Electronic Transactions on Numerical Analysis.
Volume 12, pp. 216-233, 2001.
Copyright © 2001, Kent State University.
ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

RETOOLING THE METHOD OF BLOCK CONJUGATE GRADIENTS[*]

A. A. DUBRULLE [†]

Abstract. Block implementations of the conjugate-gradients method for the solution of a linear system must deal with linear dependences that may appear in the descent or residual blocks in the course of the iteration. New algorithms presented here avoid rank estimation and deflation through the use of changes of bases and algorithmic reformulations that eliminate rank near defects. The transformations include a robust process of nonunitary orthogonalisation in the metric of a symmetric positive-definite matrix.

Key words. Linear equations, conjugate gradients, orthogonalisation.

AMS subject classifications. 65F10, 65F25.

1. Introduction.

The method of conjugate gradients with preconditioning is a method of choice for the solution of large, sparse systems of linear equations with symmetric positive-definite matrix and vector right-hand side. In spite of its widespread use and the considerable body of work dedicated to various improvements or extensions, efforts towards the development of dependable block generalisations have been comparatively few. One of the main difficulties encountered in block designs concerns rank defects that may appear in the descent or residual matrices. O'Leary [6] proposes to tackle this problem by rank monitoring of the descent matrix with a QR factorisation and by discarding dependent vectors. A variable-block method in [5] uses oblique projectors to reduce block size adaptively. In [2], Broyden gives necessary and sufficent conditions for the absence of breakdown in an analysis based on Krylov sequences.

I take here a different tack by using changes of bases that supplement rank defects and enforce linear independence. One approach uses factorisations of the descent blocks and an alternate form of the iteration for which the ill-conditioned factors disappear. Another consists of a reformulation of the common iteration that accommodates a QR factorisation of the residual blocks and implicitly eliminates the effects of rank deficiencies.

The material is organised as follows. Section 2 provides the necessary background on the common form of the conjugate-gradients algorithm and its block generalisation. The alternate formulation that lends itself to transparent transformations of the descent matrix is described in Section 3. Sections 4 and 5 discuss the use of nonunitary orthogonalisation for a consistent generalisation of the vector algorithm. Section 6 describes another generalisation of the vector algorithm where the common form of the iteration is retooled for the orthogonalisation of the residual matrix. Sample results of experiments are summarised in Section 7. Overall, this work should be considered a preliminary investigation and not a design of bulletproof software.

For simplicity of exposition, preconditioning, which independently applies to the methods discussed here, is omitted.

The notation is that commonly used in numerical linear algebra. Given a matrix G, g_j denotes its jth column, and g_{ij}, its generic element. $I_{(n,m)}$ is the matrix of the leading m columns of the identity matrix of order n, with $m \leq n$.

2. The common block algorithm.

The conjugate-gradients method iteratively builds the solution of the linear system with positive-definite matrix and vector right-hand side $Ax = b$, $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, from projections on a Krylov basis. The k^{th} iterate $x^{(k)}$, $x^{(0)} = 0$,

[*] Received December 1, 2000. Accepted for publication July 23, 2001. Recommended by L. Reichel.

[†] E-mail: `na.dubrulle@na-net.ornl.gov`

minimizes the functional

$$f(y) = \|A^{1/2}(x-y)\|_2, \qquad y \in \operatorname{span}\{A^i b\}_{i=0}^{k-1}$$

The common form of the algorithm (see [4] and [3]) is displayed in Figure 2.1. The sets of

$$\boxed{\begin{array}{c} \text{CG} \\ \hline x^{(0)} = 0, \quad r^{(0)} = b, \quad r^{(-1)} = e_1, \quad v^{(0)} = 0, \\ \left.\begin{aligned} \sigma^{(k-1)} &= \frac{r^{(k-1)T} r^{(k-1)}}{r^{(k-2)T} r^{(k-2)}}, \\ v^{(k)} &= r^{(k-1)} + \sigma^{(k-1)} v^{(k-1)}, \\ \tau^{(k)} &= \frac{r^{(k-1)T} r^{(k-1)}}{v^{(k)T} A v^{(k)}}, \\ x^{(k)} &= x^{(k-1)} + \tau^{(k)} v^{(k)}, \\ r^{(k)} &= r^{(k-1)} - \tau^{(k)} A v^{(k)}, \end{aligned}\right\} \quad k = 1, 2, \ldots . \end{array}}$$

FIG. 2.1. *Conjugate gradients, common form.*

residuals $\{r^{(k)}\}$ and descent vectors $\{v^{(k)}\}$ satisfy the following orthogonality relations:

$$\begin{aligned} v^{(i)T} A v^{(j)} &= 0, & i &\neq j, \\ r^{(i)T} r^{(j)} &= 0, & i &\neq j, \\ v^{(i)T} r^{(j)} &= 0, & i &\leq j. \end{aligned} \tag{2.1}$$

In exact arithmetic, the algorithm terminates for $k \leq n$, but not in finite-precision computations, for which it should be considered purely iterative. Details of convergence can be found in [3].

The generalisation of the vector iteration to a system with full-rank matrix right-hand side,

$$AX = B, \qquad A \in \mathbb{R}^{n \times n}, \qquad B \in \mathbb{R}^{n \times m}, \qquad m \ll n,$$

derives from the minimisation of the following matrix functional by the k^{th} iterate $X^{(k)}$, $X^{(0)} = 0$:

$$f(Y) = \|A^{1/2}(X-Y)\|_F, \qquad Y \in \operatorname{span}\{A^i B\}_{i=0}^{k-1}.$$

It is defined by the equations in Figure 2.2. The descent matrices $V^{(k)}$ and residuals $R^{(k)}$ satisfy the orthogonality relations:

$$\begin{aligned} V^{(i)T} A V^{(j)} &= 0, & i &\neq j, \\ R^{(i)T} R^{(j)} &= 0, & i &\neq j, \\ V^{(i)T} R^{(j)} &= 0, & i &\leq j. \end{aligned} \tag{2.2}$$

The case $m = 1$ coincides with the vector iteration. Orthogonality properties imply that at most $\lfloor n \div m \rfloor$ blocks of independent descent directions and residuals can exist. Consequently,

ETNA
Kent State University
etna@mcs.kent.edu

BCG

$$X^{(0)} = 0, \quad R^{(0)} = B, \quad R^{(-1)} = I_{(n,m)}, \quad V^{(0)} = 0,$$

$$\left.\begin{aligned} S^{(k-1)} &= (R^{(k-2)T} R^{(k-2)})^{-1} R^{(k-1)T} R^{(k-1)}, \\ V^{(k)} &= R^{(k-1)} + V^{(k-1)} S^{(k-1)}, \\ T^{(k)} &= (V^{(k)T} A V^{(k)})^{-1} R^{(k-1)T} R^{(k-1)}, \\ X^{(k)} &= X^{(k-1)} + V^{(k)} T^{(k)}, \\ R^{(k)} &= R^{(k-1)} - A V^{(k)} T^{(k)}, \end{aligned}\right\} \quad k = 1, 2, \ldots .$$

