ZASTOSOWANIA MATEMATYKI
APPLICATIONES MATHEMATICAE
20, 3 (1990), pp. 423–434

J. KUCZYŃSKI (Warszawa)

IMPLEMENTATION OF THE gmr ALGORITHM FOR LARGE SYMMETRIC EIGENPROBLEMS*

Abstract. We present an implementation of the generalized minimal residual (gmr) algorithm for finding an eigenpair of a large symmetric matrix. We report some numerical results for this algorithm and compare them with the results obtained for the Lanczos algorithm. A Fortran implementation of the gmr algorithm can be obtained from the Institute of Computer Science of the Polish Academy of Sciences and is also available via anonymous FTP as "pub/gmrval" on Columbia.edu 128.59.16.1 on the Arpanet. The input of this subroutine is a matrix which has been partially reduced to a tridiagonal form. Such a form can be obtained by the Lanczos process.

1. Introduction. The usual procedure for finding an eigenpair of a large symmetric matrix A is to approximate eigenpairs of A from its behaviour in a given subspace of small dimension. The most popular method of this type is the Lanczos algorithm which gives approximations of eigenvectors in the Krylov subspace. It is known (see [4]) that the Lanczos algorithm does not produce an approximate eigenpair of A with minimal residual. The generalized minimal residual algorithm (the gmr algorithm) was introduced in [3]. It finds the eigenpair with minimal residual in a Krylov subspace. The gmr algorithm enjoys certain theoretical optimality properties. The residuals of the gmr algorithm are never greater than the residuals of the Lanczos algorithm and sometimes they are much smaller. Since the cost of both algorithms is essentially the same, the gmr algorithm seems to be preferable.

In this paper we present an implementation of the gmr algorithm for real symmetric matrices. By applying k steps of the Lanczos process, a symmetric

* This paper has been partly written while the author visited the Department of Computer Science, Columbia University. This research was supported in part by the National Science Foundation under Grant DCR-82-14322.

matrix A is partially reduced to a tridiagonal form, i.e.

$$Q_{k+1}^T A Q_k = D_k,$$

where Q_k is an $(n \times k)$-matrix with orthonormal columns and D_k is a $((k+1) \times k)$-tridiagonal matrix. We assume that coefficients of the matrix D_k have been already computed.

The implementation was tested for many matrices. We report results for matrices with specifically chosen coefficients as well as for random matrices. Numerical tests confirm the theoretical advantages of the gmr algorithm over the Lanczos algorithm. For all matrices the computed residuals of the gmr algorithm are never greater than the corresponding residuals of the Lanczos algorithm and sometimes they are much smaller. The sequences of residuals generated by the gmr algorithm are always nonincreasing, while the sequences produced by the Lanczos algorithm do not enjoy this property. The Lanczos algorithm often increases significantly the residuals from one step to the next one.

For matrices with specifically chosen coefficients, the gmr algorithm is significantly more efficient than the Lanczos algorithm. For random matrices the gmr algorithm is only slightly better than the Lanczos algorithm.

2. The gmr algorithm. In this section we define the gmr algorithm and introduce some of its properties which are useful for implementation.

Let A be an $n \times n$ real symmetric matrix. For a given vector $b \in R^n$, $\|b\| = 1$ ($\|\cdot\| = \|\cdot\|_2$), consider the k-th Krylov subspace

$$K_k = \operatorname{span}(b, Ab, \ldots, A^{k-1}b), \quad k > 0.$$

Let

$$E_k = \{(x, \varrho)\colon x \in K_k, \|x\| = 1, \varrho \in R\}.$$

Define $k+1$ real numbers $c_0^*, c_1^*, \ldots, c_{k-1}^*$ and ϱ^* as

$$\|(A-\varrho^* I)(c_0^* b + c_1^* Ab + \ldots + c_{k-1}^* A^{k-1}b)\| = \min\{\|(A-\varrho I)x\|\colon (x, \varrho) \in E_k\}.$$

The gmr algorithm produces a pair (x_k, ϱ_k) given by

$$x_k = c_0^* b + c_1^* Ab + \ldots + c_{k-1}^* A^{k-1}b, \quad \varrho_k = \varrho^*.$$

In other words, the gmr algorithm finds the normalized vector x_k from the subspace K_k and the real number ϱ_k for which the residual

$$r_k = \min\{\|Ax - \varrho x\|\colon (x, \varrho) \in E_k\} = \|Ax_k - \varrho_k x_k\|$$

is as small as possible.

