IMA Journal of Numerical Analysis (2010) **30**, 94–107
doi:10.1093/imanum/drn085
Advance Access publication on June 8, 2009

A preconditioned Newton algorithm for the nearest correlation matrix

RÜDIGER BORSDORF[†]
Department of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany

AND

NICHOLAS J. HIGHAM[‡]
School of Mathematics, The University of Manchester, Manchester M13 9PL, UK

[Received on 21 April 2008; revised on 1 December 2008]

Dedicated to the memory of A. R. Mitchell, 1921–2007.

Various methods have been developed for computing the correlation matrix nearest in the Frobenius norm to a given matrix. We focus on a quadratically convergent Newton algorithm recently derived by Qi and Sun. Various improvements to the efficiency and reliability of the algorithm are introduced. Several of these relate to the linear algebra: the Newton equations are solved by minres instead of the conjugate gradient method, as it more quickly satisfies the inexact Newton condition; we apply a Jacobi preconditioner, which can be computed efficiently even though the coefficient matrix is not explicitly available; an efficient choice of eigensolver is identified; and a final scaling step is introduced to ensure that the returned matrix has unit diagonal. Potential difficulties caused by rounding errors in the Armijo line search are avoided by altering the step selection strategy. These and other improvements lead to a significant speed-up over the original algorithm and allow the solution of problems of dimension a few thousand in a few tens of minutes.

Keywords: correlation matrix; positive semidefinite matrix; Newton's method; preconditioning; rounding error; Armijo line search conditions; alternating projections method.

1. Introduction

Correlation matrices are real, symmetric positive semidefinite matrices with ones on the diagonal. They arise in situations where correlations between pairs of random variables are computed, and also when pairwise similarity measures between objects are formed and suitably scaled, for example, in machine learning (Qi *et al.*, 2007). It is common, in practice, to be faced with an *approximate* correlation matrix: a matrix that is supposed to be a correlation matrix but for a variety of possible reasons is not. In finance, for example, the correlations may be between stocks measured over a period of time and missing data (perhaps due to a company not trading for the whole period) may compromise the correlations and lead to a non-positive semidefinite matrix. Again in finance, a practitioner may wish to explore the effect on a portfolio of assigning correlations between certain assets differently from the historical values, but this again can destroy the semidefiniteness of the matrix. The use of approximate correlation matrices in these applications can render the methodology invalid and lead to negative variances and volatilities being computed (Finger, 1997; Turkay *et al.*, 2003; Qi & Sun, 2007).

[†]Email: ruediger.borsdorf@s2003.tu-chemnitz.de
[‡]Corresponding author. Email: higham@ma.man.ac.uk

The prevalence of approximate correlation matrices has led to much interest in the problem of computing the nearest correlation matrix to a given matrix $A \in \mathbb{R}^{n\times n}$, that is, solving the problem

$$\min\{\|A - X\|_F \colon X = X^{\mathrm{T}}, X \geqslant 0, \mathrm{Diag}(X) = e\}, \tag{1.1}$$

where for symmetric matrices X and Y, $X \geqslant Y$ denotes that $X - Y$ is positive semidefinite, $\mathrm{Diag}(X)$ is the vector of diagonal elements of X, e is the vector of ones and the Frobenius norm $\|X\|_F = \mathrm{trace}(X^{\mathrm{T}}X)^{1/2}$. As $\mathbb{R}^{n\times n}$ is a Hilbert space with inner product $\langle X, Y\rangle = \mathrm{trace}(X^{\mathrm{T}}Y)$ and the constraints in (1.1) are closed convex sets, (1.1) has a unique solution (Deutsch, 2001, Theorem 3.5).

The nearness problem (1.1) has been extensively studied over the last twenty years. Much of the literature is concerned with *ad hoc* methods that are not guaranteed to solve the problem. An early example is a method of Knol & ten Berge (1989) that writes $X = Y^{\mathrm{T}}Y$ and iteratively minimizes the objective function over each unit 2-norm column of Y. More recently, Lurie & Goldberg (1998) used the Gauss–Newton method to minimize $\|A - R^{\mathrm{T}}R\|_F^2$, where R is upper triangular with columns of unit 2-norm.

Higham (2002b) used convex analysis to give a characterization of the solution and described an alternating projections algorithm that converges linearly to the solution. This algorithm has several attractive features. First, it is very simple to implement, requiring just matrix additions and computation of eigendecompositions. Second, it can take advantage of the property, proved in Higham (2002b), that if $a_{ii} \geqslant 1$ for all i and A has many negative eigenvalues (as is likely in finance applications) then the solution has at least as many zero eigenvalues (so is of low rank). Third, it is readily adapted to solve the problem with additional constraints that require X to belong to a convex set, such as a constraint that holds any set of elements of X fixed (Borsdorf, 2007, Chapter 7). The main drawback of the alternating projections algorithm is its possibly slow convergence.

Malick (2004) studied a problem more general than (1.1) in which the positive semidefinite matrices are replaced by a convex set and the constraints on the diagonal of X are replaced by general linear constraints. He dualized the linear constraints and applied a quasi-Newton method to the dual problem. Boyd & Xiao (2005) explored similar ideas. When applied to the problem (1.1), the methods in both of these papers can be expected to be at best linearly convergent because the dual objective function is not twice continuously differentiable. A breakthrough was subsequently made by Qi & Sun (2006), who derived a quadratically convergent Newton method for (1.1), again by working with the dual problem. The proof of quadratic convergence relies heavily on the theory of semismooth optimization. Interior point methods can also be applied to classes of problems containing (1.1). Toh (2008) developed such a method that requires the solution of dense linear systems of dimension about $n^2/2$, constructed preconditioners for the systems and applied the method to (1.1).

