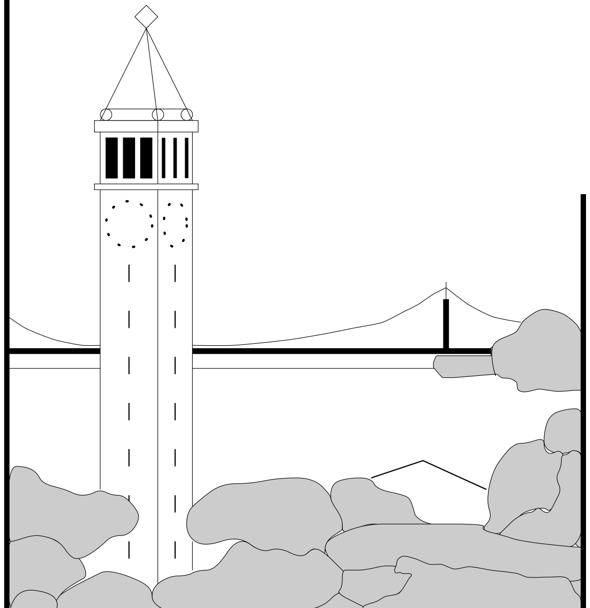
Rational Krylov, a practical algorithm for large sparse nonsymmetric matrix pencils

Axel Ruhe



Report No. UCB/CSD-95-871

April 1995

Computer Science Division (EECS)
University of California
Berkeley, California 94720

RATIONAL KRYLOV, A PRACTICAL ALGORITHM FOR LARGE SPARSE NONSYMMETRIC MATRIX PENCILS

AXEL RUHE

1

Abstract. The Rational Krylov algorithm computes eigenvalues and eigenvectors of a regular not necessarily symmetric matrix pencil. It is a generalization of the shifted and inverted Arnoldi algorithm, where several factorizations with different shifts are used in one run. It computes an orthogonal basis and a small Hessenberg pencil. The eigensolution of the Hessenberg pencil approximates the solution of the original pencil. Different types of Ritz values and harmonic Ritz values are described and compared. Periodical purging of uninteresting directions reduces the size of the basis, and makes it possible to get many linearly independent eigenvectors and principal vectors to pencils with multiple eigenvalues. Relations to iterative methods are established.

Results are reported for two large test examples. One is a symmetric pencil coming from a finite element approximation of a membrane, the other a nonsymmetric matrix modeling an idealized aircraft stability problem.

Key words. matrix, eigenvalue, Krylov, Lanczos, Arnoldi, sparse, nonsymmetric

AMS subject classifications. 65F15

1. Introduction and review.

Let us assume that you have an eigenvalue problem,

$$(A - \lambda B)x = 0 \ , \tag{1}$$

and that you want to compute a few eigenvectors x_i associated with eigenvalues λ_i in a region of the complex plane that may be close to some critical vibration frequences, interesting energy levels or a boundary of stability. The matrices are too large to be treated by standard similarity transformations, as can be found in the LAPACK library [1], but we have a routine available for direct solution of linear systems with shifted coefficient matrices $(A - \mu B)$, most often a sparse Gaussian LU-factorization.

Traditionally you would have two choices, either you apply inverse iteration with Rayleigh quotient shift, this converges fast, but needs several factorizations for each eigenvalue, or you factorize just once and run Lanczos or Arnoldi on the shifted and inverted problem, see [5, 7, 9]. The Rational Krylov algorithm [15, 16, 17, 18] is an attempt to combine the virtues of these two approaches, it iterates with *several* shifts μ_j to build up *one* orthogonal basis from which approximations to *several* eigenvalues may be computed.

You might remember that Krylov space methods, like Lanczos and Arnoldi, can be interpreted in terms of polynomials of the matrix operating on the starting vector, and that the rich theory of orthogonal polynomials can be used to explain their convergence behaviour, see the exposition in [12]. In the Rational Krylov algorithm, we will use rational functions,

$$r(\lambda) = \frac{p_j(\lambda)}{(\lambda - \mu_1)(\lambda - \mu_2)\ldots(\lambda - \mu_j)} = c_0 + \frac{c_1}{\lambda - \mu_1} + \ldots + \frac{c_j}{\lambda - \mu_j} \ . \tag{2}$$

The poles, μ_i, are the shifts, and the zeros of the numerator approximate the eigenvalues. A careful choice of shifts gives a much faster convergence to eigenvalues close

[1] Department of Computing Science, Chalmers Institute of Technology and the University of Göteborg, S-41296 Göteborg, Sweden (`ruhe@cs.chalmers.se`) Work done while visiting University of California Berkeley.

to the poles. The first form describes a sequential variant, where information gained up to step j determines the next shift μ_{j+1}, this will be the primary object of this exposition, but note that the second form indicates how a parallel variant, where all the shifts $\mu_1 \dots \mu_j$ are applied at the same time, can be implemented. Rational functions are also natural in model reduction problems in Control theory, see the recent report by Gallivan, Grimme and Van Dooren [6].

After formulating the algorithm in §2.1 and deriving its basic recursion in §2.2, we will study different ways to get approximate eigenvalues and -vectors in §2.3 and §2.4. It appears that the theory of harmonic Ritz values, expounded by Paige, Parlett and Van der Vorst [11], gives a good framework for understanding this. It also gives us a good way to choose a continuation vector when a new shift μ_j has been calculated. Then, in §2.5, we show that one variant of RKS, where a new shift is used in every step, gives precisely the same subspaces as one variant of the Jacobi Davidson algorithm, recently proposed by Van der Vorst *et al* [21], where the iterative linear system solver in the inner iteration is run to convergence for every outer iteration. In §2.6, we show how the size of the computed basis can be reduced by implicit restart, as described by Sorensen [22] for the Arnoldi algorithm, we will use a technique closely related to the locking and purging strategy studied by Lehoucq and Sorensen [10].

In §3, we discuss implementation details of a more heuristic nature, such as the questions of when a refactorization or a purge of vectors may be profitable.

In §4, we report on numerical tests. We use a symmetric pencil from a finite element approximation of a membrane eigenvalue problem to illustrate how double eigenvalues are handled, and show how very few iterations may give remarkably accurate eigenvector approximations. We will also compute long sequences of eigenvalues of the notorious TOLOSA matrix [3].

The tests bear out some of the expectations raised by the theory. The use of several shifts μ_j leads to fewer iteration steps j, but the added factorizations take time. The Rational Krylov approach comes to advantage, compared to shifted and inverted Arnoldi, when many eigenvalues are requested, as well as when the eigenvalue problem is nonsymmetric and ill conditioned. A new shift, closer to the eigenvalue sought, then leads to much faster convergence.