We now present the properties of the gmr algorithm which are useful for its implementation. Without loss of generality, assume that the vectors b, $Ab, \ldots, A^k b$ are linearly independent. Let $q_1, q_2, \ldots, q_{k+1}$ be an orthonormal

basis, the so-called *Lanczos basis*, of the subspace K_k such that

$$Aq_i = \beta_i q_{i+1} + \alpha_i q_i + \beta_{i-1} q_{i-1}, \quad i = 1, 2, \ldots, k,$$

$$\alpha_i = (Aq_i, q_i), \quad \beta_i = \|Aq_i - \alpha_i q_i - \beta_{i-1} q_{i-1}\|, \quad i = 1, 2, \ldots, k, \quad \beta_0 = 0.$$

Let $Q_i = [q_1, q_2, \ldots, q_i]$ and $e_k = [0, \ldots, 0, 1]^T$. Then the $((k+1)\times k)$-matrix D_k,

$$(1) \qquad D_k = Q_{k+1}^T A Q_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & 0 \\ \beta_1 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \\ 0 & & & & \beta_k \end{bmatrix} = \begin{bmatrix} H_k \\ \beta_k e_k^T \end{bmatrix},$$

is tridiagonal.

For $x \in K_k$, we thus have

$$x = \sum_{i=1}^{k} c_i q_i, \quad c_i \in R.$$

Setting $c_0 = c_{k+1} = c_{k+2} = 0$ we get

$$(2) \quad r_k^2 = \min\Big\{ \sum_{i=1}^{k+1} (c_{i-1}\beta_{i-1} + c_i\alpha_i - c_i\varrho + \beta_i c_{i+1})^2 : \varrho \in R,\ c_i \in R,\ \sum_{i=1}^{k} c_i^2 = 1 \Big\}$$

$$= \min\big\{\min\{\|D_k(\varrho)c\|^2 : \|c\| = 1\} : \varrho \in R\big\}$$

$$= \min\big\{\lambda_{\min}(D_k^T(\varrho) D_k(\varrho)) : \varrho \in R\big\},$$

where $D_k(\varrho) = D_k - \varrho I$, D_k is defined by (1), and $\lambda_{\min}(X)$ denotes the smallest eigenvalue of the matrix X. Hence at the k-step of the gmr algorithm we want to find a number ϱ^* for which the smallest eigenvalue of the matrix $D_k^T(\varrho) D_k(\varrho)$ is minimal. Let $c^* = [c_1^*, \ldots, c_k^*]^T$ be the corresponding eigenvector of $D_k^T(\varrho^*) D_k(\varrho^*)$. Then the vector

$$x^* = \sum_{i=1}^{k} c_i^* q_i$$

is a unit vector from K_k for which the minimum in (2) is attained.

In order to find the smallest eigenvalue of $D_k^T(\varrho) D_k(\varrho)$ we proceed as follows.

Let $H_k(\varrho) = H_k - \varrho I$, where H_k is defined by (1). Then

$$(3) \qquad D_k^T(\varrho) D_k(\varrho) = H_k^2(\varrho) + \beta_k^2 e_k e_k^T.$$

Thus we want to find the smallest eigenvalue of the matrix $H_k^2(\varrho)$ modified by the very special rank one perturbation $\beta_k^2 e_k e_k^T$. We shall use Golub's theorem about the eigenvalues of a matrix which is perturbed by a rank one matrix.

THEOREM ([1]). *Let*

$$G = \operatorname{diag}(g_i), \quad i = 1, 2, \ldots, n,$$

and

$$z = [z_1, \ldots, z_n]^T, \quad \|z\| = 1, \quad \tilde{G} = G + \alpha z z^T.$$