Qi & Sun (2006) built from their theory a globally convergent Newton algorithm for finding the nearest correlation matrix and illustrated its performance on a small set of artificial test problems. The purpose of our work is to improve the efficiency and reliability of the algorithm through a careful analysis of its component steps. The main improvements we make are as follows.

- The Newton equations are solved by minres instead of the conjugate gradient (CG) method, since minres minimizes the residual that appears in the inexact Newton condition.
- We show how to efficiently apply a Jacobi preconditioner to the Newton equations—a nontrivial task since the coefficient matrix is not explicitly available.
- The line search is modified so as to perform reliably in finite precision arithmetic when the convergence tolerance is close to the machine precision.

- We show experimentally that the choice of eigensolver can have a significant effect on the computational cost and identify a suitable choice.
- We introduce a final scaling step, justified by a distance bound, that ensures that the returned matrix has unit diagonal.

The outline of the paper is as follows. In Section 2 we present background on the dual problem and the Newton method and we state the algorithm of Qi and Sun. In Section 3 we develop our refinements to the algorithm. The improved algorithm is described in Section 4 and some numerical experiments are reported in Section 5. Concluding remarks are in Section 6.

2. Newton algorithm of Qi and Sun

In this section we summarize the key results from the analysis of Qi & Sun (2006) that will be needed later and we state the algorithm of Qi and Sun.

The problem obtained by dualizing the linear constraints in the nearest correlation matrix problem (1.1) is the unconstrained convex optimization problem (Malick, 2004; Qi & Sun, 2006)

$$\min_{y\in\mathbb{R}^n} f(y) := \frac{1}{2}\|(A+\operatorname{diag}(y))_+\|_F^2 - e^{\mathrm{T}}y. \tag{2.1}$$

Here $\operatorname{diag}(y)$ for $y \in \mathbb{R}^n$ denotes the diagonal matrix whose diagonal elements are those of the vector y, while $\operatorname{diag}(A)$ for $A \in \mathbb{R}^{n\times n}$ denotes $\operatorname{diag}([a_{11}, a_{22}, \dots, a_{nn}])$. (Recall that Diag, introduced in Section 1, maps matrices onto vectors.) The operator $(\cdot)_+$ projects onto the positive semidefinite matrices: for symmetric $C \in \mathbb{R}^{n\times n}$ with spectral decomposition $C = Q\Lambda Q^{\mathrm{T}}$ ($Q^{\mathrm{T}}Q = I$ and $\Lambda = \operatorname{diag}(\lambda_i)$), $C_+ = Q\operatorname{diag}(\max(\lambda_i, 0))Q^{\mathrm{T}}$ is the nearest positive semidefinite matrix to C in the Frobenius norm (Higham, 1988). The following lemma collects some key properties of the dual problem obtained by Malick (2004) (see also Micchelli & Utreras (1988, Lemma 2.1) for a proof of some of these properties in greater generality).

LEMMA 2.1 The dual problem (2.1) has the following properties:

(a) f is convex and continuously differentiable and has a unique minimizer;

(b) the gradient ∇f is given by

$$\nabla f(y) = \operatorname{Diag}((A+\operatorname{diag}(y))_+) - e \tag{2.2}$$

and is Lipschitz continuous with Lipschitz constant 1;

(c) the solutions y_* of the dual problem (2.1) and X_* of the primal problem (1.1) are related by

$$X_* = (A+\operatorname{diag}(y_*))_+. \tag{2.3}$$

Note that the lemma shows that the original constrained problem with $(n^2-n)/2$ variables is equivalent to an unconstrained problem with just n variables.

To find y_* we need to solve $g(y_*) = 0$, where $g(y) = \nabla f(y)$. Noting that g is not differentiable, we denote by ∂g the generalized Jacobian, which is defined since ∇f is Lipschitz continuous. Qi and Sun applied the generalized Newton iteration

$$y_{k+1} = y_k - V_k^{-1}g(y_k), \quad V_k \in \partial g(y_k), \ \ k = 0\colon \infty. \tag{2.4}$$

For a general g this iteration need not converge. However, by exploiting the strong semismoothness of the operator $(\cdot)_+$, Qi and Sun were able to prove quadratic convergence of the iteration for $g = \nabla f$.

THEOREM 2.2 (Qi and Sun) Let y_* denote the minimizer of (2.1). All $V \in \partial g(y_*)$ are positive definite and the generalized Newton method (2.4) converges quadratically to y_* for any choice of V_k if y_0 is sufficiently close to y_*.

In order to implement the method we need to be able to compute a generalized Jacobian $V \in \partial g(y)$. Qi and Sun showed that such a matrix is given implicitly by

$$V_y h = \mathrm{Diag}(P_y(W_y \circ (P_y^{\mathrm{T}} H P_y))P_y^{\mathrm{T}}). \tag{2.5}$$

Here $\circ$ denotes the Hadamard product ($X \circ Y = (x_{ij}y_{ij})$), $h \in \mathbb{R}^n$ and $H = \mathrm{diag}(h)$, P_y is an orthogonal matrix calculated from the spectral decomposition of $A + \mathrm{diag}(y)$:

$$A + \mathrm{diag}(y) = P_y \mathrm{diag}(\lambda(y)) P_y^{\mathrm{T}}, \tag{2.6}$$

with $\lambda(y)$ the vector of eigenvalues, and W_y depends on the eigenvalues $\lambda(y)$ in the way we now describe. Let $\lambda(y)$ be in descending order and define the sets $\alpha = \{i\colon \lambda_i(y) > 0\}$, $\beta = \{i\colon \lambda_i(y) = 0\}$ and $\gamma = \{i\colon \lambda_i(y) < 0\}$. Then the matrix W_y is defined by

