IMPLEMENTATION OF THE GMR ALGORITHM FOR LARGE SYMMETRIC EIGENPROBLEMS

CUCS-198-85

Jacek Kuczyński

Department of Computer Science, Columbia University

visiting from

Institute of Computer Science, Polish Acedemy of Sciences

October 1985

This research was supported in part by the National Science Foundation under Grant DCR-82-14322.

Table of Contents

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 is included. The input of this subroutine is a matrix which has been partially reduced to tridiagonal form. Such a form can be obtained by the Lanczos process. The Fortran subroutine is also available via anonymous FTP as "pub/gmrval" on Columbia.edu [128.59.16.1] on the Arpanet.

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 Parlett [80], 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 Kuczyński [85]. It finds the eigenpair with minmimal 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 preferable.

In this paper we present an implementation of the gmr algorithm for real symmetric matrices. Applying k steps of the Lanczos process, a symmetric matrix A is partially reduced to tridiagonal form, i.e., $Q_{k+1}{}^T A Q_k = D_k$, where Q_k is an nxk matrix with orthonormal columns and D_k is (k+1)xk tridiagonal matrix. We assume that coefficients of the matrix D_k have been already computed. We present a Fortran implementation of the gmr algorithm for given coefficients of D_k.

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. Often the Lanczos algorithm significantly

increases the residuals from one step to the next.

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

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

Let

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

Define the (k+1) real numbers $c^*_0,c^*_1,...,c^*_{k-1}$ and ρ^* as

$$||(A - \rho^* I)\ (c^*_0 b+c^*_1 Ab+...+c^*_{k-1}A^{k-1}b|| = \min\{\ ||(A - \rho I)x||\colon (x,\rho) \in E_k\ \}.$$

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

$$x_k = c^*_0 b+c^*_1 Ab+...+c^*_{k-1}A^{k-1}b\ ,\quad \rho_k = \rho^*.$$

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

$$r_k = \min\{\ ||Ax - \rho x||\colon (x,\rho) \in E_k\ \} = ||Ax_k - \rho_k x_k|| \qquad (2.1)$$

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 $b,Ab,...,A^k b$ are linearly independent. Let $q_1,q_2,...q_{k+1}$ be an orthonormal basis, the so called Lanczos basis, of the subspace K_k such that

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

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

Let $Q_i = [q_1,q_2,\ldots,q_i]$ and $e_k = [0,\ldots,01]^T$. Then the $(k+1)\times k$ matrix D_k

$$D_k = Q_{k+1}^T A Q_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & 0 \\ \beta_1 & \alpha_2 & & & \\ & & & & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \\ & 0 & & & \beta_k \end{bmatrix} = \begin{bmatrix} H_k \\ \beta_k\, e_k^T \end{bmatrix} \qquad (2.2)$$

is tridiagonal.

For $x \in K_k$, we thus have $x = \sum_{i=1}^{k} c_i q_i$, $c_i \in \mathbf{R}$. Setting $c_0 = c_{k+1} = c_{k+2} = 0$ we get

$$r^2_k = \min\{ \sum_{i=1}^{k+1} (c_{i-1}\beta_{i-1}+c_i\alpha_i - c_i\rho+\beta_i c_{i+1})^2 \colon \rho \in \mathbf{R},\ c_i \in \mathbf{R},\ \sum_{i=1}^{k} c^2_i = 1\}$$

$$= \min\{\ \min\{\ \|D_k(\rho)c\|^2 \colon \|c\| = 1\ \} : \rho \in \mathbf{R}\ \}$$

$$= \min\{\ \lambda_{\min}(D_k^T(\rho)D_k(\rho)) : \rho \in \mathbf{R}\ \}, \qquad (2.3)$$

where $D_k(\rho) = D_k - \rho I$, D_k is defined by (2.2) and $\lambda_{\min}(X)$ denotes the smallest eigenvalue of the matrix X. Hence at the k-th step of the gmr algorithm we want to find a number ρ^* for which the smallest eigenvalue of the matrix $D_k^T(\rho)D_k(\rho)$ is minimal. Let $c^* = [c^*_1,\ldots,c^*_k]^T$ be the corresponding eigenvector of $D_k^T(\rho^*)D_k(\rho^*)$. Then the vector $x^* = \sum_{i=1}^{k} c^*_i q_i$ is a unit vector from K_k for which the minimum in (2.3) is attained.

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

Let $H_k(\rho) = H_k - \rho I$, where H_k is defined by (2.2). Then

$$D_k^T(\rho)D_k(\rho) = H_k^2(\rho) + \beta_k^2 e_k e_k^T. \tag{2.4}$$

Thus we want to find the smallest eigenvalue of the matrix $H_k^2(\rho)$ 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 (Golub [73])

Let $G = \mathrm{diag}(g_i)$, $i = 1,2,...,n$ and $z = [z_1,...,z_n]^T$, $||z|| = 1$, $G = G+\alpha zz^T$. If the g_i are distinct, α is nonzero and all components of the vector z are nonzero then the eigenvalues of G are the zeros of