FIG. 2.2. *Block conjugate gradients, common form.*

BCGdQ

$$X^{(0)} = 0, \quad R^{(0)} = B, \quad R^{(-1)} = I_{(n,m)}, \quad P^{(0)} = 0, \quad C^{(0)} = I,$$

$$\left.\begin{aligned} S^{(k-1)} &= C^{(k-1)} (R^{(k-2)T} R^{(k-2)})^{-1} R^{(k-1)T} R^{(k-1)}, \\ P^{(k)} C^{(k)} &= R^{(k-1)} + P^{(k-1)} S^{(k-1)}, \\ T^{(k)} &= (P^{(k)T} A P^{(k)})^{-1} C^{(k)-T} R^{(k-1)T} R^{(k-1)}, \\ X^{(k)} &= X^{(k-1)} + P^{(k)} T^{(k)}, \\ R^{(k)} &= R^{(k-1)} - A P^{(k)} T^{(k)}, \end{aligned}\right\} \quad k = 1, 2, \ldots .$$

FIG. 2.3. *Block conjugate gradients, common form, with QR factorisation of the descent matrix.*

in finite-precision computations, the matrices $S^{(k)}, T^{(k)} \in \mathbb{R}^{m \times m}$ should be expected to approach singularity or nullity for some value of $k > 0$, causing a breakdown of the process. To remedy this, O'Leary [6] suggests to incorporate in the algorithm a QR factorisation of $V^{(k)}$ to monitor column dependences and eliminate redundant descent directions, thereby decreasing m. The factorisation

$$V^{(k)} = P^{(k)} C^{(k)}, \qquad P^{(k)T} P^{(k)} = I, \qquad c_{i\,j}^{(k)} = 0 \quad \forall \quad i > j, \qquad \|(C^{(k)})^{-1}\| < \infty,$$

produces a set $\{P^{(i)}\}$ that obviously possesses the same orthogonality properties as the set $\{V^{(i)}\}$. The substitution of the above expression $V^{(k)}$ in BCG and a little algebra yield algorithm BCGdQ[1] of Figure 2.3. This formulation improves the condition of the system to be solved for $T^{(k)}$. Yet, estimating the rank of $C^{(k)}$ for elimination of linear dependences in $V^{(k)}$ is a delicate operation that one would rather avoid. The next section addresses that problem.

3. Alternate formulation.

Figure 3.1 displays a variant of the vector iteration that derives directly from the basic orthogonality relations (2.1). This alternate formulation appears in the original paper by Hestenes and Stiefel [4]. Its block version BCGA shown in Figure 3.2 has the interesting property that any factorisation

$$V^{(k)} = P^{(k)} C^{(k)}, \qquad C^{(k)} \in \mathbb{R}^{m \times m}, \qquad \|(C^{(k)})^{-1}\| < \infty, \tag{3.1}$$

[1] Note that the matrices $S^{(k)}$ and $T^{(k)}$ are not the same as those of the previous algorithm.

```
CGA
$x^{(0)} = 0,\quad r^{(0)} = b,\quad \sigma^{(0)} = 1,\quad v^{(0)} = 0,$
$\left.\begin{aligned} v^{(k)} &= r^{(k-1)} + \sigma^{(k-1)} v^{(k-1)}, \\ \tau^{(k)} &= \frac{v^{(k)T} r^{(k-1)}}{v^{(k)T} A v^{(k)}}, \\ x^{(k)} &= x^{(k-1)} + \tau^{(k)} v^{(k)}, \\ r^{(k)} &= r^{(k-1)} - \tau^{(k)} A v^{(k)}, \\ \sigma^{(k)} &= -\frac{v^{(k)T} A r^{(k)}}{v^{(k)T} A v^{(k)}}, \end{aligned}\right\} \quad k = 1, 2, \ldots\cdot$
```

FIG. 3.1. *Conjugate gradients, alternate form.*

```
BCGA
$X^{(0)} = 0,\quad R^{(0)} = B,\quad S^{(0)} = I,\quad V^{(0)} = 0,$
$\left.\begin{aligned} V^{(k)} &= R^{(k-1)} + V^{(k-1)} S^{(k-1)}, \\ T^{(k)} &= (V^{(k)T} A V^{(k)})^{-1} V^{(k)T} R^{(k-1)}, \\ X^{(k)} &= X^{(k-1)} + V^{(k)} T^{(k)}, \\ R^{(k)} &= R^{(k-1)} - A V^{(k)} T^{(k)}, \\ S^{(k)} &= -(V^{(k)T} A V^{(k)})^{-1} V^{(k)T} A R^{(k)}, \end{aligned}\right\} \quad k = 1, 2, \ldots\cdot$
```

FIG. 3.2. *Block conjugate-gradients, alternate form.*

produces an algorithm BCGAdF, where $C^{(k)}$ does not appear explicitly (see Figure 3.3)[2]. Hence, if the decomposition (3.1) is such that all or part of some ill-conditioning of $V^{(k)}$ is transferred to $C^{(k)}$, $S^{(k)}$ and $T^{(k)}$ are bound to have better condition numbers than their BCGA homologues. Yet, one must keep in mind that this magic disappearance of a potential ill-conditioning hinges on the assumption that $C^{(k)}$ has an inverse, which in theory is not true after $\lfloor n \div m \rfloor$ iterations. Finite-precision computations, however, are not likely to make $C^{(k)}$ exactly singular, and the formulas of BCGAdF remain mathematically valid. Now, if $C^{(k)}$ is numerically singular, $P^{(k)}$ may contain some spurious descent vectors artificially created by the factorisation scheme. If these vectors reside in a subspace already visited, the corresponding corrections to $X^{(k-1)}$ are small because of the effect of $R^{(k-1)}$ in the computation of $T^{(k)}$. If they do not, they significantly contribute to $X^{(k)}$, a desirable situation, albeit serendipitous. These considerations constitute the basis for an implicit treatment of singularities that altogether circumvents the difficulties associated with rank estimation, and entirely trusts iteration control to the magnitude of the residuals. In the absence of rank defect, the set $\{P^{(i)}\}$ has the same orthogonality properties as the set $\{V^{(i)}\}$.