$$W_y = \begin{bmatrix} E_{\alpha\alpha} & E_{\alpha\beta} & \mathcal{T} \\ E_{\beta\alpha} & 0 & 0 \\ \mathcal{T} & 0 & 0 \end{bmatrix}, \quad \mathcal{T} = \left(\frac{\lambda_i(y)}{\lambda_i(y) - \lambda_j(y)} \right)_{i \in \alpha, j \in \gamma}, \tag{2.7}$$

where $E_{\alpha\beta}$ denotes the matrix of ones of dimension $|\alpha| \times |\beta|$. It is easy to show that $V_y \geqslant 0$. We have

$$\begin{aligned} h^{\mathrm{T}} V_y h &= \mathrm{trace}(H P_y (W_y \circ (P_y^{\mathrm{T}} H P_y)) P_y^{\mathrm{T}}) \\ &= \mathrm{trace}(\widetilde{H}(W_y \circ \widetilde{H})), \quad \text{where } \widetilde{H} = P_y^{\mathrm{T}} H P_y, \\ &= \|\widetilde{W} \circ \widetilde{H}\|_F^2, \quad \text{where } \widetilde{W} \circ \widetilde{W} = W_y, \\ &\geqslant 0, \end{aligned} \tag{2.8}$$

using the symmetry of $\widetilde{H}$.

The matrix V_y can be explicitly computed by setting $h = e_i$ in (2.5), for $i = 1\colon n$, where e_i is the ith unit vector. But, since evaluating (2.5) for a single h costs $\mathcal{O}(n^3)$ operations, obtaining V_y costs a prohibitively expensive $\mathcal{O}(n^4)$ operations. We are therefore restricted to solving the Newton equation by methods that require matrix–vector products only.

The following algorithm implements the method above as an inexact Newton method (the linear system (2.4) is solved only approximately) and it uses a line search strategy and globalization techniques. The algorithm is globally convergent and is essentially the same as Algorithm 5.1 of Qi & Sun (2006).

ALGORITHM 2.3 Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a convergence tolerance tol, this algorithm computes the nearest correlation matrix X to A in the Frobenius norm. On termination $\|\nabla f(y_k)\|_2 \leqslant$ tol (see (2.2)). The algorithm is quadratically convergent.

```
Step 1: Initialization: $y_0 \in \mathbb{R}^n$, $\eta \in (0,1)$, $\rho, \sigma \in (0, 1/2]$ and $k = 0$.
Step 2: Calculate $\nabla f(y_k)$. If $\|\nabla f(y_k)\|_2 \leqslant$ tol then set $X = (A + \mathrm{diag}(y_k))_+$ and quit.
```

Step 3: Compute a spectral decomposition (2.6) of $A + \operatorname{diag}(y_k)$ and form the matrix W_{y_k} from (2.7).

Step 4: Determine the new direction d_k by applying an iterative method (using (2.5) to compute $V_k d$) to

$$V_k d = -\nabla f(y_k), \tag{2.9}$$

terminating when both the conditions

$$\|\nabla f(y_k) + V_k d_k\|_2 \leqslant \eta_k \|\nabla f(y_k)\|_2, \tag{2.10}$$

$$-\frac{\nabla f(y_k)^{\mathrm{T}}}{\|d_k\|_2} \cdot \frac{d_k}{\|d_k\|_2} \geqslant \eta_k \tag{2.11}$$

are satisfied, where $\eta_k = \min(\eta, \|\nabla f(y_k)\|_2)$. If either one of these conditions cannot be satisfied then let

$$d_k = -B_k^{-1} \nabla f(y_k), \tag{2.12}$$

where B_k is any symmetric positive definite matrix with $\{\|B_k\|_2\}$ and $\{\|B_k^{-1}\|_2\}$ uniformly bounded.

Step 5: Choose an appropriate step length α_k by applying Armijo backtracking: find the smallest nonnegative integer m_k such that

$$f(y_k + \rho^{m_k} d_k) \leqslant f(y_k) + \sigma \rho^{m_k} \nabla f(y_k)^{\mathrm{T}} d_k \tag{2.13}$$

is satisfied.

Step 6: Set $\alpha_k = \rho^{m_k}$, $y_{k+1} = y_k + \alpha_k d_k$ and $k \leftarrow k + 1$. Go to Step 2.

In Section 3 we develop several refinements that improve the efficiency and robustness of the basic algorithm.

3. Refinements

3.1 *Linear equation solver*

Qi & Sun (2006) took the CG method as the solver for the Newton system (2.9), motivated by the fact that V_k is positive semidefinite for all k and positive definite for sufficiently large k (see (2.8) and Theorem 2.2). The stopping criterion (2.10) is based on the norm of the residual $r_k^{(j)} = \nabla f(y_k) + V_k d_k^{(j)}$ of the iterates $d_k^{(j)}$, but the CG method minimizes the *error* $d - d_k^{(j)}$ in the V_k norm ($\|x\|_{V_k} = (x^{\mathrm{T}} V_k x)^{1/2}$) on the jth iteration rather than the residual, and it can produce very nonmonotonic residuals. Moreover, the possible singularity of the coefficient matrix V_k can cause problems for the CG method.

Instead of CG, we use the minres method of Paige & Saunders (1975). This method minimizes the residual norm $\|r_k^{(j)}\|_2$ on the jth iteration and so produces a monotonically decreasing sequence of residuals. Moreover, unlike CG, minres is defined for indefinite systems and thus it should be more stable than CG in finite precision arithmetic for nearly singular or numerically indefinite matrices. Minres requires only one matrix–vector product per iteration, but it requires a few more vector operations than CG.