Purging of vectors to decrease the basis size increases the number of steps, but the orthogonalization gets faster when the basis is smaller. In symmetric and well behaved cases, the convergence rate remains linear, and it clearly pays off to do purging. In ill conditioned cases, the larger basis size appeared to be needed, it looked like the algorithm had to build up a sufficiently large subspace, in order not to loose track of an evolving eigenvector approximation. For very large matrices, the extra cost of keeping a large basis is small, compared to the matrix vector operations, a rule of thumb may be to keep about the same number of vectors in the basis as there are nonzero elements in each row of the sparse factored matrix.

The reader is urged to get the online companion report [19], all pictures are much better in color!

A word about notation: We let V_j stand for a matrix with j columns, the first j columns of V if nothing else is stated, A_{jk} is a $j \times k$ matrix, but we avoid subscripts when all rows or columns are referred to. Column j of the matrix V is v_j and row k is, with a slight notational overkill, $\underline{v}_k^T$. A superscript, like $\lambda^{(j)}$, is used to distinguish the value at step j of a quantity that changes from step to step. We denote by A^H the conjugate transpose of the matrix A. We will always use the Euclidean vector norm, denoted by $\|\ \|$.

2. The Rational Krylov iteration.

2.1. Algorithm. Starting with a vector v_1, we build up an orthonormal basis V_j, one column vector v_j at a time, using the following:

Algorithm RKS

```
1. Start:
    1. Choose shift $\mu_1$.
    2. Choose vector $v_1$, where $\|v_1\| = 1$ (random normal).
    3. Set $t_1 = e_1$ .
2. For $j = 1, 2, \ldots$ until convergence
    1. Set $r = V_j t_j$, (continuation vector).
    2. Operate $r := (A - \mu_j B)^{-1} B r$
    3. Orthogonalize $r := r - V_j h_j$ where $h_j = V_j^H r$, (Gram Schmidt).
    4. Get new vector $v_{j+1} = r / h_{j+1,j}$, where $h_{j+1,j} = \|r\|$, (normalize).
    5. Compute approximate solutions, $\lambda_i^{(j)}$ and $s_i^{(j)}$, and test for convergence.
    6. If $j$ large enough, purge and retain $j = j_1$ vectors.
    7. If motivated, get new $\mu_{j+1}$ newshift.
    8. Get $t_{j+1}$, (continuation combination).
3. Compute eigenvectors, $x_i = V_j s_i$.
```

End.

Let us follow this algorithm prove some important relations and discuss some open issues.

The choice of new shifts during the run to build up one basis V_j, is what makes Rational Krylov different from the shifted and inverted Arnoldi algorithm [2, 20, 14]. We use all information gathered up to step j, to determine the shift for step $j + 1$ in step 2.7 , but for economic reasons we keep the shift μ_j constant for several steps, since then we can use the same factorized matrix

$$A - \mu_j B = LU \tag{3}$$

in all of those. See the discussion in §3!

The use of the continuation combination $V_j t_j$ in step 2.1 is another change from Arnoldi, where one always takes the last vector v_j. It makes a difference when a new shift is taken in step j, we will explain how to choose the vector t_j in step 2.8 in §2.4.

The basic recursion (5), which shows how the original matrix pencil (A, B), (1), the computed basis V_j, and Hessenberg pencil (K, H) hang together, will be derived in §2.2.

The computation of eigenvalue and -vector approximations from the (K, H) pencil in step 2.5 will be explained in §2.3, and in §2.4 some interesting choices for approximations are compared.

The purge operation in step 2.6 is described in §2.6.

The choice of starting vector in step 1.2 is not very critical in algorithms that use shifted and inverted matrices. We used a vector of normally distributed random numbers.

We always made sure we had orthogonality to full accuracy in step 2.3. Classical Gram Schmidt with one reorthogonalization was sufficient for that purpose.

2.2. Basic recursion. Now let us follow what happens during a typical step j of ALGORITHM RKS. Eliminate the intermediate vector r, used in steps 2.1 – 2.4, and get

$$V_{j+1}h_j = (A - \mu_j B)^{-1} B V_j t_j \ ,$$

the vector h_j now has length $j+1$. Multiply from the left by $(A - \mu_j B)$,

$$(A - \mu_j B) V_{j+1} h_j = B V_j t_j \ . \tag{4}$$

Separate terms with A to the left and B to the right,

$$A V_{j+1} h_j = B V_{j+1} (h_j \mu_j + t_j) \ ,$$

with a zero added to the bottom of the vector t_j to give it length $j+1$.

This is the relation for the j th step, now put the corresponding vectors from the previous steps in front of this and get,

$$A V_{j+1} H_{j+1,j} = B V_{j+1} K_{j+1,j} \ , \tag{5}$$

with two $(j+1) \times j$ Hessenberg matrices, $H_{j+1,j} = [h_1 h_2 \ldots h_j]$ containing the Gram Schmidt orthogonalization coefficients, and

$$K_{j+1,j} = H_{j+1,j} \mathrm{diag}(\mu_i) + T_{j+1,j} \tag{6}$$

with the triangular matrix $T_{j+1,j} = [t_1 t_2 \ldots t_j]$, built up from the continuation combinations used in step 2.1.

2.3. How to get approximate eigensolutions. We have considerable freedom to choose how to get approximate eigenvalues and eigenvectors in step 2.5 of ALGORITHM RKS. We may take any j dimensional subspace of V_{j+1} and use it as a basis for a Ritz procedure. Let a basis of this subspace be $V_{j+1} Q_{j+1,j}$, with $Q_{j+1,j}$ the leading j columns of a $(j+1) \times (j+1)$ unitary matrix Q_{j+1}. Look at the basic recursion (5), and get

$$A V_{j+1} Q_{j+1} Q_{j+1}^H H_{j+1,j} = B V_{j+1} Q_{j+1} Q_{j+1}^H K_{j+1,j} \ .$$

The Ritz values will now be found by computing eigenvalues of the leading $j \times j$ block of the transformed (K, H) pencil $(Q_{j+1}^H K_{j+1,j}, Q_{j+1}^H H_{j+1,j})$.

To be more specific, let us take Q from the QR factorization of *any* combination of H and K, say $K - \sigma H$. Note that we can subtract a multiple of BVH from the basic recursion (5), and get

$$(A - \sigma B) V H = B V (K - \sigma H) \ ,$$

and then

$$(A - \sigma B) V Q_{j+1} Q_{j+1}^H H = B V Q_{j+1} R_{j+1,j} \ .$$

The last row of $R_{j+1,j}$ is zero, so we can multiply with the inverse of $R_{j,j}$ from the right and get,

$$(A - \sigma B) W_{j+1} \tilde{H}_{j+1,j} = B W_j \ , \tag{7}$$

where the new basis W is given by,

$$W_{j+1} = V_{j+1} Q_{j+1} \ ,$$

and the new matrix,

$$\tilde{H}_{j+1,j} = Q_{j+1}^H H_{j+1,j} R_{j,j}^{-1} \ ,$$

represents the pencil (A, B) in this new basis. Note that this is the same recursion as one gets from shifted and inverted Arnoldi with shift σ,

$$(A - \sigma B)^{-1} B W_j = W_{j+1} \tilde{H}_{j+1,j} \ , \tag{8}$$

with the, very important, difference that $\tilde{H}$ is not a Hessenberg matrix and W is not a Krylov sequence. It is not even built up one column at a time, since there is a new Q matrix each step.