If the g_i are distinct, α is nonzero and all components of the vector z are nonzero, then the eigenvalues of $\tilde{G}$ are the zeros of

$$\chi(t) = 1 + \alpha \sum_{i=1}^{n} z_i^2/(g_i - t).$$

Let $H_k(\varrho) = U_k(\Lambda_k - \varrho I)U_k^T$ be the spectral decomposition of the matrix $H_k(\varrho)$, $\Lambda_k = \operatorname{diag}(\lambda_i)$, where λ_i are eigenvalues of H_k. From (3) we have

$$D_k^T(\varrho) D_k(\varrho) = U_k[(\Lambda_k - \varrho I)^2 + \beta_k^2 U_k^T e_k e_k^T U_k] U_k^T.$$

Let $z = [z_1, \ldots, z_n]^T = U_k^T e_k$. Then z is the last row of the matrix U_k. It is well known (see [4], pp. 129 and 124) that if $\beta_i \neq 0$, $i = 1, \ldots, k-1$, then all elements of the vector z are nonzero and all the λ_i, $i = 1, \ldots, k$, are distinct. Assume also that $\beta_k \neq 0$ and ϱ is chosen is such a way that

$$(\lambda_i - \varrho)^2 \neq (\lambda_j - \varrho)^2 \quad \text{for } i \neq j.$$

Applying Golub's theorem to the matrix $(\Lambda_k - \varrho I)^2$ and to the vector z we see that the eigenvalues of the matrix $D_k^T(\varrho) D_k(\varrho)$ are the zeros of the function χ_ϱ,

$$\chi_\varrho(t) = 1 + \beta_k^2 \sum_{i=1}^{k} z_i^2/[(\lambda_i - \varrho)^2 - t].$$

If we denote by $\zeta(\varrho)$ the smallest zero of the function χ_ϱ, then (2) yields

$$r_k^2 = \min\{\zeta(\varrho): \varrho \in R\}.$$

Thus in order to find the minimal residual it is sufficient to compute the global minimum of the function ζ. The implementation of the gmr algorithm presented in the next section is based on this property.

3. Implementation of the gmr algorithm. The implementation of the gmr algorithm is described as follows.

Having the matrix D_k defined by (1) we compute the global minimum of the function ζ by performing the following steps:

(a) Compute all eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$ of the tridiagonal matrix H_k and the last components $z_1, z_2, \ldots, z_k$ of all its eigenvectors. Order them so that $\lambda_1 < \lambda_2 < \ldots < \lambda_k$.

(b) Define k intervals I_i:

$$I_1 = (-\infty, (\lambda_1+\lambda_2)/2),\ I_2 = ((\lambda_1+\lambda_2)/2, (\lambda_2+\lambda_3)/2), \ldots,$$

$$I_{k-1} = ((\lambda_{k-2}+\lambda_{k-1})/2, (\lambda_{k-1}+\lambda_k)/2),\ I_k = ((\lambda_{k-1}+\lambda_k)/2, +\infty).$$

(c) Calculate the limits of the function ζ at the endpoints of I_i,

$$\lim_{\varrho\to(\lambda_i+\lambda_{i+1})/2} \zeta(\varrho) = (\lambda_i-\lambda_{i+1})^2/4, \quad i = 1, \ldots, k-1.$$

(d) For each interval I_i, find the infimum of the function ζ for $i = 1, \ldots, k$.

(e) Take as the global minimum of ζ the smallest value among numbers obtained in (c) and (d); take ϱ_k as the argument of the global minimum.

We now briefly discuss the steps of the above algorithm.

To perform step (a) we can use technique described in [2]. Since we are interested in eigenvalues and only in the last components of the eigenvectors, we can calculate them in cost proportional to k^2.

Steps (b), (c) and (e) are simple and they do not require explanation. The cost of performing each of them is proportional to k.

Let us now discuss step (d). In order to find the minimum of the function ζ in I_i we propose using the iterative parabola method. It is known that ζ satisfies a Lipschitz condition with constant $4\|A\|$ and is analytic in a neighbourhood of a minimum point. Having computed values $\zeta(\varrho^{(i-2)})$, $\zeta(\varrho^{(i-1)})$, $\zeta(\varrho^{(i)})$, construct an interpolating polynomial w of the second degree (parabola) such that

$$w(\varrho^{(j)}) = \zeta(\varrho^{(j)}) \quad \text{for } j = i-2, i-1, i.$$

Assume that w' is not a constant. Then take $\varrho^{(i+1)}$ as the unique zero of w',

$$w'(\varrho^{(i+1)}) = 0, \quad i = 0, 1, 2, \ldots$$

It is well known that if starting points $\varrho^{(-2)}$, $\varrho^{(-1)}$, $\varrho^{(0)}$ are sufficiently close to the point ϱ_k at which the function ζ attains its minimum and $\zeta''(\varrho_k) \neq 0$, then the sequence $\{\varrho^{(i)}\}$ produced by the parabola method converges with order 1.32 to the point ϱ_k.

