# Eigenvalue Algorithms with Several Factorizations – A Unified Theory yet?

Axel Ruhe*

June 23, 1997

**Abstract**

A general discussion of applications, where numerical solution of large scale matrix eigenvalue problems occur, is given and the most common solution methods are classified. Three recent algorithms, rational Krylov, Jacobi Davidson and truncated RQ are compared. A reformulation of the Arnoldi algorithm as a standard transformation into Hessenberg form, is used to show the equivalence between rational Krylov and truncated RQ. A small numerical example is given, and unsolved questions and further developments are discussed.

## 1 Introduction, when do we need eigenvalues?

Numerical computation of eigenvalues is a critical link in the applied mathematics problem solving chain for quite a few different scientific and engineering endeavors. Let us start by putting it into context, the better to understand what is needed and what is possible.

Normally we have a question in the world out there, for which we want to find answers. We start by using theories from the relevant scientific fields, let it be physics, chemistry, mechanics, electrical engineering or economics, to formulate a mathematical model. This model will consist of equations for equilibrium and input/output relations, as well as differential equations for rates of change and dynamics. The applied mathematician replaces these continuous and infinite dimensional models with discrete and finite dimensional approximations, and here our matrices come up. Most often we get linear systems, for equilibrium and static situations, but when we deal with dynamics, or want to see what happens when we move from or approach an equilibrium, we will need eigenvalues.

The results of the eigenvalue computation is interpreted in the mathematical model in the form of response functions, or visualized in spectral diagrams. In

*Department of Mathematics, Chalmers Institute of Technology and the University of Göteborg, S-41296 Göteborg, Sweden (ruhe@cs.chalmers.se). Talk given at *Workshop on the use of iterative methods for large-scale Eigenvalue Problems* at Argonne National Laboratory, May 14, 1997. Partial support given by TFR, the Swedish Research Council for Engineering Sciences, Dnr 222, 96-555.

some cases the eigenvalues themselves are interesting, for instance as energy levels in a molecule, in others they are just used to get answers to questions given in other terms.

It is important for us, who deal with numerical computations, to know that it is only one link in a long chain. On the other hand, we can work much better if we have some understanding of what the other links look like, what kind of data we can expect, and which accuracy we have to deliver.

Seen in a longer historical perspective, scientific computation has evolved from a one person handicraft into an industrially organized work with division of labour, specialization and formal interfaces between different specializations and levels.

In the old times, let me say around 1960, anybody needing to compute had to do his own programming in FORTRAN or machine code. A quantum chemist had his HÜCKEL, where a Schrödinger eigenvalue problem was solved, by means of a Jacobi algorithm to give energy levels. A structural engineer used DETERMINANT SEARCH to determine vibrational modes in a finite element model of a piping system. Later, say in 1980, it was realized that these, and several other computations, could all be formulated as matrix eigenvalues, and that it could be of advantage to divide the task, and include STLM, a spectral transformation Lanczos code, in ASKA, a finite element package for aircraft design. Today in 1997, we have continued on that path, and now each acronym in the chain PETSC, ARPACK, LAPACK, BLAS, FORTRAN, IEEE-754 is developed and maintained by its own group of specialists, each program package has to call the next and deliver well specified results back.

Nowadays, it is in the software industry that we see the Tayloristic division of labour. Charlie Chaplin drew his nuts all day in "Modern Times" in 1936. Now nuts are drawn by a robot, while Charlie's spiritual grandson sits as a part of a million man hour project at ERICSSON, to program his 162 bit chunk of a burst in a digital cellular phone.

After these generalities, let us be more specific and ask, when do we need to solve,

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

for eigenvalues $\lambda_k$ and eigenvectors $x_k$?

The most general answer is, that it is on the first step from a static to a dynamic model of whatever we study. Small perturbations of the static, equilibrium, position will be formed as eigenvectors and move as the eigenvalues indicate. The foremost mathematical modelling tool is the Laplace transform, where we map a description in time and space to one involving decay rates and oscillation frequences, these are the real and imaginary parts of the eigenvalues. Yes in this business, complex numbers are simple, separating them into real and imaginary parts, is a complication.