3.2 *Preconditioning*

As we noted in Section 2, the coefficient matrix V_k is not explicitly available. Nothing is known about the eigensystem of V_k, apart from the non-negativity of the eigenvalues, and preconditioning the system (2.9) is therefore a challenge. However, it is possible to compute the diagonal elements of V_k in $\mathcal{O}(n^3)$ operations and thereby to apply the Jacobi preconditioner. Recall that the Jacobi preconditioner for a positive definite matrix A is $D = \mathrm{diag}(A)$ and the preconditioned matrix is $D^{-1/2}AD^{-1/2}$, which has 2-norm condition number within a factor n of the minimum over all diagonal congruences, by a result of van der Sluis (1969) (Higham, 2002a, Corollary 7.6). The Jacobi preconditioner is a reasonable choice in view of the existence of residual bounds for minres that depend on $\kappa_2(A)$ (Greenbaum, 1997, Chapter 3; Elman *et al.*, 2005, Chapter 6).

To see how to compute $\mathrm{diag}(V_k)$ let $h = e_i$, $H = e_i e_i^{\mathrm{T}}$ and $P_k^{\mathrm{T}} = [p_1, p_2, \dots, p_n]$. Then, from (2.5), the (i, i) element of V_k is given by

$$\begin{aligned} v_{ii} &= e_i^{\mathrm{T}} P_k(W_k \circ P_k^{\mathrm{T}} H P_k) P_k^{\mathrm{T}} e_i = p_i^{\mathrm{T}}(W_k \circ p_i p_i^{\mathrm{T}}) p_i \\ &= p_i^{\mathrm{T}} \mathrm{diag}(p_i) W_k \mathrm{diag}(p_i) p_i = q_i^{\mathrm{T}} W_k q_i, \end{aligned}$$

where $q_i = p_i \circ p_i$. Thus the diagonal elements v_{ii} can be computed as follows:

$$\begin{aligned} Q_k &= [q_1, q_2, \dots, q_n] = P_k \circ P_k, && n^2 \text{ flops}, \\ M_k &= [m_1, m_2, \dots, m_n] = W_k Q_k, && \leqslant 2n^3 \text{ flops}, \\ v_{ii} &= q_i^{\mathrm{T}} m_i, \quad i = 1\colon n, && 2n^2 \text{ flops}. \end{aligned}$$

The dominant cost is therefore the matrix–matrix multiplication giving M_k. In forming M_k the zero and ee^{T} blocks of W_k (see (2.7)) can be exploited to reduce the cost.

To allow for a possibly singular V_k and the effects of rounding errors we set all diagonal entries less than a predefined positive tolerance to that tolerance.

3.3 *Armijo backtracking*

We are aiming for an algorithm that is capable of computing the nearest correlation matrix to full machine accuracy, and so we wish to allow the convergence tolerance tol in Algorithm 2.3 to be of the order of the unit roundoff u. However, the Armijo backtracking can break down for small tolerances. To see why, consider a twice continuously differentiable function $\phi\colon \mathbb{R}^n \to \mathbb{R}$ and the expansion

$$\phi(x + p) = \phi(x) + \nabla\phi(x)^{\mathrm{T}} p + \frac{1}{2} p^{\mathrm{T}} \nabla^2 \phi(x + tp)^{\mathrm{T}} p, \quad t \in (0, 1).$$

If $|\phi(x)| = 1$, $\|p\|_2 < u^{1/2}$, $\|\nabla\phi(x)\|_2 < u^{1/2}/2$ and $\|\nabla^2\phi(x + tp)\|_2 < 1$ then $|\phi(x + p) - \phi(x)| < u|\phi(x)|$, and so $fl(\phi(x + p)) = fl(\phi(x))$. In this situation x may still be some distance from a minimizer of ϕ—albeit perhaps only one or two steps away for a quadratically converging method—yet the Armijo condition cannot be verified because the function values it needs to compare are indistinguishable in floating-point arithmetic. In numerical experiments we have found that this problem with the Armijo condition can cause Algorithm 2.3 to fail to converge in finite precision arithmetic.

To avoid this problem, when $fl(f(y_k+\rho^{m_k}d_k))$ and $fl(f(y_k))$ are close enough that rounding errors are dominating we take the inexact Newton direction with step length 1, provided that the resulting y_{k+1} satisfies

$$\frac{\|\nabla f(y_{k+1})\|_2}{\|\nabla f(y_k)\|_2} \leqslant 1-\mu \quad \text{for some } \mu \in (0,1), \tag{3.1}$$

where the latter condition ensures that useful progress is made towards the minimizer. Or, if (3.1) is not satisfied we take the steepest descent direction with step length 1.

The next result provides support for the test (3.1) by showing that it is satisfied for sufficiently large k.

LEMMA 3.1 For sufficiently large k in Algorithm 2.3 we have $y_{k+1} = y_k + d_k$, with d_k the inexact Newton direction and

$$\frac{\|\nabla f(y_{k+1})\|_2}{\|\nabla f(y_k)\|_2} = \mathcal{O}(\|d_k\|_2).$$

Proof. From the proof of Theorem 5.3 of Qi & Sun (2006), we know that for all sufficiently large k the inexact Newton step is taken with step $\alpha_k = 1$, that d_k satisfies (2.10) and (2.11), that

$$\|y_{k+1}-y_*\|_2 = \mathcal{O}(\|y_k-y_*\|_2^2), \tag{3.2}$$

$$\|y_k-y_*\|_2 \leqslant \|d_k\|_2 + \mathcal{O}(\|d_k\|_2^2) \tag{3.3}$$

and also that there exists a $\rho > 0$ so that

$$\|\nabla f(y_k)\|_2\|d_k\|_2 \geqslant -\nabla f(y_k)^{\mathrm{T}}d_k \geqslant \rho\|d_k\|_2^2. \tag{3.4}$$