Two obvious algorithms derive from the BCGAdF template. One, BCGAdQ, uses a QR factorisation so that $P^{(k)}$ has orthonormal columns. The other, BCGAdL, builds $P^{(k)}$ as a

[2] The matrices $S^{(k)}$ and $T^{(k)}$ are different from those of the previous algorithms.

BCGAdF

$$X^{(0)} = 0, \quad R^{(0)} = B, \quad S^{(0)} = I, \quad P^{(0)} = 0,$$

$$\left.\begin{aligned} P^{(k)}C^{(k)} &= R^{(k-1)} + P^{(k-1)}S^{(k-1)}, \\ T^{(k)} &= (P^{(k)T}AP^{(k)})^{-1}P^{(k)T}R^{(k-1)}, \\ X^{(k)} &= X^{(k-1)} + P^{(k)}T^{(k)}, \\ R^{(k)} &= R^{(k-1)} - AP^{(k)}T^{(k)}, \\ S^{(k)} &= -(P^{(k)T}AP^{(k)})^{-1}P^{(k)T}AR^{(k)}, \end{aligned}\right\} \quad k = 1, 2, \ldots \cdot$$

FIG. 3.3. *Block conjugate-gradients, alternate form, with factorisation of the descent matrix.*

```
function [v,av]=NUMGS(v,av)
m=size(v,2);
for k=1:m-1
   t=sqrt(1/(v(:,k)'*av(:,k)));
   v(:,k)=t*v(:,k);
   av(:,k)=t*av(:,k);
   j=k+1:m;
   w=v(:,k)'*av(:,j);
   v(:,j)=v(:,j)-v(:,k)*w;
   av(:,j)=av(:,j)-av(:,k)*w;
end
t=sqrt(1/(v(:,m)'*av(:,m)));
v(:,m)=t*v(:,m);
av(:,m)=t*av(:,m);
return
```

FIG. 4.1. *In-situ modified Gram-Schmidt nonunitary orthogonalisation. The entry parameters are V and AV, and the results, P and AP such that $P^TAP = I$ in the absence of rank defect in V.*

row permutation of a lower-trapezoidal matrix by LU decomposition with partial pivoting (see Section 5 for a bound on the condition number of such a matrix).

The generality of the decomposition (3.1) naturally leads to yet another form of iteration of lesser complexity described in the next section.

4. Nonunitary orthogonalisation.

If the arbitrary factorisation (3.1) were engineered to make $P^{(k)}$ A-orthogonal, obvious simplifications would ensue for the computation of $T^{(k)}$ and $S^{(k)}$ in an algorithm of the BCGAdF type. It turns out that this A-orthogonalisation can be performed without additional reference to A other than for the unavoidable multiplication $AV^{(k)}$.

PROPOSITION 4.1. *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite. Given full-rank matrices $V \in \mathbb{R}^{n\times m}$, $m \le n$, and $W = AV$, the matrices $P, Q \in \mathbb{R}^{n\times m}$ such that*

$$V = PC, \quad c_{ij} = 0 \;\forall\; i > j, \qquad Q^TP = I, \qquad Q = AP, \tag{4.1}$$

can be computed without reference to A.

Proof. The proof is by construction of a generalised modified Gram-Schmidt (MGS) scheme based on the elementary A-projector

$$\Pi_i = I - p_i q_i^T, \qquad q_i = Ap_i, \qquad q_i^T p_i = 1,$$

which transforms an arbitrary vector into a vector A-orthogonal to p_i. Starting with

$$p_1 = \alpha_1 v_1, \qquad q_1 = \alpha_1 w_1,$$

we derive p_2 by projection of v_2 on the A-orthogonal complement of p_1,

$$p_2 = \alpha_2 \Pi_1 v_2, \qquad q_2 = \alpha_2 \Pi_1 w_2, \qquad q_2^T p_2 = 1,$$

where α_1 and α_2 are defined by A-normalisation. Letting

$$P_j = \sum_{i=1}^{j} p_i e_i^T, \qquad Q_j = \sum_{i=1}^{j} q_i e_i^T,$$

we observe that our proposition is verified for matrices in $\mathbb{R}^{n\times 2}$:

$$Q_2^T P_2 = I, \qquad Q_2 = AP_2.$$

We now assume that P_k and Q_k have been computed so that $Q_k^T P_k = I$ and $Q_k = AP_k$ for $k < m$. Because of orthogonality, these matrices have the formal properties

$$I - P_k Q_k^T = \Pi_k \dots \Pi_1, \qquad I - Q_k P_k^T = \Pi_k^T \dots \Pi_1^T,$$

as the elementary projectors commute. In finite precision computations, however, these identities are not generally verified[3]. The computation of P_{k+1} and Q_{k+1} follows from:

$$\left.\begin{aligned} p_{k+1} &= \alpha_{k+1} \Pi_k \dots \Pi_1 v_{k+1}, \\ q_{k+1} &= \alpha_{k+1} \Pi_k^T \dots \Pi_1^T w_{k+1}, \end{aligned}\right\} \quad q_{k+1}^T p_{k+1} = 1. \tag{4.2}$$

These recurrence relations constitute a generalisation of the MGS procedure that builds $P = P_m$ and $Q = Q_m = AP_m$ to verify equations (4.1). C is the upper-triangular Cholesky factor of $V^T AV$. □

An *in situ* MATLAB implementation NUMGS of the above procedure[4] is displayed in Figure 4.1. Substituting $V^{(k)} = P^{(k)} C^{(k)}$ in BCGA with $P^{(k)}$ A-orthogonal, we obtain algorithm BCGAdA[5] displayed in Figure 4.2, where the equation defining $P^{(k)} C^{(k)}$ represents the computation of $P^{(k)}$ and $AP^{(k)}$. Since BCGAdA builds all its descent directions A-orthogonal, it constitutes a more consistent generalisation of the vector iteration[6] than BCGAdQ and BCGAdL. We must expect losses or orthogonality in the presence of nearly-dependent columns in $V^{(k)}$. Assuming that the computation of $AV^{(k)}$ is accurate to working precision ε, a bound for the MGS process in [3] yields

$$\|P^{(k)T} AP^{(k)} - I\|_2 \approx \varepsilon\kappa_2(A^{1/2} V^{(k)}) \le \varepsilon\kappa_2(A^{1/2})\,\kappa_2(V^{(k)}), \tag{4.3}$$

[3] The replacement of $\Pi_k \dots \Pi_1$ with $I - P_k Q_k^T$ in equations (4.2) would produce a classical Gram-Schmidt scheme, which is more unstable.

[4] A generalisation to the case where A is symmetric, nonsingular, but indefinite, is obtained by substituting the requirement $|Q^T P| = I$ for $Q^T P = I$ in Proposition 4.1, and by replacing in the proof the projector Π_i by $\tilde{\Pi}_i = I - \theta_i p_i q_i^T, \quad \theta_i = \mathrm{sgn}(p_i^T q_i)$.