Consider now the i-th interval

$$I_i = ((\lambda_{i-1}+\lambda_i)/2,\ (\lambda_i+\lambda_{i+1})/2)$$

and let $\varrho \in I_i$. Then it is easy to see that the smallest zero of the function χ_ϱ lies in the interval J_i. Here

$$J_1 = ((\lambda_1-\varrho)^2, (\lambda_2-\varrho)^2),$$

$$J_i = \big((\lambda_i-\varrho)^2, \min\big((\lambda_{i-1}-\varrho)^2, (\lambda_{i+1}-\varrho)^2\big)\big), \quad i = 2, \ldots, k-1,$$

$$J_k = ((\lambda_k-\varrho)^2, (\lambda_{k-1}-\varrho)^2).$$

Note that the end points of the intervals J_i, $i = 1, 2, \ldots, k$, are the smallest two arguments for which the function χ_ϱ has poles. In order to find the smallest zero of the function χ_ϱ we use bisection to the equation $\chi_\varrho(t) = 0$. One can also use other methods safeguarded with bisection. To find the minimum of the function ζ in the interval I_i we perform a few (up to 6) steps of the parabola iterative method starting from λ_i and two other points chosen close to λ_i. If at any step of the parabola method we obtain the point outside I_i, then we terminate and take as the minimum the smallest computed value of $\zeta(\varrho)$ in the I_i. It is easy to see that the cost of this step is proportional to k^2. Thus the cost of performing all the steps (a)–(e) is of order k^2.

Having values ϱ_k and $\lambda_{\min}(D_k^T(\varrho_k)D_k(\varrho_k))$ we can perform one step of the Wielandt algorithm to get the corresponding eigenvector $c^* = [c_1^*, \ldots, c_k^*]^T$ of $D_k^T(\varrho_k)D_k(\varrho_k)$. Some technical tricks for performing one step of Wielandt's method without computing $D_k^T(\varrho_k)D_k(\varrho_k)$ effectively are given in the Appendix. Using this technique we can calculate the corresponding eigenvector c^* performing $0(k)$ arithmetic operations. The cost of computing the vector

$$x_k = \sum_{i=1}^{k} c_i^* q_i$$

is of order nk operations.

We end this section by the following remark. We have assumed that we were given the coefficients of the matrix D_k and we dealt only with this matrix. If the coefficients $\alpha_1, \ldots, \alpha_k$ and $\beta_1, \ldots, \beta_k$ are not known, they and the orthonormal basis $q_1, q_2, \ldots, q_{k+1}$ can be found using the Lanczos process applied to the Krylov subspace, i.e., to the vectors $b, Ab, \ldots, A^k b$. Formulas for α_i, β_i and q_i given in the previous section are, in general, very sensitive to roundoff errors and some reorthogonalization process is necessary. We will not discuss this subject here. The reader is referred to [4], where the detailed description of the selective reorthogonalization technique can be found. We stress that the cost of constructing the basis $q_1, q_2, \ldots, q_{k+1}$ and coefficients α_i and β_i is proportional to nk, which is much more than k^2 for $k \ll n$.

4. Numerical results and comparison with the Lanczos algorithm. In this section we present some numerical results for the gmr algorithm and compare them with the results obtained for the Lanczos algorithm. This algorithm (see [4], p. 257) also uses the Krylov information

$$N_k(A, b) = [b, Ab, \ldots, A^k b].$$

The Lanczos algorithm, in fact, disregards the last codiagonal element β_k in (1) since β_k is only used to estimate the accuracy of the approximations. It deals with the resulting $(k \times k)$-matrix H_k. The algorithm produces pairs $(Q_k u_i, \lambda_i)$, $i = 1, 2, \ldots, k$, where (u_i, λ_i), $i = 1, 2, \ldots, k$, are all eigenpairs of the matrix H_k, as approximations of eigenpairs of A. The cost of the Lanczos