It follows from (3.3) and (3.4) that, for sufficiently large k,

$$\frac{\|y_k-y_*\|_2^2}{\|\nabla f(y_k)\|_2} \leqslant \frac{\|y_k-y_*\|_2^2}{\rho\|d_k\|_2} \leqslant \frac{1}{\rho}\|d_k\|_2 + \mathcal{O}(\|d_k\|_2^2) = \mathcal{O}(\|d_k\|_2). \tag{3.5}$$

From (3.2), using the Lipschitz property in Lemma 2.1(b) of $\nabla f(y)$ and (3.5), we deduce that

$$\begin{aligned}\frac{\|\nabla f(y_{k+1})\|_2}{\|\nabla f(y_k)\|_2} &= \frac{\|\nabla f(y_{k+1})-\nabla f(y_*)\|_2}{\|\nabla f(y_k)\|_2} \leqslant \frac{\|y_{k+1}-y_*\|_2}{\|\nabla f(y_k)\|_2}\\ &= \mathcal{O}\left(\frac{\|y_k-y_*\|_2^2}{\|\nabla f(y_k)\|_2}\right) = \mathcal{O}(\|d_k\|_2),\end{aligned}$$

which completes the proof. □

3.4 *Accuracy of the solution*

A subtle problem with Algorithm 2.3 is that it does not yield a matrix with unit diagonal because the constraints $\mathrm{Diag}(X) = e$ are not explicitly enforced. Indeed, if $\nabla f(y) \neq 0$ on termination then it is clear from (2.2) and (2.3) that the returned matrix does not have unit diagonal. If we simply set the diagonal elements to 1 then we may destroy the definiteness. We could then restore definiteness by projecting onto the nearest positive semidefinite matrix, which changes the diagonal. Iterating this procedure essentially gives the alternating projections algorithm (Higham, 2002b).

We will adopt a simpler and less expensive approach. We replace the final iterate X by

$$\widetilde{X} = D^{-1/2} X D^{-1/2}, \quad D = \mathrm{diag}(X),$$

which has unit diagonal. Note that this is precisely the transformation that takes a covariance matrix into the associated correlation matrix. Since this transformation is a congruence, it preserves the definiteness of X. However, it can increase the distance from A. The next lemma provides a bound on the increase.

LEMMA 3.2 If $X \geqslant 0$ is the output of Algorithm 2.3 and $D = \mathrm{diag}(X) > 0$ then

$$\|A - D^{-1/2} X D^{-1/2}\|_F \leqslant \|A - X\|_F + \frac{\mathrm{tol}}{1-\mathrm{tol}} \|X\|_F. \tag{3.6}$$

Proof. We have

$$\|A - D^{-1/2} X D^{-1/2}\|_F \leqslant \|A - X\|_F + \|X - D^{-1/2} X D^{-1/2}\|_F. \tag{3.7}$$

Our aim is to bound the second term of (3.7) by using $\|\nabla f(y_k)\|_2 \leqslant \mathrm{tol}$, where y_k is the final iterate of Algorithm 2.3. From (2.2) and (2.3) it follows that

$$D = \mathrm{diag}(\nabla f(y_k)) + I. \tag{3.8}$$

With $G = X - D^{-1/2} X D^{-1/2}$, we have

$$\begin{aligned} g_{ij}^2 &= \left(x_{ij} - (\nabla f(y_k)_i + 1)^{-1/2} x_{ij} (\nabla f(y_k)_j + 1)^{-1/2} \right)^2 \\ &= x_{ij}^2 \left(1 - (\nabla f(y_k)_i + 1)^{-1/2} (\nabla f(y_k)_j + 1)^{-1/2} \right)^2. \end{aligned} \tag{3.9}$$

Using $|\nabla f(y_k)_i| \leqslant \mathrm{tol}$ for all i yields

$$\frac{1}{1+\mathrm{tol}} \leqslant (\nabla f(y_k)_i + 1)^{-1/2} (\nabla f(y_k)_j + 1)^{-1/2} \leqslant \frac{1}{1-\mathrm{tol}}. \tag{3.10}$$

Hence, in order to find an upper bound for g_{ij}^2 it is enough to maximize the function $f(s) = (1 - 1/(1+s))^2 = s^2/(1+s)^2$ over $s \in [-\mathrm{tol}, \mathrm{tol}]$. The maximum is attained at $s = -\mathrm{tol}$ and so we obtain from (3.9) that $g_{ij}^2 \leqslant x_{ij}^2 \mathrm{tol}^2/(1-\mathrm{tol})^2$, giving $\|G\|_F \leqslant \|X\|_F \mathrm{tol}/(1-\mathrm{tol})$. The required bound then follows. □

The bound (3.6) is very satisfactory: it says that the increase in the distance $\|A - X\|_F$ induced by the normalization of the diagonal is at most about $\mathrm{tol}\|X\|_F \lesssim n\mathrm{tol}$, and we expect $n\mathrm{tol} \ll \|A - X\|_F$ in applications.

3.5 *Choice of eigensolver*

Algorithm 2.3 requires a full eigenvalue decomposition of the symmetric matrix $A + \mathrm{diag}(y)$ for every evaluation of $f(y)$ and $\nabla f(y)$, and thus at least one eigenvalue decomposition per iteration. This is a major part of the total cost of the method, and so it is essential to minimize its cost.

There are three main algorithmic options, for which the NAG Library and LAPACK codes are `f08fa`/`dsyev`, `f08fc`/`dsyevd` and `f08fd`/`dsyevr`. All three algorithms reduce the symmetric matrix to a tridiagonal matrix but then proceed differently: `f08fa` uses the QR algorithm, `f08fc` uses the divide and conquer algorithm and `f08fd` uses the dqds algorithm and multiple relatively robust representations (MRRR). In our numerical experiments we compare these algorithms.

4. The modified algorithm

The following modification of Algorithm 2.3 incorporates the improvements described in Section 3.