[5] The matrices $S^{(k)}$ and $T^{(k)}$ are different from those of the previous algorithms.

[6] The generalisation is not quite complete, as the columns of a residual block are not orthogonal.

ETNA
Kent State University
etna@mcs.kent.edu

BCGAdA

$$X^{(0)} = 0, \quad R^{(0)} = B, \quad S^{(0)} = I, \quad P^{(0)} = 0,$$

$$\left.\begin{aligned}
&P^{(k)}C^{(k)} = R^{(k-1)} + P^{(k-1)}S^{(k-1)},\\
&T^{(k)} = P^{(k)T}R^{(k-1)},\\
&X^{(k)} = X^{(k-1)} + P^{(k)}T^{(k)},\\
&R^{(k)} = R^{(k-1)} - AP^{(k)}T^{(k)},\\
&S^{(k)} = -P^{(k)T}AR^{(k)},
\end{aligned}\right\} \quad k = 1, 2, \ldots \cdot$$

FIG. 4.2. *Block conjugate-gradients, alternate form, with A-orthogonalisation of the descent matrix.*

where κ_2 is the ℓ_2 condition number. In experiments, algorithm BCGAdA proves rather temperamental, either by working well or by failing in big ways. Typically, if the tolerance η of the stopping criterion

$$\max_j \frac{\|r_j^{(k)}\|_2}{\|b_j\|_2} \leq \eta \tag{4.4}$$

is out of the reach of the initial convergent phase of the iteration, unfettered divergence sets in. This phenomenon is explained by the inability of the nonunitary orthogonalisation process to furnish adequate replacements for dependent descent vectors. We discuss in the next section an approach that corrects this deficiency.

BCGAdFA

$$X^{(0)} = 0, \quad R^{(0)} = B, \quad S^{(0)} = I, \quad P^{(0)} = 0,$$

$$\left.\begin{aligned}
&\Omega^{(k)}\Delta^{(k)} = R^{(k-1)} + P^{(k-1)}S^{(k-1)},\\
&P^{(k)}\Gamma^{(k)} = \Omega^{(k)},\\
&T^{(k)} = P^{(k)T}R^{(k-1)},\\
&X^{(k)} = X^{(k-1)} + P^{(k)}T^{(k)},\\
&R^{(k)} = R^{(k-1)} - AP^{(k)}T^{(k)},\\
&S^{(k)} = -P^{(k)T}AR^{(k)},
\end{aligned}\right\} \quad k = 1, 2, \ldots \cdot$$

FIG. 5.1. *Block conjugate-gradients, alternate form, A-orthogonalisation of the descent matrix with conditioning.*

5. Robust nonunitary orthogonalisation.

An investigation of existing methods of nonunitary orthogonalisation quickly reveals that those based on the computation of $V^{(k)T}AV^{(k)}$ are poor choices. For example, consider the well-known scheme, perhaps the least objectionable in that category, which calls on the symmetric singular-value decomposition

$$\Psi^{(k)}D^{(k)}\Psi^{(k)T} = V^{(k)T}AV^{(k)}, \qquad P^{(k)} = V^{(k)}\Psi^{(k)}D^{(k)-1/2}.$$

In the presence of a numerical rank defect in $V^{(k)}$, the values $\sqrt{d_{ii}^{(k)}}$ close to zero must be expected to incur absolute errors of the order of $0.5\varepsilon\|V^{(k)T}AV^{(k)}\|_2/d_{ii}^{(k)1.5}$ for a working precision ε (see, for example, [7]). Under such conditions, no computation of $P^{(k)}$ can be reliable. Even for monitoring linear dependences in $V^{(k)}$, this approach is not recommended since the smallest $\sqrt{d_{ii}^{(k)}}$ value, which is of crucial importance, can be erroneous by a quantity of the order of the A-condition number of $V^{(k)}$. Clearly, the design of a dependable scheme must be found elsewhere.

That NUMGS works well when $V^{(k)}$ is well-conditioned is consistent with inequality (4.3), and suggests a simple two-stage approach to building a robust algorithm. It consists of first computing a safe basis for a subspace of dimension m containing the range of $V^{(k)}$, and then passing this basis to NUMGS for A-orthogonalisation. The first stage, or conditioning transformation, can be carried out with a factorisation

$$V^{(k)} = \Omega^{(k)}\Delta^{(k)}, \qquad \Omega^{(k)} \in \mathbb{R}^{n\times m}, \tag{5.1}$$

such that $\Omega^{(k)}$ is well-conditioned. The second stage consists of the application of NUMGS to $\{\Omega^{(k)}, A\Omega^{(k)}\}$:

$$\Omega^{(k)} = P^{(k)}\Gamma^{(k)}, \qquad P^{(k)T}AP^{(k)} = I, \qquad \gamma_{i\,j}^{(k)} = 0 \quad \forall \quad i > j.$$

A combination of these equations yields

$$V^{(k)} = P^{(k)}C^{(k)}, \qquad C^{(k)} = \Gamma^{(k)}\Delta^{(k)}.$$

Figure 5.1 describes BCGAdFA, a block conjugate-gradients algorithm with robust A-orthogonalisation where matrices $\Delta^{(k)}$ and $\Gamma^{(k)}$ "vanish" from the computation. Left multiplication of the first equation in BCGAdFA by $P^{(k)T}A$ yields

$$C^{(k)} = P^{(k)T}AR^{(k-1)}.$$

This identity shows that, predictably, $C^{(k)}$ converges to negligibility like $R^{(k-1)}$:

$$\|C^{(k)}\|_2 \leq \|R^{(k-1)}\|_A.$$

By inequality (4.3), the best conditioning transformation (5.1) is a thin Householder QR decomposition for which $\Omega^{(k)}$ has orthogonal columns, and $\Delta^{(k)}$ and $C^{(k)}$ are upper-triangular. This sure-fire approach, which provides replacements for the defective columns of $V^{(k)}$, is costly of computation and may just be overkill. An obvious alternative is an LU decomposition with partial pivoting, for which $\Omega^{(k)}$ is a row permutation of a lower-trapezoidal matrix that we denote by $L^{(k)}$. Since its suitability hinges on a question of condition expressed by inequality (4.3), we develop in the following an upper bound on its ℓ_2 condition number.