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

Let $H_k(\rho) = U_k(\Lambda_k - \rho I)U_k^T$ be the spectral decomposition of the matrix $H_k(\rho)$, $\Lambda_k = \mathrm{diag}(\lambda_i)$, where λ_i are eigenvalues of H_k. From (2.4) we have

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

Let $z = [z_1,...,z_k]^T = U_k^T e_k$. Then z is the last row of the matrix U_k. It is well known, see Parlett [80, p. 129 and 124], that if $\beta_i \neq 0$, $i = 1,...,k-1$, then all elements of the vector z are nonzero and all the λ_i, $i = 1,...,k$, are distinct. Assume also that $\beta_k \neq 0$ and ρ is chosen in such a way that $(\lambda_i - \rho)^2 \neq (\lambda_j - \rho)^2$ for $i \neq j$. Applying Golub's theorem to the matrix $(\Lambda_k - \rho I)^2$ and to the vector z we get that the eigenvalues of the matrix $D_k^T(\rho)D_k(\rho)$ are the zeros of the function χ_ρ,

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

If we denote by $\varsigma(\rho)$ the smallest zero of the function χ_ρ then (2.3) yields

$$r_k^2 = \min \{ \varsigma(\rho) : \rho \in \mathbf{R} \} .$$

Thus in order to find the minimal residual it is sufficient to compute the global minimum of the function ς. 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 matrix D_k defined by (2.2) we compute the global minimum of the function ς by performing the steps:

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

(b) define k intervals I_i: $I_1 = (-\infty, (\lambda_1+\lambda_2)/2)$, $I_2 = ((\lambda_1+\lambda_2)/2, (\lambda_2+\lambda_3)/2),...,$ $I_{k-1} = ((\lambda_{k-2}+\lambda_{k-1})/2, (\lambda_{k-1}+\lambda_k)/2)$, $I_k = ((\lambda_{k-1}+\lambda_k)/2, +\infty)$.

(c) calculate the limits of the function ς at the endpoints of I_i,

$$\lim_{\rho\to(\lambda_i+\lambda_{i+1})/2} \varsigma(\rho) = (\lambda_i - \lambda_{i+1})^2/4 , \qquad i = 1,...,k-1.$$

(d) for each interval I_i, find the infimum of the function ς, $i = 1,...,k$.

(e) take as the global minimum of ς, 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 by Golub and Welsch [69]. 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 the 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 ς in I_i we

propose using the iterative parabola method. It is known that ς satisfies a Lipschitz condition with constant $4||A||$ and is analytic in a neighbourhood of a minimum point. Having computed values $\varsigma(\rho^{(i-2)})$, $\varsigma(\rho^{(i-1)})$, $\varsigma(\rho^{(i)})$, construct an interpolating polynomial w of the second degree (parabola) such that

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

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

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

It is well known that if starting points $\rho^{(-2)}$, $\rho^{(-1)}$, $\rho^{(0)}$ are sufficiently close to the point ρ_k in which function ς attains its minimum and $\varsigma''(\rho_k) \neq 0$ then the sequence $\{\rho^{(i)}\}$ 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 $\rho \in I_i$. Then it is easy to see that the smallest zero of the function χ_ρ lies in the interval J_i . Here $J_1 = ((\lambda_1 - \rho)^2,\ (\lambda_2 - \rho)^2)$, $J_i = ((\lambda_i - \rho)^2,\ \min((\lambda_{i-1} - \rho)^2,\ (\lambda_{i+1} - \rho)^2))$, $i=2,...,k-1$, $J_k = ((\lambda_k - \rho)^2,\ (\lambda_{k-1} - \rho)^2)$. Note that the endpoints of the intervals J_i, $i = 1,2,...,k$, 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 on the equation $\chi_\rho(t) = 0$. One can also use other methods safeguarded with bisection. To find the minimum of the function ς 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 of I_i, then we terminate and take as the minimum the smallest computed value of $\varsigma(\rho)$ 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), (b), (c), (d) and (e) is of order k^2.

Having values ρ_k and $\lambda_{min}(D_k^T(\rho_k)D_k(\rho_k))$ we can perform one step of the Wielandt

algorithm to get the corresponding eigenvector $c^* = [c^*_1,...,c^*_k]^T$ of $D_k^T(\rho_k)D_k(\rho_k)$. Some technical tricks for performing one step of Wielandt's method without computing $D_k^T(\rho_k)D_k(\rho_k)$ effectively are given in Appendix A. Using this technique we can calculate the corresponding eigenvector c^* performing O(k) arithmetic operations. The cost of computing 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 $\alpha_1,...,\alpha_k$ and $\beta_1,...,\beta_k$ are not known, they and the orthonormal basis $q_1,q_2,...,q_{k+1}$ can be found using the Lanczos process applied to the Krylov subspace, i.e., to the vectors $b,Ab,...A^{k+1}b$. 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 the book of Parlett [80], where the detailed description of the selective reothogonalization technique can be found. We stress that the cost of constructing basis $q_1,q_2,...,q_{k+1}$ and coefficients α_i and β_i is proportional to nk, which is much more than k^2 for $k << n$.

