}} & \cdots & \sqrt{\mathbf{d}}\bar{\mathbf{r}}_{\mathbf{k}} & \cdots \\ \mathbf{0} & \cdots & \mathbf{0} & \sqrt{\boldsymbol{\delta}}\mathbf{x}_{\mathbf{i}} & \cdots & \sqrt{\boldsymbol{\delta}}\mathbf{x}_{\mathbf{k}} & \cdots \end{array}$$

E as linhas transformadas podem ser escritas como:

$$\begin{array}{cccccccc} \mathbf{0} & \cdots & \mathbf{0} & \sqrt{\mathbf{d}'} & \cdots & \sqrt{\mathbf{d}'}\bar{\mathbf{r}}'_{\mathbf{k}} & \cdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \cdots & \sqrt{\boldsymbol{\delta}'}\mathbf{x}'_{\mathbf{k}} & \cdots \end{array}$$

em que:

$$\mathbf{d}' = \mathbf{d} + \boldsymbol{\delta}\mathbf{x}_{\mathbf{i}}^{2}$$

$$\boldsymbol{\delta}' = \mathbf{d}\boldsymbol{\delta}/\mathbf{d}'$$

$$\bar{\mathbf{c}} = \mathbf{d}/\mathbf{d}' \qquad \bar{\mathbf{s}} = \boldsymbol{\delta}\mathbf{x}_{\mathbf{i}}/\mathbf{d}'$$

$$\mathbf{x}'_{\mathbf{k}} = \mathbf{x}_{\mathbf{k}} - \mathbf{x}_{\mathbf{i}}\bar{\mathbf{r}}_{\mathbf{k}} \qquad \bar{\mathbf{r}}'_{\mathbf{k}} = \bar{\mathbf{c}}\bar{\mathbf{r}}_{\mathbf{k}} + \bar{\mathbf{s}}\mathbf{x}_{\mathbf{k}}$$

Uma das principais vantagens que se pode ter pelo emprego de um algoritmo como rotações GG, em que se processam os dados por linhas, é que, dada uma decomposição inicial da matriz de dados, há a possibilidade de armazenar o sistema triangular para posteriores atualizações.

*Descrição do modelo*

QUAAS & POLLAK (1980), tomando como base um modelo animal, apresentaram um modelo animal reduzido (RAM) para avaliação genética. Em essência, o RAM resulta da definição de um modelo que produz equações de modelos mistos (MME) equivalentes às de um modelo animal, após absorção das equações para os valores genéticos aditivos dos animais que não são pais (VAN VLECK, 1990). Há então uma redução na ordem das MME, de magnitude igual ao número de animais que ainda não produziram progênie. Usando-se o subscrito *p* para denotar animais com progênie e o subscrito *n* para os que não têm progênie, e pressupondo que cada indivíduo possui apenas um registro de produção, o RAM pode ser representado da seguinte forma:

$$\begin{bmatrix} \mathbf{y}_p \\ \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} \mathbf{X}_p \\ \mathbf{X}_n \end{bmatrix} \mathbf{b} + \begin{bmatrix} \mathbf{Z} \\ \mathbf{T} \end{bmatrix} \mathbf{a}_p + \begin{bmatrix} \boldsymbol{\varepsilon}_p \\ \mathbf{e} \end{bmatrix}$$

em que:

**y** é o vetor de observações;

**b** é o vetor de efeitos fixos;

**a** é o vetor de efeito aleatório genético aditivo;

**ε** é o vetor de efeito residual aleatório;

**X** e **Z** são matrizes de incidência;

Pela formulação do modelo, temos que: $\mathbf{a}_n = \mathbf{T}\mathbf{a}_p + \boldsymbol{\theta}$

Em que **T** é a matriz que relaciona cada indivíduo em $\mathbf{a_n}$ a seus pais em $\mathbf{a_p}$ e $\boldsymbol{\theta}$ é o vetor do efeito aleatório da segregação mendeliana.

Logo, fica claro que: $\mathbf{e} = \boldsymbol{\varepsilon}_n + \boldsymbol{\theta}$

Com: $\mathbf{Var}\begin{pmatrix} \boldsymbol{\varepsilon}_p \\ \mathbf{e} \end{pmatrix} = \begin{pmatrix} \mathbf{I}\sigma^2_{\varepsilon} & \tilde{\mathbf{0}} \\ \tilde{\mathbf{0}} & \mathbf{I}\sigma^2_{\varepsilon} + \mathbf{B}\sigma^2_a \end{pmatrix}$ e $\mathbf{Var}(a_p) = \mathbf{A}_p\sigma^2_a$

As MME de um modelo animal reduzido são:

$$\begin{bmatrix} \mathbf{X}_p^T\mathbf{X}_p + \mathbf{X}_n^T\mathbf{R}_n^{-1}\mathbf{X}_n & \mathbf{X}_p^T\mathbf{Z}_p + \mathbf{X}_n^T\mathbf{R}_n^{-1}\mathbf{T} \\ \mathbf{Z}_p^T\mathbf{X}_p + \mathbf{T}^T\mathbf{R}_n^{-1}\mathbf{X}_n & \mathbf{Z}_p^T\mathbf{Z}_p + \mathbf{T}^T\mathbf{R}_n^{-1}\mathbf{T} + \mathbf{A}_p^{-1}\boldsymbol{\lambda} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{a}}_p \end{bmatrix} = \begin{bmatrix} \mathbf{X}_p^T\mathbf{y}_p + \mathbf{X}_n^T\mathbf{R}_n^{-1}\mathbf{y}_n \\ \mathbf{Z}_p^T\mathbf{y}_p + \mathbf{T}^T\mathbf{R}_n^{-1}\mathbf{y}_n \end{bmatrix}$$

em que $\boldsymbol{\lambda} = \dfrac{\sigma^2_e}{\sigma^2_a}$

Após obtidas as soluções para **b** e $\mathbf{a_p}$, soluções para $\mathbf{a_n}$ podem ser obtidas por meio das seguintes fórmulas:

$$\hat{\mathbf{a}}_n = \mathbf{T}\hat{\mathbf{a}}_p + \hat{\boldsymbol{\theta}}$$

$$\hat{\boldsymbol{\theta}} = (\mathbf{Z'}_n\,\mathbf{Z}_n + \mathbf{B}^{-1}\boldsymbol{\lambda})^{-1}(\mathbf{y}_n - \mathbf{X}_n\hat{\mathbf{b}} - \mathbf{T}\hat{\mathbf{a}}_p)$$ (SCHAEFFER, 2002)

*Processamento seqüencial das observações*

FRIES (2000) apresentou um procedimento para o processamento seqüencial de observações, que produzem as mesmas soluções do RAM, porém sem a necessidade de formar a matriz $\mathbf{A_p}$. O procedimento consiste em agregar a informação referente às relações de parentesco por meio da adição de pseudo-observações. Uma descrição mais detalhada é feita a seguir.

De maneira semelhante ao RAM, tome-se o modelo animal: $\mathbf{y = Xb + Ia + \varepsilon}$.

O fundamento para adição das pseudo-observações deriva da definição do valor genético como sendo: metade do valor genético do pai + metade do valor genético da mãe + um efeito de segregação mendeliana, i.e., $\hat{\mathbf{a}}_i = \frac{1}{2}\hat{\mathbf{a}}_S + \frac{1}{2}\hat{\mathbf{a}}_D + \hat{\boldsymbol{\theta}}_i$.

Em notação matricial: $\mathbf{y = Xb + 0.5(Z_S a_S + Z_D a_D) + e}$ com $\mathbf{e = \theta + \varepsilon}$

A diferença entre este procedimento e a formulação do RAM é que “em vez de se utilizarem os resíduos para estimar o efeito da segregação mendeliana apenas no caso de animais jovens, reutiliza-se a informação contida nos resíduos quando animais jovens produzem sua própria progênie” (FRIES, 2000).

Por questão de simplicidade, admita-se que o único efeito fixo no modelo seja uma média geral. Temos então que, no ponto de partida de um programa de melhoramento, uma observação feita em um animal da população base pode ser expressa como:

$$\mathbf{y_i = \mu + a_i + \varepsilon_i}$$

Percebe-se que a única contribuição de genética aditiva sendo considerada para a explicação do dado fenótipo é o valor genético do próprio indivíduo.

Em seguida, os animais da população base produzem sua própria progênie, cujo desempenho pode ser representado como:

$$\mathbf{y_i = \mu + \frac{1}{2}a_{si} + \frac{1}{2}a_{di} + e_i}$$

A partir dessa geração, quando animais com informação de desempenho próprio e de genealogia produzem sua progênie, torna-se necessário combinar essas três fontes de informação. Para tanto, definem-se as pseudo-observações, que são representações das observações desses animais como:

$$\mathbf{y_i = \mu + \frac{1}{2}a_{si} + \frac{1}{2}a_{di} + \theta_i + \varepsilon_i}$$

Uma estimativa da segregação mendeliana a ser usada nessa decomposição do valor genético é dada por:

$$\hat{\boldsymbol{\theta}}_i = (\tfrac{1}{2}\sigma_a^2 / \sigma_e^2)\hat{e} = k\hat{e}$$

Temos que:

$$\hat{a}_i = \tfrac{1}{2}\hat{a}_{si} + \tfrac{1}{2}\hat{a}_{di} + \hat{\theta}_i = \tfrac{1}{2}\hat{a}_{si} + \tfrac{1}{2}\hat{a}_{di} + k\hat{e}_i$$

$$\hat{a}_i = \tfrac{1}{2}\hat{a}_{si} + \tfrac{1}{2}\hat{a}_{di} + k(y_i - \hat{\mu} - \tfrac{1}{2}\hat{a}_{si} - \tfrac{1}{2}\hat{a}_{di})$$

Logo, a pseudo-observação a ser adicionada é da forma:

$$ky_i = k\hat{\mu} + [(k-1)/2]\hat{a}_{si} + [(k-1)/2]\hat{a}_{di} + \hat{a}_i$$

*O exemplo de FRIES (2000)*

Para ilustrar como formam-se as matrizes de dados, processando-se as informações da maneira seqüencial definida acima, tome-se como exemplo o conjunto de dados na Tabela 1 (adaptado de FRIES, 2000).

Tabela 1. Conjunto de dados ilustrativo do procedimento seqüencial

| animal | pai | mãe | y |
|---|---|---|---|
| 1 | 0 | 0 | 10 |
| 2 | 0 | 0 | 9 |
| 3 | 0 | 0 | 8 |
| 4 | 0 | 0 | 7 |
| 5 | 1 | 2 | 9 |
| 6 | 1 | 2 | 10 |
| 7 | 3 | 4 | 8 |
| 8 | 5 | 6 | 11 |

Por questão de simplicidade, pressuponha-se que o único efeito fixo no modelo seja uma média geral, e que a razão da variância residual pela variância genética aditiva seja igual a 1. Dessa forma:

$$\lambda = \frac{\sigma_\varepsilon^2}{\sigma_a^2} = 1$$

$$\sigma_e^2 = \tfrac{1}{2}\sigma_a^2 + \sigma_\varepsilon^2$$

$$k = \tfrac{1}{3}$$

Os animais 1, 2, 3 e 4 são da população base, portanto as observações referentes a seus registros de produção contêm apenas os termos correspondentes aos seus valores genéticos aditivos. Os animais 5, 6, 7 e 8 têm informação de produção e genealogia, portanto seus desempenhos são expressos como função dos valores genéticos dos pais, mais a segregação mendeliana. Como os animais 5 e 6 são também pais do animal 8, duas pseudo-observações são adicionadas para combinar as suas informações de desempenho e de parentesco. Seja **X** o vetor de incidência da média geral, **W** a matriz de incidência do efeito genético aditivo e **y** o vetor de observações, chame-se **Q** a matriz de dados contendo todas as observações reais e as pseudo-observações, i.e., **Q** = { **X** | **W** | **y** }. Temos que:

$$\mathbf{Q} = \left[\begin{array}{ccccccc|c|c}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 10 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 9 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 8 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 7 & 1 \\
\hline
1 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 9 & 1{,}5 \\
1 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 10 & 1{,}5 \\
1 & 0 & 0 & 1/2 & 1/2 & 0 & 0 & 8 & 1{,}5 \\
1 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 11 & 1{,}5 \\
\hline
1/3 & -1/3 & -1/3 & 0 & 0 & 1 & 0 & 3 & 1/3 \\
1/3 & -1/3 & -1/3 & 0 & 0 & 0 & 1 & 10/3 & 1/3
\end{array}\right] \quad \begin{array}{l} r_{ii} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

$\mathbf{Var}(\varepsilon)$ — linhas 1–4

$\mathbf{Var(e) = Var}(\theta + \varepsilon)$ — linhas 5–8

$\mathbf{Var}(\theta)/\mathbf{Var(e)}$ — linhas 9–10

Formando-se as MME correspondentes a esse modelo, tem-se:

$$\begin{bmatrix} \mathbf{X}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}^T\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{W}^T\mathbf{R}^{-1}\mathbf{X} & \mathbf{W}^T\mathbf{R}^{-1}\mathbf{W}+\mathbf{G}* \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\mu}} \\ \hat{\mathbf{a}}_p \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{W}^T\mathbf{R}^{-1}\mathbf{y} \end{bmatrix} \quad \text{com} \quad \mathbf{G}* = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Vê-se que a informação de parentesco é agregada pelas pseudo-observações e por uma matriz **G*** contendo 1 nas posições diagonais dos animais base com registro de produção e 0 em todas as outras.

Cada informação é ponderada ($r_{ii}$) de acordo com o termo de resíduo correspondente a cada tipo de observação, como se observa na coluna à direita da matriz **Q**. Os devidos pesos podem ser atribuídos a cada observação por meio do algoritmo para implementação de mínimos quadrados ponderados, via rotações GG, descrito em GENTLEMAN (1974).

Como o método numérico empregado aqui aplica uma seqüência de rotações planares nas linhas de dados, a montagem das MME não se faz necessária. Temos então que aplicar algumas pré-multiplicações e adicionar algumas outras pseudo-observações ao sistema de modo que o processo fique equivalente à montagem das MME. A matriz de dados a fatorar é então:

$$\left\{ \begin{array}{cc||c} \mathbf{R}^{-0.5}\mathbf{X} & \mathbf{R}^{-0.5}\mathbf{W} & \mathbf{R}^{-0.5}\mathbf{y} \\ \tilde{\mathbf{0}} & (\mathbf{G}*)^{-0.5} & \tilde{\mathbf{0}} \end{array} \right\}$$

## Resultados e discussão

*Estrutura de armazenamento das matrizes*

Para armazenamento da triangular superior **R** para sucessivas atualizações, definiu-se uma ordenação das colunas da matriz de dados, de modo a minimizar preenchimento e possibilitar a predição do padrão de esparsidade do sistema após

cada atualização. Tal predição foi feita de acordo com a chamada *regra givens local* (COLEMAN et al., 1986).

Sejam $\mathbf{X}$, $\mathbf{Z}_s$ e $\mathbf{Z}_d$ matrizes formadas pelas colunas de incidência de um efeito fixo de grupo de contemporâneos e um efeito genético aditivo aleatório de animais com progênie (touros e vacas), respectivamente, a ordem das colunas foi definida na forma $\mathbf{Z}_d \,||\, \mathbf{X} \,||\, \mathbf{Z}_s$ com mães e grupos aninhados em rebanhos. Dada essa ordenação, a matriz triangular superior $\mathbf{R}$ assume a forma de três sub-matrizes bloco-diagonais (correspondentes aos componentes 'vaca x vaca', 'grupo x grupo' e 'vaca x grupo'), em que cada bloco contém a informação de um rebanho, e três sub-matrizes densas (correspondentes aos componentes 'vaca x touro', 'grupo x touro' e 'touro x touro'), conforme apresentado na Figura 1.

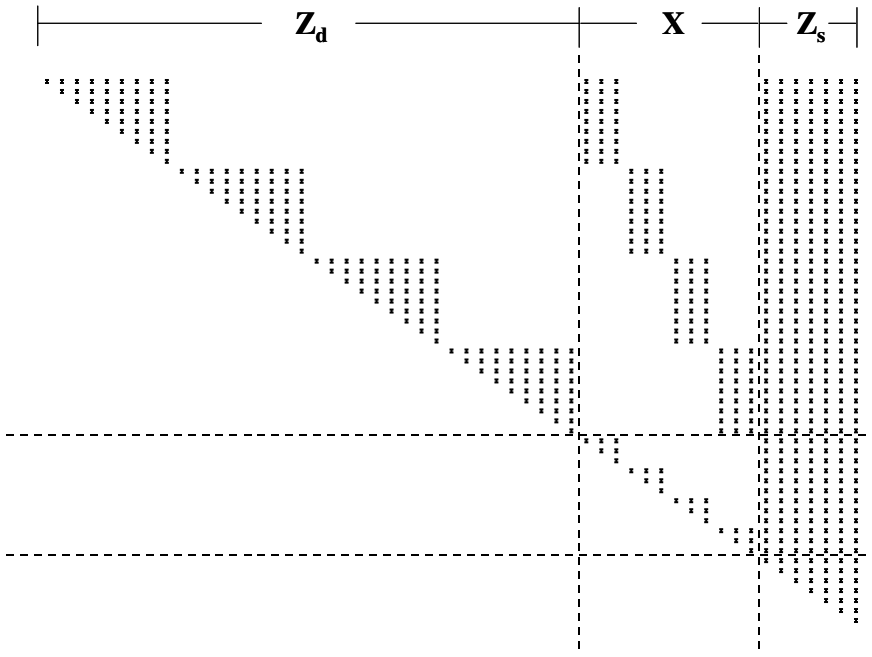


Figura 1. Padrão de preenchimento na matriz triangular superior **R**.