ALGORITHM 4.1 Given a matrix $A \in \mathbb{R}^{n\times n}$ and a convergence tolerance tol, this algorithm computes the nearest correlation matrix X to A in the Frobenius norm. On termination $\|\nabla f(y_k)\|_2 \leqslant$ tol (see (2.2)). The algorithm is quadratically convergent.

```
Step 1: Initialization: $\eta = 0.5$, $\varphi = 10^{-6}$, $\mu \in (0, 1)$, $\rho, \sigma \in (0, 1/2]$ and $k = 0$.
Step 2: Set $A \leftarrow (A + A^{\mathrm{T}})/2$ if $A$ is nonsymmetric. Set $a_{ii} = 1$, for $i = 1\colon n$, and $y_0 = 0$.
Step 3: Calculate $\nabla f(y_k)$. If $\|\nabla f(y_k)\|_2 \leqslant$ tol, set $X = D^{-1/2}\widetilde{X}D^{-1/2}$, where $\widetilde{X} = (A + \mathrm{diag}(y_k))_+$
        and $D = \mathrm{diag}(\widetilde{X})$, and quit.
Step 4: Compute a spectral decomposition of $A + \mathrm{diag}(y_k)$ and form the matrix $W_{y_k}$ from (2.7) and the
        Jacobi preconditioner $D_k$ (see Section 3.2).
Step 5: Determine the new direction $d_k$ by applying minres to the preconditioned linear system (using
        (2.5) to compute $V_k d$)

        $$D_k^{-1/2} V_k D_k^{-1/2} \cdot D_k^{1/2} d = -D_k^{-1/2} \nabla f(y_k), \qquad (4.1)$$

        terminating when both the conditions

        $$\|\nabla f(y_k) + V_k d_k\|_2 \leqslant \min(\eta, \|\nabla f(y_k)\|_2)\|\nabla f(y_k)\|_2, \qquad (4.2)$$

        $$-\frac{\nabla f(y_k)^{\mathrm{T}}}{\|d_k\|_2} \cdot \frac{d_k}{\|d_k\|_2} \geqslant \min(\varphi, \|\nabla f(y_k)\|_2) \qquad (4.3)$$

        are satisfied. If either of these conditions cannot be satisfied then let $d_k = -\nabla f(y_k)$.
Step 6: (Choice of step using Armijo backtracking.)
        For $m = 0\colon \infty$
            If $f(y_k + \rho^m d_k) \leqslant f(y_k) + \sigma\rho^m \nabla f(y_k)^{\mathrm{T}} d_k$ then set $\alpha_k = \rho^m$ and go to Step 7.
            If $f(y_k + \rho^m d_k)$ and $f(y_k)$ are 'nearly equal' then, if

            $$\frac{\|\nabla f(y_k + \alpha_k d_k)\|_2}{\|\nabla f(y_k)\|_2} \leqslant 1 - \mu \qquad (4.4)$$

            then set $\alpha_k = 1$ and go to Step 7, else set $d_k = -\nabla f(y_k)$ and $\alpha_k = 1$ and go to Step 7.
        end
Step 7: Set $y_{k+1} = y_k + \alpha_k d_k$ and $k \leftarrow k + 1$. Go to Step 3.
```

A few comments are in order.

(a) Since all matrices agreeing with A on the off-diagonal have the same nearest correlation matrix, we set the diagonal of A to unity at the start. When $y_0 = 0$ this gives immediate convergence if the resulting matrix is positive semidefinite.

(b) Step 2 projects onto the symmetric matrices (Higham, 1988) and allows the algorithm to work for nonsymmetric inputs A (Borsdorf, 2007, Theorem 4.91).

(c) Our use of $\varphi \ll \eta$ in (4.3) and (4.2) encourages the use of inexact Newton directions over the steepest descent direction, which our numerical experiments have shown leads to faster run times.

(d) In Step 6 a suitable test for two floating-point numbers a and b being 'nearly equal' is $|a-b| < \gamma\, u(1+|a|+|b|)$, where γ is a constant that we set to 100.

(e) Some other straightforward tests that terminate when rounding errors start to dominate are omitted to avoid clutter but are included in our implementation tested in Section 5.

(f) Qi & Sun (2006) showed how their Newton algorithm can be adapted for the problem in which the constraint $X \geqslant 0$ in (1.1) is replaced by $X \geqslant \beta I$, where $\beta \in (0,1)$. Algorithm 4.1 can likewise be adapted for this so-called 'calibration of correlation matrices' problem.

5. Numerical experiments

We now present some numerical experiments that illustrate the behaviour of Algorithm 4.1 and compare it with Algorithm 2.3 and the alternating projections method. The tests were carried out in MATLAB R2006b on a 2.4-GHz AMD Athlon under linux (Tables 1 and 2) and MATLAB R2007b on a 2.2-GHz AMD Athlon under Windows XP (Tables 3 and 4). The unit roundoff is $u = 2^{-53} \approx 1.1 \times 10^{-16}$. We invoked certain NAG Fortran Library (Mark 21) codes via the NAG Toolbox for MATLAB (Beta 1 under linux and Beta 2 under Windows) (NAG Toolbox for MATLAB).

The codes tested are as follows.

- `nearcor`: a MATLAB implementation of Algorithm 2.3 written by the authors of Qi & Sun (2006) and used in the testing in that paper.
- `nearcor_new`: our MATLAB implementation of Algorithm 4.1. We take $\mu = 0.9$, $\rho = 0.5$ and $\sigma = 10^{-4}$. We use a MATLAB implementation of minres provided by the authors of Elman *et al.* (2005).