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 Parlett [80,p.257], also uses the Krylov information

$$N_k(A,b) = [b,Ab,...,A^kb].$$

The Lanczos algorithm, in fact, disregards the last codiagonal element β_k in (2.2) since β_k is only used to estimate the accuracy of the approximations. It deals with the resulting kxk matrix H_k. The algorithm produces pairs $(Q_k u_i,\lambda_i)$, $i = 1,2,...,k$, where (u_i,λ_i), $i = 1,2,...,k$, 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

$$\begin{aligned} r_k^L &= \min\{ \|AQ_k u_i - \lambda_i Q_k u_i\|: 1 \le i \le k\} \\ &= |\beta_k| \min\{ |u_{ki}|: 1 \le i \le k\} \le |\beta_k|, \end{aligned}$$

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 $r_k^G \le r_k^L$. 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 $k = 1,2,...,n$. This intuition is incorrect. As shown in Kuczyński [85] the small difference in the formulas leads to completely different values for the residuals of the two

algorithms. See Example 4.1.

For all examples, both the gmr and Lanczos algorithm are tested for k = 1,2,....,n and their residuals are compared. Without loss of generality we confine ourselves to tridiagonal matrices. For simplicity we set the vector $b = [1,0,...,0]^T$. 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 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 4.1

Let $\alpha_i = 0$, $i = 1,2,...,101$, $\beta_i = 0.5$, $i = 1,2,...,100$, $i \neq 1,11,21,...,91$ and $\beta_1 = \beta_{11} = \beta_{21} = ... = \beta_{91} = 0.05$. 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 $\{r^L_{2k-1}\}$, for $k \geq 10$, of the Lanczos residuals is nonincreasing. The gmr residuals r^G_{2k-1} are 2 or 3 times smaller than r^L_{2k-1}. Both algorithms terminate at the 71-st step by reaching residuals smaller than $_{10}$-8. For even indices larger than 18, the Lanczos algorithm does not take full advantage of the available information and produced large residuals. For instance $r^L_{64} = 4.2_{10}$-4, $r^L_{66} = 3.9_{10}$-4, $r^L_{68} = 5.0_{10}$-4, $r^L_{70} = 1.2_{10}$-3, while $r^L_{68} = r^L_{65} = r^L_{67} = r^L_{69} = 4.9_{10}$-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 in the 70-th step. By contrast, we stress that the residuals of the gmr algorithm are $r^G_{65} = 2.8_{10}$-8 $\geq r^G_{66} \geq r^G_{67} \geq r^G_{68} \geq r^G_{69} \geq r^G_{70} = 2.0_{10}$-8.

Example 4.2

Let $\alpha_i = 0$ and $\beta_i = 1/2$ for $i = 1,2,...,n$ for $n \geq 800$. For this matrix both algorithms produce decreasing sequences of residuals. Table 4.1 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.

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

Table 4.1

Example 4.3

The increase of the Lanczos residuals observed in Example 4.1 occurs quite often. For instance, for a tridiagonal matrix of dimension 100 defined as follows: $\alpha_1 = \alpha_2 = 1/3$, $\alpha_3 = \alpha_4 = -1/3$, $\alpha_5 = \alpha_6 = 1/3,...,$ $\alpha_{99} = \alpha_{100} = -1/3$ and $\beta_i = (-1)^{i+1}1/3$, $i = 1,2,...,99$, the Lanczos algorithm increases the residual error at every fourth step and the increase is very large. For instance $r^L_{50} = 1.8_{10}$-3, $r^L_{54} = 1.6_{10}$-3, $r^L_{58} = 1.4_{10}$-3, $r^L_{62} = 1.3_{10}$-3, while all other residuals from the step 49 to 61 vary between 4.5_{10}-7 and 2.2_{10}-8.

Example 4.4

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 Kuczyński [85] proves that for any symmetric matrix A and $k < n$

$$r_k^G \leq ||A||/k$$

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

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

Similarly, for the Lanczos algorithm, the bounds are

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

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

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

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

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

From theory we know that $p_k^G \geq 1$ and $p_k^L \geq 1/2$. 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 slighty 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 from the same residuals equal to 1_{10}-3 and at the 500-th step they reach: $r^L_{500} = 1.1_{10}$-2 and $r^G_{500} = 3.6_{10}$-4. The sequences p_k^L and p_k^G decrease very slowly for $k \geq 50$. For the Lanczos algorithm we obtain $p^L_{50} = 0.79$, $p^L_{250} = 0.74$, $p^L_{500} = 0.72$. For the gmr algorithm we get: $p^G_{50} = 1.33$, $p^G_{250} = 1.29$, $p^G_{500} = 1.27$.

(ii) We also tested the 201x201 matrix with codiagonal elements $\beta_i = \sqrt{i/(n-1)}/2$, $i = 1,2,\ldots200$. The gmr residuals decrease very slowly at every step. For instance, $r_1^G = 3.5_{10}$-2,

$r^G_{50} = 7.7_{10}$-3, $r^G_{100} = 5.5_{10}$-3, $r^G_{150} = 4.5_{10}$-3, $r^G_{200} = 3.9_{10}$-3. the Lanczos residuals are constant for all 200 steps, $r_1^L = r_2^L = \ldots = r^L_{200} = 3.5_{10}$-2. The sequences p_k^L and p_k^G are both decreasing. For the Lanczos algorithm we obtain: $p^L_{20} = 1.12$, $p^L_{50} = 0.85$, $p^L_{100} = 0.73$, $p^L_{150} = 0.67$, $p^L_{175} = 0.65$, $p^L_{200} = 0.631$; while for the gmr algorithm $p^G_{20} = 1.48$, $p^G_{50} = 1.24$, $p^G_{100} = 1.13$, $p^G_{150} = 1.08$, $p^G_{175} = 1.06$, $p^G_{200} = 1.046$.