Cada bloco, correspondendo a cada rebanho, nas três sub-matrizes bloco-diagonais são representados na figura como matrizes densas. Aqui cabe definir o que vem a ser uma matriz densa e uma matriz esparsa, no contexto deste trabalho.

Seguindo a definição de WILKINSON (1965), uma matriz pode ser considerada esparsa quando o número de zeros é tão grande que ela “paga” para que se explore essa esparsidade. Dada essa definição de esparsidade, deixe-se claro que, quando se diz que cada bloco é uma matriz densa, não se quer dizer que haja 100% de caselas preenchidas.

De fato, se em uma primeira avaliação as ordens das colunas de vacas e grupos forem ordenadas de maneira eficiente, cada bloco pode ser bastante esparso. O que pode vir a torná-los densos são as atualizações, pois a eficiente ordenação feita num primeiro momento pode perder completamente o efeito quando novas observações são rotadas com **R**. Como não se pode prever quais vacas vão produzir progênie em safras futuras, a pressuposição de que os blocos são densos realmente se faz necessária.

Explorando a Figura 1, percebe-se que, pela concatenação dos componentes ‘vaca x vaca’, vaca x grupo’, ‘vaca x touro’, ‘grupo x grupo’ e ‘grupo x touro’ de cada rebanho com a parte ‘touro x touro’, tem-se uma matriz triangular superior para cada rebanho. Cada uma dessas matrizes é independente das triangulares dos outros rebanhos, exceto pela parte ‘touro x touro’, que fornece a conexão entre rebanhos. ROSO et al. (2004), comparando algumas formas de medir grau de conectabilidade entre grupos de touros de diferentes centrais de teste de desempenho, verificaram que as ligações genéticas devidas a touros respondiam por 94,5% das conexões entre os grupos.

Aninhar vacas em rebanhos permite uma abordagem multi-frontal (DUFF & REID, 1983) do problema e apresenta a vantagem de que as rotações podem ser aplicadas um rebanho por vez. O método de decomposição multi-frontal é aplicável nas situações em que se têm matrizes da forma:

$$\mathbf{R} = \sum_{\mathbf{k}} \mathbf{B}^{(\mathbf{k})}$$

No caso aqui apresentado, cada matriz frontal $\mathbf{B}^{(k)}$ representa a contribuição de um rebanho para a matriz **R** geral. Com isso, cada sub-problema correspondente a um

rebanho pode ser armazenado em estrutura densa (vetorizada), dispensado o custo extra ("overhead") com endereçamento indireto que se tem com o uso de estrutura esparsa, e.g., lista ligada.

Na presente aplicação, a somatória é válida apenas para as triangulares de cada rebanho, sem a parte 'touro x touro', portanto tem-se na verdade uma trapezóide independente para cada rebanho. A parte 'touro x touro', comum a todos eles, pode ser armazenada em arquivo à parte, sendo lida e gravada a cada atualização de rebanho.

Possivelmente, a divisão de **R** nesse conjunto de k trapezóides dê margem à otimização do processo por meio de paralelismo. Para tanto, seria preciso definir uma forma de armazenar as linhas de dados de cada rebanho que remanescessem com líder não nulo em coluna de touro, e depois rotá-las num último passo. Neste trabalho porém, como se tinha em mente um procedimento passível de aplicação em equipamentos simples (com apenas um processador) não se buscou nenhuma estratégia nesse sentido.

*Estratégia operacional*

O sistema de atualização de EBVs foi desenvolvido por meio de um conjunto de programas, em Fortran 90, seguindo uma estratégia que funciona em três etapas. Na primeira etapa, inicializam-se todos os rebanhos participantes, isto é, todas as observações de cada rebanho disponíveis até o momento são incorporadas ao sistema. As duas etapas seguintes consistem da atualização da triangular superior e da solução do sistema triangular, sempre que se tenha interesse em incorporar novas observações.

Da maneira como o sistema foi montado, é preciso que se verifique, antes de proceder à inicialização dos rebanhos, o número total de touros, incluindo todos os rebanhos. As identificações dos touros devem ser recodificadas em números inteiros que variam de 1 até o número total de touros. Esses novos códigos serão gravados em arquivo, para que as identificações sejam consistentes entre rebanhos.

Na etapa de inicialização, o primeiro conjunto de dados de cada rebanho é rotado com a correspondente trapezóide (inicialmente nula) e com a triangular 'touro x touro'. As identificações de vacas e grupos são recodificadas em números inteiros que

variam de 1 até o número total de vacas no rebanho, e 1 até o número total de grupos no rebanho. Durante a execução de inicialização de cada rebanho, quando se faz a alocação de memória para a triangular superior do rebanho, a parte 'touro x touro' é lida do arquivo e carregada na porção inferior dessa triangular, logo abaixo da trapezóide. Como a parte 'touro x touro' é comum a todos os rebanhos, essa parte da matriz é armazenada separadamente. Dois arquivos são usados: um contendo a diagonal, com número de registros igual ao número total de touros (ns) em todos os rebanhos, e outro contendo a parte acima da diagonal, com número de registros igual a [ns*(ns-1)/2].

Nesta etapa, criam-se os arquivos para armazenamento da trapezóide de cada rebanho. Após rotadas todas as linhas de dados, a trapezóide de cada rebanho é então gravada, também em dois arquivos: um contendo a diagonal, com número de registros igual ao número de vacas e grupos no rebanho, e outro contendo a parte da trapezóide acima da diagonal, com número de registros igual a {[nt*(nt-1)/2]-[ns*(ns-1)/2]}, em que nt é a ordem da matriz triangular do dado rebanho. A parte 'touro x touro', atualizada a cada vez que se inclui um rebanho, é gravada em seu arquivo, substituindo a anterior.

Nas etapas de atualização, rotam-se as novas observações realizadas em um dado rebanho. As identificações de novos touros, grupos e vacas são recodificadas, seguindo a numeração máxima do passo anterior. A trapezóide do rebanho e a triangular 'touro x touro' são redimensionadas e as observações são rotadas com a triangular do rebanho. Havendo informação de novos touros, as trapezóides dos outros rebanhos também são redimensionadas.

Para minimizar o espaço em disco necessário para armazenamento, bem como aumentar a velocidade na leitura e gravação dos dados, todos os arquivos usados para armazenamento intermediário de dados são não formatados e de acesso direto.

Quando se verifica que novos touros ou vacas também apareceram no programa como produtos, as respectivas pseudo-observações são rotadas. Para tanto é necessário manter arquivos com identificação e dados de genealogia e produção de bezerros. Visando otimizar a busca, mantém-se um arquivo de fêmeas para cada rebanho e um arquivo geral de machos, também não formatados e de acesso direto.

A última etapa do processo é a solução do sistema triangular geral. Para tanto, obtêm-se primeiramente as soluções para touros e depois as soluções para grupos e vacas de cada rebanho. Dadas as soluções para touros, as retro-substituições nas trapezóides de cada rebanho exigem apenas que sejam carregados os arquivos com as trapezóides, portanto os requerimentos de memória de alta velocidade se mantêm nos níveis dos tamanhos de cada rebanho.

No mesmo programa em que se calculam as soluções, também são calculadas as acurácias dos EBVs de touros. O cálculo é feito por meio da seguinte fórmula:

$$\mathbf{r_i = \sqrt{1 - d_i \lambda}}$$ MRODE (1996)

em que:

$\mathbf{r_i}$ é a acurácia do EBV do i-ésimo touro;

$\mathbf{d_i}$ é o i-ésimo elemento da diagonal da inversa da parte 'touro x touro' das MME;

$\boldsymbol{\lambda}$ é a razão da variância residual pela genética aditiva.

Como se percebe pela fórmula, apenas os elementos da diagonal da inversa são necessários para o cálculo das acurácias. A obtenção dos elementos foi feita pelo emprego do algoritmo "COV" descrito em LAWSON & HANSON (1974). Implementando-se apenas os sete primeiros passos do algoritmo, calculam-se apenas os elementos da diagonal da inversa.

Opcionalmente, o sistema permite que se calculem também os PEVs dos EBVs de touros e os PEVs das diferenças entre EBVs de touros (PEVDs). Neste caso, os elementos de fora da diagonal da inversa da parte 'touro x touro' também precisam ser calculados. Os PEVDs são obtidos por meio da seguinte fórmula:

$$\mathbf{PEVD_{ij} = Var(EBV_i) + Var(EBV_j) - 2 * Cov(EBV_i, EBV_j)}$$

As (co)variâncias de EBVs de touros são obtidas multiplicando-se a parte 'touro x touro' da inversa da "mão esquerda" das MME pelo quadrado médio do erro. Vale salientar que no processo de rotações GG, a soma de quadrados do resíduo é

facilmente acumulada ao final da eliminação de cada linha de dado. A inversa da parte 'touro x touro' foi obtida da porção 'touro x touro' do fator de Cholesky, obtido da triangular superior, seguindo procedimento descrito em WAUGH & DWYER (1945).

*Testes, validação e desempenho*

Para testar e validar os resultados obtidos com o sistema de atualização, tomou-se um conjunto de dados de ganho de peso da desmama ao sobreano do programa de melhoramento Conexão Delta G, cedido pela empresa GenSys Consultores Associados S/S Ltda. Os dados foram editados de modo que o conjunto final continha 32.891 observações em produtos de 103 touros e 18.166 vacas, distribuídos em 3.652 grupos de contemporâneos, em onze anos sucessivos de avaliação em oito rebanhos. As freqüências de observações por ano e rebanho são apresentadas na Tabela 2.

Tabela 2. Distribuição das observações nos rebanhos e anos de avaliação

| Ano | Rebanho | | | | | | | | Totais |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| 1 | 211 | 293 | 148 | 568 | 565 | 518 | 395 | 167 | 2.865 |
| 2 | 638 | 62 | 21 | 652 | 508 | 625 | 358 | 207 | 3.071 |
| 3 | 499 | 179 | 82 | 387 | 570 | 664 | 334 | 115 | 2.830 |
| 4 | 499 | 327 | 490 | 525 | 448 | 681 | 282 | 124 | 3.376 |
| 5 | 371 | 499 | 321 | 562 | 285 | 609 | 96 | 229 | 2.972 |
| 6 | 581 | 383 | 419 | 657 | 566 | 575 | 209 | 192 | 3.582 |
| 7 | 710 | 535 | 339 | 590 | 552 | 534 | 179 | 269 | 3.708 |
| 8 | 66 | 703 | 297 | 611 | 521 | 419 | 211 | 338 | 3.166 |
| 9 | 90 | 556 | 414 | 576 | 368 | 346 | 168 | 317 | 2.835 |
| 10 | 111 | 582 | 409 | 200 | 353 | 390 | 176 | 342 | 2.563 |
| 11 | 52 | 511 | 213 | 55 | 362 | 290 | 135 | 305 | 1.923 |
| Totais | 3.828 | 4.630 | 3.153 | 5.383 | 5.098 | 5.651 | 2.543 | 2.605 | 32.891 |

O conjunto final foi subdividido em 88 arquivos de dados correspondentes aos onze anos de avaliação nos oito rebanhos. Cada um dos arquivos foi processado de maneira independente e incorporado ao sistema, simulando uma seqüência de 88 atualizações sucessivas de avaliação. Os tempos gastos em cada rodada de avaliação são apresentados na Tabela 3.

Ao final, os resultados foram comparados aos obtidos em análise em um só passo, por meio de programa que também aplica rotações GG, utilizando o conjunto de dados inteiro. Os resultados também foram validados pela comparação das soluções obtidas nas análises feitas pelo sistema de atualização com as soluções obtidas usando-se o programa MTDFREML (BOLDMAN et al., 1995), com modelo animal completo. As soluções obtidas pelos três procedimentos foram as mesmas.

Tabela 3. Tempo de CPU (em segundos) em cada rodada de avaliação

| Ano | Rebanho | | | | | | | | Totais |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |
| 1 | 0,07 | 0,11 | 0,04 | 0,42 | 0,35 | 0,30 | 0,18 | 0,05 | 1,52 |
| 2 | 2,49 | 2,81 | 2,87 | 3,59 | 4,33 | 5,14 | 5,72 | 5,89 | 32,84 |
| 3 | 1,85 | 7,38 | 0,17 | 2,35 | 9,05 | 10,00 | 10,62 | 11,03 | 52,45 |
| 4 | 12,22 | 13,98 | 13,61 | 15,56 | 21,36 | 17,58 | 17,82 | 0,41 | 112,54 |
| 5 | 19,49 | 22,41 | 21,27 | 6,63 | 3,47 | 31,24 | 1,70 | 0,63 | 106,84 |
| 6 | 28,31 | 30,06 | 30,98 | 9,39 | 35,92 | 8,66 | 2,01 | 0,83 | 146,16 |
| 7 | 39,80 | 4,09 | 2,77 | 11,44 | 53,46 | 49,26 | 2,35 | 1,11 | 164,28 |
| 8 | 15,78 | 50,96 | 3,70 | 55,81 | 9,35 | 57,66 | 55,76 | 1,50 | 250,52 |
| 9 | 16,64 | 8,53 | 59,39 | 65,03 | 73,76 | 10,92 | 64,03 | 1,89 | 300,19 |
| 10 | 17,56 | 10,83 | 6,53 | 17,64 | 11,16 | 12,38 | 71,67 | 2,31 | 150,08 |
| 11 | 18,10 | 12,53 | 75,26 | 17,04 | 12,48 | 12,96 | 3,51 | 2,75 | 154,63 |
| Totais | 172,31 | 163,69 | 216,59 | 204,90 | 234,69 | 216,10 | 235,37 | 28,40 | 1.472,05 |

Os tempos apresentados na Tabela 3 são apenas os usados para atualizar a trapezóide do respectivo rebanho e a triangular 'touro x touro'. Para computar o tempo total necessário para atualizar as soluções em cada uma das rodadas é preciso somar o tempo gasto com a solução do sistema triangular geral. O tempo de execução do programa que resolve o sistema triangular geral por retro-substituição e calcula as acurácias das predições de valor genético dos touros foi de 37,30 segundos, para solução do sistema depois de todas as atualizações.

Percebe-se pela confrontação das informações apresentadas nas Tabelas 2 e 3 que o tempo de execução não é apenas função do número de observações a serem incorporadas ao sistema. Isso fica bem evidenciado em algumas comparações:

- No ano 10, foram necessários 71,67 segundos para incorporar 176 observações ao rebanho 7, enquanto foram gastos apenas 2,31 segundos para acrescentar 342 observações aos dados do rebanho 8. Essa diferença se explica pelo fato de não haver nenhum touro novo no arquivo de atualização do rebanho 8;
- No ano 7, incorporaram-se arquivos com 535 e 552 observações aos rebanhos 2 e 5, respectivamente. Para tanto foram gastos 4,09 e 53,46 segundos, respectivamente. A explicação da diferença é que, no arquivo do rebanho 5, um macho produzido dentro do programa aparecia pela primeira vez como touro.

Esses e alguns outros contrastes, omitidos por questão de brevidade, indicaram que o tempo necessário para atualização é função: do tamanho do rebanho em questão; da presença ou não de novos touros nos novos dados; e do volume de pseudo-observações a serem rotadas, sobretudo de touros.

O tempo total de execução do programa MTDFREML, somando os tempos de execução dos três aplicativos, com o conjunto de dados inteiro, foi de 150,31 segundos. Tomando como 37,30 segundos o tempo necessário para solução do sistema triangular geral, os tempos gastos com a 10ª atualização nos oito rebanhos variou de 40,05 a 112,56 segundos.

Como o sistema admite a atualização das soluções mesmo após adição de somente uma observação, o tempo gasto com a atualização pode ser praticamente o necessário para as retro-substituições apenas. Espera-se que esse tempo seja favorável mesmo quando comparado a outros métodos/softwares que se utilizem dos bancos de dados inteiros.