$L^{(k)}$ has the highest condition number when it coincides with the matrix $\hat{L}$ resulting from the LU decomposition of an $n \times m$ matrix that produces the largest possible element growth factor in the course of the computation (see [8], p. 212):

$$\hat{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -1 & -1 & -1 & -1 & \cdots & -1 & 1 & 0 \\ -1 & -1 & -1 & -1 & \cdots & -1 & -1 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -1 & -1 & -1 & -1 & \cdots & -1 & -1 & 1 \end{bmatrix}$$

To derive a bound on the condition number of $\hat{L}$ we start with

$$\|\hat{L}\|_2 \leq \|\hat{L}\|_F < \sqrt{mn}, \tag{5.2}$$

which follows from the definition of the matrix.

An upper bound for $\|\hat{L}^{-1}\|_2$ is obtained by building a lower-triangular matrix $\Upsilon \in \mathbb{R}^{m\times m}$ that has the same condition number as $\hat{L}$ and a known inverse. The Frobenius norm of this inverse provides the desired bound, as follows.

We first apply to $\hat{L}$ a Householder transformation that modifies row m and annihilates rows $m+1, ..., n$. Υ is obtained by removing the null rows from the result. These operations preserve the condition of $\hat{L}$ and its Frobenius norm. We have

$$\Upsilon = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -1 & -1 & -1 & -1 & \cdots & -1 & 1 & 0 \\ -\mu & -\mu & -\mu & -\mu & \cdots & -\mu & -\mu & \mu \end{bmatrix}, \qquad \mu = \sqrt{n-m+1}.$$

This matrix has the following inverse, which is the Cholesky factor of $(\hat{L}^T\hat{L})^{-1}$,

$$\Upsilon^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 2 & 1 & 1 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 2^{m-3} & 2^{m-4} & 2^{m-5} & \cdots & 1 & 1 & 0 \\ 2^{m-2} & 2^{m-3} & 2^{m-4} & \cdots & 2 & 1 & \mu^{-1} \end{bmatrix}.$$

Summations of geometric progressions of common ratio 4 easily yield $\|\Upsilon^{-1}\|_F$. A simple upper bound ensues, where, for simplicity, the contribution of the last diagonal element is replaced by unity:

$$\|\Upsilon^{-1}\|_2 < \|\Upsilon^{-1}\|_F < \frac{1}{3}\sqrt{4^m + 6m - 1}.$$

ETNA
Kent State University
etna@mcs.kent.edu

¿From inequality (5.2), it follows that

$$\kappa_2(L^{(k)}) \leq \kappa_2(\hat{L}) < \frac{1}{3}\sqrt{mn(4^m + 6m - 1)}.$$

This bound, which adequately approximates $\kappa_2(\hat{L})$ when n is sizeably larger than m, is not small. For $m = 10$ and $n = 2000$, $\kappa_2(\hat{L})$ is of the order of 10^4. Yet, it grossly exceeds the condition numbers that one should expect in practice. This is not surprising since $\hat{L}$ is an artificial construction to illustrate element growth in Gaussian elimination, and a very unlikely occurrence in normal computations. By contrast, the LU decomposition of a random 2000×10 matrix with elements normally distributed in $\mathcal{N}(0; 1)$ produces a lower-trapezoidal factor with condition number generally less than 10. In any case, a wealth of studies of Gaussian elimination and its long history of widespread use tell us that LU conditioning is a fairly safe bet.

In summary, the preferred robust scheme with LU conditioning consists of two phases:

1. Computation of $L^{(k)}$ by LU factorisation with partial pivoting:
   $V^{(k)} = L^{(k)}U^{(k)}$,
2. Computation of $P^{(k)}$ from $\{L^{(k)}, AL^{(k)}\}$ with NUMGS:
   $L^{(k)} = P^{(k)}\Gamma^{(k)}$.

Again, $C^{(k)} = \Gamma^{(k)}U^{(k)}$ is upper-triangular. In numerical experiments, LU and QR conditionings of the descent blocks yield equally good results, and do not require more work than the approach based on the singular-value decomposition of $V^{(k)T}AV^{(k)}$. Tests show good accuracy measured by the deviation of $P^{(k)T}AP^{(k)}$ from the identity and by the small subdiagonal elements of $C^{(k)}$ computed as

$$C^{(k)} = P^{(k)T}AV^{(k)}.$$

The following example illustrates the effect of LU conditioning, where A is a Wishart matrix, $n = 500$, $\kappa_2(A) = 1.44 \times 10^6$, and $V^{(k)}$ is defined for $m = 10$ by

$$v_{ij}^{(k)} = \left(\frac{i}{n}\right)^j, \qquad \kappa_2(V^{(k)}) = 1.55 \times 10^7.$$

(i) Simple A-orthogonalisation of $V^{(k)}$ (sub($*$) designates the subdiagonal part of a matrix):

$$\|I - P^{(k)T}AP^{(k)}\|_2 = 2.24 \times 10^{-9}, \qquad \|\text{sub}(P^{(k)T}AV^{(k)})\|_2 = 8.23 \times 10^{-7}.$$

(ii) Robust A-orthogonalisation of $V^{(k)}$ with LU conditioning:

$$\|I - P^{(k)T}AP^{(k)}\|_2 = 2.55 \times 10^{-15}, \qquad \|\text{sub}(P^{(k)T}AV^{(k)})\|_2 = 9.38 \times 10^{-13}.$$

The BCGAdFA template defines the variants BCGAdLA and BCGAdQA for LU and QR conditionings.

6. Orthogonalisation of the residual matrix.

In Section 4, we saw how an appropriate change of basis produced an A-orthogonal descent matrix, a property formally consistent with the vector iteration. Orthogonality of the residual vectors is another such property that we propose to enforce with another change of basis.

In algorithm BCG of Section 2, consider the thin QR factorisation of the generic block of residuals

$$R^{(j)} = Q^{(j)}C^{(j)}, \qquad Q^{(j)T}Q^{(j)} = I, \qquad \|(C^{(j)})^{-1}\| < \infty,$$

and the associated transformation of the descent block that defines $P^{(k)}$:

$$V^{(k)} = P^{(k)} C^{(k-1)}.$$

Replacement of these formulas in algorithm BCG and a little algebra yield algorithm BCGrQ[7] of Figure 6.1, where the equation defining the product $Q^{(k)}S^{(k)}$ represents a thin Householder QR decomposition. The identity

$$S^{(k)} = C^{(k)}(C^{(k-1)})^{-1}, \tag{6.1}$$

which follows from this decomposition, provides a stable formula for the computation of $C^{(k)}$. The discussion of a possible singularity of $C^{(k)}$ is the same as that for the factori-

BCGrQ

$$X^{(0)} = 0, \quad Q^{(0)}C^{(0)} = B, \quad S^{(0)} = I, \quad P^{(0)} = 0,$$
$$\left.\begin{aligned} P^{(k)} &= Q^{(k-1)} + P^{(k-1)}S^{(k-1)T}, \\ T^{(k)} &= (P^{(k)T}AP^{(k)})^{-1}, \\ X^{(k)} &= X^{(k-1)} + P^{(k)}T^{(k)}C^{(k-1)}, \\ Q^{(k)}S^{(k)} &= Q^{(k-1)} - AP^{(k)}T^{(k)}, \\ C^{(k)} &= S^{(k)}C^{(k-1)}, \end{aligned}\right\} \quad k = 1, 2, \dots\cdot$$

FIG. 6.1. *Block conjugate-gradients with orthogonalisation of the residual matrix.*

sation of the descent block in Section 3. In this algorithm, $C^{(k-1)}$, which contains all the information on the magnitude of the residuals, appears only for the purpose of scaling the contribution of $P^{(k)}$ to the solution.