Let us, all the same, take the eigensolution of the square matrix $\tilde{H}_{j,j}$,

$$\tilde{H}_{j,j} s = s\theta \ , \tag{9}$$

precisely as in Arnoldi, to get the approximate eigenvalues

$$\lambda_i^{(j)} = \sigma + \frac{1}{\theta_i}, \ i = 1, \dots, j \ ,$$

and vectors

$$y_i^{(j)} = W_j s_i = V_{j+1} Q_{j+1,j} s_i \ , \ i = 1, \dots, j \ . \tag{10}$$

We see that,

$$\tilde{H}_{j,j} = W_j^H (A - \sigma B)^{-1} B W_j \ ,$$

so the approximations θ_i are Ritz values of the shifted and inverted pencil $(A - \sigma B)^{-1} B$ over the subspace spanned by W_j, which is contained in the space spanned by the computed basis V_{j+1}.

The transformed residual of the vector $y_i^{(j)}$ (10) is,

$$\begin{aligned}
(A - \sigma B)^{-1} B y_i^{(j)} - y_i^{(j)} \theta_i &= (A - \sigma B)^{-1} B W_j s_i - W_j s_i \theta_i \\
&= W_{j+1} \left(\tilde{H}_{j+1,j} - \begin{bmatrix} \tilde{H}_{j,j} \\ 0 \end{bmatrix} \right) s_i && (11) \\
&= w_{j+1} \underline{\tilde{h}}_{j+1}^T s_i \\
&= w_{j+1} \omega_i^{(j)} \ , && (12)
\end{aligned}$$

where the second equality (11) follows from the transformed basic recursion (8) and of the small eigenvalue problem (9). Note that the residuals for *all* the approximate eigenvalues are in the same direction w_{j+1}, which is orthogonal to W_j, the basis of the subspace from which the eigenvector approximations are computed. Our approximations $\lambda^{(j)}$ can be interpreted as Harmonic Ritz values of $B^{-1}(A - \sigma B)$, to use the terminology introduced in [11].

The norms of the transformed residuals are $|\omega_i^{(j)}|$ (12), and can be computed directly from the small, j dimensional, eigenproblem (9). We can use the same technique as in Lanczos or Arnoldi, to check when eigenvalues are about to converge, and do the time consuming computation of the long eigenvector approximations, $y_i^{(j)}$ (10), just once, when it has converged to sufficient accuracy.

2.4. Choice of approximations. Let us now consider some interesting choices of the factor σ:

2.4.1. Current shift. First take $\sigma = \mu_j$, the current shift. We realize that the last row of $K - \mu H$ is zero, see the expression (6), and remember that H, the matrix of orthogonalization coefficients, is by its construction upper Hessenberg and T, the matrix of starting combinations, must be upper triangular. This means that we may take the unitary matrix Q_{j+1} as the unit matrix, and get the eigenvalue approximations from the upper square part of the (K, H) pencil,

$$K_{j,j}s = \alpha H_{j,j}s \ , \tag{13}$$

now giving the eigenvalue approximations $\lambda_i^{(j)} = \alpha_i$. We let these be the standard eigenvalue approximations, and use them as a default. Remember that they are harmonic Ritz values of $B^{-1}(A - \mu B)$. For symmetric pencils, they will move towards the shift μ_j as long as the shift is kept constant, see [11]. We have observed a similar behaviour in the general case.

2.4.2. Infinity. The second choice is $\sigma = \infty$, which corresponds to making a QR factorization of $H_{j+1,j} = Q_{j+1}R_{j+1,j}$. In an obvious fashion we get,

$$AV_{j+1}Q_{j+1}R_{j+1,j} = BV_{j+1}K_{j+1,j} \ ,$$

$$AW_j = BW_{j+1}\tilde{K}_{j+1,j} \ , \tag{14}$$

with $W = VQ$ and $\tilde{K} = Q^H K R_{j,j}^{-1}$. The eigenvalues γ_i of $\tilde{K}_{j,j}s = s\gamma$, are the standard Ritz values of $B^{-1}A$, with Ritz vectors,

$$u_i = W_j s_i \ . \tag{15}$$

This property made this choice the favorite of the earlier contribution [16], but this study and other recent works like [21] have indicated that harmonic Ritz values are better, especially when it comes to choosing new shifts in the continuation of the algorithm. At convergence the difference disappears.

2.4.3. Next shift. The third interesting choice of σ is μ_{j+1}, the shift we are going to use in the *next* step. In that step, we intend to multiply with the matrix $(A - \mu_{j+1}B)^{-1}B$, to add a new direction to the basis spanned by V_{j+1} or equivalently W_{j+1}. Inserting $\sigma = \mu_{j+1}$ in the relation (8), we get $W_{j+1}\tilde{H}_{j+1,j} = (A - \mu_{j+1}B)^{-1}BW_j$ so any vector in the span of W_j will not add to the span of W_{j+1}. So let us choose the remaining, last, vector in the modified basis,

$$w_{j+1} = V_{j+1}q_{j+1} \ ,$$

as the vector to operate on in this next step.

Thus let step 2.8 in Algorithm RKS read,

$$t_{j+1} = \begin{cases} e_{j+1} & \text{if}\, \mu_{j+1} = \mu_j \ , \\ q_{j+1} = Q_{j+1}e_{j+1} & \text{otherwise} \end{cases} \ , \tag{16}$$

where, in the second case, Q_{j+1} is obtained from the QR factorization,

$$(K_{j+1,j} - \mu_{j+1}H_{j+1,j}) = Q_{j+1}R_{j+1,j} \ .$$

It might be interesting to note that the last element of this vector is zero, if the shift μ_{j+1} is chosen as one of the eigenvalues α_i of the small pencil (13). This is due to the fact that only the last element is nonzero in the last row of $K_{j+1,j} - \alpha_i H_{j+1,j}$, so the first $j-1$ Householder reflections in the QR factorization do not touch this last row. After these reflections, $P_{j-1}P_{j-2}\dots P_1(K_{j,j} - \alpha_i H_{j,j})$ is upper triangular, with a zero in the bottom right corner, provided that α_i is an eigenvalue of the pencil $(K_{j,j}, H_{j,j})$. The last reflection is then simply a transposition of the two last coordinates, and Q_{j+1} gets its bottom right element zero.

It is also evident from algebraic considerations, that we cannot simply continue with v_{j+1} when we have chosen a new shift as a Ritz value from the span of V_j, since then the next rational function (2) will have a common factor in the numerator and denominator. See the earlier study [17]!