(iii) The small values of p_k^L and p_k^G are also obtained for the 200x200 matrix with $\beta_i = \log(i+1)/(2\log(n))$, i = 1,2,...,199, on the codiagonal. A few results for both algorithms are shown in Table 4.2

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

Table 4.2

For large k, the sequence p_k^G is quite close to 1. We believe that for larger dimension n, the sequence p_k^G would be even closer to one. 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 4.3 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.

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

Table 4.3

Example 4.5 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 $r_k^L = 20\ r^L_{k-1}$, 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 sequence 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 201x201 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 4.4 presents the average number of steps needed by the Lanczos and gmr algorithms for a few values of ϵ.

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

Table 4.4

These tests suggest that for random matrices the efficiency of both algorithms is nearly the same.

5. Appendix A

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

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

which appears in the Wielandt algorithm.

Assume that the matrices $H_k^2(\rho) - \lambda I$ and $D_k^T(\rho)D_k(\rho) - \lambda I$ 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 $A = H_k^2(\rho) - \lambda I$ and the vectors $u = v = e_k$, we obtain

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

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

Let $s = (H_k^2(\rho) - \lambda I)^{-1}w = (H_k(\rho - \lambda I)^{-1}\ (H_k(\rho) + \lambda I)^{-1}w$. Then