Mesmo apresentando uma vantagem numérica, alguns segundos a mais ou a menos em uma rodada de avaliação podem não fazer diferença muito marcante, do ponto de vista prático. Fica claro então que o maior benefício que se pode ter pelo emprego de um método direto como as rotações GG é a possibilidade de cálculo das acurácias, PEVs e PEVDs. Quando se implementa um método iterativo para resolver as MME, geralmente se usa algum algoritmo que fornece uma aproximação dos elementos da diagonal da inversa. Porém, tais algoritmos geralmente não fornecem uma aproximação dos elementos de fora da diagonal, e esses são necessários para comparações de diferenças entre determinados animais ou grupos de animais (KENNEDY & TRUS, 1993).

*Viabilidade*

Um fator determinante da viabilidade de uso do sistema é a demanda por memória de alta velocidade. Como o sistema trata a triangular 'touro x touro' e as trapezóides de cada rebanho como matrizes densas, o procedimento é mais indicado para programas com um grande número de pequenos rebanhos. Caso contrário, o uso do sistema pode ser bastante comprometido ou mesmo inviabilizado, dependendo das dimensões das matrizes frontais. Quanto menor a quantidade e maior o tamanho dos rebanhos, menos vantajoso, ou mesmo viável, o emprego do sistema. Em um caso extremo, i.e., apenas um grande rebanho, a abordagem multi-frontal perderia completamente o sentido. Nessa situação, seria mais indicado o emprego de estratégia em um passo só, do tipo 'fatoração simbólica → reordenação do sistema → fatoração numérica → solução', como em GEORGE & HEATH (1980).

Para ilustrar o quanto o tamanho dos rebanhos pode limitar o uso do sistema, tomem-se os seguintes exemplos:

- Seja um rebanho com dados de 2.000 vacas em 250 grupos, participante de um programa em que se avaliam 1.000 touros. O vetor usado para armazenar a triangular superior desse rebanho demanda ~40,3Mb de memória de alta velocidade, em precisão dupla.
- Seja um rebanho com dados de 20.000 vacas em 2.500 grupos, participante de um programa em que se avaliam 5.000 touros. O vetor usado para armazenar a triangular superior desse rebanho demanda ~2,8Gb de memória de alta velocidade, em precisão dupla.

Na Tabela 4, são apresentados os números de parâmetros, após a última atualização, e as exigências em memória para armazenamento das trapezóides de cada um dos oito rebanhos do conjunto de dados usado para testar o sistema. O número total de touros, após a última atualização, foi 103.

Tabela 4. Números finais de parâmetros e exigências em memória de alta velocidade, em cada um dos oito rebanhos usados para testar o sistema

| Rebanho | Nº final de vacas | Nº final de GCs | Memória exigida (Mb) |
|---|---|---|---|
| 1 | 3.497 | 346 | 59,37 |
| 2 | 2.461 | 487 | 35,48 |
| 3 | 2.139 | 312 | 24,85 |
| 4 | 2.739 | 962 | 55,17 |
| 5 | 2.242 | 604 | 33,15 |
| 6 | 2.332 | 662 | 36,56 |
| 7 | 1.442 | 170 | 11,19 |
| 8 | 1.314 | 109 | 8,85 |

A análise feita com o conjunto de dados inteiro, por meio do programa MTDFREML, exigiu apenas 5,8Mb de memória de alta velocidade, i.e., o menor tempo de execução apresentado pelo sistema de atualização foi pago com maior demanda por memória. Em muitas situações, a quantidade de memória exigida por um programa é o

fator que determina a viabilidade de seu uso. Dada a crescente velocidade com que se aumenta a disponibilidade de memória computacional, a preocupação com economia de memória pode deixar de ser importante em muitas situações (e.g., programas de melhoramento com um grande número de rebanhos pequenos). Isso pode permitir o resgate de muitos métodos computacionais interessantes (e.g., que possibilitam cálculo de PEVs e PEVDs) mas que não eram empregados por limitação de hardware.

## Conclusões

Rotações GG podem ser empregadas em um sistema para atualização de predições de valor genético sob modelo animal reduzido, em que se aninham os efeitos de vacas dentro de rebanhos. A abordagem multi-frontal usada para armazenar uma trapezóide para cada rebanho torna o sistema ágil e faz com que os requerimentos de memória convencional sejam mantidos em níveis baixos.

## Referências

BOLDMAN, K.G.; KRIESE, L.A.; VAN VLECK, L.D.; VAN TASSELL, C.P.; KACHMAN, S.D. **A manual for use of MTDFREML**. A set of programs to obtain estimates of variances and covariances [Draft]. U.S. Department of Agriculture, Agricultural Research Service. 1995.

COLEMAN, T.F.; EDENBRANDT, A.; GILBERT, J.R. Predicting Fill for Sparse Orthogonal Factorization. **Journal of the Association for Computing Machinery**, v.33, n.3, p.517-532. 1986.

CHRISTENSEN, L.G. Basic principles in the estimation of breeding values by direct updating. **Z. Tierzüchtg. Züchtgsbiol.**, v.98, p.132-150. 1981.

DUFF, I.S; REID, J.K. The multifrontal solution of indefinite sparse symmetric linear equations. **ACM Trans. Math. Softw.**, v.9, p.302-325. 1983.

FRIES, L.A. Aggregating ancestral, performance and progeny information into EBVs from a sire & dam model. **Asian-Australasian J. Anim. Sci.**, v.13, p.203-206. 2000.

GENTLEMAN, W.M. Least Squares Computations by Givens Transformations Without Square Roots. **Journal of the Institute of Mathematics and its Applications**, v.12, p.329-336. 1973.

GENTLEMAN, W.M. Algorithm AS 75: Basic procedures for large, sparse or weighted linear least problems. **Applied Statistics**, v.23, p.448-454. 1974.

GEORGE, A.; HEATH, M.T. Solution for sparse linear least squares problems using Givens rotations. **Linear Algebra and Appl.**, v.34, p.69-83. 1980.

GIVENS, W. Numerical Computation of the Characteristic Values of a Real Symmetric Matrix. **Oak Ridge National Laboratory Report**, Oak Ridge, ORNL – 1574, 1954.

JAMROZIK, J.; SCHAEFFER, L.R. Procedures for updating solutions to animal models as data accumulate. **J. Dairy Sci.**, v.74, p.1993-2000. 1991.

KENNEDY, B.W.; TRUS, D. Considerations on genetic connectedness between management units under an animal model. **J. Anim. Sci.**, v.71, p.2341-2352. 1993.

LAWSON, C.L.; HANSON, R.J. **Solving Least Squares Problems**. New Jersey: Prentice-Hall series in automatic computation, 1974. 340p.

MRODE, R.A. **Linear Models for the Prediction of Animal Breeding Values**. Wallingford: CAB International, 1996. 187p.

QUAAS, R.L.; POLLAK, E.J. Mixed model methodology for farm and ranch beef cattle testing programs. **J. Anim. Sci.**, v.51, p.1277-1287. 1980.

ROSO, V.M.; SCHENKEL, F.S.; MILLER, S.P. Degree of connectedness among groups of centrally tested beef bulls. **Can. J. Anim. Sci.**, v.84, p.37-47. 2004.

SCHAEFFER, L.R. **ANSC637 Set 17 – Selection bias**. [online] Disponível na Internet via WWW. URL: http://www.aps.uoguelph.ca/~lrs/ANSC637/LRS17/. 2002.

VAN VLECK, L.D. Absorption of equations for non-parents for an animal model with maternal effects and genetic groups. **J. Anim. Sci.**, v.68, p.4014-4025. 1990.

WAUGH, F.V.; DWYER, P.S. Compact computation of the inverse of a matrix. **The Annals of Mathematical Statistics**, v.16, p.259-271. 1945.

WILKINSON, J.H. **The algebraic eigenvalue problem**. Oxford University Press, 1965.

## CAPÍTULO 3 – USO DE UM MÉTODO NUMÉRICO DIRETO PARA ESTIMAÇÃO DE EFEITOS DE SNPs EM VALORES GENÉTICOS

**RESUMO** – Neste trabalho, consideraram-se as rotações de Gentleman-Givens (GG) como um método numérico direto para fatoração das matrizes de dados em problemas que envolvem estimação de efeitos de polimorfismos em determinados nucleotídeos (SNPs) nos valores genéticos de indivíduos. Possíveis vantagens desse método em relação ao método do Gradiente Conjugado (GC) foram investigadas. Duas situações, com dez repetições de conjuntos de dados cada, foram simuladas para comparação entre os métodos. As situações diferiram entre si com relação ao número de indivíduos genotipados: 2.000 indivíduos foram simulados na situação 1 e 10.000 indivíduos na situação 2. As médias de tempos de processamento foram: 2.210,04 e 286,52 segundos para GC e GG respectivamente, na situação 1; e 36.474,98 e 2.277,78 segundos para GC e GG respectivamente, na situação 2. As médias de requerimentos de memória foram: 103,11 e 356,06 Mbytes para GC e GG respectivamente, na situação 1; e 177,30 e 433,99 Mbytes para GC e GG respectivamente, na situação 2. Se a estimação dos efeitos dos SNPs nos valores genéticos dos indivíduos for feita uma vez por ano ou a cada dois anos, com o propósito de predizer valor genético com informação genômica, então a diferença em tempos de execução apresentada aqui não parece ser de muita importância. Porém, se grande número de análises precisarem ser feitas para testar diferentes modelos e métodos estatísticos a serem empregados no processo de estimação, então pode-se economizar tempo pelo uso das rotações GG. Outros aspectos positivos do método são estabilidade numérica e possibilidade de atualizar as soluções e calcular os erros padrão das estimativas.

**Palavras-Chave:** marcador molecular, método direto, rotações de Givens, seleção genômica

## Introdução

Muitas aplicações em melhoramento animal envolvem a solução de sistemas de equações lineares com um grande número de incógnitas. Quando se usam modelos do tipo modelo animal uni ou multi-característica, por exemplo, o número de parâmetros cresce em função do número de animais sendo avaliados, o que não muito raramente se encontra na casa das centenas de milhares. As matrizes de coeficientes que advém desse tipo de problema são muito grandes para serem armazenadas em memória convencional. Por esse motivo, métodos iterativos, sejam nos dados ou nas equações de modelos mistos, são bastante empregados em tais problemas.

O desenvolvimento de métodos acurados para detecção de variação em seqüências de determinados genes desencadeou uma série de estudos (MEUWISSEN et al., 2001; XU, 2003) que procuram associar tais variações à variação fenotípica em características de interesse econômico. Painéis de alta densidade para genotipagem de milhares de polimorfismos em determinados nucleotídeos (SNPs) encontram-se disponíveis em nível comercial e seus custos tendem a decrescer com o tempo, o que pode tornar a seleção genômica um procedimento viável num futuro não muito distante (SCHAEFFER, 2006). O número de incógnitas, nos modelos considerados para estimação dos efeitos de SNPs no mérito genético total dos indivíduos, é função do número de marcadores genotipados, que deverá ser constante no tempo. Este aspecto e a crescente taxa de avanço em capacidade dos computadores pode tornar o uso de um método direto uma escolha viável para esse tipos de aplicação.

Neste trabalho, consideramos a aplicação de um algoritmo de implementação das rotações modificadas de Givens para fatoração QR das matrizes de dados em análises de estimação de efeitos de SNPs. Possíveis vantagens e limitações desse método em relação a um método iterativo foram investigadas.

## Material e métodos

*Dados*

Os genomas simulados consistiram de 25 cromossomos com 100cM de comprimento cada. Cada um deles continha 400 SNPs e 40 QTLs distribuídos aleatoriamente. As taxas de mutação e de falta de informação sobre o genótipo do marcador foram de 1e-7 e 0,005, respectivamente. Outros parâmetros empregados no processo de simulação são apresentados na Tabela 1.

Tabela 1. Parâmetros usados na simulação

| | |
|---|---|
| Herdabilidade | 0,3 |
| Variância fenotípica | 10,0 |
| Número de progênies por parto | 1 |
| Número de gerações | 6 |
| Taxa de reposição de machos | 50% |
| Taxa de reposição de fêmeas | 20% |
| Critério de seleção | EBV |

Duas situações, com dez repetições de conjuntos de dados cada, foram simuladas•. As situações diferiram entre si com relação ao número de indivíduos genotipados na última geração: 2.000 indivíduos foram simulados na situação 1 e 10.000 indivíduos na situação 2. Em cada repetição, excluíram-se das análises os marcadores cuja menor freqüência alélica era inferior a 0,1.

*Procedimento estatístico*

Um modelo de regressão linear múltipla (ZENG, 1993; XU, 2003) foi empregado na estimação dos efeitos dos SNPs nos méritos genéticos dos indivíduos. A equação do modelo é descrita a seguir:

$$\mathbf{y_i = \mu + \sum_{j=1}^{p} x_{ij} b_j + e_i} \qquad [1]$$

em que:

$\mathbf{y_i}$ é o valor genético predito do i-ésimo animal;

$\boldsymbol{\mu}$ é uma média geral;

$\mathbf{x_{ij}}$ é o coeficiente correspondente ao genótipo do j-ésimo SNP no i-ésimo animal;

$\mathbf{b_j}$ é o regressor do efeito do j-ésimo SNP;

**p** é o número de SNPs genotipados;

$\mathbf{e_i}$ é um termo residual aleatório.

Os coeficientes $x_{ij}$ foram definidos como 0 para o genótipo $A_1A_1$, 1 para o genótipo $A_1A_2$ e 2 para o genótipo $A_2A_2$. De acordo com ZENG (1993), os regresores $b_j$ levam em conta os efeitos de todos os QTLs entre os marcadores j-1 e j+1. Aqui não se faz nenhuma pressuposição a respeito das posições dos QTLs, mas pressupõe-se que as relações entre os efeitos dos genótipos de todos os marcadores são estritamente aditivas. Nesse sentido, a região do genoma representada por um dado marcador pode ser interpretada praticamente como um cromossomo e os $b_j$ como estimativas dos efeitos médios de substituição alélica nos marcadores.

Quando o número de marcadores é maior que o número de indivíduos genotipados, mínimos quadrados ordinários ou ponderados não se prestam à estimação dos coeficientes de regressão. Algum outro método de estimação como mínimos quadrados parciais, componentes principais, regressão de cumeeira ou alguma forma de procedimento bayesiano, deve então ser empregado. A questão acerca de qual procedimento estatístico, ou mesmo modelo, a empregar ainda não atingiu um consenso. Um método de regressão de cumeeira foi empregado.

---

• A simulação foi feita por meio de programa desenvolvido por SARGOLZAEI & SCHENKEL (não publicado).

Montadas as equações normais, adiciona-se uma matriz de parâmetros de cumeeira à mão esquerda das equações para que se possa resolver o sistema. Em notação matricial, as equações a resolver são:

$$\begin{bmatrix} \iota^t W \iota & \iota^t W X \\ X^t W \iota & X^t W X + \Phi \end{bmatrix} \begin{bmatrix} \mu \\ \beta \end{bmatrix} = \begin{bmatrix} \iota^t W y \\ X^t W y \end{bmatrix} \qquad [2]$$

em que:

$\iota$ é um vetor de 1s, de ordem igual ao número de indivíduos genotipados;

$W$ é uma matriz diagonal com $w_{ii}$ igual à acurácia na medida do EBV do i-ésimo indivíduo;

$\Phi$ é uma matriz de parâmetros de cumeeira.

A maior motivação para o uso de um procedimento por regressão de cumeeira é geralmente a presença de multicolinearidade, i.e., dependências lineares entre as variáveis explanatórias. No presente caso porém, a adição de quantidades à diagonal da matriz de coeficientes foi feita com o propósito de incorporar informação suficiente ao sistema para que se pudesse resolvê-lo. Por questão de simplicidade, e já que o objetivo deste trabalho foi a comparação de métodos numéricos e não estatísticos, as matrizes Φ e W adotadas aqui foram duas matrizes identidade.