In this realm we use eigenvalues to get the solution of a linear system of ordinary differential equations over time

$$x(t) = \Sigma_j x_j e^{\lambda_j t}\,,$$

as well as to study the response or transfer function of a linear dynamical system,

$$H(z) = E^T(G + zC)^{-1}B\,,$$

but also to follow a deformation path,

$$f(x, \lambda)\,,$$

for increasing load $\lambda$. This last is a nonlinear eigenvalue problem, take partial derivatives and we get a linear one.

# 2 How to compute eigenvalues

There is essentially two approaches to get eigenvalues, *transformation* and *iterative* methods. Transformation methods make a sequence of similarity transformations, until we reach a diagonal form,

$$A = X\Lambda X^{-1}\,,$$

showing all the eigenvalues and eigenvectors. Transformation methods can be used when you can store the whole matrix in one array in your computer and want all eigenvalues. The program package LAPACK[1] and MATLAB give transformation methods as their primary choice.

In an iterative method, on the other hand, a sequence of vectors is computed,

$$x^{(1)}, x^{(2)}, \ldots, x^{(k)} \Rightarrow x_j\,,$$

hopefully converging towards an eigenvector $x_j$.

A hybrid, which has occurred to be the most fruitful one for the large and sparse problems that is our main interest today, is *subspace* methods, where a low dimensional subspace is built and updated during the computation, and eigenvalues are found from the small section matrix,

$$V^H A V\,,$$

where $V$ is a matrix of orthonormal basis vectors of the subspace in question.

In most cases the subspaces are chosen as *Krylov* subspaces,

$$K^j(C, x) = \{x, Cx, C^2x, \ldots, C^{j-1}x\}\,, \tag{2}$$

obtained by repeated application of a matrix operator $C$.

We will spend what remains of this contribution to discuss different aspects of such Krylov subspace methods and concentrate on:

1. *Matrix preparation.* How do we get the operating matrix $C$ from the matrix pair $(A, B)$ of the original pencil (1).

2. *Subspace generation.* How do we add and remove vectors from the basis $V$, and do we insist on keeping it orthogonal?

3. *Eigenvalue approximation.* How do we get approximate eigenvalues when the basis is given?

The standard, and well understood, matrix preparation is the shift invert spectral transformation,

$$C = (A - \mu B)^{-1} B\,, \tag{3}$$

but now we will discuss also algorithms where different shifts or poles, $\mu_j$ are applied in different iterations $j$, as well as what happens, if we replace the direct sparse Gaussian elimination factorization needed for the inverse in (3) with an incomplete factorization,

$$A - \mu B = LU + R\,.$$

The subspaces are traditionally generated by adding one vector at a time and orthogonalizing, like in the symmetric Lanczos [9] or Arnoldi [2] algorithms,

$$CV_j = V_{j+1} H_{j+1,j}\,, \tag{4}$$

but also more elaborate schemes, like implicit restart with lock and purge, have been proposed recently, see [11], where the basis $V$ is transformed from the right and partitioned into,

$$CV = [V_1 V_2 V_3] \begin{bmatrix} T & H_{1,2} & \dots \\ 0 & H_{2,2} & \dots \\ & e_1 e_k^T h_{k+1,k} & H_{3,3} \end{bmatrix},$$

where $V_1$ are locked vectors, already converged, $V_2$ are used for furhter computation in implicit restart, and $V_3$ are purged and leave the basis.

Finally, the standard way of finding approximate eigenvalues is to compute the Ritz values,

$$\lambda(H_{j,j})\,,$$

but this has been compared to other orthogonality conditions,

$$(AV - \lambda BV)s \perp W\,,$$

Petrov Galerkin approximations [5] or harmonic Ritz values [13] of different flavours.

# 3 Current approaches

We will concentrate our study on three recently proposed methods, that give different ways to operate with several shifts $\mu_j$, while forming one basis $V$. The Rational Krylov method [14, 15, 16, 17, 18] was developed expressly for this purpose, see also the recent works by Lehoucq, Meerbergen and coworkers, [10], where it is used with inexact solutions of linear systems in the inner iteration.