2.5. The Jacobi Davidson algorithm. We are now ready to show that one specific choice of approximations and shifts in our ALGORITHM RKS gives the same sequence of subspaces as one specific variant of the algorithm described by Van der Vorst & Co [21]. In ALGORITHM RKS, choose the second alternative in §2.4, standard Ritz values γ_i of the original pencil, for eigenvalue approximations in every step j, and use this as shift the next step $j+1$. In the Jacobi Davidson algorithm [21], choose the standard Ritz variant and run the iterative linear system solver to completion in each outer iteration j.

The proof is by induction, the reader may need to have a description of Jacobi Davidson like [21] available, to follow the rest of this subsection.

First, assume that both algorithms are started with the same vector $v_1 = w_1$. Then look at step j of both algorithms and let the basis W_j be given.

The algorithm in [21] takes a Ritz vector u from the span of W_j. It then adds the direction t to the basis, where t is computed from the linear system,

$$(I - uu^H)(A - \gamma I)(I - uu^H)t = -(A - \gamma I)u \ , \tag{17}$$

(In [21], only the case $B = I$ is treated, but the generalization is obvious). The new direction t is chosen orthogonal to u, $(I - uu^H)t = t$, so the last factor in the left hand side has no effect. Expand the first projection and get $(A - \gamma I)t - uu^H(A - \gamma I)t = -(A - \gamma I)u$, or

$$(A - \gamma I)(t + u) = u\zeta \ , \quad \zeta = u^H(A - \gamma I)t \ ,$$

so the direction of $(A - \gamma I)^{-1}u$ is added to the space W_j in the next iteration, provided that the system (17) is solved to full accuracy.

In ALGORITHM RKS, use the second choice of approximate eigenvalues, γ_i of the previous subsection, and assume that $B = I$ and $H_{j+1,j}$ is factorized, so that (14) reads,

$$\begin{aligned} AW_j &= W_{j+1}\tilde{K}_{j+1,j}, \\ (A - \gamma I)W_j &= W_{j+1}(\tilde{K}_{j+1,j} - \gamma I), \\ W_j &= (A - \gamma I)^{-1}W_{j+1}Q_{j+1}R_{j+1,j} \ , \end{aligned}$$

where the last QR factorization is the one used to determine continuation vector (16) when γ is the next shift. The next step of ALGORITHM RKS will add $r = (A - \gamma I)^{-1}W_{j+1}q_{j+1}$ to the basis. This makes sure that any vector in $(A - \gamma I)^{-1}W_j$, and consequently also $(A - \gamma I)^{-1}u$ for the Ritz vector u (15), is contained in the next subspace.

2.6. Purging and restart. When the size of the basis V_j grows too large for comfort, we divide it into two parts, where the first one contains all converged eigenvectors, together with some that are about to converge to eigenvalues in the interesting region of the complex plane. We then continue the iteration, keeping only the first part in the basis. This can be done so that the basic recursion relation (5) holds all the time in the following way.

We first transform (5) into a standard problem by multiplying with $H_{j,j}^{-1}$ from the right,

$$AV_{j+1}\left[\begin{array}{c} I \\ \underline{h}_{j+1}^{T}H_{j,j}^{-1}\end{array}\right] = BV_{j+1}\left[\begin{array}{c} K_{j,j}H_{j,j}^{-1} \\ \underline{k}_{j+1}^{T}H_{j,j}^{-1}\end{array}\right].$$

We then compute the Schur triangular form $T_{j,j}$ of this standard eigenproblem $K_{j,j}H_{j,j}^{-1}$, and order it so that the j_1 interesting eigenvalues are on top,

$$U^H K_{j,j}H_{j,j}^{-1}U = \left[\begin{array}{cc} T_{j_1,j_1} & T_{j_1,j-j_1} \\ 0 & T_{j-j_1,j-j_1}\end{array}\right]. \tag{18}$$

There are several ways of reordering T. We may run exact shift QR transformations, first shifting with those eigenvalues that are candidates for purging and placing them in the bottom, but we prefer to do it from the top down, using RQ transformations, and first get the converged eigenvalues as leading diagonal elements. We run this RQ iteration until no change is observed in the leading diagonal element, to make sure that the block $T_{j-j_1,j}$ really is zero to working accuracy.

Some care has to be exercised, see the analysis in [13], to avoid disorder. There it is shown that tiny small last elements in the eigenvectors cause disorder in the QR iteration. Consequently, in the RQ iteration, we must not take eigenvalues with a tiny *first* component in their eigenvectors, i. e. such that are not represented in the starting vector. We can still get several linearly independent eigenvectors to a multiple eigenvalue. The first time we use such a multiple eigenvalue as shift in the RQ iteration, we get the Schur vector that is represented in the starting vector, the second time we use the same shift, we get one more column in the Schur form, now with a vector that is represented in the second basis vector, and so on. See the careful discussion of this issue in [10]!

We now multiply the basis V_j from the right by the unitary matrix $U = [U_{j,j_1}\, U_{j,j-j_1}]$ of the Schur decomposition, and discard those vectors that correspond to the second block, $U_{j,j-j_1}$. We are back at the basic recursion (5), now with a basis of lower dimension j_1+1,

$$A[V_jU_{j,j_1}\; v_{j+1}]\left[\begin{array}{c} I_{j_1} \\ \underline{h}_{j+1}^{T}H_{j,j}^{-1}U_{j,j_1}\end{array}\right] = B[V_jU_{j,j_1}\; v_{j+1}]\left[\begin{array}{c} T_{j_1,j_1} \\ \underline{k}_{j+1}^{T}H_{j,j}^{-1}U_{j,j_1}\end{array}\right], \tag{19}$$

and can continue with the expressions in brackets as the new V, H and K. Note that the last vector v_{j+1} is not changed during all these manipulations. When purging occurs, it is natural to compute the continuation vector (16) *after* the purging step.

3. Implementation details. We have developed a program that attempts to compute all eigenvalues in a rectangle in the complex plane with opposite corners given by the two complex numbers *lb* and *ub*. We are also given a point *goal*, preferably close to one end of the region, and a forward direction, pointing into the region. This point is used as the first shift μ_1.

In each step j, we compute Ritz values using the pencil (13), and flag them as converged when the quantity $\omega_i^{(j)}$ (12) is small enough, say $\omega_i^{(j)} < 100 macheps$.

We keep the same shift μ_j for several steps j, determined by how expensive a new factorization (3) is, compared to continuation for a few more steps. We have simply chosen to set a parameter *cstep*, which tells us to wait until *cstep* eigenvalues have converged, before we choose a new shift. We also check that all eigenvalues behind the current shift have converged, this makes sure the process runs in the chosen forward direction, and that no new copies of multiple eigenvalues will be left behind. When we decide on taking a new shift μ_{j+1}, we take it as the mean of the *cstep* closest current forward Ritz values that have not yet converged. We also prescribe a maximum number of steps to keep a shift, most often 20, as well as a minimum number, most often 5.