*Métodos numéricos*

O método direto empregado neste estudo foi um algoritmo para implementação de decomposição ortogonal (QR) por linhas, conhecido como Rotações de Givens (GIVENS, 1954). De acordo com BAI et al. (2001), transformações ortogonais são bastante robustas e numericamente estáveis. O procedimento padrão de rotações de Givens requerem uma operação de radiciação a cada rotação e quatro de multiplicação para cada par de elementos fora da diagonal nas linhas pivô (r) e de trabalho (x):

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} r_1 & \cdots & r_k \\ x_1 & \cdots & x_k \end{bmatrix}=\begin{bmatrix} r'_1 & \cdots & r'_k \\ 0 & \cdots & x'_k \end{bmatrix}$$

$$c=\frac{r_1}{\sqrt{r_1^2+x_1^2}} \qquad s=\frac{x_1}{\sqrt{r_1^2+x_1^2}} \qquad [3]$$

$$r'_i = cr_i + sx_i$$

$$x'_i = -sr_i + cx_i$$

Seguindo o mesmo princípio que deu origem às versões livres de radiciação da decomposição de Cholesky e da ortogonalização de Gram-Schmidt, GENTLEMAN, (1973) propôs uma modificação no procedimento com objetivo de economizar a operação de radiciação em [3]. A modificação consiste em se chegar não a uma matriz R, mas sim uma matriz diagonal D e uma matriz triangular superior unitária $\overline{R}$ tais que:

$$R = D^{0.5}\overline{R} \qquad [4]$$

O algoritmo modificado então rota uma linha do produto $D^{0.5}\overline{R}$ com uma linha escalonada de X:

$$\begin{matrix} 0 & \cdots & 0 & \sqrt{d} & \cdots & \sqrt{d}\bar{r}_k & \cdots \\ 0 & \cdots & 0 & \sqrt{\delta}x_i & \cdots & \sqrt{\delta}x_k & \cdots \end{matrix}$$

E as linhas transformadas podem ser escritas como:

$$\begin{matrix} 0 & \cdots & 0 & \sqrt{d'} & \cdots & \sqrt{d'}\bar{r}'_k & \cdots \\ 0 & \cdots & 0 & 0 & \cdots & \sqrt{\delta'}x'_k & \cdots \end{matrix}$$

em que:

$$d' = d + \delta x_i^2$$

$$\delta' = d\delta / d' \qquad [5]$$

$$\bar{c} = d/d' \qquad \bar{s} = \delta x_i / d'$$

$$x'_k = x_k - x_i\bar{r}_k \qquad \bar{r}'_k = \bar{c}\bar{r}_k + \bar{s}x_k$$

Além de evitar a operação de radiciação em [3], as fórmulas em [5] também requerem apenas três multiplicações para cada par de elementos envolvidos na rotação. Na verdade, com uma pequena modificação adicional, uma outra multiplicação ainda poderia ser evitada, mas isso comprometeria a estabilidade numérica (BAI et al., 2001).

Como o método implica a aplicação de uma seqüência de rotações planares nas linhas da matriz de dados, a montagem das equações normais não é necessária. Deve-se então aplicar algumas pré-multiplicações e adicionar algumas pseudo-observações ao sistema de modo que o processo se torne equivalente à montagem da equações em [2]. A matriz de dados se torna:

$$\begin{Bmatrix} \mathbf{W}^{0.5}\mathbf{\iota} & \mathbf{W}^{0.5}\mathbf{X} \\ \tilde{\mathbf{0}} & \mathbf{\Phi}^{0.5} \end{Bmatrix} \begin{Vmatrix} \mathbf{W}^{0.5}\mathbf{y} \\ \tilde{\mathbf{0}} \end{Vmatrix} \qquad [6]$$

Essa foi então a matriz de dados fatorada no produto QR, usando o algoritmo para aplicação das rotações de Gentleman-Givens (GG) apresentado em GENTLEMAN (1974).

O método usado como referência para avaliar o desempenho das rotações GG foi o método do Gradiente Conjugado (GC), um método bastante conhecido e empregado em trabalhos de melhoramento animal (e.g., LIDAUER et al., 1999; TSURUTA et al., 2001; STRANDÉN et al., 2002).

**Resultados e discussão**

Nas Tabelas 2, 3, 4 e 5 são apresentados o tempo total de execução e demanda por memória de alta velocidade, bem como o número de marcadores excluídos em cada repetição, pelos métodos GC e GG em ambas situações simuladas.

Tabela 2. Tempos de execução e demanda por memória no método GC, situação 1

| Repetição | Nº de SNPs excluídos | Tempo de CPU (segundos) | Memória (Mb) |
|---|---|---|---|
| 1 | 592 | 2123,74 | 103,16 |
| 2 | 637 | 2190,76 | 102,27 |
| 3 | 586 | 2090,78 | 103,28 |
| 4 | 607 | 2161,80 | 102,86 |
| 5 | 608 | 2209,52 | 102,84 |
| 6 | 580 | 2243,49 | 103,40 |
| 7 | 615 | 2174,20 | 102,70 |
| 8 | 573 | 2244,24 | 103,54 |
| 9 | 560 | 2352,50 | 103,80 |
| 10 | 590 | 2309,37 | 103,20 |

Tabela 3. Tempos de execução e demanda por memória no método GG, situação 1

| Repetição | Nº de SNPs excluídos | Tempo de CPU (segundos) | Memória (Mb) |
|---|---|---|---|
| 1 | 592 | 285,72 | 356,26 |
| 2 | 637 | 280,10 | 352,95 |
| 3 | 586 | 281,04 | 356,70 |
| 4 | 607 | 279,87 | 355,15 |
| 5 | 608 | 294,13 | 355,08 |
| 6 | 580 | 293,21 | 357,14 |
| 7 | 615 | 272,97 | 354,56 |
| 8 | 573 | 284,94 | 357,66 |
| 9 | 560 | 283,47 | 358,62 |
| 10 | 590 | 309,77 | 356,40 |

Tabela 4. Tempos de execução e demanda por memória no método GC, situação 2

| Repetição | Nº de SNPs excluídos | Tempo de CPU (segundos) | Memória (Mb) |
|---|---|---|---|
| 1 | 545 | 41260,58 | 176,76 |
| 2 | 513 | 45687,80 | 177,65 |
| 3 | 495 | 31446,43 | 178,15 |
| 4 | 538 | 32937,64 | 176,96 |
| 5 | 620 | 30319,18 | 174,69 |
| 6 | 509 | 40981,07 | 177,76 |
| 7 | 538 | 32362,40 | 176,96 |
| 8 | 501 | 34615,64 | 177,98 |
| 9 | 516 | 40065,44 | 177,56 |
| 10 | 481 | 35073,60 | 178,53 |

Tabela 5. Tempos de execução e demanda por memória no método GG, situação 2

| Repetição | Nº de SNPs excluídos | Tempo de CPU (segundos) | Memória (Mb) |
|---|---|---|---|
| 1 | 545 | 2493,76 | 432,39 |
| 2 | 513 | 2246,41 | 435,01 |
| 3 | 495 | 2148,43 | 436,49 |
| 4 | 538 | 2278,43 | 432,97 |
| 5 | 620 | 2155,16 | 426,28 |
| 6 | 509 | 2213,64 | 435,34 |
| 7 | 538 | 2292,55 | 432,97 |
| 8 | 501 | 2328,81 | 436,00 |
| 9 | 516 | 2300,33 | 434,77 |
| 10 | 481 | 2320,24 | 437,64 |

As médias de tempos de execução foram: 2.210,04 e 286,52 segundos para GC e GG respectivamente, na situação 1; e 36.474,98 e 2.277,78 segundos para GC e GG respectivamente, na situação 2. As médias de requerimentos de memória foram: 103,11 e 356,06 Mb para GC e GG respectivamente, na situação 1; e 177,30 and 433,99 Mb para GC e GG respectivamente, na situação 2.

A principal desvantagem verificada na aplicação de GG foi a maior quantidade de memória requerida para as análises. No algoritmo implementado neste trabalho, a matriz triangular superior R foi armazenada de forma vetorizada, requerendo [np*(np+1)]/2 posições de memória em um arranjo de tipo real (8 bytes), em que np é o número de parâmetros no modelo. Na implementação de GC, a mão esquerda das equações pode ser armazenada em um arranjo de mesma dimensão, mas de tipo inteiro curto (2 bytes). Note que os coeficientes em X apenas assumem valor 0, 1 ou 2 e que ambas as matrizes, a de coeficientes em GC e a triangular superior em GG, são densas. Se considerarmos um painel de 10.000 SNPs, o requerimento em memória de alta velocidade vai de ~95Mb no método GC para ~381Mb na implementação de GG.

Com o contínuo aumento em capacidade computacional, essa diferença tende a se tornar cada vez menos importante no futuro, especialmente porque uma vez definido um modelo apropriado e um conjunto de SNPs a ser considerado, o número de parâmetros tende a ser constante no tempo.

Se a estimação dos efeitos dos SNPs precisar ser feita uma vez ao ano ou a cada dois anos, então as diferenças em tempos de processamento aqui apresentadas também não parecem ser de grande importância. Contudo, se uma intensa série de análises precisar ser conduzida para testar diferentes modelos e métodos estatísticos para definição de procedimentos padrão na estimação de efeitos de SNPs, então pode-se economizar tempo pelo uso do método GG.

Uma propriedade importante de um algoritmo como GG para fatoração QR por linhas é que ele permite tirar total vantagem de operações aritméticas já feitas. As soluções podem ser atualizadas sem a necessidade de processar novamente linhas de dados já rotadas em passos anteriores. O procedimento é ilustrado na Figura 1.

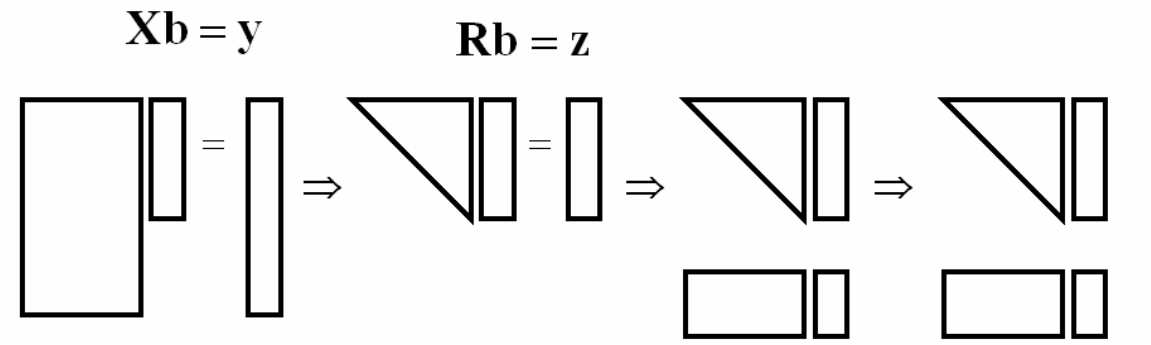


Figura 1. Esquema de atualização em fatoração QR por linhas

Após rotar um conjunto inicial de linhas de dados do sistema **Xb=y**, a matriz triangular superior $\mathbf{R=Q^tX}$ e a mão direta transformada $\mathbf{z=Q^ty}$ podem ser armazenadas em arquivo para posteriores atualizações. Uma vez que se disponibilize informação de novos indivíduos genotipados, apenas essas novas observações precisam ser rotadas com R e as soluções atualizadas podem ser obtidas por retro-substituição no sistema triangular **Rb=z**. Apenas um sistema triangular precisa ser resolvido porque as transformações ortogonais são também aplicadas à mão direita das equações.

Além da adição de novas observações, a exclusão de observações do sistema também é possível. As informações podem ser excluídas rotando-se as linhas de dados com os mesmos pesos atribuídos quando de suas inclusões, porém com sinal trocado. Todavia, de acordo com GENTLEMAN (1973), qualquer método para remoção de observações do sistema é potencialmente instável e deve ser feito com precaução. Neste trabalho, não incluíram-se rotinas para atualizações nas comparações entre os métodos porque o método GG se mostrou superior em tempo de processamento mesmo considerando-se a fatoração do sistema inteiro.

Conforme mencionado anteriormente, definições a respeito do modelo e do método estatístico mais apropriados a serem empregados na estimação dos efeitos dos SNPs nos valores genéticos ainda não estão próximas de um padrão a seguir. Inúmeras abordagens diferentes ainda devem ser testadas e se um método numérico puder representar uma vantagem relevante em termos de tempo de execução, esse processo de testes e experimentações pode ser bastante otimizado. O método GG

pode facilmente acomodar diferentes maneiras de definição da informação externa a ser incorporada à análise, bastando para tanto montar a matriz Φ correspondente a ser considerada em [6].

Uma vez que os efeitos dos SNPs tenham sido acuradamente estimados, então valores genéticos preditos com base em informação de marcadores do genoma inteiro (MEUWISSEN et al., 2001), chamados GEBVs por SCHAEFFER (2006), podem ser calculados como a soma dos efeitos estimados dos genótipos (ou haplótipos, dependendo da abordagem) correspondentes aos genótipos observados em todos os marcadores, para cada indivíduo. Considerando-se o modelo de regressão de XU (2003), então o vetor $\hat{\mathbf{g}}$ correspondente aos GEBVs pode ser predito como:

$$\hat{\mathbf{g}} = \mathbf{X}\hat{\mathbf{b}} \qquad [7]$$

com:

$$\mathbf{Var}(\hat{\mathbf{g}}) = \mathbf{diag}[\mathbf{X}(\mathbf{X}^{\mathbf{t}}\mathbf{X})^{-1}\hat{\sigma}_{\mathbf{e}}^{2}\mathbf{X}^{\mathbf{t}}] \qquad [8]$$

Uma outra propriedade de transformações ortogonais é que a matriz triangular superior R obtida por fatoração QR na matriz de dados é equivalente ao fator de Cholesky da matriz de coeficientes $X^tX$ das equações normais. Dada essa propriedade, é possível calcular os elementos de $(X^tX)^{-1}$, necessários em [8], diretamente da matriz R sem que seja preciso montar explicitamente o sistema de equações normais. Como $X^tX = R^tR$ então $(X^tX)^{-1} = R^{-1}R^{-t}$. Para simplificar a notação, chamemos a matriz $(X^tX)^{-1}$ de M, com suas colunas $m_1$, $m_2$,… $m_n$. Temos que $RM = R^{-t}$. Então, como R é triangular superior e $R^{-t}$ é triangular inferior, e notando que o i-ésimo elemento da diagonal de $R^{-t}$ é o recíproco do i-ésimo elemento da diagonal de R, é possível computar os elementos das colunas $m_n$ até $m_1$ do último para o primeiro, uma coluna por vez. Essa demonstração é dada em WAUGH & DWYER (1945).

Para ilustrar como funciona o algoritmo, apresenta-se na Figura 2 uma subrotina para implementação do procedimento acima descrito, escrita em linguagem Fortran 77. As variáveis R, RTRINV, NCOL e MAXCOL são: o fator de Cholesky, a inversa, o número de colunas em R e o número máximo de colunas em R, respectivamente.

```
      SUBROUTINE INVERSA(R,RTRINV,NCOL,MAXCOL)
C=============================================
C
      DOUBLE PRECISION  R(MAXCOL,*) , RTRINV(MAXCOL,*)
      DOUBLE PRECISION  AUX1, AUX2
      INTEGER NCOL
C
C
C     ------------------------------------------------------
C     Calculo da inversa de X'X = R'R, considerando  X = QR
C     ------------------------------------------------------
C
C
      DO I = NCOL , 1 , -1
           AUX1 = 0.0D0
           IF (I.LT.NCOL) THEN
                DO L = I+1 , NCOL
                     AUX1 = AUX1 + R(I,L)*RTRINV(I,L)
                END DO
           END IF
           RTRINV(I,I) = (1/R(I,I) - AUX1)/R(I,I)
           DO J = I-1 , 1 , -1
                AUX2 = 0.0D0
                DO K = J+1 , NCOL
                     AUX2 = AUX2 - R(J,K)*RTRINV(K,I)
                END DO
                RTRINV(J,I) = AUX2/R(J,J)
                RTRINV(I,J) = RTRINV(J,I)
           END DO
      END DO
C
C
      RETURN
      END
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
```

Figura 2. Subrotina em Fortran 77 para cálculo da inversa da “mão esquerda”

É evidente que nenhum do dois arranjos definidos no código precisam ser declarados como matrizes, dado que ambas são simétricas. Na subrotina acima

apresentada, tanto o fator de Cholesky quanto a inversa são definidos em arranjos bi-dimensionais apenas com o intuito de ilustrar o algoritmo com maior clareza.

## Conclusões

Rotações GG demandaram menos tempo de processamento e mais memória para armazenamento de dados que o método GC. Essas diferenças podem ser de pouca importância se: i) as exigências de memória para GG puderem ser facilmente supridas; ii) as análises não forem feitas com muita freqüência.

Alguns aspectos positivos do método direto são: estabilidade numérica e a possibilidade de atualizar as soluções e calcular os erros padrão das estimativas.

## Referências

BAI, Z.Z.; DUFF, I.S.; WATHEN, A.J. A class of incomplete orthogonal factorization methods. I: methods and theories. **BIT Numerical Mathematics**, v.41, n.1, p.53-70. 2001.

GENTLEMAN, W.M. Least Squares Computations by Givens Transformations Without Square Roots. **Journal of the Institute of Mathematics and its Applications**, v.12, p.329-336. 1973.

GENTLEMAN, W.M. Algorithm AS 75: Basic procedures for large, sparse or weighted linear least problems. **Applied Statistics**, v.23, p.448-454. 1974.