algorithm is essentially the same as the cost of the gmr algorithm. It is known that the smallest residual r_k^L of the Lanczos algorithm satisfies

$$r_k^L = \min\{\|AQ_k u_i - \lambda_i Q_k u_i\|: \ 1 \leqslant i \leqslant k\}$$

$$= |\beta_k| \min\{|u_{ki}|: \ 1 \leqslant i \leqslant k\} \leqslant |\beta_k|,$$

where u_{ki} is the last, k-th, component of the vector u_i. It is also known that

$$r_k^L = \min\{\sqrt{\|Ax\|^2 - (Ax, x)^2}: \ x \in K_k, \|x\| = 1, (A - (Ax, x)I)x \perp K_k\}.$$

The residual of the k-th step of the gmr algorithm is given by

$$r_k^G = \min\{\sqrt{\|Ax\|^2 - (Ax, x)^2}: \ x \in K_k, \ \|x\| = 1\}.$$

It is easy to see that $r_k^G \leqslant r_k^L$. Moreover, it is known that $r_1^G = r_1^L$ and $r_n^G = r_n^L = 0$. This and similarity of the formulas for residuals might suggest that r_k^L should be close to r_k^G for $k = 1, 2, \ldots, n$. This intuition is incorrect. As shown in [3] the small difference in the formulas leads to completely different values for the residuals of the two algorithms (see Example 1).

For all examples, both the gmr and Lanczos algorithms are tested for $k = 1, 2, \ldots, n$ and their residuals are compared. Without loss of generality we confine ourselves to tridiagonal matrices. For simplicity we set the vector $b = [1, 0, \ldots, 0]^T$. Numerical tests were performed on a DEC-20 computer with 8 decimal accuracy at the Computer Science Department of Columbia University. Some tests were also performed on a DEC-20 computer at the Computer Science Department of the University of Utah in Salt Lake City and on VAX 750 computer at AT & T Bell Laboratory in Murray Hill. We first report the results for the following matrix.

EXAMPLE 1. Let

$$\alpha_i = 0, \quad i = 1, 2, \ldots, 101,$$

$$\beta_i = 0.5, \quad i = 1, 2, \ldots, 100, \ i \neq 1, 11, 21, \ldots, 91,$$

and

$$\beta_1 = \beta_{11} = \beta_{21} = \ldots = \beta_{91} = 0.05.$$

The sequence of residuals of the gmr algorithm is strictly decreasing, while the sequence of residuals of the Lanczos algorithm does not have this property. In fact, only the subsequence $\{r_{2k-1}^L\}$, for $k \geqslant 10$, of the Lanczos residuals is nonincreasing. The gmr residuals r_{2k-1}^G are 2 or 3 times smaller than r_{2k-1}^L. Both algorithms terminate at the 71-st step by reaching residuals smaller than 10^{-8}. For even indices larger than 16, the Lanczos algorithm does not take full advantage of the available information and produced large residuals. For instance,

$$r_{64}^L = 4.2_{10}-4, \quad r_{66}^L = 3.9_{10}-4, \quad r_{68}^L = 5.0_{10}-4, \quad r_{70}^L = 1.2_{10}-3,$$

while

$$r_{63}^L = r_{65}^L = r_{67}^L = r_{69}^L = 4.9_{10}-8.$$

This means that at the 69-th step the Lanczos algorithm guarantees 7 correct decimal digits, while at the next step only 3. The Lanczos algorithm increases the residual more than 24000 times at the 70-th step. By contrast, we stress that the residuals of the gmr algorithm are

$$r_{65}^G = 2.8_{10}-8 \geqslant r_{66}^G \geqslant r_{67}^G \geqslant r_{68}^G \geqslant r_{69}^G \geqslant r_{70}^G = 2.0_{10}-8.$$

EXAMPLE 2. Let $\alpha_i = 0$ and $\beta_i = 1/2$ for $i = 1, 2, \ldots, n$ for $n \geqslant 800$. For this matrix both algorithms produce decreasing sequences of residuals. Table 1 shows how many steps one has to perform using the gmr (G) and Lanczos (L) algorithms to get residuals not greater than ε. The gmr algorithm uses significantly fewer steps.