¿From the orthogonality relations (2.2), $P^{(i)T}Q^{(j)} = 0$ for $i \leq j$. Since $Q^{(i)T}$ has orthogonal columns, it follows that

$$Q^{(k-1)T}P^{(k)} = I, \qquad P^{(k)T}AP^{(k)} = Q^{(k-1)T}AP^{(k)}.$$

It is possible to derive from BCGAdF an algorithm BCGAdFArQ (Figure 6.2) that combines both robust A-orthogonalisation of the descent matrix and orthogonalisation of the residual matrix. Such a construction, which constitutes a true generalisation of the vector iteration, generates two algorithms, BCGAdQArQ and BCGAdLArQ, for LU and QR conditionings.

7. Summary report of experiments.

Results of sample MATLAB experiments are displayed in the Appendix. For easier reading, I recapitulate how algorithm name relates to implementation.

- *Common form of the method*
    - BCG: the plain algorithm (for reference).
    - BCGdQ: a slight improvement of BCG by orthogonalisation of the descent matrix (for reference).
    - BCGrQ: retooled algorithm based on the orthogonalisation of the residual matrix.

[7] As usual, the matrices $S^{(k)}$ and $T^{(k)}$ are not the same as in previous algorithms

```
BCGAdFArQ
$X^{(0)} = 0, \quad Q^{(0)}C^{(0)} = B, \quad S^{(0)} = I, \quad P^{(0)} = 0,$
    $P^{(k)}F^{(k)} = Q^{(k-1)} + P^{(k-1)}S^{(k-1)},$
    $T^{(k)} = P^{(k)T}Q^{(k-1)},$
    $X^{(k)} = X^{(k-1)} + P^{(k)}T^{(k)}C^{(k-1)},$
    $Q^{(k)}Z^{(k)} = Q^{(k-1)} - AP^{(k)}T^{(k)},$        } $k = 1, 2, \ldots.$
    $C^{(k)} = Z^{(k)}C^{(k-1)},$
    $S^{(k)} = -P^{(k)T}AQ^{(k)},$
```

FIG. 6.2. *Block conjugate-gradients, alternate form, with orthogonalisation of the residual matrix and robust A-orthogonalisation of the descent matrix. The expression $P^{(k)}F^{(k)}$ represents the robust A-orthogonalisation, and $Q^{(k)}Z^{(k)}$, a thin QR decomposition.*

- *Alternate form of the method*
  Unless specified otherwise, transformations apply to the descent blocks.
  - BCGAdQ: QR factorisation.
  - BCGAdL: LU factorisation with partial pivoting.
  - BCGAdQA: Robust A-orthogonalisation with QR conditioning.
  - BCGAdLA: Robust A-orthogonalisation with LU conditioning.
  - BCGAdQArQ: Robust A-orthogonalisation of the descent matrix with QR conditioning and orthogonalisation of the residual matrix.
  - BCGAdLArQ: Robust A-orthogonalisation of the descent matrix with LU conditioning and orthogonalisation of the residual matrix.

To illustrate these methods, I ran five sets of experiments for systems of orders ranging from 200 to 800 with the following types of matrices:

1. Matrices generated by random unitary congruence from positive eigenvalues normally distributed with null mean and unit variance. The condition numbers were set to 10^6.
2. Shifted Wishart matrices.
   The Wishart matrices are obtained from the multiplication of square matrices of entries normally distributed with null mean and unit variance by their transposes [1]. Appropriate shifts of the diagonal produce condition numbers close to 10^5.
3. Symmetric tridiagonal matrices of diagonal $\{2+t, \ldots, 2+t, t\}$ and unit codiagonal. These matrices have an approximate condition number $4.5/t$, where t is set to 10^{-5}.
4. Shifted Wilkinson tridiagonal matrices of even orders.
   Wilkinson's tridiagonal matrix W_n^- [8] has unit codiagonal and diagonal elements $d_i = \lfloor n/2 \rfloor - i + 1$, $1 \leq i \leq n$. For the values of n considered here, condition numbers of the order of 10^5 result by subtracting $n^2/(2n + 1.01)$ from the diagonal.
5. Squares of symmetric tridiagonal matrices with random diagonal elements uniformly distributed in $(-1/8, +1/8)$ and unit codiagonal.

Concerning the operation counts (Mflop) reported in the Appendix, note that the structures of the above tridiagonal and pentadiagonal matrices are not exploited in the matrix multiplications performed by the algorithms.

The right-hand sides vectors have random elements uniformly distributed in $(-1,+1)$ for $m = 10$. The criterion used for stopping the iteration is

$$\max_j \frac{\|r_j^{(k)}\|_2}{\|b_j\|_2} \le \eta,$$

with η set to 10^{-12}. For BCGrQ, $c_j^{(k)}$ replaces $r_j^{(k)}$ in the inequality. In the results displayed in the Appendix, the residual represents

$$\max_j \frac{\|b_j - Ax_j\|_2}{\|b_j\|_2}.$$

A number of iterations that attains the set limit $[n \div 3]$ is flagged by '+++'.

In the experiments reported here as well as in others, BCGrQ consistently shows the least sensitivity to matrix condition and delivers the best overall performance for number of iterations, residual size, and count of operations. The reasons for this superior behavior are not entirely clear. BCGdLArQ and BCGdQArQ generally place second for number of iterations.

8. Conclusion. The main result of this exploratory work is that retooling the block method for appropriate changes of bases is an effective means to generate reliable algorithms free of tricky rank monitoring and repair of singularities. An alternate form of the method allows for transparent transformations of the descent matrix, including one that ensures A-orthogonality. The best algorithm, however, follows from the common method by orthogonalisation of the residual matrix. A true generalisation of the vector iteration that combines A-orthogonalisation of the descent matrix and orthogonalisation of the residual matrix is competitive for count of iterations, but more costly for count of operations.