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

Thus we have

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

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

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

where

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

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

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

6. Appendix B

```
00100 C
00200 C
00300 C  SUBROUTINE GMRVAL
00400 C
00500 C
00600       SUBROUTINE GMRVAL(K,ALFA,BETA,Z1,VAL,WL,IERR)
00700 C
00800 C     SUBROUTINE GMRVAL IMPLEMENTS THE GENERALIZED MINIMAL RESIDUAL
00900 C  ALGORITHM FOR THE REAL SYMMETRIC EIGENPROBLEM DESCRIBED IN [1]. IT
01000 C  FINDS AN APPROXIMATION OF AN EIGENPAIR OF A REAL SYMMETRIC MATRIX A
01100 C  OF DIMENSION N USING PARTIAL INFORMATION OF A . INFORMATION OF
01200 C  THE MATRIX A IS GIVEN BY THE (K+1)-ST KRYLOV SUBSPACE, I.E.,
01300 C  BY THE I-TH POWER OF A ON THE NONZERO VECTOR B , I=0,1,....,K,
01400 C  K < N , ASSUME THAT WE HAVE THE ORTHONORMAL BASIS
01500 C  Q[1],Q[2],....,Q[K+1] OF THIS SUBSPACE, THE SO CALLED LANCZOS BASIS,
01600 C  WHICH PARTIALLY REDUCES THE MATRIX A TO THE TRIDIAGONAL FORM,
01700 C  I.E., THE (K+1)XK MATRIX
01800 C
01900 C
02000 C     D=(Q[1],Q[2],....,Q[K+1])' A (Q[1],Q[2],....,Q[K])=
02100 C
02200 C                          [ X X              ]
02300 C                          [ X X X    0       ]
02400 C                          [   X X X          ]
02500 C                          [     ...          ]
02600 C                     =    [        ...       ]
02700 C                          [          ...     ]
02800 C                          [ 0      X X X     ]
02900 C                          [          X X     ]
03000 C                          [            X     ]
03100 C
03200 C
03300 C  IS TRIDIAGONAL. HAVING THIS TRIDIAGONAL RECTANGULAR MATRIX THE
03400 C  SUBROUTINE GMRVAL FINDS A NORMALIZED VECTOR Z1 FROM THE K-TH
03500 C  KRYLOV SUBSPACE AND A REAL NUMBER VAL FOR WHICH THE NORM OF THE
03600 C  RESIDUAL ||A Z1 - VAL Z1|| IS AS SMALL AS POSSIBLE. THE USER IS
03700 C  SUPPOSED TO SUPPLY THE DIMENSION, K , AND COEFFICIENTS OF THE
03800 C  TRIDIAGONAL MATRIX D . THE SUBROUTINE GIVES THE NORM OF THE
03900 C  RESIDUAL, AN APPROXIMATION OF AN EIGENVALUE AND COEFFICIENTS OF THE
04000 C  EIGENVECTOR IN THE BASIS Q[1],Q[2],....,Q[K] .
04100 C     THE COEFFICIENTS OF THE TRIDIAGONAL MATRIX D CAN BE COMPUTED
04200 C  BY THE LANCZOS ALGORITHM WITH SELECTIVE REORTHOGONALIZATION APPLIED
04300 C  TO THE VECTORS GENERATING THE KRYLOV INFORMATION (SEE [2]) . THE
04400 C  COST OF THE SUBROUTINE GMRVAL IS PROPORTIONAL TO K*K . THE COST
04500 C  OF PRODUCING ORTHONORMAL BASIS Q[I] AND TRIDIAGONAL MATRIX D IS
04600 C  USUALLY MUCH LARGER, SINCE IT IS PROPORTIONAL TO N*K . THE
04700 C  DETAILED DESCRIPTION OF THE ALGORITHM WHICH FINDS THE VECTOR Z1
04800 C  AND THE NUMBER VAL MAY BE FOUND IN [1].
04900 C
05000 C
05100 C     INPUT PARAMETERS
05200 C
05300 C
05400 C  K      INTEGER VARIABLE WHICH IS THE DIMENSION OF THE KRYLOV
05500 C         SUBSPACE; K MUST BE POSITIVE BUT NOT GREATER THAN 1000 .
05600 C
05700 C  ALFA   ONE DIMENSIONAL REAL ARRAY OF SIZE K WHICH CONTAINS
05800 C         DIAGONAL ELEMENTS OF THE RECTANGULAR (K+1)XK MATRIX D .
05900 C
06000 C  BETA   ONE DIMENSIONAL REAL ARRAY OF SIZE K WHICH CONTAINS
06100 C         CODIAGONAL ELEMENTS OF THE MATRIX D .
06200 C
06300 C     THE EXECUTION OF THE SUBROUTINE DOES NOT CHANGE THE ELEMENTS OF
06400 C  THE MATRIX D .
06500 C
06600 C
06700 C     OUTPUT PARAMETERS
06800 C
06900 C
07000 C  IERR   INTEGER VARIABLE SIGNALING HOW THE CALCULATIONS WERE
07100 C         TERMINATED; IF IERR IS EQUAL TO ZERO THEN THE BEST
07200 C         APPROXIMATION OF AN EIGENPAIR WAS FOUND. OUR SUBROUTINE