TABLE 1

ε	$5_{10}-1$	$1_{10}-1$	$5_{10}-2$	$1_{10}-2$	$5_{10}-3$	$1_{10}-3$	$5_{10}-4$	$1_{10}-4$
# L	1	7	12	36	58	170	270	780
# G	1	6	9	21	30	69	98	221

EXAMPLE 3. The increase of the Lanczos residuals observed in Example 1 occurs quite often. For instance, for a tridiagonal matrix of dimension 100 defined as follows:

$$\alpha_1 = \alpha_2 = 1/3, \quad \alpha_3 = \alpha_4 = -1/3, \quad \alpha_5 = \alpha_6 = 1/3, \quad \ldots, \quad \alpha_{99} = \alpha_{100} = 1/3$$

and

$$\beta_i = (-1)^{i+1}(1/3), \quad i = 1, 2, \ldots, 99,$$

the Lanczos algorithm increases the residual error at every fourth step and the increase is very large. For instance,

$$r_{50}^L = 1.8_{10}-3, \quad r_{54}^L = 1.6_{10}-3, \quad r_{58}^L = 1.4_{10}-3, \quad r_{62}^L = 1.3_{10}-3,$$

while all other residuals from the step 49 to 61 vary between $4.5_{10}-7$ and $2.2_{10}-8$.

EXAMPLE 4. One of the goals of testing is to establish empirically how fast residuals of the gmr and Lanczos algorithms converge for symmetric matrices. For the gmr algorithm it has been proved in [3] that for any symmetric matrix A and $k < n$

$$r_k^G \leqslant \|A\|/k,$$

and for any $k < n$ there exists a real symmetric matrix A for which

$$r_k^G \geqslant \|A\|/2k.$$

Similarly, for the Lanczos algorithm the bounds are

$$r_k^L \leqslant \|A\|/\sqrt{k},$$

and for any $k < n$ there exists a symmetric matrix A for which

$$r_k^L \geqslant \|A\|/(\sqrt{k}+1).$$

We want to find out how sharp these bounds are for specific matrices with norm bounded by unity. In order to measure the speed of convergence define the sequences $\{p_k^G\}$, $\{p_k^L\}$ as

$$r_k^G = (k^{p_k^G})^{-1}, \quad r_k^L = (k^{p_k^L})^{-1}, \quad k = 2, 3, \ldots, n-1.$$

From theory we know that $p_k^G \geqslant 1$ and $p_k^L \geqslant 1/2$. We computed p_k^G and p_k^L for many tridiagonal matrices with norm bounded by unity. The smallest values of p_k^G and p_k^L were obtained for matrices with zeros on the main diagonal and with slightly increasing codiagonal elements. We report three examples of such matrices.

(i) For the matrix of dimension 501 with codiagonal elements β_i equal to $i/(2(n-1))$, the gmr residuals decrease at every second step, while the Lanczos residuals do not decrease at all. Both algorithms begin at the same residuals equal to $1_{10}-3$ and at the 500-th step they reach

$$r_{500}^L = 1.1_{10}-2 \quad \text{and} \quad r_{500}^G = 3.6_{10}-4.$$

The sequences p_k^L and p_k^G decrease very slowly for $k \geqslant 50$. For the Lanczos algorithm we obtain

$$p_{50}^L = 0.79, \quad p_{250}^L = 0.74, \quad p_{500}^L = 0.72.$$

For the gmr algorithm we get

$$p_{50}^G = 1.33, \quad p_{250}^G = 1.29, \quad p_{500}^G = 1.27.$$

(ii) We also tested the (201×201)-matrix with codiagonal elements

$$\beta_i = \sqrt{i/(n-1)}/2, \quad i = 1, 2, \ldots, 200.$$

The gmr residuals decrease very slowly at every step. For instance,

$$r_1^G = 3.5_{10}-2, \quad r_{50}^G = 7.7_{10}-3, \quad r_{100}^G = 5.5_{10}-3,$$

$$r_{150}^G = 4.5_{10}-3, \quad r_{200}^G = 3.9_{10}-3.$$

The Lanczos residuals are constant for all 200 steps:

$$r_1^L = r_2^L = \ldots = r_{200}^L = 3.5_{10}-2.$$

The sequences p_k^L and p_k^G are both decreasing. For the Lanczos algorithm we obtain

$$p_{20}^L = 1.12, \quad p_{50}^L = 0.85, \quad p_{100}^L = 0.73,$$
$$p_{150}^L = 0.67, \quad p_{175}^L = 0.65, \quad p_{200}^L = 0.631;$$

while for the gmr algorithm we have

$$p_{20}^G = 1.48, \quad p_{50}^G = 1.24, \quad p_{100}^G = 1.13,$$
$$p_{150}^G = 1.08, \quad p_{175}^G = 1.06, \quad p_{200}^G = 1.046.$$