REFERENCES

[1] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis*, Wiley, New York, 1958.

[2] C. G. Broyden, *A breakdown of the block-CG method*, Optim. Methods Softw., 7 (1996), pp. 41–55.

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1996.

[4] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.

[5] A. A. Nikishin and A. Y. Yeremin, *Variable block CG algorithms for solving large sparse symmetric positive-definite linear systems on parallel computers*, SIAM J. Matrix Anal. Appl., 16 (1995), No. 4, pp. 1135–1153.

[6] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[7] L. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[8] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

Appendix.

Matrices with eigenvalues normally distributed

n	m	$\kappa(A)$	η	Algorithm	Iterations max= $[n/3]$	Residual	Mflop
200	10	10^6	10^{-12}	BCG	+++	5.00×10^{-5}	68
				BCGdQ	+++	1.04×10^{-6}	75
				BCGrQ	35	1.43×10^{-11}	38
				BCGAdQ	42	1.59×10^{-11}	48
				BCGAdL	43	1.37×10^{-11}	47
				BCGAdQA	43	1.75×10^{-11}	51
				BCGAdLA	42	1.59×10^{-11}	47
				BCGAdQArQ	40	1.52×10^{-11}	50
				BCGAdLArQ	41	1.37×10^{-11}	49
400	10	10^6	10^{-12}	BCG	+++	4.40×10^{-4}	483
				BCGdQ	114	1.60×10^{-11}	437
				BCGrQ	58	1.28×10^{-11}	213
				BCGAdQ	73	1.53×10^{-11}	285
				BCGAdL	75	1.44×10^{-11}	282
				BCGAdQA	74	1.51×10^{-11}	295
				BCGAdLA	72	1.50×10^{-11}	276
				BCGAdQArQ	52	1.39×10^{-11}	272
				BCGAdLArQ	53	1.40×10^{-11}	275
600	10	10^6	10^{-12}	BCG	+++	1.59×10^{-6}	1568
				BCGdQ	+++	4.41×10^{-10}	1629
				BCGrQ	76	1.51×10^{-11}	601
				BCGAdQ	88	1.53×10^{-11}	727
				BCGAdL	89	1.48×10^{-11}	716
				BCGAdQA	89	1.51×10^{-11}	745
				BCGAdLA	86	1.59×10^{-11}	701
				BCGAdQArQ	81	1.56×10^{-11}	694
				BCGAdLArQ	83	1.59×10^{-11}	693
800	10	10^6	10^{-12}	BCG	+++	7.26×10^{-7}	3644
				BCGdQ	126	1.79×10^{-11}	1771
				BCGrQ	101	1.85×10^{-11}	1388
				BCGAdQ	110	1.79×10^{-11}	1563
				BCGAdL	109	1.72×10^{-11}	1517
				BCGAdQA	112	1.78×10^{-11}	1609
				BCGAdLA	107	1.74×10^{-11}	1506
				BCGAdQArQ	105	1.79×10^{-11}	1535
				BCGAdLArQ	103	1.81×10^{-11}	1476

Shifted Wishart matrices

n	m	$\kappa(A)$	η	Algorithm	Iterations max= $[n/3]$	Residual	Mflop
200	10	$\approx 10^5$	10^{-12}	BCG	+++	1.56×10^{-10}	68
				BCGdQ	61	1.64×10^{-12}	68
				BCGrQ	37	1.64×10^{-12}	39
				BCGAdQ	50	1.58×10^{-12}	58
				BCGAdL	50	1.59×10^{-12}	54
				BCGAdQA	49	1.69×10^{-12}	58
				BCGAdLA	49	1.64×10^{-12}	55
				BCGAdQArQ	50	1.60×10^{-12}	63
				BCGAdLArQ	51	1.60×10^{-12}	61
400	10	$\approx 10^5$	10^{-12}	BCG	+++	4.39×10^{-10}	483
				BCGdQ	125	2.11×10^{-12}	479
				BCGrQ	75	1.87×10^{-12}	277
				BCGAdQ	111	2.08×10^{-12}	434
				BCGAdL	111	2.09×10^{-12}	417
				BCGAdQA	111	1.94×10^{-12}	442
				BCGAdLA	111	2.14×10^{-12}	426
				BCGAdQArQ	113	2.24×10^{-12}	468
				BCGAdLArQ	113	2.01×10^{-12}	452
600	10	$\approx 10^5$	10^{-12}	BCG	+++	2.67×10^{-10}	1568
				BCGdQ	+++	1.84×10^{-12}	1629
				BCGrQ	106	1.73×10^{-12}	841
				BCGAdQ	177	1.85×10^{-12}	1462
				BCGAdL	177	1.98×10^{-12}	1423
				BCGAdQA	176	1.71×10^{-12}	1473
				BCGAdLA	146	1.75×10^{-12}	1190
				BCGAdQArQ	138	1.87×10^{-12}	1049
				BCGAdLArQ	171	1.93×10^{-12}	1293
800	10	$\approx 10^5$	10^{-12}	BCG	+++	1.04×10^{-9}	3644
				BCGdQ	+++	5.84×10^{-11}	3753
				BCGrQ	158	2.09×10^{-12}	2179
				BCGAdQ	247	2.37×10^{-12}	3510
				BCGAdL	246	2.43×10^{-12}	3424
				BCGAdQA	246	2.50×10^{-12}	3533
				BCGAdLA	247	2.46×10^{-12}	3476
				BCGAdQArQ	219	2.37×10^{-12}	3217
				BCGAdLArQ	177	2.04×10^{-12}	2546

Symmetric tridiagonal matrices with diagonal $\{2+t,\dots,t\}$, $t=10^{-5}$, and unit codiagonal