07300 C         REQUIRES THE COMPUTATION OF EIGENVALUES AND LAST COMPONENTS
07400 C         OF EIGENVECTORS OF THE KXK TRIDIAGONAL SUBMATRIX. IN ORDER
07500 C         TO COMPUTE THEM WE USE A SLIGHTLY MODIFIED SUBROUTINE
07600 C         IMTQL2 FROM [3] . IF THIS SUBROUTINE DOES NOT GIVE DESIRED
07700 C         EIGENELEMENTS AFTER 30 STEPS OF THE QL ALGORTIHM THEN
07800 C         COMPUTATIONS ARE TERMINATED. THE VARIABLE IERR IS THEN
07900 C         SET TO ONE. IN THIS CASE, NO RESULTS HAVE BEEN COMPUTED.
08000 C
```

```
C  Z1    ONE DIMENSIONAL REAL ARRAY OF SIZE  K ; IF  IERR=0  IT
C       CONTAINS THE COMPONENTS OF THE BEST POSSIBLE APPROXIMATION
C       OF THE NORMALIZED EIGENVECTOR IN THE  K-TH KRYLOV SUBSPACE
C       IN THE BASIS  Q[1],Q[2],....Q[K] .
C
C  VAL    REAL VARIABLE; IF  IERR=0  IT CONTAINS AN APPROXIMATION OF
C       AN EIGENVALUE.
C
C  WL     REAL VARIABLE; IF  IERR=0  IT CONTAINS THE NORM OF THE
C       RESIDUAL  ||A Z1 - VAL Z1|| .
C
C
C    REFERENCES
C
C  [1]  KUCZYNSKI,J., IMPLEMENTATION OF THE GMR ALGORITHM FOR LARGE
C         SYMMETRIC EIGENPROBLEM, DEPT. OF COMPUTER SCIENCE REPORT,
C         COLUMBIA UNIVERSITY, 1985.
C
C  [2]  PARLETT,B.N., THE SYMMETRIC EIGENVALUE PROBLEM, PRENTICE HALL,
C         INC. ENGLEWOOD CLIFFS, 1980.
C
C  [3]  SMITH,B.T., J.M.BOYLE, B.S.GARBOW, Y.IKEBE, V.C.KLEMA,
C         C.B.MOLER, MATRIX EIGENSYSTEM ROUTINES - EISPACK GUIDE,
C         SPRINGER VERLAG, BERLIN, HEIDELBERG, NEW YORK, 1974.
C
C
C
      INTEGER I,IERR,I1,J,K,K1,K2,M1
      REAL A,B,BK,C,TO,TOL,TOL0,VAL,WL,W0,W1,W2,X,XL,X0,X1,X2
      DIMENSION ALFA(K),BETA(K),Z1(K)
      DIMENSION BETA1(1000),U(1000),WW(1000)
      REAL ABS,AMAX1,AMIN1,SQRT
      LOGICAL CHECK
C
      IERR=0
      VAL=ALFA(1)
      Z1(1)=1.0
      WL=ABS(BETA(1))
      IF (K.EQ.1) RETURN
C
C  COMPUTE MACHINE PRECISION
C
      TOL=1.0
   10 IF (1.0+TOL.EQ.1.0) GO TO 20
      TOL=TOL/2.0
      GO TO 10
   20 TOL=2.0*TOL
C
      DO 30 I=2,K
        Z1(I)=0.0
   30 CONTINUE
C
C  CHECK IF THE CODIAGONAL, BETA, CONTAINS A SMALL ELEMENT
C
      M1=K-1
      DO 40 I=1,M1
        J=I
        IF (ABS(BETA(I)).LE.TOL*(ABS(ALFA(I))+ABS(ALFA(I+1)))) GO TO 50
   40 CONTINUE
      J=J+1
   50 K1=J
      IF (K1.EQ.1) RETURN
C
C  COMPUTE ALL EIGENVALUES AND LAST COMPONENTS OF THE EIGENVECTORS OF
C  THE LEADING SUBMATRIX, I.E., COMPUTE  WW(I) AND U(I) FOR I=1,...,K.
C
      M1=K1-1
      DO 60 I=1,M1
        WW(I)=ALFA(I)
        BETA1(I)=BETA(I)
        U(I)=0.0
   60 CONTINUE
      U(K1)=1.0
      WW(K1)=ALFA(K1)
      CALL QRVAL(K1,WW,BETA1,U,IERR,TOL)
C
C  CHECK IF THE MODIFIED EISPACK'S ROUTINE FOUND DESIRED EIGENELEMENTS
C
      IF (IERR.NE.0) RETURN
C
      BK=BETA(K1)**2
      IF (ABS(BETA(K1)).GT.TOL*AMAX1(ABS(WW(1)),ABS(WW(K1)))) GO TO 70
C
```

```
      WL=SQRT(BK)*U(K1)
      XL=WW(K1)
      GO TO 160
C
   70 DO 80 I=1,K1
        U(I)=U(I)**2
   80 CONTINUE
C
C  FIND THE SMALLEST RESIDUAL IN THE KRYLOV SUBSPACE
C
      W1=WW(1)
      W2=WW(2)
      XL=(W1+W2)*.5
      WL=(W1-W2)*(W1-W2)*.25
      IF (K1.EQ.2) GO TO 100
C
      DO 90 I=3,K1
        W1=WW(I-1)
        W2=WW(I)
        X0=(W1+W2)*.5
        W0=(W1-W2)*(W1-W2)*.25
        WL=AMIN1(WL,W0)
        IF (W0.EQ.WL) XL=X0
   90 CONTINUE
C
  100 DO 150 I1=1,K1
C
C  PERFORM A FEW STEPS OF THE INTERPOLATING METHOD TO GET BETTER
C  (SMALLER) RESIDUAL (SEE [1])
C
      I=I1
      X0=WW(I)
      TOL0=TOL*(ABS(X0)+1.0)
      X1=X0+2.0*TOL0
      CALL READY(X1,K1,I,WW,W1,WL,XL,U,BK,TOL)
      X2=X0-2.0*TOL0
      CALL READY(X2,K1,I,WW,W2,WL,XL,U,BK,TOL)
      K2=1
  110 CALL READY(X0,K1,I,WW,W0,WL,XL,U,BK,TOL)
      A=W2/(X0-X2)/(X1-X2)
      B=W1/(X0-X1)/(X2-X1)
      C=W0/(X1-X0)/(X2-X0)
      X=A*(X0+X1)+B*(X0+X2)+C*(X1+X2)
      A=A+B+C
      A=A*2.0
      IF (ABS(A).LE.0.01*TOL) GO TO 150