We decide to purge and restart using the assumption that the Ritz values converge linearly with j. We had some feeling that if we kept more vectors, the values should converge in fewer steps, but our experiments have given remarkably linear convergence curves after an initial stage when the algorithm gathers information. We have to weigh the cost of computing a new basis, $W_{j_1} = V_j U_{j_1}$, see (19), which is $n \times j \times j_1$ flops, against the cost of keeping the $j - j_1$ purgable vectors for another k steps, which is $4n(j - j_1)k$ flops, if one reorthogonalization is always done in step 2.3 of ALGORITHM RKS. We can count how many Ritz values there are inside and outside the rectangle of interest, and so get a value of j_1, the number of kept vectors, before we actually set about to do the purging. Remember that j always is so small that we can neglect all operations that do not involve vectors of length n . If we purge, the number of vectors will grow again, if nothing special happens until j is back at its original value, i.e. we run for $k = (j - j_1)$ steps before we are back at the same decision situation. Under these assumptions, it pays to purge when,

$$4(j - j_1)^2 > j \times j_1 \,,$$

that is when,

$$j > \left(\frac{9 + \sqrt{17}}{8}\right) j_1 \approx 1.6 j_1 \,. \tag{20}$$

We used this relation, and most often purged all but *jkept* of the unconverged values.

We continue this way, at each purge increasing the size of the converged block in the Schur form (18), until no more unconverged Ritz values are to be found in the rectangular region given by *lb* and *ub*, and then we flag the whole algorithm as converged.

4. Two numerical examples.

We have tested the Rational Krylov algorithm using MATLAB4 on SUN SPARC, Hewlett Packard 700 series and IBM RS6000/590 workstations. The linear system computations in step 2.2 of ALGORITHM RKS were done with the sparse matrix option in MATLAB4. Reorthogonalization was done in step 2.3 whenever necessary. We took advantage of the complex arithmetic in MATLAB, even when we had real matrices.

This setup is simple to program, but does not squeeze the maximum performance out of the machines. A `.mex` file implementation of the simple but heavy Gram Schmidt orthogonalization in step 2.3, which used the IBM ESSL [8] implementation of the BLAS routines, ran about 10 times faster than our MATLAB code for that step. We actually used it in the runs reported in the upper half of table 4, that is why it is faster than the lower half, where a smaller matrix is treated.

Our first test example is a symmetric positive definite pencil, describing a finite element model of a membrane. We used the `numgrid` and `delsq` routines from MAT-

TABLE 1

Summary of time and space needed for different parameter settings. Asterisk indicates that a stipulated maximum was hit.

Settings			Resources needed				
Problem	*jkept*	*cstep*	Shifts	Purges	*jstep*	*jmax*	Time
L-membrane	∞		1	3	157	85	112.28
$[0, 500]$	15		1	4	108	56	46.82
Arnoldi	10		1	6	104	40	33.69
(22 eigv)	5		1	33	202	31	49.73
RKS	∞	2	7	0	72	65	77.52
	10	2	9	1	72	53	65.39
	5	2	10	12	100	33	62.80
$[0, 1000]$, Arn	10		1	22	355	67	124.20
(49 eigv) RKS	10	2	11	3	124	93	167.01
Tolosa	∞		1	4	131	50*	102.23
Arnoldi	15		1	6	174	50*	118.48
(23 eigv)	10		1	16	206	41	114.94
	5		1	42	244	31	100.29
RKS	15	2	9	2	64	50*	49.72
	10	2	11	5	97	42	55.71
	5	2	11	14	103	32	40.52
Cluster, 28 eigv	10	2	4	12	150*	34	71.82

LAB, slightly modified to get the 9 point stencil given by bilinear finite elements over a quadrilateral mesh.

We tried several shapes of membranes, and noted that the simplest square membrane gave the program some challenges, since it has many pairs of double eigenvalues. Here we choose to report runs on the standard L shaped membrane. It has some double eigenvalues; the first pair is at $\lambda_8 = \lambda_9 \approx 197$. If we take the element size $h = 1/64$, we will get matrices of size $n = 2945$. We have varied the parameters discussed in the previous sections. First we fixed the shift at $goal = 0$ all the time, this is shifted and inverted Arnoldi. In the next run we did purge as soon as the size j satisfied the condition (20) for some chosen values of *jkept*. See the summary in table 4!

It must be stressed that ours was not a highly optimized code, we ran Arnoldi simply by inhibiting purges and new shifts in our RKS code. Of the 112 seconds used in the first line, we spent 87 to compute Ritz values etc, we actually solved the small eigenproblem (13) in every step j. A simpler Arnoldi program, (See MATLAB PDE-TOOL [4]!), got the same results in just 25.93 seconds.

Under the line we show results for Rational Krylov proper. We see that fewer steps *jstep* and vectors *jmax* are needed, but that the total timing is slightly worse due to the time needed for factorizations. Repeated purging, triggered by a smaller value of *jkept*, saves basis vectors for both methods, but we need more iteration steps j to get all eigenvalues in the interval.

It might be instructive to follow how the eigenvalue approximations converge. In figure 1 they are plotted with the time going upwards. We let the color indicate the size of the residual as estimated by $\omega_i^{(j)}$ (12) and use the `jet` color map. The blue (dark) points are far from convergence, the yellow (lighter) ones are on their way, and

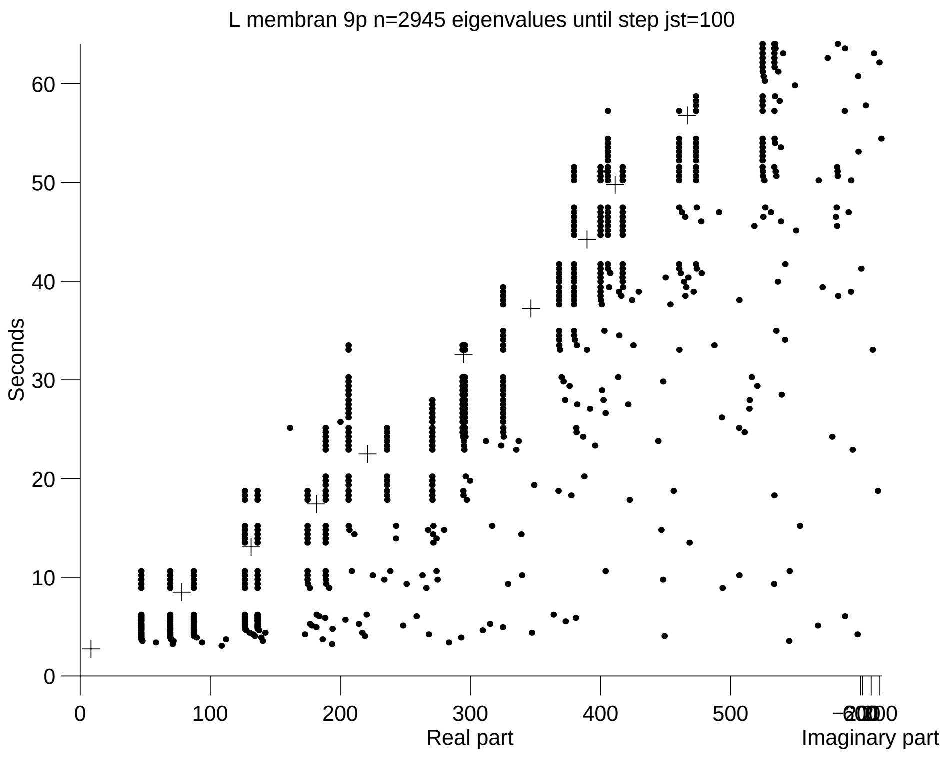


FIG. 1. *Eigenvalues for all steps of RKS applied to the finite element model of L-membrane matrix. Accuracy indicated by color. Position of shifts marked by +.*

when they get red (dark again) the $\omega_i^{(j)}$ approaches the tolerance.