The Jacobi Davidson method, see [3, 12, 21, 5], was first designed to make it possible to use an iterative linear system solver as inner iteration, it has later been shown that it generates a Rational Krylov sequence when the systems are solved to full accuracy [18].

The third method we will study is Truncated RQ (TRQ), proposed by Sorensen and Yang [22]. It updates an entire subspace, by applying a shifted and inverted operator (3) to one carefully chosen vector. Elaborate numerical tests are reported in [22], they show that TRQ has very similar convergence properties, compared to RKS and JD.

# 4 Variations on Arnoldi

[1] Let us now establish the relation between our different methods by fitting them into the same framework, and why not choose their common ancestor, the Arnoldi algorithm (4)? We will need to apply Arnoldi both forwards and backwards, let us describe this in a MATLAB like terminology.

Arnoldi is not (yet) built into MATLAB. However, the command `hess` can be used in the following way,

$$\begin{bmatrix} 0 & 0,\ldots,0 \\ e_1 & H \end{bmatrix} = \texttt{hess}\begin{bmatrix} 0 & 0,\ldots,0 \\ v_1 & A \end{bmatrix}. \tag{5}$$

It is well known that this will generate the same basis $V$ and Hessenberg matrix $H$ as Arnoldi started on $v_1$, even though `hess` uses Householder transformations instead of Gram Schmidt, this approach was proposed by Golub, Underwood and Wilkinson [6], to make sure Lanczos will produce vectors orthogonal to full accuracy.

We can use `hess` also to get one step of implicitly shifted QR,

$$\begin{bmatrix} 0 & 0,\ldots,0 \\ e_1 x & T^{(j+1)} \end{bmatrix} = \texttt{hess}\begin{bmatrix} 0 & 0 & 0 & 0 & & & 0 \\ \alpha_1-\mu & \alpha_1 & \beta_1 & 0 & & & \\ \beta_1 & \beta_1 & \alpha_2 & \beta_2 & & & \\ 0 & 0 & \beta_2 & \ddots & \ddots & & \\ & & & \ddots & \ddots & \beta_{n-1} \\ 0 & & & & \beta_{n-1} & \alpha_n \end{bmatrix},$$

where $T^{(j+1)}$ stands for the transformed tridiagonal matrix and $\mu$ is the QR shift. The same procedure can, of course, be applied to a nonsymmetric Hessenberg matrix.

Now this device can be permuted in several interesting ways. This transformation (5) takes a starting column, $v_1$, and sweeps it from left to right, so let us call it the left-right variant. By proper permutations and transpositions we get three more variants, precisely as in Gaussian elimination. If we let the index vector `ir` stand for reversion, the MATLAB call,

[1] Note the title of Saad's work [19], which was the one reviving the interest in Arnoldi!

```
[P, H]=hess(A(ir,ir)');
H=H(ir,ir)';
P=P(ir,ir);
```

produces an upper Hessenberg matrix, starting in the lower right corner and sweeping the last row up. This down-up variant will be specially interesting in this study, since it corresponds to the RQ iteration.

We can develop the third, up-down variant by adding a row on top, this gives a way to determine a tridiagonal matrix with given eigenvalues $\lambda_j$ and first components of eigenvectors $\alpha_j$ by calling,

```
[P, H]=hess([0 a'; a diag(lambda)]);
T=H(2:n+1,2:n+1);
```

This inverse eigenvalue algorithm was described by de Boor and Golub [4] using Lanczos, while Gragg and Harrod [8] gave this more stable variant.

I have not yet seen in which context the fourth, right-left, variant has been described, but sure there is some case when it can be put to use.

## 4.1 Rational Krylov as Arnoldi

We can use the down-up variant of the Hessenberg transformation to fit the RKS recursion into any Arnoldi we please, in the following way:

Take the basic recursion of RKS, see [18],

$$AV_{j+1}H_{j+1,j} = BV_{j+1}K_{j+1,j} \ , \tag{6}$$

where $V$ is the computed orthogonal basis, $H_{j+1,j}$ is the $(j+1) \times j$ upper trapezoidal matrix of orthogonalization coefficients and, $K = H\mathrm{diag}(\mu_i) + T$ is the other trapezoidal matrix in the reduced pencil, here $\mu_i$ are the shifts (poles) and the upper triangular $T$ has the continuation combinations as columns.