n	m	$\kappa(A)$	η	Algorithm	Iterations max= $[n/3]$	Residual	Mflop
200	10	$\approx 4.5\times 10^5$	10^{-12}	BCG	+++	7.21×10^{-5}	68
				BCGdQ	+++	1.30×10^{-10}	75
				BCGrQ	25	5.50×10^{-12}	26
				BCGAdQ	40	6.02×10^{-12}	46
				BCGAdL	36	6.44×10^{-12}	39
				BCGAdQA	40	7.92×10^{-12}	48
				BCGAdLA	36	5.80×10^{-12}	40
				BCGAdQArQ	24	2.63×10^{-12}	30
				BCGAdLArQ	24	8.16×10^{-12}	28
400	10	$\approx 4.5\times 10^5$	10^{-12}	BCG	+++	1.85×10^{-7}	482
				BCGdQ	+++	2.15×10^{-7}	510
				BCGrQ	41	4.69×10^{-12}	150
				BCGAdQ	45	8.81×10^{-12}	176
				BCGAdL	40	5.54×10^{-12}	150
				BCGAdQA	44	4.24×10^{-12}	175
				BCGAdLA	40	5.52×10^{-12}	153
				BCGAdQArQ	41	6.50×10^{-12}	167
				BCGAdLArQ	41	4.21×10^{-12}	162
600	10	$\approx 4.5\times 10^5$	10^{-12}	BCG	+++	1.94×10^{-7}	1568
				BCGdQ	62	4.17×10^{-12}	505
				BCGrQ	61	1.85×10^{-12}	481
				BCGAdQ	61	4.67×10^{-12}	504
				BCGAdL	61	7.25×10^{-12}	490
				BCGAdQA	61	3.99×10^{-12}	511
				BCGAdLA	61	1.04×10^{-11}	497
				BCGAdQArQ	57	4.72×10^{-12}	486
				BCGAdLArQ	62	6.96×10^{-12}	516
800	10	$\approx 4.5\times 10^5$	10^{-12}	BCG	+++	6.14×10^{-8}	3644
				BCGdQ	65	2.90×10^{-12}	914
				BCGrQ	58	1.80×10^{-12}	791
				BCGAdQ	68	7.76×10^{-12}	966
				BCGAdL	69	5.48×10^{-12}	960
				BCGAdQA	68	1.71×10^{-12}	977
				BCGAdLA	68	2.48×10^{-12}	957
				BCGAdQArQ	58	2.71×10^{-12}	842
				BCGAdLArQ	58	4.37×10^{-12}	825

Shifted Wilkinson tridiagonal matrices W_n^- of even orders

n	m	$\kappa(A)$	η	Algorithm	Iterations max= $[n/3]$	Residual	Mflop
200	10	1.03×10^5	10^{-12}	BCG	43	6.01×10^{-12}	44
				BCGdQ	45	4.25×10^{-13}	50
				BCGrQ	22	2.74×10^{-13}	23
				BCGAdQ	28	6.64×10^{-13}	32
				BCGAdL	27	9.96×10^{-13}	29
				BCGAdQA	28	7.98×10^{-13}	33
				BCGAdLA	27	8.64×10^{-13}	30
				BCGAdQArQ	22	2.34×10^{-13}	27
				BCGAdLArQ	22	2.62×10^{-13}	26
400	10	2.47×10^5	10^{-12}	BCG	+++	1.94×10^{-7}	482
				BCGdQ	68	5.93×10^{-13}	261
				BCGrQ	42	1.51×10^{-13}	154
				BCGAdQ	46	8.11×10^{-13}	180
				BCGAdL	45	5.87×10^{-13}	169
				BCGAdQA	46	8.41×10^{-13}	183
				BCGAdLA	56	8.07×10^{-13}	215
				BCGAdQArQ	42	5.39×10^{-13}	172
				BCGAdLArQ	42	2.50×10^{-13}	166
600	10	3.96×10^5	10^{-12}	BCG	+++	5.86×10^{-7}	1568
				BCGdQ	116	8.66×10^{-13}	945
				BCGrQ	60	6.43×10^{-13}	473
				BCGAdQ	63	6.31×10^{-13}	520
				BCGAdL	64	7.14×10^{-13}	514
				BCGAdQA	63	6.67×10^{-13}	527
				BCGAdLA	64	4.97×10^{-13}	522
				BCGAdQArQ	60	4.91×10^{-13}	512
				BCGAdLArQ	60	8.04×10^{-13}	499
800	10	5.46×10^5	10^{-12}	BCG	+++	5.16×10^{-8}	3644
				BCGdQ	126	8.85×10^{-13}	1771
				BCGrQ	72	5.80×10^{-13}	986
				BCGAdQ	90	7.75×10^{-12}	1279
				BCGAdL	74	9.18×10^{-12}	1030
				BCGAdQA	77	9.47×10^{-12}	1106
				BCGAdLA	77	6.86×10^{-12}	1083
				BCGAdQArQ	71	9.53×10^{-13}	1033
				BCGAdLArQ	71	8.54×10^{-13}	1013

Squares of symmetric tridiagonal matrices with random diagonal elements uniformly distributed in $(-1/8, +1/8)$ and unit codiagonal

n	m	$\kappa(A)$	η	Algorithm	Iterations max= $[n/3]$	Residual	Mflop
200	10	2.64×10^4	10^{-12}	BCG	+++	1.05×10^{-6}	68
				BCGdQ	+++	1.39×10^{-12}	75
				BCGrQ	24	6.84×10^{-13}	25
				BCGAdQ	+++	1.51×10^{-11}	77
				BCGAdL	+++	3.07×10^{-12}	73
				BCGAdQA	+++	1.39×10^{-12}	80
				BCGAdLA	51	9.25×10^{-13}	57
				BCGAdQArQ	39	9.40×10^{-13}	49
				BCGAdLArQ	40	1.08×10^{-12}	48
400	10	5.14×10^5	10^{-12}	BCG	+++	1.89×10^{-8}	483
				BCGdQ	+++	2.10×10^{-9}	510
				BCGrQ	63	4.34×10^{-12}	232
				BCGAdQ	+++	1.93×10^{-11}	520
				BCGAdL	100	5.83×10^{-12}	376
				BCGAdQA	93	6.25×10^{-12}	370
				BCGAdLA	89	6.36×10^{-12}	341
				BCGAdQArQ	79	5.69×10^{-12}	326
				BCGAdLArQ	74	5.58×10^{-12}	295
600	10	4.24×10^5	10^{-12}	BCG	+++	1.18×10^{-6}	1568
				BCGdQ	+++	2.28×10^{-4}	1629
				BCGrQ	72	4.01×10^{-12}	569
				BCGAdQ	128	7.88×10^{-12}	1057
				BCGAdL	128	7.58×10^{-12}	1029
				BCGAdQA	126	7.99×10^{-12}	1055
				BCGAdLA	127	7.76×10^{-12}	1036
				BCGAdQArQ	80	4.74×10^{-12}	685
				BCGAdLArQ	79	5.00×10^{-12}	659
800	10	2.87×10^5	10^{-12}	BCG	+++	9.98×10^{-8}	3644
				BCGdQ	+++	2.87×10^{-3}	3753
				BCGrQ	98	2.10×10^{-12}	1346
				BCGAdQ	+++	6.25×10^{-12}	3794
				BCGAdL	180	3.96×10^{-12}	2505
				BCGAdQA	179	3.94×10^{-12}	2571
				BCGAdLA	216	4.90×10^{-12}	3040
				BCGAdQArQ	106	2.38×10^{-12}	1550
				BCGAdLArQ	127	3.00×10^{-12}	1823