New shifts are marked with crosses (plus signs), and in this run we had $cstep = 2$, which placed the shifts just between pairs of eigenvalues. The time needed for the refactorizations shows up as empty space just below (before) the shift marks.

The reader has to take into account that the alignment is not perfect, the first shift is at exact 0, and the pile of dots shown just to the right of 200 is actually the eigenvalue at $\lambda \approx 197$.

The pattern shown here is very typical, a new Ritz value comes in at each step, but it takes only one or two steps for it to stabilize, and then it stays. Look at the double eigenvalue at $\lambda \approx 197$. The first copy converges after 23 seconds, and is treated as converged in the purge at second 25. Then the second copy comes in from the left, see the two dark dots to the left of $\lambda = 200$ at second 24, and stays in the computation until it converges and is set aside in the purge at the 33 rd second. All its simple neighbors are then since long converged, but note that we have postponed the new shift at $\lambda \approx 300$ until everything to the left of the previous shift at 213 has been flagged as converged. A similar event occurs at the second double eigenvalue at $\lambda \approx 397$, and after 58 seconds we have no more unconverged approximations to the left of $ub = 500$. We then start again with a random orthogonal vector and the projected matrix, and run until we are sure that no new Ritz values will be found in the region. After 10 steps, at second 62, the smallest Ritz value stabilizes at $\lambda \approx 516$, and we expect no more to be found to the left of $ub = 500$.

We have plotted the sizes of the residual estimates in figure 2. When a new shift

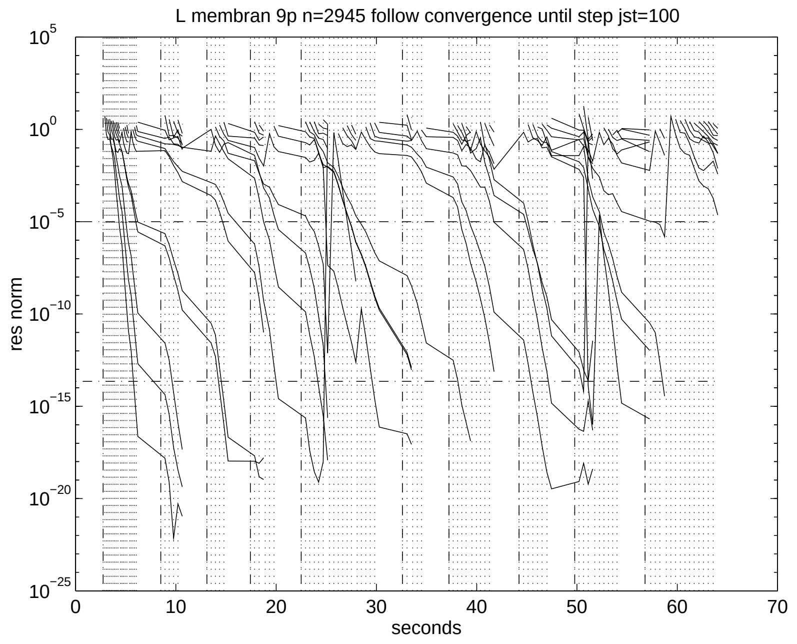


FIG. 2. *Residual estimates as function of time. Each step marked by dotted vertical line, refactorization by dashdotted line. Tolerances horizontal lines.*

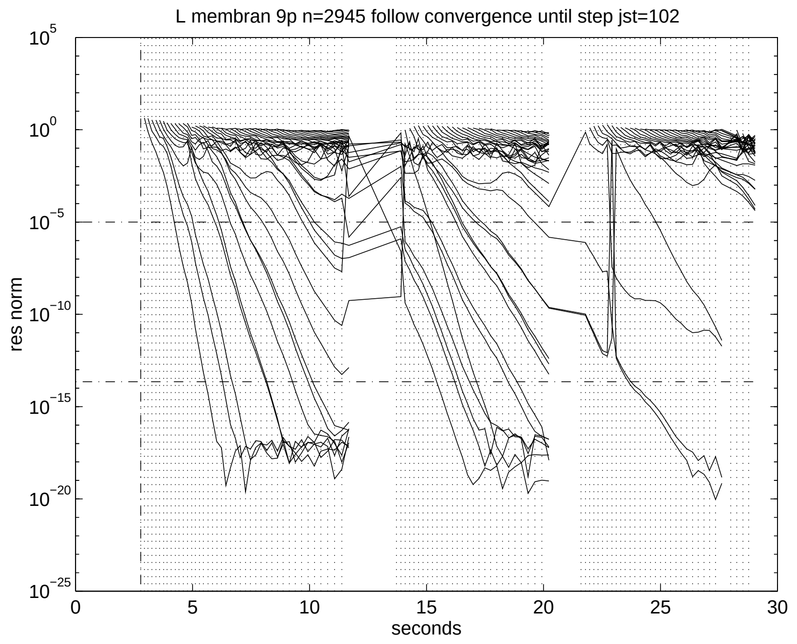


FIG. 3. *Residual estimates for Arnoldi with two big purges.*

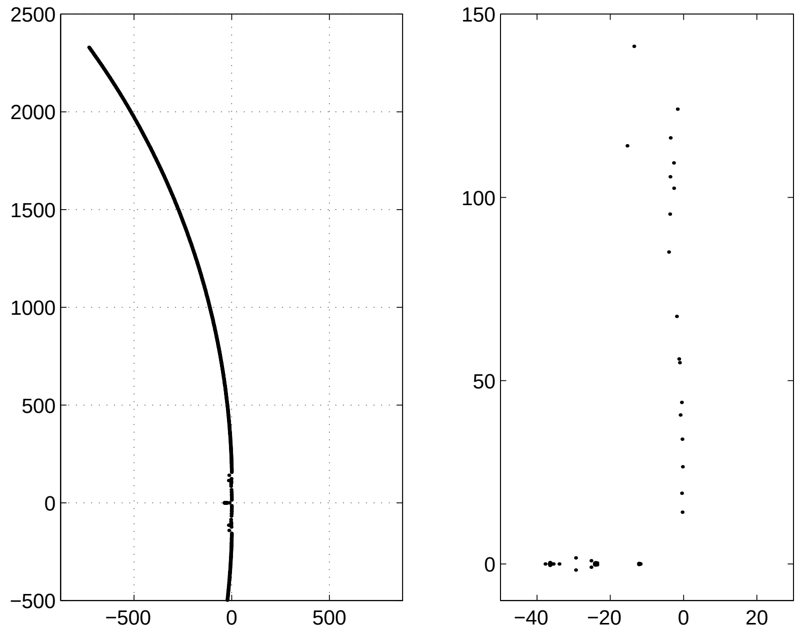


FIG. 4. *Tolosa matrix, overview and detail of spectrum. Matrix is real so only upper half plane shown.*