Factorize,

$$H_{j+1,j} = Q_{j+1} \begin{bmatrix} R_{j,j} \\ 0 \end{bmatrix} ,$$

and multiply the basic recursion (6) from the right by the inverse of the triangular factor,

$$AV_{j+1}Q_{j+1} = BV_{j+1}Q_{j+1}Q^H_{j+1}K_{j+1,j}R^{-1}_{j,j} \ . \tag{7}$$

Apply the down-up variant of the Hessenberg algorithm to the full $(j+1) \times j$ matrix formed by the last three factors here,

$$Q^H_{j+1}K_{j+1,j}R^{-1}_{j,j} = P_{j+1,j+1}\tilde{K}_{j+1,j}P^H_{j,j} \ , \tag{8}$$

giving an upper trapezoidal matrix $\tilde{K}_{j+1,j}$, if we strike out the last zero column added in the down-up variant of (5), and note that the reversed `hess` transformation, $P_{j+1,j+1}$, has the unit vector $e_{j+1}$ as its last column.

Multiply the transformed basic recursion (7) with the orthogonal $P_{j+1,j}$ from the right and get,

$$\begin{aligned} AV_{j+1}Q_{j+1}P_{j+1,j} &= BV_{j+1}Q_{j+1}P_{j+1,j+1}\tilde{K}_{j+1,j} && (9) \\ AW_j^{(j)} &= BW_{j+1}^{(j)}\tilde{K}_{j+1,j}^{(j)} , && (10) \end{aligned}$$

an Arnoldi recursion with the new orthonormal basis $W$. The superscript $(j)$ is there to caution that for each step $j$, a new basis and a new trapezoidal reduced matrix is implied. This is in contrast to the standard Arnoldi algorithm, and the original formulation (6) of the RKS recursion, where new columns and rows are added to $V$, $H$ and $K$ in each step, without changing elements already computed.

Note that we could use the same reformulation to get any shifted and inverted Arnoldi from RKS, see [18], but we have chosen this direct unshifted variant, because it conforms with the formulation of the TRQ algorithm [22].

As described here, $O(nj^2)$ arithmetic operations are performed in each step $j$. Let us now device an updating variant that only needs $O(nj)$.

In a typical step, we start from an Arnoldi recursion (10), up to step $j-1$,

$$AV_{j-1} = BV_jK_{j,j-1} ,$$

and choose a shift $\mu_j$ from the eigenvalues of $K_{j,j-1}$. Multiply a linear combination $t_j$ of the basis vectors $V_j$ with a shifted and inverted matrix, and orthogonalize with Gram Schmidt to get,

$$V_{j+1}h_j = (A-\mu_jB)^{-1}BV_jt_j ,$$

adding one column $v_{j+1}$ to the orthonormal basis. Multiply by $(A-\mu_jB)$ and formulate it all as a Rational Krylov recursion (6),

$$AV_{j+1}\begin{bmatrix} I_{j-1} & h_j \\ 0 & h_{j,j} \\ 0 & h_{j+1,j} \end{bmatrix} = BV_{j+1}\begin{bmatrix} K_{j,j-1} & t_j + h_j\mu_j \\ 0 & h_{j+1,j}\mu_j \end{bmatrix} .$$

There is only one rotation, $Q_{j+1,j}$ in the $(j, j+1)$ plane, needed to make the left hand side matrix upper triangular,

$$AV_{j+1}Q_{j+1,j}\begin{bmatrix} I_{j-1} & h_j \\ 0 & r_{j,j} \\ 0 & 0 \end{bmatrix} = BV_{j+1}Q_{j+1,j}\begin{bmatrix} & K_{j-1,j-1} & t_j + h_j\mu_j \\ 0 & \tilde{k}_{j,j-1} & \tilde{k}_{j,j} \\ 0 & \tilde{k}_{j+1,j-1} & \tilde{k}_{j+1,j} \end{bmatrix} , \quad (11)$$

affecting only the lowest two by two submatrix of the right hand side factor.

We will get into the single matrix recursion (7), by modifying only the last column $k_j$ of the $K$ matrix into $(k_j - K_{j+1,j-1}h_j)/r_{j,j}$.