(iii) The small values of p_k^L and p_k^G are also obtained for the (200×200)-matrix with

$$\beta_i = \log(i+1)/(2\log(n)), \quad i = 1, 2, \ldots, 199,$$

on the codiagonal. A few results for both algorithms are shown in Table 2.

TABLE 2

k	25	50	75	100	125	150	175	180	190	199
r_k^L	$5.1_{10}-2$	$3.9_{10}-2$	$3.3_{10}-2$	$3.0_{10}-2$	$2.7_{10}-2$	$2.5_{10}-2$	$2.3_{10}-2$	$2.3_{10}-2$	$2.3_{10}-2$	$2.2_{10}-2$
r_k^G	$2.1_{10}-2$	$1.3_{10}-2$	$9.5_{10}-3$	$7.4_{10}-3$	$6.2_{10}-3$	$5.3_{10}-3$	$4.6_{10}-3$	$4.5_{10}-3$	$4.3_{10}-3$	$4.1_{10}-3$
p_k^L	0.93	0.83	0.79	0.76	0.75	0.74	0.726	0.725	0.721	0.719
p_k^G	1.21	1.11	1.08	1.06	1.05	1.05	1.042	1.041	1.039	1.038

For large k, the sequence p_k^G is quite close to 1. We believe that for a larger dimension n the sequence p_k^G would be even closer to 1. Observe that for the last two matrices the Lanczos sequence p_k^L is relatively close to 1/2. We believe that there exists a symmetric matrix for which the sequence p_k^L approaches 1/2.

For the same matrix as before, Table 3 shows how many steps are needed to reduce the first residual

$$r_1^G = r_1^L = 6.5_{10}-2$$

by a factor of q using the Lanczos or gmr algorithms.

TABLE 3

q	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# L	78	200	200	200	200	200	200	200	200	200	200	200	200	200	200
# G	10	23	38	51	63	77	89	104	117	130	144	156	172	186	200

EXAMPLE 5. *Random matrices.* We tested many tridiagonal matrices with coefficients generated pseudo-randomly with uniform distribution in the interval $[-1/3, 1/3]$. We do not observe large differences between residuals of both algorithms. However, very often the sequence of Lanczos residuals is not strictly decreasing, though the increase is rather small. In general, the k-th residual r_k^L does not exceed the $(k-1)$-st residual multiplied by 3 or 4. However, for a few matrices, $r_k^L = 20 r_{k-1}^L$ for some k.

Both algorithms were efficient. For random matrices of dimension 201 they computed the residuals smaller than $4_{10}-8$ after about 25 steps. Fast convergence of both algorithms for random matrices can be easily explained. Indeed, the sequences of numbers generated pseudo-randomly from the interval $[-1/3, 1/3]$ are unlikely to be increasing, and almost surely some codiagonal elements are small. These two properties make the residuals of both algorithms small.

Both algorithms were tested for 80 random (201×201)-matrices. For each matrix the gmr residuals are smaller than the corresponding Lanczos residuals. The differences between them are usually insignificant. For each of eighty matrices we compute the number of steps needed to make the residual less than ε. Table 4 presents the average number of steps needed by the Lanczos and gmr algorithms for a few values of ε. These tests suggest that for random matrices the efficiency of both algorithms is nearly the same.