is introduced, those in the neighborhood start dropping much faster. There is some confusion when the new copy of $\lambda \approx 197$ comes in at second 24. The convergence is essentially linear when the shift is kept constant. This is quite a bit more evident in the Arnoldi run shown in figure 3. This is for *jkept* = 15, where two big purges are done at seconds 12 and 20. Our exact shift RQ algorithm for this needs many operations, and it shows in the timing here. A more economical scheme, as developed in [10], will be faster.

On the last 2 rows in the upper half of table 4, we show what happened when we computed the 49 eigenvalues smaller that 1000. Now RKS needs just one third as many steps as Arnoldi, that has quite a hard time of finding both copies of the last double eigenvalue at $\lambda \approx 997$.

Let us also report some runs on the test matrix TOLOSA taken from [3]. It is typical for those matrices one obtains when calculating the stability of an aircraft structure, and for $n = 2000$, the size we tested, its spectrum is plotted in figure 4 . It took us 16 minutes on our fastest IBM to compute these exact eigenvalues. We just used the MATLAB `eig` command,

In our tests we sought the eigenvalues in the upper left end of the spectrum by choosing $goal = -750 + 2390i$. These eigenvalues are the worst conditioned and are also critical for the actual design.

We report some tests on computing the 23 eigenvalues in the region bounded by $-750 < \mathrm{Re}\lambda < -650$, $2200 < \mathrm{Im}\lambda < 2400$. See the summary in the bottom half of table 4! For the case with frequent purges, *jkept* = 5, we also show how the Ritz

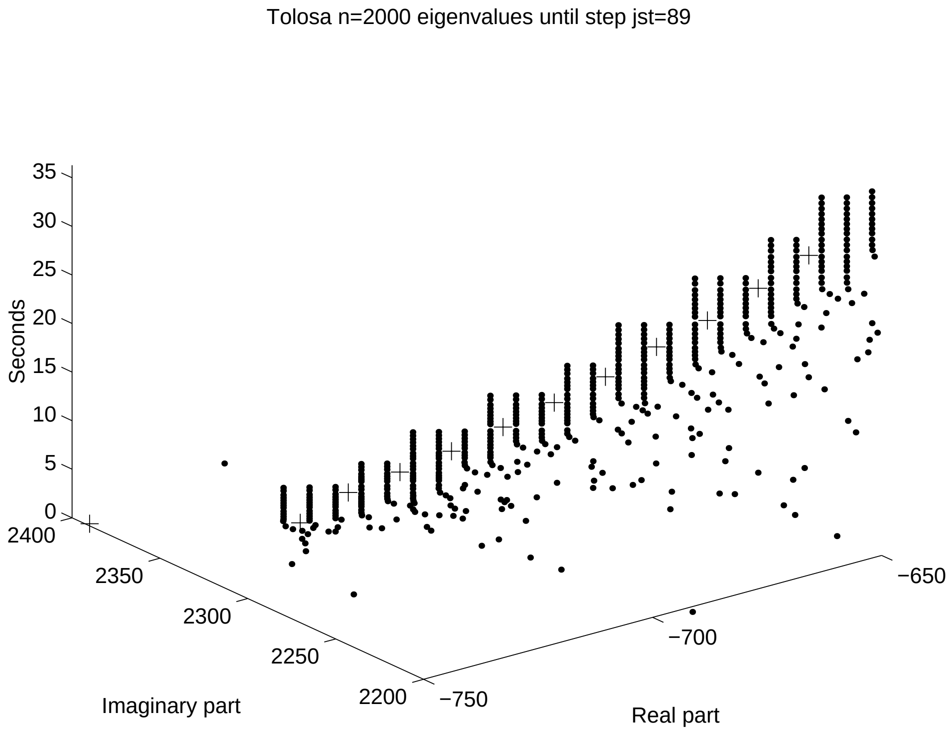


FIG. 5. *Tolosa matrix, follow eigenvalue approximations, perspective.*

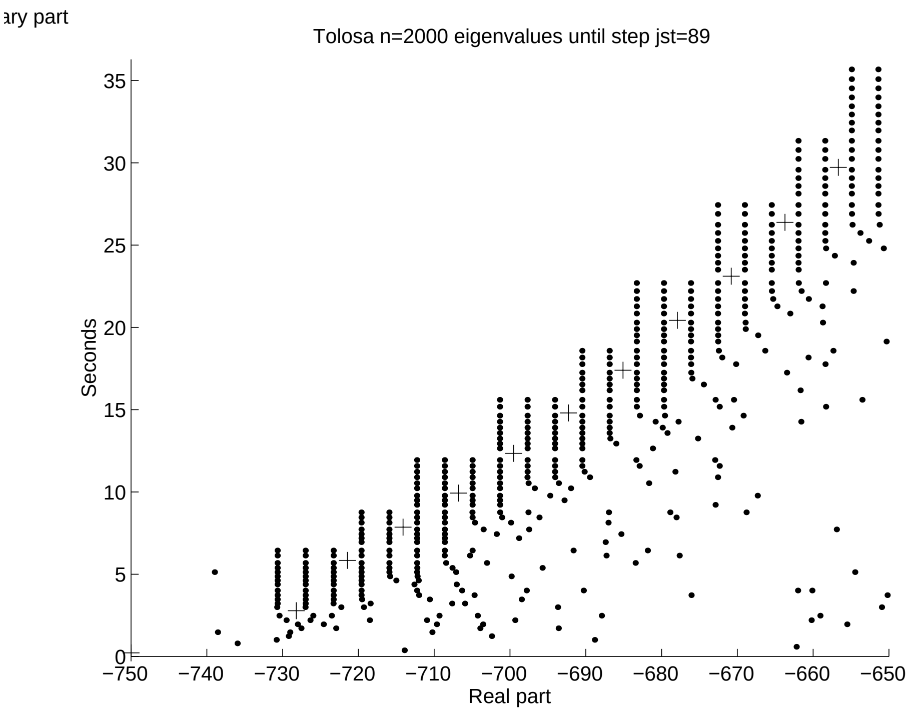


FIG. 6. *Tolosa matrix, follow eigenvalue approximations, view from imaginary direction.*