The down up variant of the Hessenberg transformation (8) now consists of only one chase to zero the fill in element $\tilde{k}_{j+1,j-1}$, and we have arrived at the next step of the Arnoldi recursion (10), using just $j$ plane rotations and one matrix vector multiplication.

### 4.2 TRQ as Arnoldi

The TRQ algorithm [22], is even more closely related to Arnoldi. Let us describe it for the simple $B = I$ case. It takes an initial Arnoldi recursion (10) of a predetermined dimension $j$, this may be generated by direct application of Arnoldi the first time. Then, in a typical iteration step, it operates with a shifted and inverted operator on a linear combination $t_j$ of the current basis,

$$w = (A - \mu I)^{-1} V_{j+1} t_j \, .$$

I have modified the notation from [22] to simplify comparison with the previous subsection. Then **Algorithm 2** in [22] is applied to orthogonalize this to the leading $j$ part of the current basis and get,

$$(A - \mu I) v_+ = V_j h_j + v_{j+1} \alpha \, .$$

This is added to the already computed Arnoldi recursion (10) to get,

$$(A - \mu I) \left[ V_j \; v_+ \right] = \left[ V_j \; v_{j+1} \right] \begin{bmatrix} H_{j,j} - \mu I & h_j \\ \beta_j e_j^T & \alpha \end{bmatrix} . \tag{12}$$

Please note that we have two different bases on the left and right. Make a RQ factorization of the Hessenberg matrix on the right,

$$(A - \mu I) \left[ V_j \; v_+ \right] = \left[ V_j \; v_{j+1} \right] \begin{bmatrix} R_{j,j} & r_j \\ 0 & \rho \end{bmatrix} Q_{j+1} \, .$$

Multiply by $Q_{j+1}^H$ from the right, to yield the modified basis,

$$W_{j+1} = \left[ V_j \; v_+ \right] Q_{j+1}^H \, .$$

We intend to throw away the last column, and then we may silently replace $v_{j+1}$ by $v_+$ in the basis on the right hand side, and transform it into the modified basis $W_{j+1}$ while completing the RQ iteration by premultiplying R with Q,

$$(A - \mu I) W_j = W_{j+1} \tilde{H}_{j+1,j} \, .$$

A straightforward but tedious study of the equations involved, shows that the same effect is obtained, if we apply the algorithm of the previous subsection and throw away the last column. There we kept all the basis vectors, and added a new one after them, but the $j$th vector of the basis after the rotation, $V_{j+1} Q_{j+1,j}$ in (11), is just the vector $v_+$ discussed here. (In the previous subsection we treated the step from a $j-1$ to a $j$ basis, here we start from $j$ and throw away one vector to keep it that size.)

## 5 A small example

We have made just a few very preliminary tests of the new variants described in this contribution using a simple minded MATLAB code.

Let us follow what happens with the symmetric matrix $A$ of size $n = 385$ one gets from a standard 5 point finite difference approximation to an L shaped membrane, generated by the commands `G=numgrid('L',25)` and `A=delsq(G)` in MATLAB. We used the same starting vector of normally distributed random numbers in all runs reported here.