GIVENS, W. Numerical Computation of the Characteristic Values of a Real Symmetric Matrix. **Oak Ridge National Laboratory Report**, Oak Ridge, ORNL – 1574, 1954.

LIDAUER, M.; STRANDÉN, I.; MÄNTYSAARI, E.A. *et al*. Solving Large Test-Day Models by Iteration on Data and Preconditioned Conjugate Gradient. **Journal of Dairy Science**, v.82, p.2788-2796. 1999.

MEUWISSEN, T.H.E.; HAYES, B.J.; GODDARD, M.E. Prediction of total genetic value using genome-wide dense marker maps. **Genetics**, v.157, p.1819-1829. 2001.

SCHAEFFER, L.R. Strategy for applying genome-wide selection in dairy cattle. **J. Anim. Breed. Genet.**, v.123, p.218-223. 2006.

STRANDÉN, I.; TSURUTA, S.; MISZTAL, I. Simple preconditioners for the conjugate gradient method: experience with test day models. **Journal of Animal Breeding and Genetics**, v.119, p.166-174. 2002.

TSURUTA, S.; MISZTAL, I.; STRANDÉN, I. Use of the preconditioned conjugate gradient algorithm as a generic solver for mixed-model equations in animal breeding applications. **Journal of Animal Science**, v.79, p.1166-1172. 2001.

WAUGH, F.V.; DWYER, P.S. Compact computation of the inverse of a matrix. **The Annals of Mathematical Statistics**, v.16, p.259-271. 1945.

XU, S. Estimating polygenic effects using markers of the entire genome. **Genetics**, v.163, p.789-801. 2003.

ZENG, Z-B. Theoretical basis for separation of multiple linked gene effects in mapping quantitative trait loci. **Proc. Natl. Acad. Sci. USA**, v.90, p.10972-10976. 1993.

## CAPÍTULO 4 – IMPLICAÇÕES

Na seção 'Resultados e discussão' do capítulo 2, embora tenham sido apresentados, comparados e discutidos os tempos de execução do sistema de atualização e do programa MTDFREML, cabe salientar aqui que a intenção do estudo não foi de comparar softwares com relação a tempo de processamento. Mesmo porque o MTDFREML é um sistema extremamente eficiente e rápido na solução das MME, além de desempenhar uma série de outras funções (e.g., estimação de componentes de variância). A condução das análises por meio dos dois programas teve por finalidade a validação dos resultados e uma avaliação do desempenho do sistema, tomando o MTDFREML como referência.

Vários outros programas encontram-se disponíveis (livres ou sob licença) para estimação de componentes de variância e/ou solução das MME. Como exemplos, podem-se citar os seguintes programas ou pacotes: **ASReml** (Arthur Gilmour et al., disponível sob licença em http://www.vsni.co.uk/products/asreml/), **VCE/PEST** (Eildert Groeneveld, disponível em http://w3.tzv.fal.de/~eg/), **WOMBAT** (Karin Meyer, disponível em http://agbu.une.edu.au/~kmeyer/wombat.html), **DMU** (Per Madsen & Just Jensen, disponível em ftp://genetics.agrsci.dk/pub/dmu/dmu.tar.gz), coleção **BLUPF90** (Ignacy Misztal et al., disponíveis em http://nce.ads.uga.edu/~ignacy/newprograms.html). Esses e outros programas foram (e vêm sendo) desenvolvidos com propósito relativamente amplo, apresentando flexibilidade em relação a modelos e número de características.

O sistema apresentado no capítulo 2 foi desenvolvido com uma aplicação bem específica em mente: modelo animal reduzido e análise uni-característica. Para essa aplicação específica, o sistema apresenta algumas propriedades interessantes. É possível e relativamente fácil adaptar o sistema a um modelo touro e vaca, porém a viabilidade de modificá-lo de forma a lidar com outros modelos mais complexos e/ou análises multi-característica precisa ser melhor estudada.

O maior benefício que se pode ter pelo emprego das rotações GG na predição de EBVs é a possibilidade de cálculo (e não apenas aproximação) das acurácias e dos PEVs das diferenças entre EBVs. A maior limitação ao emprego do método na predição

de EBVs parece ser a necessidade de pressupor vacas aninhadas em rebanhos. Nas situações em que essa não for uma pressuposição aceitável, a abordagem multi-frontal delineada neste trabalho deixa de ser possível, comprometendo a viabilidade de predição do padrão de esparsidade e, conseqüentemente, o emprego do método.

Nesta tese, consideraram-se duas aplicações em melhoramento animal em que se testou a viabilidade do emprego das rotações GG. O método pode contudo se mostrar interessante em uma série de outras aplicações. Dada a maneira como se processam as observações (por linhas) e como se armazenam os dados (em uma triangular superior de ordem igual ao número de parâmetros), fica bem claro que uma aplicação bem adequada ao emprego do método é o caso em que se usa um número muito grande de observações para estimar um número não muito grande de parâmetros. Ou seja, na solução de um sistema retangular de equações lineares bastante “fino” e “alto”, e que gera um fator de Cholesky denso. Nessa linha, cabe citar a experiência do autor com o emprego de rotações de Givens na estimação de efeitos genéticos em animais cruzados, usando regressão de cumeeira. Maiores detalhes sobre essa aplicação podem ser encontrados na dissertação de mestrado do autor, defendida nesta mesma unidade.

Vantagem maior se pode tirar ainda do método se novas observações nos mesmos parâmetros se tornam disponíveis ao longo do tempo. Ou seja, o sistema de equações não precisa ser necessariamente “fino” e “alto” no início, mas assim se tornar ao longo do tempo, pelo acúmulo de observações. Essa foi a idéia que inspirou o emprego do método na aplicação descrita no capítulo 3.

*Alguns pontos a serem melhorados/aperfeiçoados no sistema de atualização*

- No estudo reportado no capítulo 2, o arquivo de dados utilizado apresentava informação completa de parentesco, ou seja, todos os animais com fenótipo medido tinham pai e mãe conhecidos. Essa não é nem de perto uma situação realística. Portanto, uma modificação importante a ser implementada no sistema é a adaptação dos programas de modo a contemplar informação perdida de parentesco.

- A recodificação dos animais é feita por meio de buscas lineares em arquivos contendo os códigos originais dos animais que já apareceram no sistema. Embora não se tenha feito formalmente um perfil de desempenho dos programas, é de se esperar que muito se ganhe pela adoção de uma estratégia mais eficiente de busca (e.g., busca binária).

**APÊNDICE A** – PREV.f90: código fonte em Fortran 90 para criar arquivos de armazenagem da triangular superior ‘touro x touro’ e gerar arquivo de parâmetros a serem usados pelo sistema de atualização.

```
!
!     PREV.f90 (Genetic Evaluations with Gentleman-Givens rotations)
!
!     By Eduardo CG Pimentel           10/23/2007
!     E-mail: pimentel@fcav.unesp.br
!
!     ------------------------------------------------------------------
!     ------------------------------------------------------------------
!
      program main
      implicit none
!
!     ------------------------------------------------------------------
!
      character(22) :: skip
!
      integer :: i,np,nrbar,rec_nr
!
      real(8) :: zero,one
!
      integer, parameter :: nherd=8
!
!     ------------------------------------------------------------------
!
      open(10,file='PARIN.dat',status='old')
      open(20,file='DIAG_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='unknown')
      open(30,file='RBAR_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='unknown')
      open(40,file='THET_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='unknown')
      open(50,file='info.txt',status='unknown')
!
!     ------------------------------------------------------------------
!
      read(10,*) skip
      read(10,*) skip
      read(10,*) skip
      read(10,*) np
!
!     ------------------------------------------------------------------
!
      data zero /0.0d0/
      data one /1.0d0/
      nrbar=(np*(np-1))/2
!
      rec_nr=0
      do i=1,np
         rec_nr=rec_nr+1
         write(20,REC=rec_nr) one   ! G* for base sires
         write(40,REC=rec_nr) zero
      end do
!
      rec_nr=0
```

```
      do i=1,nrbar
         rec_nr=rec_nr+1
         write(30,REC=rec_nr) zero
      end do
!
      rec_nr=0
      write(50,'(i40,1x,a35)') nherd,'  ! Total number of herds       '
      write(50,'(f40.30,1x,a35)') zero,'! Residual sum of squares     '
      write(50,'(i40,1x,a35)') np,'     ! Total number of observations'
      write(50,'(i40,1x,a35)') rec_nr,' ! N of males with info        '
!
      stop
      end program main
!
!     ----------------------------------------------------------------------
!     ----------------------------------------------------------------------
!
```

**APÊNDICE B** – INIT.f90: código fonte em Fortran 90 para inicializar cada rebanho no sistema de atualização. Cria a trapezóide correspondente ao rebanho e atualiza a triangular superior ‘touro x touro’.

```
!
!     INIT.f90 (Genetic Evaluations with Gentleman-Givens rotations)
!
!     By Eduardo CG Pimentel           10/23/2007
!     E-mail: pimentel@fcav.unesp.br
!
!     ---------------------------------------------------------------
!     ---------------------------------------------------------------
!
      program main
      implicit none
!
!     ---------------------------------------------------------------
!
      character(20) :: data_file,daid_file,cgid_file,herd_file
      character(20) :: rbar_file,diag_file,thet_file,fown_file
      character(10) :: date,time,zone,herd
      character(15) :: an,si,da
      character(1) :: gend
!
      integer :: i,j,k,nrec,gr,nsire,maxid,np,nrbar,ifault,count,isisi
      integer :: countdam,countcg,nobs,nownif,nownim,nherd,rec_nr
      integer :: ival(8),fval(8)
!
      real(4) :: cputime_ini,cputime_fin,cputime
      real(8) :: ob,weight,yelem,sserr
!
      character(15), allocatable :: anim(:),sire(:),dam(:),si_orID(:)
      character(1), allocatable :: gender(:)
!
      integer, allocatable :: codanim(:),codsire(:),coddam(:)
      integer, allocatable :: cg(:),codcg(:),Gdam(:)
!
      real(8), allocatable :: obs(:),xrow(:),d(:),rbar(:),thetab(:)
!
!     ---------------------------------------------------------------
!
      interface includ
         subroutine includ(np,nrbar,weight,xrow,yelem,&
                           d,rbar,thetab,sserr,ifault)
            integer, intent(in) :: np,nrbar
            integer, intent(out) :: ifault
            real(8), intent(in) :: weight,yelem
            real(8), intent(inout) :: sserr
            real(8), intent(inout) :: xrow(:),d(:),rbar(:),thetab(:)
         end subroutine includ
      end interface includ
!
!     ---------------------------------------------------------------
!
      call cpu_time(cputime_ini)
      call date_and_time (date,time,zone,ival)
!
!     ---------------------------------------------------------------
```

```
!
      open(10,file='PARIN.dat',status='old')
!
!     ------------------------------------------------------------------
!
      read(10,*) herd
      read(10,*) data_file
      read(10,*) nrec
      read(10,*) nsire
      close(10)
!
      herd=adjustl(herd)
      daid_file='DAID_'//trim(herd)//'.cod'
      cgid_file='CGID_'//trim(herd)//'.cod'
      herd_file='HERD_'//trim(herd)//'.par'
      rbar_file='RBAR_'//trim(herd)//'.bin'
      diag_file='DIAG_'//trim(herd)//'.bin'
      thet_file='THET_'//trim(herd)//'.bin'
      fown_file='FOWN_'//trim(herd)//'.bin'
!
!     ------------------------------------------------------------------
!
      open(20,file=data_file,status='old')
      open(30,file=daid_file,status='new')
      open(40,file=cgid_file,status='new')
      open(50,file=herd_file,status='new')
      open(60,file=rbar_file,access='direct',&
           recl=8,form='unformatted',status='new')
      open(70,file=diag_file,access='direct',&
           recl=8,form='unformatted',status='new')
      open(80,file='SIRES.cod',status='old')
      open(90,file='RBAR_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(100,file='DIAG_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(110,file=fown_file,access='direct',&
           recl=35,form='unformatted',status='new')
      open(120,file='MALE_PROD.bin',access='direct',&
           recl=45,form='unformatted',status='unknown')
      open(130,file='THET_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(140,file=thet_file,access='direct',&
           recl=8,form='unformatted',status='new')
      open(150,file='info.txt',status='old')
!
!     ------------------------------------------------------------------
!
      read(150,'(i40)') nherd     ! Total number of herds
      read(150,'(f40.30)') sserr  ! Residual sum of squares
      read(150,'(i40)') nobs    ! Total number of observations
      read(150,'(i40)') nownim        ! N of males with info
      nownif=0
      close(150)
!
```

```
      allocate(anim(1:nrec),sire(1:nrec),dam(1:nrec))
      allocate(cg(1:nrec),gender(1:nrec),obs(1:nrec))
!
      print *,' '
      print *,' '
      print *,'    Reading input data file...  '
      print *,' '
!
      count=0
 100  read(20,*,end=110) an,si,da,gr,gend,ob
      count=count+1
      anim(count)=an; sire(count)=si; dam(count)=da; cg(count)=gr
      gender(count)=gend; obs(count)=ob
      go to 100
 110  continue
      close(20)
!
      print *,' '
      print *,'    Recoding dams, groups and sires...  '
      print *,' '
!
      maxid=0; countdam=0; countcg=0
!
      allocate(coddam(1:nrec),codanim(1:nrec))
      allocate(codcg(1:nrec),codsire(1:nrec),Gdam(1:nrec))
      coddam=0; codanim=0; codcg=0; codsire=0; Gdam=1
!
! --> Recoding dam IDs <--
!
      OUTER_1: do i=1,nrec
                  maxid=maxid+1
                  coddam(i)=maxid
                  INNER1_1: do j=1,i-1
                              if(dam(j)==dam(i))then
                                coddam(i)=coddam(j)
                                Gdam(i)=0
                                maxid=maxid-1
                                cycle OUTER_1
                              end if
                            end do INNER1_1
                  INNER2_1: do k=1,nrec
                              if(anim(k)==dam(i))then
                                codanim(k)=coddam(i)
                                Gdam(i)=0
                                exit INNER2_1
                              end if
                            end do INNER2_1
                  write(30,'(a15)') dam(i)
               end do OUTER_1
!
      close(30)
!
      countdam=maxid
      maxid=0
```

```
!
! --> Recoding group IDs <--
!
      OUTER_2: do i=1,nrec
                  maxid=maxid+1
                  codcg(i)=maxid
                  INNER_2: do j=1,i-1
                              if(cg(j)==cg(i))then
                                codcg(i)=codcg(j)
                                maxid=maxid-1
                                cycle OUTER_2
                              end if
                           end do INNER_2
                  write(40,'(i10)') cg(i)
               end do OUTER_2
!
      countcg=maxid
      close(40)
!
      allocate(si_orID(1:nsire))
!
      do i=1,nsire
         read(80,'(a15)') si_orID(i)
      end do
      close(80)
!
! --> Recoding sire IDs <--
!
      OUTER_3: do i=1,nrec
                  INNER1_3: do j=1,nsire
                              if(si_orID(j)==sire(i))then
                                codsire(i)=j
                                exit INNER1_3
                              end if
                           end do INNER1_3
                  INNER2_3: do k=1,nsire
                              if(si_orID(k)==anim(i))then
                                codanim(i)=k
                                exit INNER2_3
                              end if
                           end do INNER2_3
               end do OUTER_3
!
! --> Storing IDs of products and their parents for further pseudos <--
!
      do i=1,nrec
         if(gender(i)=='M'.and.codanim(i)==0)then
           nownim=nownim+1
           write(120,REC=nownim) &
           anim(i),codsire(i),coddam(i),codcg(i),obs(i),herd
         end if
         if(gender(i)=='F'.and.codanim(i)==0)then
           nownif=nownif+1
           write(110,REC=nownif) &
```

```
            anim(i),codsire(i),coddam(i),codcg(i),obs(i)
          end if
       end do
       close(110)
       close(120)
!
       deallocate (anim,sire,dam,cg,gender,si_orID)
!
! --> Creating this herd's part in the upper triangular <--
!
       np=countdam+countcg+nsire
       nrbar=(np*(np-1))/2
       allocate(xrow(1:np),d(1:np),thetab(1:np),rbar(1:nrbar))
       xrow=0.0d0; d=0.0d0; thetab=0.0d0; rbar=0.0d0
!
! --> Retrieving 'sire x sire' part of the upper triangular <--
!
       isisi=(countdam+countcg)*(np+np-(countdam+countcg+1))/2+1
       rec_nr=0
       do i=isisi,nrbar
          rec_nr=rec_nr+1
          read(90,REC=rec_nr) rbar(i)
       end do
!
       rec_nr=0
       do j=countdam+countcg+1,np
          rec_nr=rec_nr+1
          read(100,REC=rec_nr) d(j)
          read(130,REC=rec_nr) thetab(j)
       end do
!
       close(90)
       close(100)
       close(130)
!
       print *,' '
       print *,'    Rotating current records (updating R)...  '
       print *,' '
!
       count=0
       do i=1,nrec
          weight=2.0d0/3.0d0
          xrow=0.0d0; yelem=obs(i)
          xrow(countdam+codcg(i))=1.0d0
          xrow(coddam(i))=0.5d0; xrow(countdam+countcg+codsire(i))=0.5d0
          call includ (np,nrbar,weight,xrow,yelem,&
                       d,rbar,thetab,sserr,ifault)
          count=count+1
          if(codanim(i)/=0.and.gender(i)=='M')then
            weight=3.0d0
            xrow=0.0d0; yelem=obs(i)/3.0d0
            xrow(countdam+codcg(i))=1.0d0/3.0d0
            xrow(countdam+countcg+codanim(i))=1.0d0
            xrow(coddam(i))=-1.0d0/3.0d0
```

```
            xrow(countdam+countcg+codsire(i))=-1.0d0/3.0d0
            call includ (np,nrbar,weight,xrow,yelem,&
                         d,rbar,thetab,sserr,ifault)
            count=count+1
          end if
          if(codanim(i)/=0.and.gender(i)=='F')then
            weight=3.0d0
            xrow=0.0d0; yelem=obs(i)/3.0d0
            xrow(codanim(i))=1.0d0; xrow(countdam+codcg(i))=1.0d0/3.0d0
            xrow(coddam(i))=-1.0d0/3.0d0
            xrow(countdam+countcg+codsire(i))=-1.0d0/3.0d0
            call includ (np,nrbar,weight,xrow,yelem,&
                         d,rbar,thetab,sserr,ifault)
            count=count+1
          end if
          if(Gdam(i)==1)then
            weight=1.0d0
            xrow=0.0d0; yelem=0.0d0
            xrow(coddam(i))=1.0d0
            call includ (np,nrbar,weight,xrow,yelem,&
                         d,rbar,thetab,sserr,ifault)
            count=count+1
          end if
        end do
!
        deallocate (codanim,codsire,coddam,codcg,obs,xrow,Gdam)
!
        print *,' '
        print *,'    Writing results...  '
        print *,' '
!
        rec_nr=0
        do i=1,isisi-1
          rec_nr=rec_nr+1
          write(60,REC=rec_nr) rbar(i)
        end do
!
        rec_nr=0
        do j=1,countdam+countcg
          rec_nr=rec_nr+1
          write(70,REC=rec_nr) d(j)
          write(140,REC=rec_nr) thetab(j)
        end do
!
        close(60)
        close(70)
        close(140)
!
        open(90,file='RBAR_SIRE.bin',access='direct',&
             recl=8,form='unformatted',status='replace')
        open(100,file='DIAG_SIRE.bin',access='direct',&
             recl=8,form='unformatted',status='replace')
        open(130,file='THET_SIRE.bin',access='direct',&
             recl=8,form='unformatted',status='replace')
```

```
!
      rec_nr=0
      do i=isisi,nrbar
         rec_nr=rec_nr+1
         write(90,REC=rec_nr) rbar(i)
      end do
!
      rec_nr=0
      do j=countdam+countcg+1,np
         rec_nr=rec_nr+1
         write(100,REC=rec_nr) d(j)
         write(130,REC=rec_nr) thetab(j)
      end do
!
      open(150,file='info.txt',status='replace')
      nobs=nobs+count
!
      write(150,'(i40,1x,a35)')nherd,'   ! Total number of herds       '
      write(150,'(f40.30,1x,a35)')sserr,'! Residual sum of squares     '
      write(150,'(i40,1x,a35)')nobs,'    ! Total number of observations'
      write(150,'(i40,1x,a35)')nownim,'  ! N of males with info        '
      write(150,'(i40,1x,a35)')nsire,'   ! Total number of sires       '
!
      close(90)
      close(100)
      close(130)
      close(150)
!
      write(50,9999) countdam,' ! Current number of dams'
      write(50,9999) countcg,'  ! Current number of CGs '
      write(50,9999) count,'    ! Current number of obs '
      write(50,9999) nownif,'   ! N of females with info'
      close(50)
!
      deallocate (d,rbar,thetab)
!
      call date_and_time (date,time,zone,fval)
      call cpu_time(cputime_fin)
      cputime=cputime_fin-cputime_ini
!
      print *,' '
      print *,' '
      print *,' '
      print *,'    Execution completed sucessfully '
      print *,' '
      print *,' '
      print *,' '
      print *,'    Initial time: ',ival(1),'/',ival(2),'/',ival(3),&
                         '  at  ',ival(5),'h',ival(6),'m',ival(7),'s'
      print *,' '
      print *,'      Final time: ',fval(1),'/',fval(2),'/',fval(3),&
                         '  at  ',fval(5),'h',fval(6),'m',fval(7),'s'
      print *,' '
      print *,' '
```

```
      print *,' '
      print *,'    Elapsed CPU time was: ',cputime,'sec'
      print *,' '
      print *,' '
      flush(6)
!
 9999 format(i40,1x,a30)
      stop
      end program main
!
!     ----------------------------------------------------------------
!     ----------------------------------------------------------------
!
      subroutine includ (np,nrbar,weight,xrow,yelem,&
                         d,rbar,thetab,sserr,ifault)
!
      integer, intent(in) :: np,nrbar
      integer, intent(out) :: ifault
      real(8), intent(in) :: weight,yelem
      real(8), intent(inout) :: sserr
      real(8), intent(inout) :: xrow(:),d(:),rbar(:),thetab(:)
!
!
!
!                          CREDIT WHERE CREDIT IS DUE
!                          ~~~~~~~~~~~~~~~~~~~~~~~~~~
!     Algorithm AS 75.1 Appl. Statist. (1974) Vol.23, No.3, p448
!     Available from StatLib at http://lib.stat.cmu.edu/apstat/
!                     ~~~~~~~~
!
!
!     Calling this subroutine updates d, rbar, thetab and sserr
!     by the inclusion of xrow, yelem with specified weight.
!
      double precision cbar,di,dpi,sbar,w,xi,xk,y,zero
      integer i,nextr,ip,k
!
      data zero/0.0d0/
!
! --> Check input parameters <--
!
      ifault=1
      if (np.lt.1.or.nrbar.lt.np*(np-1)/2) return
      ifault=0
!
      w=weight
      y=yelem
      do 30 i=1,np
!
!                  --> Skip unnecessary transformations <--
!     (test on exact zeros must be used or stability can be destroyed)
!
         if (w.eq.zero) return
         if (xrow(i).eq.zero) go to 30
```

```
         xi=xrow(i)
         di=d(i)
         dpi=di+w*xi*xi
         cbar=di/dpi
         sbar=w*xi/dpi
         w=cbar*w
         d(i)=dpi
         if (i.eq.np) go to 20
         nextr=(i-1)*(np+np-i)/2+1
         ip=i+1
         do 10 k=ip,np
            xk=xrow(k)
            xrow(k)=xk-xi*rbar(nextr)
            rbar(nextr)=cbar*rbar(nextr)+sbar*xk
            nextr=nextr+1
10       continue
20       xk=y
         y=xk-xi*thetab(i)
         thetab(i)=cbar*thetab(i)+sbar*xk
30    continue
      sserr=sserr+w*y*y
      return
      end subroutine includ
!
!     ----------------------------------------------------------------
!     ----------------------------------------------------------------
!
```