TABLE 1 *Comparison of different iterative methods in* `nearcor_new`, *with* $\mathrm{tol} = 10^{-7} n$

	cor1399			cor3120	
	CG	minres		CG	minres
No preconditioning			No preconditioning		
T_{tot}	213.4	170.9	T_{tot}	2799	1736
T_{mvp}	146.3	104.3	T_{mvp}	2100	1138.6
T_{eig}	62.6	61.7	T_{eig}	662.0	562.8
Iters	7	7	Iters	7	6
# mvp	42	30	# mvp	57	31
With preconditioning			With preconditioning		
T_{tot}	142.4	111.0	T_{tot}	1197	905.7
T_{mvp}	76.6	45.2	T_{mvp}	624.2	331.1
T_{eig}	52.8	52.8	T_{eig}	468.3	469.2
Iters	6	6	Iters	5	5
# mvp	22	13	# mvp	17	9
Time Pre.	8.83	8.9	Time Pre.	72.9	72.89

TABLE 2 *Ratios of total time taken by* `nearcor_new` *with* `f08fa` *and* `f08fd` *to total time for* `nearcor_new` *with* `f08fc`

	`f08fa`	`f08fc`	`f08fd`
cor1399	2.1	1.0	1.9
cor3120	2.3	1.0	2.2

TABLE 3 *Time in seconds for the new code* `nearcor_new`, `nearcor` (*Qi and Sun*) *and alternating projections, with* tol $= 10^{-7}n$

	`nearcor_new`		`nearcor`		Alternating projections	
	Time	Iterations	Time	Iterations	Time	Iterations
cor1399	96.9	5	378.5	5	529.0	62
cor3120	814.0	4	5256.7	4	–	–
Risk-daily	0.39	0	0.47	0	1.02	2
Risk-monthly	0.36	0	0.53	0	1.22	2

TABLE 4 *Time in seconds for the new code* `nearcor_new`, `nearcor` (*Qi and Sun*) *and alternating projections, with* tol $= 2nu$

	`nearcor_new`		`nearcor`		Alternating projections	
	Time	Iterations	Time	Iterations	Time	Iterations
cor1399	171.9	7	537.1	7	4251.0	494
cor3120	1533.8	6	15685.0	9	–	–
Risk-daily	5.73	5	–	–	30.92	55
Risk-monthly	15.94	10	–	–	18.66	27

- The MATLAB implementation of the alternating projections method used in the testing in Higham (2002b).

We use four test matrices, all of which are approximate correlation matrices with unit diagonal.

- cor1399: This is a matrix of dimension 1399 of stock data provided by a fund management company. It is highly rank deficient and its off-diagonal entries are in the interval $[-0.9644, 1.1574]$. (Missing data can result in off-diagonal entries of magnitude greater than 1, depending on how the matrix is constructed.)
- cor3120: This is a matrix from the same source as the first one. It has dimension $n = 3120$ and has full rank. The off-diagonal elements are in the interval $[-0.6250, 1.0751]$.
- Risk-daily, Risk-monthly: These are matrices from the RiskMetrics database (Educational data sets, http://www.riskmetrics.com/stddownload_edu.html). The documentation says that, 'The data sets contain consistently calculated volatilities and correlation forecasts for use in estimating market risk. The asset classes covered are government bonds, money markets, swaps, foreign exchange and equity indices (where applicable) for 31 currencies, and commodities.' We obtained two matrices for a 1-day and a 1-month horizon assigned to 15 July 2006, which have dimension 387 and a smallest eigenvalue of -7.92×10^{-6} and -4.91×10^{-6}, respectively.

First, we investigate the influence of the iterative solver and preconditioning. With $\mathrm{tol} = 10^{-7}n$, we solved the first two problems using `nearcor_new`, and again with `nearcor_new` with the CG method replacing minres, using the NAG CG suite `f11gd/f11ge/f11gf`, and in each case solving both with and without preconditioning. The results are in Table 1, where we report T_{tot}: the total run time (in seconds), T_{mvp}: the time taken to compute all the matrix–vector products $V_k h$ (see (2.5)), T_{eig}: the time to compute the spectral decompositions, Iters: the number of outer (inexact Newton) iterations, # mvp: the number of matrix–vector products required and Time Pre.: the time to compute the preconditioner.

Several comments can be made on Table 1. First, minres leads to a faster algorithm than the CG method both with and without preconditioning and results in one fewer iteration for the cor3120 matrix without preconditioning. The reduction in time is largely accounted for by the fewer matrix–vector products. Second, preconditioning brings a useful speed-up, amounting for cor3120 to a 48% reduction in time with minres and a 58% reduction with CG. Third, the eigenvalue computations take 32% of the total time for cor3120 with unpreconditioned minres, rising to 52% of the time with preconditioning.

We now take a look at the effect of the choice of eigensolver (see Section 3.5). The results in Table 1 are based on the use of NAG code `f08fc` (divide and conquer). Table 2 reports the results for solving the same problems as in Table 1 using `nearcor_new` with this eigensolver, `f08fa` (the QR algorithm—as used by MATLAB for the `eig` function) and `f08fd` (dqds/MRRR), using preconditioning in each case. The results shown are ratios of total times spent to run Algorithm 4.1 with each eigensolver normalized by the run time with the fastest eigensolver `f08fc`. We see that with `f08fc` the algorithm is twice faster than if the other two eigensolvers are used, for these matrices, and, indeed, `f08fc` is also the fastest in similar tests that we have performed with different matrices (Borsdorf, 2007). We found this difference quite surprising, and it shows the importance of trying different algorithmic variants of the basic linear algebra ‘black boxes’. The performance of these codes for tridiagonal matrices was investigated by Demmel *et al.* (2008), who observed that the performance of divide and conquer and dqds/MRRR depends strongly on the particular matrices to which they are applied.