We ran RKS starting with a shift $\mu = 0$ and applying a new shift every step after the first 2 steps. This way we changed shifts much more often than recommended in [18], but the convergence was very fast, flagging 6 eigenvalues as converged after $j = 15$ steps. Let us list what happened:

```
Backwards Arnoldi RKS
  j jsh Shift        Eigenvalue appr  omega     resnorm  k(h)
  1  1  0.00000e+00  1.9033744216e+00 4.93e-01 1.15e-15 7.31e-01
  2  1  0.00000e+00  9.9380121840e-02 4.61e-01 1.78e-15 1.25e+00
  3  1  9.93801e-02  8.6007792339e-02 4.38e-01 2.09e-15 8.12e-01
  4  1  8.60078e-02  6.7307749585e-02 1.18e-01 8.30e-15 3.30e+00
  5  1  6.73077e-02  6.7241155915e-02 1.62e-05 5.85e-14 2.19e+01
  6  1  6.72412e-02  6.7241155911e-02 5.63e-16 5.62e-10 1.94e+05
  7  2  1.04911e-01  1.0491116253e-01 1.08e-10 1.52e-09 2.23e+03
  8  2  1.04911e-01  1.0491116253e-01 9.68e-18 1.88e-09 9.37e-01
  9  3  1.36297e-01  1.3629669486e-01 7.17e-09 2.49e-07 7.29e+02
 10  3  1.36297e-01  1.3629669486e-01 3.98e-15 3.05e-07 1.09e+00
 11  4  2.02796e-01  2.0279520146e-01 1.42e-08 6.96e-06 5.95e+02
 12  4  2.02796e-01  2.0279520146e-01 2.19e-14 1.05e-05 1.33e+00
 13  5  2.20269e-01  2.2026935329e-01 8.97e-14 1.40e-03 1.35e+04
 14  6  2.83651e-01  2.8348793801e-01 4.00e-05 5.33e-03 4.53e+01
 15  6  2.83488e-01  2.8348793890e-01 1.07e-13 6.88e-03 1.22e+00
```

Here, `j` is the step, size of basis, `jsh` the eigenvalue computed, `omega` is the estimate of the residual used to flag convergence, `resnorm` gives the difference between the right and left sides of the transformed basic recursion (10), and `k(h)` the condition of the upper triangular matrix multiplied from the right in (7). Note that each time this one is ill conditioned, we will get a larger discrepancy in the previous column. This behaviour has to be studied, a probable, but yet untried remedy is to avoid the multiplication from the right and keep the left hand side factor triangular and the right hand side Hessenberg, this is the intermediate form of the QZ algorithm.

These reservations made, let us show some of the $K$ matrices. After $j = 4$, we have

```
k =
   9.9974e-02   1.2727e-02
   1.2727e-02   7.2393e-02   5.2688e-03
            0   5.2688e-03   2.1530e-01   9.1025e-02
            0            0   9.1025e-02   1.9131e+00
            0            0            0  -1.6748e+00
j=5
```

```
k =
   6.7242e-02   1.7171e-04
   1.7171e-04   1.0491e-01   5.4056e-04
            0   5.4056e-04   1.9680e-01   3.0140e-02
            0            0   3.0140e-02   3.0142e-01   3.3474e-01
            0            0            0   3.3474e-01   2.1940e+00
            0            0            0            0  -1.7942e+00

j=6
k =

   6.7241e-02   3.9619e-14
   2.0749e-14   1.0491e-01   1.7579e-04
            0   1.7579e-04   1.7787e-01   3.4337e-02
            0            0   3.4337e-02   1.7205e-01   4.7463e-02
            0            0            0   4.7463e-02   6.6195e-01   5.0935e-01
            0            0            0            0   5.0935e-01   2.9516e+00
            0            0            0            0            0  -1.8588e+00
```

We have stroken out the elements above the tridiagonal, in the case $j = 6$ they were of larger size as indicated in the table.

Note that the matrices, apart from that, are symmetric and that all elements change in each iteration. The first vector in the basis should, in principle, yield the same recursion if one just applied Arnoldi, but it is very special, actually an instance of a vector for which Lanczos converges slowly, see [20]!

Let us finally look at TRQ with a basis size of $j = 5$. We started to build up the basis by inverse iteration Arnoldi, then we continued for in all 15 steps, updating the basis. Let us list what happened:

```
Backwards Arnoldi TRQ
  j jsh Shift        Eigenvalue appr  omega     resnorm  k(h)
  1  1  0.00000e+00  1.9033744216e+00 4.93e-01 1.15e-15 7.31e-01
  2  1  0.00000e+00  9.9380121840e-02 4.61e-01 1.78e-15 1.25e+00
  3  1  0.00000e+00  7.0526385436e-02 4.15e-01 3.25e-15 1.22e+00
  4  1  0.00000e+00  6.7712802632e-02 1.89e-01 4.39e-15 8.71e-01
  5  1  0.00000e+00  6.7247323127e-02 4.86e-02 4.99e-15 1.37e+00
  5  1  6.72473e-02  6.7247323127e-02 1.25e-07 1.87e-13 1.13e+02
  5  1  6.72412e-02  6.7247323127e-02 1.55e-21 1.18e-08 6.06e+06
  5  2  1.04915e-01  1.0566480579e-01 9.37e-09 5.12e-07 2.52e+03
  5  2  1.04915e-01  1.0566480579e-01 9.67e-14 1.44e-07 2.95e-01
  5  3  1.36305e-01  2.1242421703e-01 1.63e-08 1.14e-05 8.91e+01
  5  3  1.36305e-01  2.1242421703e-01 7.98e-13 7.39e-07 1.11e-01
  5  4  2.02902e-01  8.5106500479e-01 7.48e-07 4.03e-06 5.42e+00
  5  4  2.02795e-01  8.5106500479e-01 7.82e-14 1.22e-04 1.01e+03
  5  5  2.20272e-01  4.3996828376e+00 3.29e-09 7.47e-08 2.68e-06
  5  5  2.20272e-01  4.3996828376e+00 5.11e-14 7.47e-08 7.59e-05
```