**APÊNDICE C** – UPDA.f90: código fonte em Fortran 90 para atualizar os dados de um rebanho no sistema de atualização. Redimensiona e atualiza a trapezóide correspondente ao rebanho e atualiza a triangular superior ‘touro x touro’.

```
!
!     UPDA.f90 (Genetic Evaluations with Gentleman-Givens rotations)
!
!     By Eduardo CG Pimentel           10/23/2007
!     E-mail: pimentel@fcav.unesp.br
!
!     ---------------------------------------------------------------------
!     ---------------------------------------------------------------------
!
      program main
      implicit none
!
!     ---------------------------------------------------------------------
!
      character(20) :: data_file,daid_file,cgid_file,herd_file
      character(20) :: rbar_file,diag_file,thet_file,fown_file
      character(10) :: date,time,zone,c_herd,i_herd
      character(15) :: an,si,da
      character(1) :: gend
!
      integer :: i,j,k,l,nrec,gr,nsire,np,nrbar,ifault,count,isisi,nextr
      integer :: cgmaxid,nherd,oldnp,oldnrbar,newnsire,hobs,newpos
      integer :: ival(8),fval(8),countdam,countcg,nobs,damaxid,ncgnewcol
      integer :: nownif,nownif2,nownim,nownim2,npseudof,diffnp,ndanewcol
      integer :: incr,npseudom,apos,spos,dpos,rec_nr
!
      real(4) :: cputime_ini,cputime_fin,cputime
      real(8) :: ob,weight,yelem,sserr
!
      character(15), allocatable :: sire(:),si_orID(:),dam(:),da_orID(:)
      character(15), allocatable :: anim(:),m_anim(:),f_anim(:)
      character(10), allocatable :: herd(:)
      character(1), allocatable :: gender(:)
!
      integer, allocatable :: codanim(:),codsire(:),coddam(:),codcg(:)
      integer, allocatable :: f_codanim(:),f_codsire(:),f_coddam(:)
      integer, allocatable :: m_codanim(:),m_codsire(:),m_coddam(:)
      integer, allocatable :: cg(:),f_codcg(:),m_codcg(:),Gda(:),Gsi(:)
!
      real(8), allocatable :: obs(:),xrow(:),d(:),rbar(:),thetab(:)
      real(8), allocatable :: f_obs(:),m_obs(:),oldrbar(:)
!
!     ---------------------------------------------------------------------
!
      interface includ
         subroutine includ(np,nrbar,weight,xrow,yelem,&
                           d,rbar,thetab,sserr,ifault)
            integer, intent(in) :: np,nrbar
            integer, intent(out) :: ifault
            real(8), intent(in) :: weight,yelem
            real(8), intent(inout) :: sserr
            real(8), intent(inout) :: xrow(:),d(:),rbar(:),thetab(:)
         end subroutine includ
      end interface includ
```

```
!
!     ------------------------------------------------------------------
!
      call cpu_time(cputime_ini)
      call date_and_time (date,time,zone,ival)
!
!     ------------------------------------------------------------------
!
      open(10,file='PARUP.dat',status='old')
!
!     ------------------------------------------------------------------
!
      read(10,*) c_herd
      read(10,*) data_file
      read(10,*) nrec
      close(10)
!
      c_herd=adjustl(c_herd)
      daid_file='DAID_'//trim(c_herd)//'.cod'
      cgid_file='CGID_'//trim(c_herd)//'.cod'
      herd_file='HERD_'//trim(c_herd)//'.par'
      rbar_file='RBAR_'//trim(c_herd)//'.bin'
      diag_file='DIAG_'//trim(c_herd)//'.bin'
      thet_file='THET_'//trim(c_herd)//'.bin'
      fown_file='FOWN_'//trim(c_herd)//'.bin'
!
!     ------------------------------------------------------------------
!
      open(20,file=data_file,status='old')
      open(30,file=daid_file,status='old',position='append')
      open(40,file=cgid_file,status='old',position='append')
      open(50,file=herd_file,status='old')
      open(60,file=rbar_file,access='direct',&
           recl=8,form='unformatted',status='old')
      open(70,file=diag_file,access='direct',&
           recl=8,form='unformatted',status='old')
      open(80,file='SIRES.cod',status='old',position='append')
      open(90,file='RBAR_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(100,file='DIAG_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(110,file=fown_file,access='direct',&
           recl=35,form='unformatted',status='old')
      open(120,file='MALE_PROD.bin',access='direct',&
           recl=45,form='unformatted',status='old')
      open(130,file='THET_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(140,file=thet_file,access='direct',&
           recl=8,form='unformatted',status='old')
      open(150,file='info.txt',status='old')
      open(160,status='scratch')
!
!     ------------------------------------------------------------------
!
```

```
      read(50,'(i40)') countdam   ! Current number of dams
      read(50,'(i40)') countcg    ! Current number of CGs
      read(50,'(i40)') hobs       ! Current number of obs
      read(50,'(i40)') nownif     ! N of females with info
!
      read(150,'(i40)') nherd     ! Total number of herds
      read(150,'(f40.30)') sserr  ! Residual sum of squares
      read(150,'(i40)') nobs      ! Total number of observations
      read(150,'(i40)') nownim    ! N of males with info
      read(150,'(i40)') nsire     ! Total number of sires
!
      damaxid=countdam; cgmaxid=countcg
      count=0; npseudof=0; npseudom=0
      close(50)
      close(150)
!
      allocate(anim(1:nrec),sire(1:nrec),dam(1:nrec))
      allocate(cg(1:nrec),gender(1:nrec),obs(1:nrec))
      cg=0; obs=0.0d0
!
      print *,' '
      print *,' '
      print *,'    Reading input data file...  '
      print *,' '
!
      i=0
 100  read(20,*,end=110) an,si,da,gr,gend,ob
      i=i+1
      anim(i)=an; sire(i)=si; dam(i)=da; cg(i)=gr
      gender(i)=gend; obs(i)=ob
      go to 100
 110  continue
      close(20)
!
      print *,' '
      print *,'    Recoding dams, groups and sires...  '
      print *,' '
!
      rewind 30
      allocate(da_orID(1:countdam))
      do i=1,countdam
         read(30,'(a15)') da_orID(i)
      end do
!
      allocate(f_anim(1:nownif),f_codanim(1:nownif),f_codsire(1:nownif))
      allocate(f_coddam(1:nownif),f_codcg(1:nownif),f_obs(1:nownif))
      f_codanim=0; f_codcg=0; f_coddam=0; f_codsire=0; f_obs=0.0d0
!
      do j=1,nownif
         read(110,REC=j) &
         f_anim(j),f_codsire(j),f_coddam(j),f_codcg(j),f_obs(j)
      end do
      close(110)
!
```

```
      allocate(coddam(1:nrec),codanim(1:nrec),Gda(1:nrec))
      allocate(codcg(1:nrec),codsire(1:nrec),Gsi(1:nrec))
      coddam=0; codanim=0; codcg=0; codsire=0; Gda=1; Gsi=1
!
! --> Recoding dam IDs <--
!
      endfile 30
      OUTER_1: do i=1,nrec
                  damaxid=damaxid+1
                  coddam(i)=damaxid
                  INNER1_1: do j=1,countdam
                              if(da_orID(j)==dam(i))then
                                coddam(i)=j; Gda(i)=0
                                damaxid=damaxid-1
                                cycle OUTER_1
                              end if
                            end do INNER1_1
                  INNER2_1: do k=1,nownif
                              if(f_anim(k)==dam(i))then
                                npseudof=npseudof+1
                                f_codanim(k)=coddam(i)
                                Gda(i)=0
                                exit INNER2_1
                              end if
                            end do INNER2_1
                  write(30,'(a15)') dam(i)
               end do OUTER_1
      deallocate(da_orID)
      ndanewcol=damaxid-countdam
      close(30)
!
      open(110,file=fown_file,access='direct',&
           recl=35,form='unformatted',status='replace')
!
      nownif2=0
      do j=1,nownif
         if(f_codanim(j)==0)then
           nownif2=nownif2+1
           write(110,REC=nownif2) &
           f_anim(j),f_codsire(j),f_coddam(j),f_codcg(j),f_obs(j)
         end if
      end do
!
! --> Recoding group IDs <--
!
      endfile 40
      OUTER_2: do i=1,nrec
                  cgmaxid=cgmaxid+1
                  codcg(i)=cgmaxid
                  INNER_2: do j=1,i-1
                              if(cg(j)==cg(i))then
                                codcg(i)=codcg(j)
                                cgmaxid=cgmaxid-1
                                cycle OUTER_2
```

```
                              end if
                           end do INNER_2
                  write(40,'(i10)') cg(i)
               end do OUTER_2
      ncgnewcol=cgmaxid-countcg
      close(40)
!
      rewind 80
      allocate(si_orID(1:nsire))
      do i=1,nsire
         read(80,'(a15)') si_orID(i)
      end do
!
      allocate(m_anim(1:nownim),m_codanim(1:nownim),m_codsire(1:nownim))
      allocate(m_coddam(1:nownim),m_codcg(1:nownim),m_obs(1:nownim))
      allocate(herd(1:nownim))
      m_codanim=0
!
      do j=1,nownim
         read(120,REC=j) &
         m_anim(j),m_codsire(j),m_coddam(j),m_codcg(j),m_obs(j),herd(j)
      end do
      close(120)
!
! --> Recoding sire IDs <--
!
      endfile 80
      count=nsire
      OUTER_3: do i=1,nrec
                  count=count+1
                  codsire(i)=count
                  INNER1_3: do l=1,i-1
                              if(sire(l)==sire(i))then
                                codsire(i)=codsire(l)
                                count=count-1; Gsi(i)=0
                                cycle OUTER_3
                              end if
                           end do INNER1_3
                  INNER2_3: do j=1,nsire
                              if(si_orID(j)==sire(i))then
                                codsire(i)=j; Gsi(i)=0
                                count=count-1
                                cycle OUTER_3
                              end if
                           end do INNER2_3
                  INNER3_3: do k=1,nownim
                              if(m_anim(k)==sire(i))then
                                npseudom=npseudom+1; Gsi(i)=0
                                m_codanim(k)=codsire(i)
                                exit INNER3_3
                              end if
                           end do INNER3_3
                  write(80,'(a15)') sire(i)
               end do OUTER_3
```

```
      newnsire=count
      close(80)
!
      open(120,file='MALE_PROD.bin',access='direct',&
           recl=45,form='unformatted',status='replace')
!
      nownim2=0
      do j=1,nownim
         if(m_codanim(j)==0)then
           nownim2=nownim2+1
           write(120,REC=nownim2) m_anim(j),&
           m_codsire(j),m_coddam(j),m_codcg(j),m_obs(j),herd(j)
         else
           write(160,9998) m_codanim(j),&
           m_codsire(j),m_coddam(j),m_codcg(j),m_obs(j),herd(j)
         end if
      end do
!
      deallocate (m_anim,m_codanim,m_codsire,m_coddam,m_codcg,m_obs)
!
! --> Storing IDs of products and their parents for further pseudos <--
!
      do i=1,nrec
         if(gender(i)=='M'.and.codanim(i)==0)then
           nownim2=nownim2+1
           write(120,REC=nownim2) &
           anim(i),codsire(i),coddam(i),codcg(i),obs(i),c_herd
         end if
         if(gender(i)=='F'.and.codanim(i)==0)then
           nownif2=nownif2+1
           write(110,REC=nownif2) &
           anim(i),codsire(i),coddam(i),codcg(i),obs(i)
         end if
      end do
      close(110)
      close(120)
!
      deallocate (anim,sire,dam,cg,gender,si_orID,herd)
!
      print *,' '
      print *,'    Re-indexing the upper triangular R...  '
      print *,' '
!
! --> Retrieving this herd's part in the upper triangular <--
!
      oldnp=countdam+countcg+nsire
      oldnrbar=(oldnp*(oldnp-1))/2
      isisi=(countdam+countcg)*(oldnp+oldnp-(countdam+countcg+1))/2+1
      allocate(oldrbar(1:oldnrbar))
!
      do i=1,isisi-1
         read(60,REC=i) oldrbar(i)
      end do
!
```

```
      rec_nr=0
      do j=isisi,oldnrbar
         rec_nr=rec_nr+1
         read(90,REC=rec_nr) oldrbar(j)
      end do
      close(60)
      close(90)
!
! --> Re-indexing this herd's part in the upper triangular <--
!
      np=damaxid+cgmaxid+newnsire
      nrbar=(np*(np-1))/2
      diffnp=np-oldnp
      allocate(xrow(1:np),d(1:np),thetab(1:np),rbar(1:nrbar))
      xrow=0.0d0; d=0.0d0; thetab=0.0d0; rbar=0.0d0
!
! --> Re-indexing the diagonal and the right hand side <--
!
      do i=1,countdam
         read(70,REC=i) d(i)
         read(140,REC=i) thetab(i)
      end do
!
      rec_nr=countdam
      do j=damaxid+1,damaxid+countcg
         rec_nr=rec_nr+1
         read(70,REC=rec_nr) d(j)
         read(140,REC=rec_nr) thetab(j)
      end do
!
      rec_nr=0
      do k=damaxid+cgmaxid+1,damaxid+cgmaxid+nsire
         rec_nr=rec_nr+1
         read(100,REC=rec_nr) d(k)
         read(130,REC=rec_nr) thetab(k)
      end do
      close(70)
      close(100)
      close(130)
      close(140)
!
! --> Re-indexing 'dam x dam', 'dam x group' and 'dam x sire' <--
!
      do i=1,countdam
         incr=diffnp*(i-1)
         nextr=(i-1)*(oldnp+oldnp-i)/2+1
         if(i<countdam)then
           do j=nextr,countdam-1+nextr-i
              rbar(j+incr)=oldrbar(j)
           end do
         end if
         if(i<countdam)then
           j=countdam-1+nextr-i
         else
```

```
            j=nextr-1
          end if
          incr=incr+ndanewcol
          do k=j+1,j+countcg
             rbar(k+incr)=oldrbar(k)
          end do
          incr=incr+ncgnewcol
          k=j+countcg
          do l=k+1,k+nsire
             rbar(l+incr)=oldrbar(l)
          end do
       end do
!
! --> Re-indexing 'group x group' and 'group x sire' <--
!
       do i=countdam+1,countdam+countcg
          nextr=(i-1)*(oldnp+oldnp-i)/2+1
          newpos=(i+ndanewcol-1)*(np+np-i-ndanewcol)/2+1
          if(i<countdam+countcg)then
            do j=nextr,countdam+countcg-1+nextr-i
               rbar(newpos)=oldrbar(j)
               newpos=newpos+1
            end do
          end if
          if(i<countdam+countcg)then
            j=countdam+countcg-1+nextr-i
          else
            j=nextr-1
          end if
          newpos=newpos+ncgnewcol
          do k=j+1,j+nsire
             rbar(newpos)=oldrbar(k)
             newpos=newpos+1
          end do
       end do
!
! --> Re-indexing 'sire x sire' <--
!
       do i=countdam+countcg+1,countdam+countcg+nsire-1
          nextr=(i-1)*(oldnp+oldnp-i)/2+1
          newpos=(i+ndanewcol+ncgnewcol-1)*&
                 (np+np-i-ndanewcol-ncgnewcol)/2+1
          do j=nextr,oldnp-1+nextr-i
             rbar(newpos)=oldrbar(j)
             newpos=newpos+1
          end do
       end do
!
       deallocate(oldrbar)
!
       print *,' '
       print *,'    Rotating current records (updating R)...  '
       print *,' '
!
```

```
      count=0
      do i=1,nrec
         weight=2.0d0/3.0d0
         xrow=0.0d0; yelem=obs(i)
         xrow(damaxid+codcg(i))=1.0d0
         xrow(coddam(i))=0.5d0; xrow(damaxid+cgmaxid+codsire(i))=0.5d0
         call includ (np,nrbar,weight,xrow,yelem,&
                      d,rbar,thetab,sserr,ifault)
         count=count+1
         if(Gda(i)==1)then
           weight=1.0d0
           xrow=0.0d0; yelem=0.0d0
           xrow(coddam(i))=1.0d0
           call includ (np,nrbar,weight,xrow,yelem,&
                        d,rbar,thetab,sserr,ifault)
           count=count+1
         end if
         if(Gsi(i)==1)then
           weight=1.0d0
           xrow=0.0d0; yelem=0.0d0
           xrow(damaxid+cgmaxid+codsire(i))=1.0d0
           call includ (np,nrbar,weight,xrow,yelem,&
                        d,rbar,thetab,sserr,ifault)
           count=count+1
         end if
      end do
!
      weight=3.0d0
      do j=1,nownif
         if(f_codanim(j)/=0)then
           xrow=0.0d0; yelem=f_obs(j)/3.0d0
           xrow(damaxid+f_codcg(j))=1.0d0/3.0d0
           xrow(f_codanim(j))=1.0d0; xrow(f_coddam(j))=-1.0d0/3.0d0
           xrow(damaxid+cgmaxid+f_codsire(j))=-1.0d0/3.0d0
           call includ (np,nrbar,weight,xrow,yelem,&
                        d,rbar,thetab,sserr,ifault)
           count=count+1
         end if
      end do
!
      deallocate (codanim,codsire,coddam,codcg,obs,xrow,Gda,Gsi)
      deallocate (f_anim,f_codanim,f_codsire,f_coddam,f_codcg,f_obs)
!
      print *,' '
      print *,'    Writing results...  '
      print *,' '
!
      open(60,file=rbar_file,access='direct',&
           recl=8,form='unformatted',status='replace')
      open(70,file=diag_file,access='direct',&
           recl=8,form='unformatted',status='replace')
      open(140,file=thet_file,access='direct',&
           recl=8,form='unformatted',status='replace')
!
```

```
      isisi=(damaxid+cgmaxid)*(np+np-(damaxid+cgmaxid+1))/2+1
!
      do i=1,isisi-1
         write(60,REC=i) rbar(i)
      end do
!
      do j=1,damaxid+cgmaxid
         write(70,REC=j) d(j)
         write(140,REC=j) thetab(j)
      end do
!
      close(60)
      close(70)
      close(140)
!
      open(90,file='RBAR_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='replace')
      open(100,file='DIAG_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='replace')
      open(130,file='THET_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='replace')
!
      rec_nr=0
      do i=isisi,nrbar
         rec_nr=rec_nr+1
         write(90,REC=rec_nr) rbar(i)
      end do
!
      rec_nr=0
      do j=damaxid+cgmaxid+1,np
         rec_nr=rec_nr+1
         write(100,REC=rec_nr) d(j)
         write(130,REC=rec_nr) thetab(j)
      end do
      close(90)
      close(100)
      close(130)
!
      open(50,file=herd_file,status='replace')
      hobs=hobs+count
!
      write(50,9999) damaxid,'  ! Current number of dams'
      write(50,9999) cgmaxid,'  ! Current number of CGs '
      write(50,9999) hobs,'     ! Current number of obs '
      write(50,9999) nownif2,'  ! N of females with info'
      close(50)
!
      deallocate (d,rbar,thetab)
!
      if(newnsire/=nsire)then
        print *,' '
        print *,'    Re-indexing upper triangulars of other herds...  '
        print *,' '
        do i=1,nherd
```

```
            write(i_herd,*) i
            i_herd=adjustl(i_herd)
            if(i_herd==c_herd)cycle
            herd_file='HERD_'//trim(i_herd)//'.par'
            rbar_file='RBAR_'//trim(i_herd)//'.bin'
            open(50,file=herd_file,status='old')
            read(50,'(i40)') countdam
            read(50,'(i40)') countcg
            close(50)
            open(60,file=rbar_file,access='direct',&
                 recl=8,form='unformatted',status='old')
            oldnp=countdam+countcg+nsire
            np=countdam+countcg+newnsire
            isisi=(countdam+countcg)*(np+np-(countdam+countcg+1))/2+1
            allocate(rbar(1:isisi-1))
            rbar=0.0d0
            rec_nr=0
            do j=1,countdam+countcg
               nextr=(j-1)*(np+np-j)/2+1
               do k=nextr,nextr+oldnp-1-j
                  rec_nr=rec_nr+1
                  read(60,REC=rec_nr) rbar(k)
               end do
            end do
            close(60)
            open(60,file=rbar_file,access='direct',&
                 recl=8,form='unformatted',status='replace')
            do l=1,isisi-1
               write(60,REC=l) rbar(l)
            end do
            deallocate(rbar)
            close(60)
          end do
        end if
!
! --> Rotating pseudo-observations of males that became sires <--
!
        if(npseudom/=0)then
          print *,' '
          print *,'    Rotating pseudo-observations of males...  '
          print *,' '
          rewind 160
          do i=1,npseudom
             read(160,9998) apos,spos,dpos,gr,ob,i_herd
             i_herd=adjustl(i_herd)
             herd_file='HERD_'//trim(i_herd)//'.par'
             rbar_file='RBAR_'//trim(i_herd)//'.bin'
             diag_file='DIAG_'//trim(i_herd)//'.bin'
             thet_file='THET_'//trim(i_herd)//'.bin'
             open(50,file=herd_file,status='old')
             read(50,'(i40)') countdam
             read(50,'(i40)') countcg
             read(50,'(i40)') hobs
             read(50,'(i40)') nownif
```

```
close(50)
np=countdam+countcg+newnsire
nrbar=(np*(np-1))/2
isisi=(countdam+countcg)*(np+np-(countdam+countcg+1))/2+1
allocate(xrow(1:np),d(1:np),thetab(1:np),rbar(1:nrbar))
xrow=0.0d0; d=0.0d0; thetab=0.0d0; rbar=0.0d0
open(60,file=rbar_file,access='direct',&
     recl=8,form='unformatted',status='old')
open(70,file=diag_file,access='direct',&
     recl=8,form='unformatted',status='old')
open(140,file=thet_file,access='direct',&
     recl=8,form='unformatted',status='old')
open(90,file='RBAR_SIRE.bin',access='direct',&
     recl=8,form='unformatted',status='old')
open(100,file='DIAG_SIRE.bin',access='direct',&
     recl=8,form='unformatted',status='old')
open(130,file='THET_SIRE.bin',access='direct',&
     recl=8,form='unformatted',status='old')
do j=1,isisi-1
   read(60,REC=j) rbar(j)
end do
rec_nr=0
do k=isisi,nrbar
   rec_nr=rec_nr+1
   read(90,REC=rec_nr) rbar(k)
end do
do j=1,countdam+countcg
   read(70,REC=j) d(j)
   read(140,REC=j) thetab(j)
end do
rec_nr=0
do k=countdam+countcg+1,np
   rec_nr=rec_nr+1
   read(100,REC=rec_nr) d(k)
   read(130,REC=rec_nr) thetab(k)
end do
close(60)
close(70)
close(140)
close(90)
close(100)
close(130)
weight=3.0d0; yelem=ob/3.0d0
xrow(countdam+countcg+apos)=1.0d0
xrow(countdam+gr)=1.0d0/3.0d0; xrow(dpos)=-1.0d0/3.0d0
xrow(countdam+countcg+spos)=-1.0d0/3.0d0
call includ (np,nrbar,weight,xrow,yelem,&
           d,rbar,thetab,sserr,ifault)
count=count+1
hobs=hobs+1
open(60,file=rbar_file,access='direct',&
     recl=8,form='unformatted',status='replace')
open(70,file=diag_file,access='direct',&
     recl=8,form='unformatted',status='replace')
```

```
            open(140,file=thet_file,access='direct',&
                 recl=8,form='unformatted',status='replace')
            do j=1,isisi-1
               write(60,REC=j) rbar(j)
            end do
            do k=1,countdam+countcg
               write(70,REC=k) d(k)
               write(140,REC=k) thetab(k)
            end do
            close(60)
            close(70)
            close(140)
            open(90,file='RBAR_SIRE.bin',access='direct',&
                 recl=8,form='unformatted',status='replace')
            open(100,file='DIAG_SIRE.bin',access='direct',&
                 recl=8,form='unformatted',status='replace')
            open(130,file='THET_SIRE.bin',access='direct',&
                 recl=8,form='unformatted',status='replace')
            rec_nr=0
            do j=isisi,nrbar
               rec_nr=rec_nr+1
               write(90,REC=rec_nr) rbar(j)
            end do
            rec_nr=0
            do k=countdam+countcg+1,np
               rec_nr=rec_nr+1
               write(100,REC=rec_nr) d(k)
               write(130,REC=rec_nr) thetab(k)
            end do
            close(90)
            close(100)
            close(130)
            open(50,file=herd_file,status='replace')
            write(50,9999) countdam,' ! Current number of dams'
            write(50,9999) countcg,'  ! Current number of CGs '
            write(50,9999) hobs,'     ! Current number of obs '
            write(50,9999) nownif,'   ! N of females with info'
            close(50)
            deallocate(xrow,d,thetab,rbar)
          end do
          close(160)
        end if
!
        open(150,file='info.txt',status='replace')
        nobs=nobs+count
        write(150,'(i40,1x,a35)')nherd,'   ! Total number of herds       '
        write(150,'(f40.30,1x,a35)')sserr,'! Residual sum of squares     '
        write(150,'(i40,1x,a35)')nobs,'    ! Total number of observations'
        write(150,'(i40,1x,a35)')nownim2,' ! N of males with info        '
        write(150,'(i40,1x,a35)')newnsire,'! Total number of sires       '
        close(150)
!
        call date_and_time (date,time,zone,fval)
        call cpu_time(cputime_fin)
```

```
      cputime=cputime_fin-cputime_ini
!
      print *,' '
      print *,' '
      print *,' '
      print *,'    Execution completed sucessfully '
      print *,' '
      print *,' '
      print *,' '
      print *,'    Initial time: ',ival(1),'/',ival(2),'/',ival(3),&
                          '  at  ',ival(5),'h',ival(6),'m',ival(7),'s'
      print *,' '
      print *,'      Final time: ',fval(1),'/',fval(2),'/',fval(3),&
                          '  at  ',fval(5),'h',fval(6),'m',fval(7),'s'
      print *,' '
      print *,' '
      print *,' '
      print *,'    Elapsed CPU time was: ',cputime,'sec'
      print *,' '
      print *,' '
      flush(6)
!
 9998 format(4(i10,1x),f20.10,1x,a10)
 9999 format(i40,1x,a30)
      stop
      end program main
!
!     ----------------------------------------------------------------------
!     ----------------------------------------------------------------------
!
      subroutine includ (np,nrbar,weight,xrow,yelem,&
                         d,rbar,thetab,sserr,ifault)
!
      integer, intent(in) :: np,nrbar
      integer, intent(out) :: ifault
      real(8), intent(in) :: weight,yelem
      real(8), intent(inout) :: sserr
      real(8), intent(inout) :: xrow(:),d(:),rbar(:),thetab(:)
!
!
!
!                          CREDIT WHERE CREDIT IS DUE
!                          ~~~~~~~~~~~~~~~~~~~~~~~~~~
!     Algorithm AS 75.1 Appl. Statist. (1974) Vol.23, No.3, p448
!     Available from StatLib at http://lib.stat.cmu.edu/apstat/
!                     ~~~~~~~
!
!
!     Calling this subroutine updates d, rbar, thetab and sserr
!     by the inclusion of xrow, yelem with specified weight.
!
      double precision cbar,di,dpi,sbar,w,xi,xk,y,zero
      integer i,nextr,ip,k
!
```

```
      data zero/0.0d0/
!
! --> Check input parameters <--
!
      ifault=1
      if (np.lt.1.or.nrbar.lt.np*(np-1)/2) return
      ifault=0
!
      w=weight
      y=yelem
      do 30 i=1,np
!
!                 --> Skip unnecessary transformations <--
!     (test on exact zeros must be used or stability can be destroyed)
!
         if (w.eq.zero) return
         if (xrow(i).eq.zero) go to 30
         xi=xrow(i)
         di=d(i)
         dpi=di+w*xi*xi
         cbar=di/dpi
         sbar=w*xi/dpi
         w=cbar*w
         d(i)=dpi
         if (i.eq.np) go to 20
         nextr=(i-1)*(np+np-i)/2+1
         ip=i+1
         do 10 k=ip,np
            xk=xrow(k)
            xrow(k)=xk-xi*rbar(nextr)
            rbar(nextr)=cbar*rbar(nextr)+sbar*xk
            nextr=nextr+1
 10      continue
 20      xk=y
         y=xk-xi*thetab(i)
         thetab(i)=cbar*thetab(i)+sbar*xk
 30   continue
      sserr=sserr+w*y*y
      return
      end subroutine includ
!
!     ----------------------------------------------------------------
!     ----------------------------------------------------------------
!
```