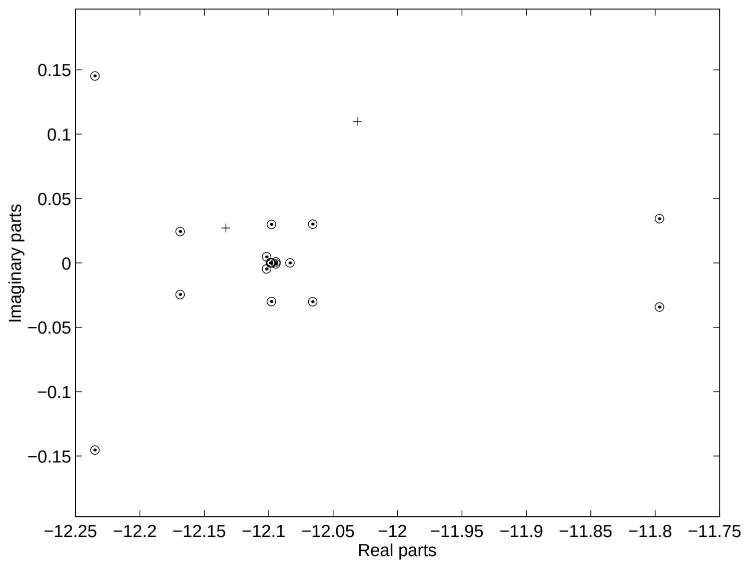


FIG. 7. *Tolosa matrix, region around eigenvalue of high multiplicity. o : computed eigenvalue, .: exact eigenvalue, +: shift.*

values converge in figures 5 and 6. The most notable difference from the previous, symmetric, example is that we need to do quite a few steps before the first eigenvalue converges, it happens at step $j = 31$ after 8 seconds. Then they will come in a few at a time, the shifts will be rather close and the starting approximations before a new shift is applied is rather good. This is the main reason for the superiority or RKS over Arnoldi for this matrix.

When we compare different frequences of purging, we get a similar effect as for the L membrane, but less pronounced. The case $jkept = 10$ ran slower than the other two, this is due to that it happened to need a full 9 steps after a restart with a random vector to decide that all eigenvalues are already found. It might be noted that Tolosa is a rather simplified test matrix, any more realistic example is expected to have more filled elements.

In this region all eigenvalues were simple. We also tested eigenvalues close to the origin which is a challenge since there is an eigenvalue with multiplicity 385 at $\lambda \approx -12.098$. Setting a region $-15 < \mathrm{Re}(\lambda) < 1$, $-1 < \mathrm{Im}(\lambda) < 15$ and letting RKS go for 150 steps, we got 28 eigenvalues, one at $\lambda \approx -0.3323 + 14.1214i$, and the rest plotted in figure 7. We plot our approximations as circles and the exact eigenvalues as dots. We got 12 copies of the multiple eigenvalue, all with nearly orthogonal vectors. After we had got all the distinct eigenvalues, we got one new copy every 5 to 7 steps, the purging makes sure that we all the time get new linearly independent eigenvectors. See the last line of table 4!

We also computed the spectrum up along the imaginary axis from $goal = 0$. In the beginning, we had to purge copies of $\lambda \approx -12.098$ that popped up, but later, when the shifts were further up along the imaginary axis, they ceased to appear and we could trot along as long as needed, all the time picking up a couple of eigenvalues for each shift. The essential one dimensionality of the spectrum, makes our heuristics for finding all eigenvalues work well for the Tolosa matrix.

Acknowledgement. This work got into shape while the author enjoyed the kind hospitality of Beresford Parlett and the stimulating environment of the LAPACK group at University of California, Berkeley.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide, Release 2.0*, SIAM, Philadelphia, 1995. 324 pages.

[2] W. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[3] F. Chatelin and S. Godet-Thobie, *Stability analysis in aeronautical industries*, in Proceedings of the 2nd Symposium on High-Performance Computing Montpellier, France,, M. Durand and F. E. Dabaghi, eds., Elsevier/North-Holland, 1991, pp. 415–422.

[4] Computer Solutions Europe AB, *The Partial Differential Toolbox*, 1995. Beta-test version.

[5] T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. Comp., 35 (1980), pp. 1251–1268.

[6] K. Gallivan, E. Grimme, and P. V. Dooren, *A rational Lanczos algorithm for model reduction*, to appear, (1995).

[7] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matr. Anal. Appl., 15 (1994), pp. 228–272.

[8] IBM, *Engineering and Scientific Subroutine Library*, version 2 release 2 ed., 1994.

[9] M. T. Jones and M. L. Patrick, *LANZ: Software for solving the large sparse symmetric generalized eigenproblem*, Tech. Rep. MCS-P158-0690, Argonne National Lab, Argonne, IL. Also available from `netlib`., 1990.

[10] R. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, Tech. Rep. TR94-13, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1995.

[11] C. Paige, B. N. Parlett, and H. V. der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Num. Lin. Alg. Appl., 2 (1995), pp. 115–133.

[12] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice Hall, 1980.

[13] B. N. Parlett and J. Le, *Forward instability of tridiagonal QR*, SIAM J. Matr. Anal. Appl., 14 (1993), pp. 279–316.

[14] A. Ruhe, *The two-sided Arnoldi algorithm for nonsymmetric eigenvalue problems*, in Matrix Pencils, LNM 973, B. Kågström and A. Ruhe, eds., Springer-Verlag, Berlin Heidelberg New York, 1983, pp. 104–120.

[15] ———, *Rational Krylov sequence methods for eigenvalue computation*, Lin. Alg. Appl., 58 (1984), pp. 391–405.

[16] ———, *Rational Krylov algorithms for nonsymmetric eigenvalue problems*, in Recent Advances in Iterative Methods, IMA Volumes in Mathematics and its Applications 60, G. Golub, A. Greenbaum, and M. Luskin, eds., Springer-Verlag, New York, 1994, pp. 149–164.

[17] ———, *Rational Krylov algorithms for nonsymmetric eigenvalue problems, II: Matrix pairs*, Lin. Alg. Appl., 197/198 (1994), pp. 283–296.

[18] ———, *The Rational Krylov algorithm for nonsymmetric eigenvalue problems. III: Complex shifts for real matrices*, BIT, 34 (1994), pp. 165–176.

[19] ———, *Showing the progress of the Rational Krylov algorithm*, tech. rep., Dept Computer Science, University of California Berkeley, 1995.

[20] Y. Saad, *Variations of Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Lin. Alg. Appl., 34 (1980), pp. 269–295.

[21] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, Tech. Rep. 856, University of Utrecht, Department of Mathematics, 1994.

[22] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matr. Anal. Appl., 13 (1992), pp. 357–385.