Also here we get deterioration in the discrepancy, but to a somewhat smaller extent than for RKS. I have yet to compare this to the original code given in [22], this discrepancy was not indicated there.

Look at some $K$ matrices, first after 6 steps when the first eigenvalue has just converged, and then at end after 15 steps, when the bottom element is just about negligible, signalling that all the 5 eigenvalues have converged:

```
k =
   6.7241e-02   9.8288e-12
  -1.0990e-19   1.0499e-01   2.3735e-03
            0   2.3735e-03   1.9546e-01   2.5507e-02
            0            0   2.5507e-02   1.6484e-01   6.3396e-02
            0            0            0   6.3396e-02   4.8926e-01
            0            0            0            0  -2.4276e-01

k =
   6.7241e-02  -4.7107e-09
  -5.2057e-10   1.8785e-01   3.5221e-02
            0   3.5221e-02   1.1989e-01   1.2474e-03
            0            0   1.2474e-03   2.0753e-01   3.0100e-02
            0            0            0   3.0100e-02   1.4901e-01
            0            0            0            0  -1.4115e-12
```

# 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. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quarterly of Applied Mathematics, 9 (1951), pp. 17–29.

[3] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, Journal of Computational Physics, 17 (1975), pp. 87–94.

[4] C. de Boor and G. H. Golub, *The numerically stable reconstruction of a Jacobi matrix from spectral data*, Lin. Alg. Appl., 21 (1978), pp. 245–260.

[5] D. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst, *Jacobi-Davidson style QR and QZ algorithms for the partial reduction of matrix pencils*, SIAM J. Sci. Comput., (1997), p. to appear.

[6] G. H. Golub, R. Underwood, and J. H. Wilkinson, *The Lanczos algorithm for the symmetric $Ax = \lambda Bx$ problem*, Tech. Rep. STAN-CS-72-270, Computer Science, Stanford University, Stanford, CA, 1972.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 2nd ed., 1989.

[8] W. B. Gragg and W. J. Harrod, *The numerically stable reconstruction of Jacobi matrices from spectral data*, Numerische Mathematik, 44 (1984), pp. 317–336.

[9] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards, 45 (1950), pp. 255–282.

[10] R. Lehoucq and K. Meerbergen, *The inexact rational Krylov sequence method*, SIAM J. Matr. Anal. Appl., 17 (1997), p. to appear.

[11] R. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matr. Anal. Appl., 17 (1996), pp. 789–821.

[12] R. B. Morgan and D. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 817–825.

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

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

[15] ——, *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.

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

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

[18] ——, *Rational Krylov, a practical algorithm for large sparse nonsymmetric matrix pencils*, Tech. Rep. UCB/CSD-95-871, Dept Computer Science, University of California Berkeley, accepted for publication in SISC, 1995.

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

[20] D. S. Scott, *How to make the Lanczos algorithm converge slowly*, Mathematics of Computation, 33 (1979), pp. 239–247.

[21] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matr. Anal. Appl., 17 (1996), pp. 401–425.

[22] D. C. Sorensen and C. Yang, *A truncated RQ-iteration for large scale eigenvalue calculations*, SIAM J. Matr. Anal. Appl., 17 (1997), p. to appear.