**APÊNDICE D** – SOLV.f90: código fonte em Fortran 90 para resolver o sistema triangular geral por retro-substituição. Gera arquivos com soluções para grupos de contemporêneos e animais e calcula as acurácias.

```
!
!     SOLV.f90 (Genetic Evaluations with Gentleman-Givens rotations)
!
!     By Eduardo CG Pimentel           10/23/2007
!     E-mail: pimentel@fcav.unesp.br
!
!     ---------------------------------------------------------------
!     ---------------------------------------------------------------
!
      program main
      implicit none
!
!     ---------------------------------------------------------------
!
      character(20) :: rbar_file,thet_file,herd_file
      character(30) :: solda_file,solcg_file
      character(10) :: date,time,zone,i_herd
!
      integer :: i,j,k,l,nsire,np,nrbar,ifault,count,isisi,nextr,npt,ip
      integer :: ival(8),fval(8),nherd,newpos,ipos,fpos,nobs,sizesisi
      integer :: orderA,index1,index2
!
      real(4) :: cputime_ini,cputime_fin,cputime
      real(8) :: sserr,sm
!
      integer, allocatable :: nda(:),ncg(:)
!
      real(8), allocatable :: betasi(:),acc(:),beta(:),thetab(:)
      real(8), allocatable :: d(:),rbar(:),A(:)

      real(8), parameter :: alpha=1.0d0   ! ratio of variances
!
!     ---------------------------------------------------------------
!
      call cpu_time(cputime_ini)
      call date_and_time (date,time,zone,ival)
!
!     ---------------------------------------------------------------
!
      open(150,file='info.txt',status='old')
!
!     ---------------------------------------------------------------
!
      read(150,'(i40)') nherd     ! Total number of herds
      read(150,'(f40.30)') sm     ! (skip) Residual sum of squares
      read(150,'(i40)') i         ! (skip) Total number of observations
      read(150,'(i40)') i         ! (skip) N of males with info
      read(150,'(i40)') nsire     ! Total number of sires
!
      print *,' '
      print *,' '
      print *,'    Backsolving for sires...  '
      print *,' '
!
```

```
      open(90,file='RBAR_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(100,file='DIAG_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(130,file='THET_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
!
      nrbar=nsire*(nsire-1)/2
      sizesisi=nrbar
      allocate(thetab(1:nsire),betasi(1:nsire))
      allocate(d(1:nsire),rbar(1:nrbar))
      d=0.0d0; rbar=0.0d0; thetab=0.0d0; betasi=0.0d0
!
      do i=1,nrbar
         read(90,REC=i) rbar(i)
      end do
      do j=1,nsire
         read(130,REC=j) thetab(j)
         read(100,REC=j) d(j)
      end do
      close(90)
      close(100)
      close(130)
!
      do j=1,nsire
         i=nsire-j+1
         betasi(i)=thetab(i)
         nextr=(i-1)*(nsire+nsire-i)/2+1
         ip=i+1
         do k=ip,nsire
            betasi(i)=betasi(i)-rbar(nextr)*betasi(k)
            nextr=nextr+1
         end do
      end do
!
      print *,' '
      print *,'    Scaling the Cholesky factor...  '
      print *,' '
!
      np=nsire
      orderA=np*(np+1)/2
      allocate(acc(1:np),A(1:orderA))
      acc=0.0d0; A=0.0d0
      d=sqrt(d)
      do i=1,np
         index1=(i-1)*(np+np-i)/2+i
         A(index1)=1.0d0/d(i)
         acc(i)=A(index1)*A(index1)
         if(i==np)exit
         index2=(i-1)*(np+np-i)/2+1
         do j=index2,index2+np-i-1
            index1=index1+1
            A(index1)=rbar(j)*d(i)
         end do
```

```
      end do
!
      print *,' '
      print *,'    Calculating accuracies...  '
      print *,' '
!
      do i=1,np-1
         do j=i+1,np
            sm=0.0d0
            do k=i,j-1
               index1=(i-1)*(np+np-i)/2+k
               index2=(k-1)*(np+np-k)/2+j
               sm=sm+A(index1)*A(index2)
            end do
            index1=(i-1)*(np+np-i)/2+j
            index2=(j-1)*(np+np-j)/2+j
            A(index1)=-sm*A(index2)
            acc(i)=acc(i)+A(index1)*A(index1)
         end do
      end do
!
      do i=1,np
         acc(i)=sqrt(1-(acc(i)*alpha))
      end do
!
      deallocate(d,rbar,thetab,A)
      open(200,file='EBV_SIRE.out',status='replace')
!
      do i=1,nsire
         write(200,'(f20.10,1x,f20.10)') betasi(i),acc(i)
      end do
      close(200)
!
      allocate(nda(1:nherd),ncg(1:nherd))
      nda=0; ncg=0
!
      npt=0
      do i=1,nherd
         write(i_herd,*) i
         i_herd=adjustl(i_herd)
         herd_file='HERD_'//trim(i_herd)//'.par'
         open(50,file=herd_file,status='old')
         read(50,'(i40)') nda(i)
         read(50,'(i40)') ncg(i)
         close(50)
         npt=npt+nda(i)+ncg(i)
      end do
!
      do l=1,nherd
         print *,' '
         print *,'    Backsolving for groups and dams (herd',l,')...  '
         print *,' '
         write(i_herd,*) l
         i_herd=adjustl(i_herd)
```

```
            rbar_file='RBAR_'//trim(i_herd)//'.bin'
            thet_file='THET_'//trim(i_herd)//'.bin'
            solda_file='EBV_DAM_herd'//trim(i_herd)//'.out'
            solcg_file='BETA_CG_herd'//trim(i_herd)//'.out'
            open(60,file=rbar_file,access='direct',&
                 recl=8,form='unformatted',status='old')
            open(140,file=thet_file,access='direct',&
                 recl=8,form='unformatted',status='old')
            np=nda(l)+ncg(l)+nsire
            nrbar=np*(np-1)/2-sizesisi
            allocate(rbar(1:nrbar),thetab(1:np),beta(1:np))
            rbar=0.0d0; thetab=0.0d0; beta=0.0d0
            do i=1,nsire
               j=i+nda(l)+ncg(l)
               beta(j)=betasi(i)
            end do
            do j=1,nrbar
               read(60,REC=j) rbar(j)
            end do
            do k=1,nda(l)+ncg(l)
               read(140,REC=k) thetab(k)
            end do
            close(60)
            close(140)
            do j=1,nda(l)+ncg(l)
               i=nda(l)+ncg(l)-j+1
               beta(i)=thetab(i)
               nextr=(i-1)*(np+np-i)/2+1
               ip=i+1
               do k=ip,np
                  beta(i)=beta(i)-rbar(nextr)*beta(k)
                  nextr=nextr+1
               end do
            end do
            open(300,file=solda_file,status='replace')
            open(400,file=solcg_file,status='replace')
            do i=1,nda(l)
               write(300,'(f20.10)') beta(i)
            end do
            do j=nda(l)+1,nda(l)+ncg(l)
               write(400,'(f20.10)') beta(j)
            end do
            deallocate(rbar,thetab,beta)
            close(300)
            close(400)
         end do
!
         deallocate(nda,ncg,betasi,acc)
!
         call date_and_time (date,time,zone,fval)
         call cpu_time(cputime_fin)
         cputime=cputime_fin-cputime_ini
!
         print *,' '
```

```
      print *,' '
      print *,' '
      print *,'    Execution completed sucessfully '
      print *,' '
      print *,' '
      print *,' '
      print *,'    Initial time: ',ival(1),'/',ival(2),'/',ival(3),&
                         '  at  ',ival(5),'h',ival(6),'m',ival(7),'s'
      print *,' '
      print *,'      Final time: ',fval(1),'/',fval(2),'/',fval(3),&
                         '  at  ',fval(5),'h',fval(6),'m',fval(7),'s'
      print *,' '
      print *,' '
      print *,' '
      print *,'    Elapsed CPU time was: ',cputime,'sec'
      print *,' '
      print *,' '
      flush(6)
!
      stop
      end program main
!
!     ------------------------------------------------------------------
!     ------------------------------------------------------------------
!
```