      X=X/A
      TO=TOL*(ABS(X)+1.0)
      IF (AMIN1(ABS(X-X0),ABS(X-X1)).LT.AMAX1(TO,TOL0).
     *    OR.K2.GT.6) GO TO 150
      IF (I.NE.1) GO TO 120
        CHECK=X.GE.(WW(1)+WW(2))*0.5
      GO TO 140
  120 IF (I.NE.K1) GO TO 130
        CHECK=X.LE.(WW(K1)+WW(K1-1))*0.5
      GO TO 140
  130 CHECK=X.LE.(WW(I)+WW(I-1))*0.5.OR.X.GE.(WW(I)+WW(I+1))*0.5
  140 IF (CHECK) GO TO 150
      K2=K2+1
      X2=X1
      W2=W1
      X1=X0
      W1=W0
      X0=X
      TOL0=TO
      GO TO 110
  150 CONTINUE
C
      VAL=XL
      WL=SQRT(WL)
C
C  CALCULATE AN EIGENVECTOR
C
  160 DO 170 I=1,K1
      Z1(I)=1.0
      U(I)=0.0
      WW(I)=ALFA(I)-XL+WL
      BETA1(I)=BETA(I)
  170 CONTINUE
      U(K1)=1.0
      CALL SOLVE(K1,WW,BETA1,Z1,U,TOL)
      DO 180 I=1,K1
        WW(I)=ALFA(I)-XL-WL
```

```
24700        BETA1(I)=BETA(I)
24800      180 CONTINUE
24900        CALL SOLVE(K1,WW,BETA1,Z1,U,TOL)
25000        B=0.0
25100        BK=BK*Z1(K1)/(1.0+BK*U(K1))
25200        DO 190 I=1,K1
25300          Z1(I)=Z1(I)-BK*U(I)
25400          B=B+Z1(I)**2
25500      190 CONTINUE
25600  C
25700  C  NORMALIZE THE COMPUTED EIGENVECTOR Z1
25800  C
25900        B=SQRT(B)
26000        DO 200 I=1,K1
26100          Z1(I)=Z1(I)/B
26200      200 CONTINUE
26300        RETURN
26400        END
26500  C
26600  C
26700  C
26800  C  AUXILIARY SUBROUTINES
26900  C
27000  C
27100  C
27200        SUBROUTINE READY(X,K,I,WW,W,WL,XL,U,BK,TOL)
27300  C
27400  C  SUBROUTINE  READY  FINDS THE MINIMAL NORM OF THE RESIDUAL
27500  C  ‖ A X - RO X ‖  FOR FIXED  RO .
27600  C
27700        INTEGER I,I1,J,K
27800        REAL A,A1,A2,B,BK,C,F,TOL,T1,T2,W,WL,X,XL
27900        DIMENSION U(1000),WW(1000)
28000        REAL ABS,AMIN1,SQRT
28100  C
28200        T1=8.0*TOL
28300        T2=TOL*TOL*0.5
28400        A1=(WW(I)-X)**2
28500        IF (I.NE.1) GO TO 10
28600          A2=(WW(2)-X)**2
28700          GO TO 30
28800      10 IF (I.NE.K) GO TO 20
28900          A2=(WW(K-1)-X)**2
29000          GO TO 30
29100      20 A2=AMIN1((WW(I-1)-X)**2,(WW(I+1)-X)**2)
29200  C
29300      30 IF ((A2-A1).LE.T1*A2.OR.A2.LE.T2) GO TO 60
29400        A=(A1+A2)*0.5
29500        F=0.0
29600        B=SQRT(A)
29700        DO 40 I1=1,K
29800          J=I1
29900          C=WW(J)-X
30000          C=(C-B)*(C+B)
30100          IF (ABS(C).GT.T2) GO TO 35
30200          IF (I.EQ.J) GO TO 50
30300          GO TO 45
30400      35    F=F+U(J)/C
30500      40 CONTINUE
30600        F=1.0+BK*F
30700  C
30800        IF (F.GT.0.0) GO TO 50
30900      45 A1=A
31000        GO TO 30
31100  C
31200      50 A2=A
31300        GO TO 30
31400  C
31500      60 W=A2
31600        WL=AMIN1(WL,W)
31700        IF (WL.EQ.W) XL=X
31800        RETURN
31900        END
32000  C
32100  C
32200  C
32300        SUBROUTINE QRVAL(N,D,E,Z,IERR,TOL)
32400  C
32500  C  SUBROUTINE  QRVAL  FINDS ALL EIGENVALUES AND LAST COMPONENTS OF ALL
32600  C  EIGENVECTORS OF A TRIDIAGONAL MATRIX. THIS IS A MODIFICATION OF
32700  C  SUBROUTINE  IMTQL  FROM [3] .
32800  C
32900        INTEGER I,IERR,II,J,K,L,M,MML,N
```

```
      REAL D(N),E(N),Z(N)
      REAL B,C,F,G,P,R,S,T,TOL
      REAL ABS,SQRT
C
      IERR=0
      IF (N.EQ.1) GO TO 1001
C
      E(N)=0.0
      DO 240 L=1,N
        J=0
  105   DO 110 M=L,N
          IF (M.EQ.N) GO TO 120
          IF (ABS(E(M)).LE.TOL*(ABS(D(M))+ABS(D(M+1)))) GO TO 120
  110   CONTINUE
  120   P=D(L)
        IF (M.EQ.L) GO TO 240
        IF (J.EQ.30) GO TO 1000
        J=J+1
C
        G=(D(L+1)-P)/(2.0*E(L))
        R=SQRT(G*G+1.0)
        T=1.0
        IF (G.LT.0.0) T=-1.0
        G=D(M)-P+E(L)/(G+T*R)
        S=1.0
        C=1.0
        P=0.0
        MML=M-L
C
        DO 200 II=1,MML
          I=M-II
          F=S*E(I)
          B=C*E(I)
          IF (ABS(F).LT.ABS(G)) GO TO 150
          C=G/F