TABLE 4

ε		$1_{10}-1$	$1_{10}-2$	$1_{10}-3$	$1_{10}-4$	$1_{10}-5$	$1_{10}-6$	$1_{10}-7$
Average number of steps	L	2.1	5.94	10.09	15.1	18.23	20.96	24.04
	G	2.06	5.44	9.16	13.88	17.95	20.79	23.6

Appendix. We describe how to perform one step of the Wielandt algorithm in order to find the eigenvector of the matrix $D_k^T(\varrho)D_k(\varrho)$ corresponding to the smallest eigenvalue. Assume that we have a sufficiently good approximation λ, $\lambda \geqslant 0$, of the smallest eigenvalue of the matrix $D_k^T(\varrho)D_k(\varrho)$. We must solve the system of linear equations

$$(D_k^T(\varrho)D_k(\varrho)-\lambda I)u = w \quad \text{for given } w \in R^k,$$

which appears in the Wielandt algorithm.

Assume that the matrices $H_k^2(\varrho)-\lambda I$ and $D_k^T(\varrho)D_k(\varrho)-\lambda I$ are nonsingular. Then from the formula of Sherman, Morrison and Woodbury

$$(A+uv^T)^{-1} = A^{-1}-1/(1+v^TA^{-1}u)A^{-1}uv^TA^{-1},$$

applied to the matrix $A = H_k^2(\varrho)-\lambda I$ and the vectors $u = v = e_k$, we obtain

$$\begin{aligned}(D_k^T(\varrho)D_k(\varrho)-\lambda I)^{-1} &= (H_k^2(\varrho)-\lambda I+\beta_k^2 e_k e_k^T)^{-1}\\ &= [I-1/(1+\omega_k)\beta_k^2(H_k^2(\varrho)-\lambda I)^{-1}e_k e_k^T](H_k^2(\varrho)-\lambda I)^{-1},\end{aligned}$$

where $\omega_k = \beta_k^2 e_k^T(H_k^2(\varrho)-\lambda I)^{-1}e_k$.

Let

$$s = (H_k^2(\varrho)-\lambda I)^{-1}w = (H_k(\varrho)-\sqrt{\lambda}I)^{-1}(H_k(\varrho)+\sqrt{\lambda}I)^{-1}w.$$

Then

$$\beta_k e_k^T(H_k^2(\varrho)-\lambda I)^{-1}w = \beta_k e_k^T s = \beta_k s_k, \quad \text{where } s = (s_1, \ldots, s_k)^T.$$

Thus we have

$$(D_k^T(\varrho)D_k(\varrho)-\lambda I)^{-1}w = s-\beta_k^2 s_k/(1+\omega_k)(H_k^2(\varrho)-\lambda I)^{-1}e_k.$$

Put $t=(t_1,\ldots,t_k)^T=(H_k^2(\varrho)-\lambda I)^{-1}e_k$. It is easy to calculate that $\omega_k=\beta_k^2 t_k$ and

$$u=(D_k^T(\varrho)D_k(\varrho)-\lambda I)^{-1}w = s-\beta_k^2 s_k/(1+\beta_k^2 t_k)t,$$

where

$$s=(H_k(\varrho)-\sqrt{\lambda}I)^{-1}(H_k(\varrho)+\sqrt{\lambda}I)^{-1}w,$$

$$t=(H_k(\varrho)-\sqrt{\lambda}I)^{-1}(H_k(\varrho)+\sqrt{\lambda}I)^{-1}e_k.$$

To solve systems of equations with matrices $H_k(\varrho)+\sqrt{\lambda}I$ and $H_k(\varrho)-\sqrt{\lambda}I$ we can use any numerically stable method (we use Gaussian elimination with partial pivoting) for solving systems of linear equations.

Acknowledgement. I am very pleased to acknowledge Professor Henryk Woźniakowski for his many helpful discussions concerning the subject of this paper. Without his comments and suggestions it would have been impossible to write this paper. I would like to thank Professor Joseph Traub, who carefully read earlier versions of this paper and suggested some improvements. Many thanks are due to Professors Frank Stenger and Kris Sikorski and Dr. David Lee for their cooperation in performing numerical tests.

References

[1] G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Rev. 15 (1973), pp. 318–334.

[2] — and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp. 23 (1969), pp. 221–230.

[3] J. Kuczyński, *On the optimal solution of large eigenpair problems*, J. Complexity 2 (1986), pp. 131–162.

[4] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice Hall, Inc., Englewood Cliffs, N. J., 1980.

JACEK KUCZYŃSKI
INSTITUTE OF COMPUTER SCIENCE
POLISH ACADEMY OF SCIENCES
P.O. BOX 22
00-901 WARSAW

Received on 1987.12.01