In Tables 3 and 4 we compare `nearcor_new` with the code `nearcor` of Qi and Sun and with the alternating projections code from Higham (2002b), with two different convergence tolerances corresponding to half and full machine precision. The main differences between `nearcor_new` and `nearcor` affecting the run time are the different eigensolvers (`f08fc` for `nearcor_new` and MATLAB’s `eig` for `nearcor`), the different iterative method for computing the Newton direction (minres versus CG), the use of the Jacobi preconditioner in `nearcor_new` and more optimized MATLAB coding in `nearcor_new`, particularly for the gradient evaluations. In the tables ‘–’ denotes that `nearcor` did not converge or that alternating projections was unable to handle the matrix in a reasonable time (we estimate a run time of several days for cor3120). The speed-up of `nearcor_new` over `nearcor` of a factor more than 6 on both problems is significant given that both codes are using the same Newton algorithm.

Finally, we mention that we have carried out more extensive tests with various classes of random matrices of dimension up to 1000. The results, reported in Borsdorf (2007, Chapter 6), are consistent with those shown here. The speed-up of `nearcor_new` over `nearcor` ranges between 1.1 and 3.0 and it shows a generally increasing trend with n.

6. Concluding remarks

Our MATLAB implementation `nearcor_new` of the Newton method of Qi and Sun can solve problems of dimension a few thousand in a few minutes. The run time is dominated by the cost of computing spectral decompositions, and most of the remaining time is spent in computing the matrix–vector products (2.5). Although an extensive computational comparison of all available methods for solving the nearest correlation matrix problem is lacking, the available evidence suggests that the Newton method is the best general purpose method. Toh (2008) reported his code having run times of over 1 h to solve a nearest correlation matrix problem of dimension 1600 (the maximum size reported therein), whereas `nearcor_new` solves a problem of twice the dimension within 30 min. The alternating projections method is very easy to implement and is attractive for small problems and modest convergence tolerances, but, in general, cannot compete with Newton's method for efficiency. Extensions of the Newton method to incorporate constraints and to Hadamard weighting have recently been developed (Qi & Sun, 2007, 2008) and we expect that the ideas herein can be profitably employed in those methods.

Acknowledgements

We thank Jörg Liesen, Marcos Raydan, David Sayers and David Silvester for helpful discussions on this work. We also thank the referees for useful suggestions.

Funding

Royal Society-Wolfson Research Merit Award (to N.J.H.); The Engineering and Physical Sciences Research Council Studentship held in the School of Mathematics, The University of Manchester (to R.B.).

REFERENCES

BORSDORF, R. (2007) A Newton algorithm for the nearest correlation matrix. *M.Sc. Thesis*, The University of Manchester, UK.

BOYD, S. & XIAO, L. (2005) Least-squares covariance matrix adjustments. *SIAM J. Matrix Anal. Appl.*, **27**, 532–546.

DEMMEL, J. W., MARQUES, O. A., PARLETT, B. N. & VÖMEL, C. (2008) Performance and accuracy of LAPACK's symmetric tridiagonal eigensolvers. *SIAM J. Sci. Comput.*, **30**, 1508–1526.

DEUTSCH, F. (2001) *Best Approximation in Inner Product Spaces*. New York: Springer.

ELMAN, H. C., SILVESTER, D. J. & WATHEN, A. J. (2005) *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*. Oxford: Oxford University Press.

FINGER, C. C. (1997) A methodology to stress correlations. *RiskMetrics Monit.*, **Fourth Quarter**, 3–11.

GREENBAUM, A. (1997) *Iterative Methods for Solving Linear Systems*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

HIGHAM, N. J. (1988) Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra Appl.*, **103**, 103–118.

HIGHAM, N. J. (2002a) *Accuracy and Stability of Numerical Algorithms*, 2nd edn. Philadelphia, PA: Society for Industrial and Applied Mathematics.

HIGHAM, N. J. (2002b) Computing the nearest correlation matrix—A problem from finance. *IMA J. Numer. Anal.*, **22**, 329–343.

KNOL, D. K. & TEN BERGE, J. M. F. (1989) Least-squares approximation of an improper correlation matrix by a proper one. *Psychometrika*, **54**, 53–61.

LURIE, P. M. & GOLDBERG, M. S. (1998) An approximate method for sampling correlated random variables from partially-specified distributions. *Manage. Sci.*, **44**, 203–218.

MALICK, J. (2004) A dual approach to solve semidefinite least-squares problems. *SIAM J. Matrix Anal. Appl.*, **26**, 272–284.

MICCHELLI, C. A. & UTRERAS, F. I. (1988) Smoothing and interpolation in a convex subset of a Hilbert space. *SIAM J. Sci. Statist. Comput.*, **9**, 728–746.

NAG Toolbox for MATLAB. Oxford: NAG Ltd. Available at http://www.nag.co.uk/.

PAIGE, C. C. & SAUNDERS, M. A. (1975) Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, **12**, 617–629.

QI, H.-D. & SUN, D. (2006) A quadratically convergent Newton method for computing the nearest correlation matrix. *SIAM J. Matrix Anal. Appl.*, **28**, 360–385.

QI, H.-D. & SUN, D. (2007) Correlation stress testing for value-at-risk: an unconstrained convex optimization approach. Manuscript.

QI, H.-D. & SUN, D. (2008) An augmented Lagrangian dual approach for the H-weighted nearest correlation matrix problem. Manuscript.

QI, H.-D., XIA, Z. & XING, G. (2007) An application of the nearest correlation matrix on web document classification. *J. Indus. Manage. Optim.*, **3**, 701–713.

TOH, K.-C. (2008) An inexact primal-dual path following algorithm for convex quadratic SDP. *Math. Program.*, **112**, 221–254.

TURKAY, S., EPPERLEIN, E. & CHRISTOFIDES, N. (2003) Correlation stress testing for value-at-risk. *J. Risk*, **5**, 75–89.

VAN DER SLUIS, A. (1969) Condition numbers and equilibration of matrices. *Numer. Math.*, **14**, 14–23.