          R=SQRT(C*C+1.0)
          E(I+1)=F*R
          S=1.0/R
          C=C*S
          GO TO 160
  150     S=F/G
          R=SQRT(S*S+1.0)
          E(I+1)=G*R
          C=1.0/R
          S=S*C
  160     G=D(I+1)-P
          R=(D(I)-G)*S+2.0*C*B
          P=S*R
          D(I+1)=G+P
          G=C*R-B
C
          F=Z(I+1)
          Z(I+1)=S*Z(I)+C*F
          Z(I)=C*Z(I)-S*F
  200   CONTINUE
C
        D(L)=D(L)-P
        E(L)=G
        E(M)=0.0
        GO TO 105
  240 CONTINUE
C
      DO 300 II=2,N
        I=II-1
        K=I
        P=D(I)
        DO 260 J=II,N
          IF (D(J).GE.P) GO TO 260
          P=D(J)
          K=J
  260   CONTINUE
C
        IF (K.EQ.I) GO TO 300
        D(K)=D(I)
        D(I)=P
C
        P=Z(I)
        Z(I)=Z(K)
        Z(K)=P
C
  300 CONTINUE
C
      GO TO 1001
```

```
41300 C
41400  1000 IERR=1
41500  1001 RETURN
41600       END
41700 C
41800 C
41900       SUBROUTINE SOLVE(K,ALFA,BETA,Z1,Z2,TOL)
42000 C
42100 C  SUBROUTINE  SOLVE  SOLVES A SYSTEM OF LINEAR EQUATIONS WITH A
42200 C  TRIDIAGONAL MATRIX FOR TWO RIGHT-HAND VECTORS.
42300 C
42400       INTEGER I,I1,J,K,K1,M
42500       REAL TOL,TOL1,X,X1,X2,Y
42600       DIMENSION ALFA(K),BETA(K),GAM(1000),Z1(K),Z2(K)
42700       REAL SQRT
42800 C
42900       TOL1=TOL*TOL*5.0
43000       X1=BETA(1)
43100       IF (K.EQ.2) GO TO 50
43200       X2=BETA(2)
43300       J=K-2
43400       DO 10 I=1,J
43500         GAM(I)=0.0
43600    10 CONTINUE
43700 C
43800       DO 40 I=1,J
43900         I1=I+1
44000         Y=ALFA(I)
44100         IF (ABS(X1).LE.ABS(Y)) GO TO 20
44200 C
44300         X=Y
44400         Y=X1
44500         ALFA(I)=Y
44600         X1=X
44700         X=BETA(I)
44800         BETA(I)=ALFA(I1)
44900         ALFA(I1)=X
45000         X=GAM(I)
45100         GAM(I)=BETA(I1)
45200         BETA(I1)=X
45300         X=Z1(I)
45400         Z1(I)=Z1(I1)
45500         Z1(I1)=X
45600         X=Z2(I)
45700         Z2(I)=Z2(I1)
45800         Z2(I1)=X
45900 C
46000    20   IF (Y.EQ.0.0) GO TO 30
46100         X=X1/Y
46200         ALFA(I1)=ALFA(I1)-X*BETA(I)
46300         BETA(I1)=BETA(I1)-X*GAM(I)
46400         Z1(I1)=Z1(I1)-X*Z1(I)
46500         Z2(I1)=Z2(I1)-X*Z2(I)
46600    30   X1=X2
46700         IF (I.NE.J) X2=BETA(I+2)
46800    40 CONTINUE
46900    50 K1=K-1
47000       Y=ALFA(K1)
47100       IF (ABS(X1).LE.ABS(Y)) GO TO 60
47200       X=Y
47300       Y=X1
47400       ALFA(K1)=Y
47500       X1=X
47600       X=BETA(K1)
47700       BETA(K1)=ALFA(K)
47800       ALFA(K)=X
47900       X=Z1(K1)
48000       Z1(K1)=Z1(K)
48100       Z1(K)=X
48200       X=Z2(K1)
48300       Z2(K1)=Z2(K)
48400       Z2(K)=X
48500    60 IF (Y.EQ.0.0) GO TO 70
48600       X=X1/Y
48700       ALFA(K)=ALFA(K)-X*BETA(K1)
48800       Z1(K)=Z1(K)-X*Z1(K1)
48900       Z2(K)=Z2(K)-X*Z2(K1)
49000 C
49100    70 Y=ALFA(K)
49200       IF (Y.EQ.0.0) Y=TOL1
49300       Z1(K)=Z1(K)/Y
49400       Z2(K)=Z2(K)/Y
49500       Y=ALFA(K1)
```

```
      IF (Y.EQ.0.0) Y=TOL1
      Z1(K1)=(Z1(K1)-BETA(K1)*Z1(K))/Y
      Z2(K1)=(Z2(K1)-BETA(K1)*Z2(K))/Y
      M=K-2
      IF (M.EQ.0) RETURN
      DO 80 I=1,M
        J=M+1-I
        Y=ALFA(J)
        IF (Y.EQ.0.0) Y=TOL1
        Z1(J)=(Z1(J)-BETA(J)*Z1(J+1)-GAM(J)*Z1(J+2))/Y
        Z2(J)=(Z2(J)-BETA(J)*Z2(J+1)-GAM(J)*Z2(J+2))/Y
   80 CONTINUE
      RETURN
      END
```

7. 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.

8. References

Golub,G.H. [73]: Some Modified Matrix Eigenvalue Problems. SIAM Reviev 15, (1973) pp 318-334.

Golub,G.H., Welsch,J.H. [69]: Calculation of Gauss Quadrature Rules. Math. Comp. **23**, (1969) pp 221-230.

Kuczyński,J. [85]: On the Optimal Solution of Large Eigenpair Problems. To appear in Journal of Complexity, **2**.

Parlett,B.N. [80]: **The Symmetric Eigenvalue Problem**. Prentice Hall, Inc., Englewood Cliffs, N.J. 1980.