**APÊNDICE E** – PEVD.f90: código fonte em Fortran 90 para obter os elementos da parte ‘touro x touro’ da inversa da “mão esquerda” das MME. Calcula os PEVs de cada EBV e os PEVs das diferenças/contrastes entre os EBVs dos touros.

```
!
!     PEVD.f90 (Genetic Evaluation with Gentleman-Givens rotations)
!
!     By Eduardo CG Pimentel           10/23/2007
!     E-mail: pimentel@fcav.unesp.br
!
!     ---------------------------------------------------------------------
!     ---------------------------------------------------------------------
!
      program main
      implicit none
!
!     ---------------------------------------------------------------------
!
      character(10) :: date,time,zone,i_herd
      character(20) :: herd_file
!
      integer :: i,j,k,ival(8),fval(8),initm,initr,nextm,rpos,npt,nherd
      integer :: nobs,np,nrbar,ninv,diagm,offm,curr,next,count,posm,posr
!
      real(8), allocatable :: d(:),rbar(:),sqd(:),chol(:)
      real(8), allocatable :: inv(:),accu(:),var(:),cov(:)
!
      real(8) :: sserr,mse,aux1,aux2,pev
!
      real(8), parameter :: alpha=1.0d0   ! ratio of variances
!
!     ---------------------------------------------------------------------
!
      open(50,file='ACC_SIRE.out',status='replace')
      open(90,file='RBAR_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(130,file='DIAG_SIRE.bin',access='direct',&
           recl=8,form='unformatted',status='old')
      open(150,file='info.txt',status='old')
      open(10,file='off_diag.inv',status='replace')
      open(20,file='PEV_SIRE.out',status='replace')
!
!     ---------------------------------------------------------------------
!
      read(150,'(i40)') nherd     ! Total number of herds
      read(150,'(f40.30)') sserr  ! Residual sum of squares
      read(150,'(i40)') nobs      ! Total number of observations
      read(150,'(i40)') j         ! (skip) N of males with info
      read(150,'(i40)') np        ! Total number of sires
!
!     ---------------------------------------------------------------------
!
      call date_and_time (date,time,zone,ival)
!
      print *,' '
      print *,' '
      print *,'    Reading upper triangular of sires...  '
      print *,' '
```

```
!
      nrbar=np*(np-1)/2
      allocate(d(1:np),rbar(1:nrbar))
      d=0.0d0; rbar=0.0d0
!
      do i=1,nrbar
         read(90,REC=i) rbar(i)
      end do
      do j=1,np
         read(130,REC=j) d(j)
      end do
      close(90)
      close(130)
!
      print *,' '
      print *,'    Scaling the Cholesky factor...  '
      print *,' '
!
      allocate(sqd(1:np),chol(1:nrbar))
      sqd=0.0d0; chol=0.0d0
      sqd=sqrt(d)
      next=1
      do i=1,np-1
         curr=next
         next=next+np-i
         do j=curr,next-1
            chol(j)=rbar(j)*sqd(i)
         end do
      end do
      deallocate(d,rbar)
!
      npt=np
      do i=1,nherd
         write(i_herd,*) i
         i_herd=adjustl(i_herd)
         herd_file='HERD_'//trim(i_herd)//'.par'
         open(30,file=herd_file,status='old')
         read(30,'(i40)') j
         read(30,'(i40)') k
         close(30)
         npt=npt+j+k
      end do
      mse=sserr/(nobs-npt)
!
      print *,' '
      print *,'    Calculating off-diagonals of inverse...  '
      print *,' '
!
      ninv=(np*(np+1))/2     ! dimension of M=inv(A'A) (incl. diag)
      allocate(inv(1:ninv),accu(1:np))
      allocate(var(1:np),cov(1:nrbar))
      inv=0.0d0; accu=0.0d0; var=0.0d0; cov=0.0d0
!
      curr=0
```

```
      initm=0
      diagm=1
      initr=nrbar
      posr=nrbar+1
      do i=np,1,-1
         aux1=0.0d0
         if (i<np) then
            posm=np-i+1
            count=0
            do j=i+1,np
               posr=posr-1
               count=count+1
               aux1=aux1+chol(posr)*inv(posm)
               posm=posm+np-count
            end do
         end if
         inv(diagm)=(1/sqd(i)-aux1)/sqd(i)
         var(i)=inv(diagm)*mse
         accu(i)=sqrt(1-(inv(diagm)*alpha))
         if (i==1) exit
         offm=diagm
         initr=initr+i-np
         rpos=initr
         initm=initm+1
         do k=i-1,1,-1
            nextm=initm
            count=0
            offm=offm+1
            aux2=0.0d0
            do while (nextm<diagm)
               aux2=aux2-chol(rpos)*inv(nextm)
               rpos=rpos-1
               count=count+1
               nextm=nextm+np-count
            end do
            do while (offm>nextm)
               aux2=aux2-chol(rpos)*inv(nextm)
               rpos=rpos-1
               nextm=nextm+1
            end do
            inv(offm)=aux2/sqd(k)
            write(10,'(2(i10,1x),f20.10)') i,k,inv(offm)
            curr=curr+1
            cov(curr)=inv(offm)*mse
         end do
         diagm=diagm+i
      end do
!
      deallocate(inv,sqd,chol)
!
      print *,' '
      print *,'    Writing results...  '
      print *,' '
!
```

```
      do i=1,np
         write(50,'(f20.10)') accu(i)
      end do
!
      curr=0
      do j=np,1,-1
         pev=var(j)
         write(20,'(2(i10,1x),f20.10)') j,j,pev
         if(j==1) exit
         do k=j-1,1,-1
            curr=curr+1
            pev=var(j)+var(k)-2*cov(curr)
            write(20,'(2(i10,1x),f20.10)') j,k,pev
         end do
      end do
!
      deallocate(var,cov)
!
      call date_and_time (date,time,zone,fval)
!
      print *,' '
      print *,' '
      print *,'    Initial time: ',ival(1),'/',ival(2),'/',ival(3),&
                           '  at  ',ival(5),'h',ival(6),'m',ival(7),'s'
      print *,' '
      print *,'      Final time: ',fval(1),'/',fval(2),'/',fval(3),&
                           '  at  ',fval(5),'h',fval(6),'m',fval(7),'s'
      print *,' '
      print *,' '
      print *,' '
!
      stop
      end program main
!
!     ------------------------------------------------------------------
!     ------------------------------------------------------------------
!
```

# Livros Grátis

( http://www.livrosgratis.com.br )

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo