A Rational Krylov Toolbox for MATLAB

Berljafa, Mario and Güttel, Stefan

2014

MIMS EPrint: **2014.56**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

A Rational Krylov Toolbox for MATLAB

Mario Berljafa* Stefan Güttel*

December 9, 2014

Contents

1 Overview and download

In its latest version the Rational Krylov Toolbox [1] contains

- an implementation of Ruhe's rational Krylov sequence method [5],
- algorithms for the implicit and explicit relocation of the poles of a rational Krylov space [2], and
- an implementation of RKFIT [2,3], a robust algorithm for rational least squares fitting.

It can be downloaded from

`http://guettel.com/rktoolbox/rktoolbox.zip`

*School of Mathematics, The University of Manchester, Alan Turing Building, Oxford Road, M13 9PL Manchester, United Kingdom, `m.berljafa@maths.man.ac.uk`, `stefan.guettel@manchester.ac.uk`

To install simply unpack the zip file and add the folder to the Matlab path.

Alternatively one can copy and paste the following two lines to the MATLAB command window. It will download and unzip the toolbox into the current MATLAB directory, and attempt to add it to the path:

```
unzip('http://guettel.com/rktoolbox/rktoolbox.zip');
addpath(fullfile(cd, 'rktoolbox')); savepath
```

2 Computing a rational Krylov basis

Relevant functions: `rat_krylov`

A rational Krylov space is a linear space of rational functions in a matrix times a vector. Let A be a square matrix, $\boldsymbol{v}$ a starting vector of the same dimension, and let $\xi_1, \xi_2, \ldots, \xi_m$ be a sequence of complex or infinite *poles* all distinct from the eigenvalues of A. Then the rational Krylov space of order $m+1$ associated with $A, \boldsymbol{v}, \xi_j$ is defined as

$$\mathcal{Q}_{m+1}(A, \boldsymbol{v}) = [q_m(A)]^{-1}\text{span}\{\boldsymbol{v}, A\boldsymbol{v}, \ldots, A^m\boldsymbol{v}\},$$

where $q_m(z) = \prod_{j=1,\xi_j\neq\infty}^{m}(z-\xi_j)$ is the common denominator of the rational functions associated with the rational Krylov space. The rational Krylov sequence method by Ruhe [5] computes an orthonormal basis V_{m+1} of $\mathcal{Q}_{m+1}(A, \boldsymbol{v})$. The basis matrix V_{m+1} satisfies a rational Arnoldi decomposition of the form

$$AV_{m+1}\underline{K_m} = V_{m+1}\underline{H_m},$$

where $(\underline{H_m}, \underline{K_m})$ is an (unreduced) upper Hessenberg pencil of size $(m+1)\times m$.

Example: Let us compute V_{m+1}, $\underline{K_m}$, and $\underline{H_m}$ using the function `rat_krylov`, and verify that the outputs satisfy the rational Arnoldi decomposition by computing the relative residual norm $\|AV_{m+1}\underline{K_m} - V_{m+1}\underline{H_m}\|_2/\|\underline{H_m}\|_2$. The matrix A will be the `tridiag` matrix of size 200 from MATLAB's `gallery`, $\boldsymbol{v} = [1, 0, \ldots, 0]^T$, and the $m = 5$ poles ξ_j are $-1, \infty, -i, 0, +i$.

```
N = 100;                                % matrix size
A = gallery('tridiag', N);
v = zeros(N, 1); v(1) = 1;              % starting vector
poles = [-1, inf, -1i, 0, 1i];          % m = 5 poles
```

```
[V, K, H] = rat_krylov(A, v, poles);
resnorm = norm(A*V*K - V*H)/norm(H)  % residual check

resnorm =
   4.8250e-15
```

As some of the poles ξ_j in this example are complex, the matrices V_{m+1}, $\underline{K_m}$, and $\underline{H_m}$ will be complex, too:

```
realcheck = [isreal(V), isreal(K), isreal(H)]

realcheck =
     0     0     0
```

However, the poles ξ_j can be reordered to appear in complex conjugate pairs using the function `cplxsort`. After reordering the poles we can call the function `rat_krylov` with the `'real'` option, thereby computing a real rational Arnoldi decomposition:

```
% group together complex-conjugate pairs
poles = cplxsort(poles);
[V, K, H] = rat_krylov(A, v, poles, 'real');
resnorm = norm(A*V*K - V*H)/norm(H)  % residual check
realcheck = [isreal(V), isreal(K), isreal(H)]

resnorm =
   6.1896e-15
realcheck =
     1     1     1
```

Rational Arnoldi decompositions are useful for several purposes. For example, the eigenvalues of the upper $m \times m$ part of the pencil $(\underline{H_m}, \underline{K_m})$ can be excellent approximations to some of A's eigenvalues [5]. Other applications include matrix function approximation and rational quadrature, model order reduction, matrix equations, and rational least squares fitting (see below).

The implementation of `rat_krylov` also supports matrix pairs (A, B) leading to decompositions of the form $AV_{m+1}\underline{K_m} = BV_{m+1}\underline{H_m}$. It is also possible to provide a user-specified inner product and linear system solver via an optional `param` structure. For more details type `help rat_krylov`.

3 Moving poles of a rational Krylov space

Relevant functions: `move_poles_expl, move_poles_impl`

There is a direct link between the starting vector $\boldsymbol{v}$ and the poles ξ_j of a rational Krylov space $\mathcal{Q}_{m+1}$. A change of the poles ξ_j to $\breve{\xi}_j$ can be interpreted as a change

of the starting vector from $\boldsymbol{v}$ to $\breve{\boldsymbol{v}}$, and vice versa. Algorithms for moving the poles of a rational Krylov space are described in [2] and implemented in the functions `move_poles_expl` and `move_poles_impl`.

Example: Let us move the $m = 5$ poles ξ_j from the above example to $\breve{\xi}_j = -j$, $j = 1, 2, \ldots, 5$.

```
poles_new = -1:-1:-5;
[KT, HT, QT, ZT] = move_poles_expl(K, H, poles_new);
```

The poles of a rational Krylov space are the eigenvalues of the lower $m \times m$ part of the pencil $(\underline{\breve{H}_m}, \underline{\breve{K}_m})$ in a rational Arnoldi decomposition $A\breve{V}_{m+1}\underline{\breve{K}_m} = \breve{V}_{m+1}\underline{\breve{H}_m}$ associated with that space [2]. By transforming a rational Arnoldi decomposition we are therefore effectively moving the poles:

```
VT = V*QT';
resnorm = norm(A*VT*KT - VT*HT)/norm(H)  % residual check
moved_poles = rat_poles(HT, KT).'
```

```
resnorm =
   1.1813e-14
moved_poles =
  -1.0000e+00 - 3.4927e-17i
  -2.0000e+00 - 8.2510e-16i
  -3.0000e+00 - 5.5969e-16i
  -4.0000e+00 + 1.5260e-15i
  -5.0000e+00 - 7.6765e-16i
```

4 Rational least squares fitting

Relevant function: `rkfit`

RKFIT is an algorithm for rational least squares fitting based on rational Krylov spaces, see [2,3]. Given matrices A and F and a vector $\boldsymbol{v}$, RKFIT attempts to find a rational function $r_m(z)$ of type (m, m) such that

$$\|F\boldsymbol{v} - r_m(A)\boldsymbol{v}\|_2 \to \min.$$

Clearly, if A is a diagonal matrix then this minimization is equivalent to a weighted discrete rational least squares problem on the eigenvalues of A, but RKFIT can handle general matrices. For example, if A has nontrivial Jordan blocks then one would also fit the derivative of $r_m(z)$ at (some of) the eigenvalues of A.

Example: Consider again the tridiagonal matrix A and the vector $\boldsymbol{v}$ from above and let $F = A^{1/2}$.

```
F = sqrtm(full(A));
exact = F*v;
```

Now let us find a rational function $r_m(z)$ of degree $m = 10$ such that $\|F\boldsymbol{v} - r_m(A)\boldsymbol{v}\|_2$ is minimal. The function `rkfit` requires an input vector of m initial poles and then tries to return an improved set of poles. If we had no clue about where to place the initial poles we can easily set them all to infinity. In the following we run RKFIT 5 times and display the relative misfit $\|F\boldsymbol{v} - r_m(A)\boldsymbol{v}\|_2/\|F\boldsymbol{v}\|_2$ after each iteration:

```
poles = inf*ones(1, 10);
for iter = 1:5
    [poles, ratfun, misfit] = rkfit(F, A, v, poles, 'real');
    rel_misfit = misfit/norm(exact);
    disp(sprintf('iter %d: %e',[iter rel_misfit]))
end
```

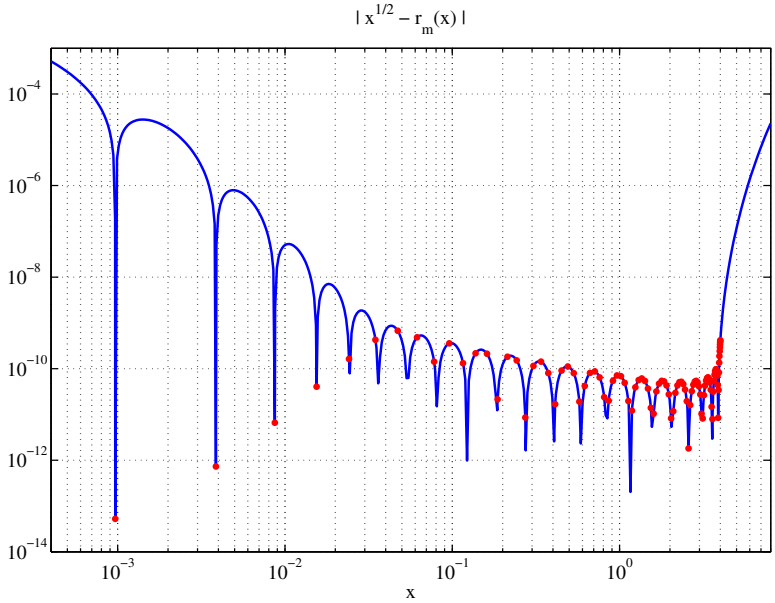
```
iter 1: 7.055604e-07
iter 2: 1.410851e-10
iter 3: 4.632047e-11
iter 4: 4.563095e-11
iter 5: 4.573434e-11
```

Apparently the rational function $r_m(A)\boldsymbol{v}$ of degree 10 approximates $A^{1/2}\boldsymbol{v}$ to about 10 decimal places. A useful output of `rkfit` is the function handle `ratfun`, which allows one to evaluate the rational function $r_m(z)$. There are two ways this function handle can be used:

- `ratfun(A,v)` evaluates $r_m(A)\boldsymbol{v}$ as a matrix function times a vector, or
- `ratfun(z)` evaluates $r_m(z)$ as a scalar function in the complex plane.

For example, here is a plot of the error $|x^{1/2} - r_m(x)|$ over the spectral interval of A (approximately $[0, 4]$), together with the values at the eigenvalues of A:

```
figure
ee = eig(full(A)).';
xx = sort([logspace(-4.3, 1, 500) , ee]);
loglog(xx,abs(sqrt(xx) - ratfun(xx))); hold on
loglog(ee,abs(sqrt(ee) - ratfun(ee)), 'r.')
axis([4e-4, 8, 1e-14, 1e-3]); xlabel('x'); grid on
title('| x^{1/2} - r_m(x) |','interpreter','tex')
```



As expected the rational function $r_m(z)$ is a good approximation of the square root over $[0, 4]$. It is, however, not a uniform approximation because we are minimizing a least squares error on the eigenvalues of A, and moreover we are implicitly using a weight function given by the components of $\boldsymbol{v}$ in A's eigenvector basis.

Example: RKFIT can also handle nonnormal matrices A and F. Let us find a rational function $r_m(z)$ of degree $m = 16$ which gives an optimal least-squares approximation to the matrix exponential times vector of the Grcar matrix:

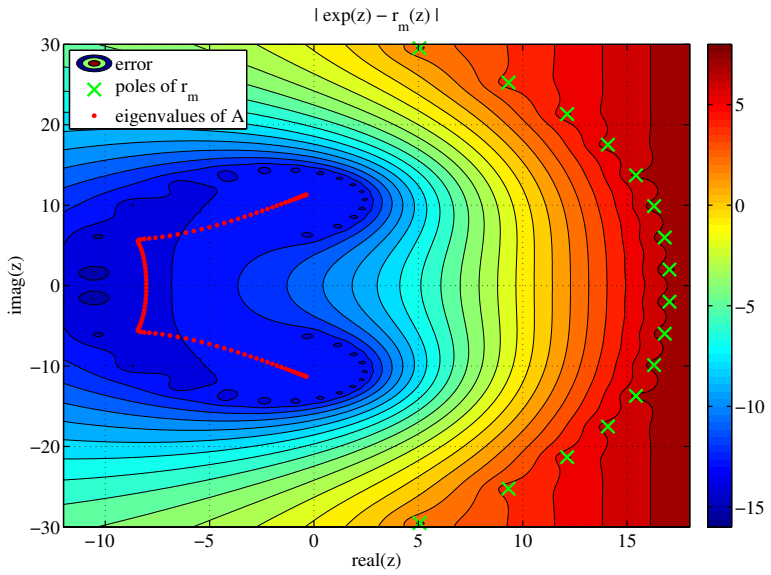
```
N = 100;
A = -5*gallery('grcar',N,3);
v = ones(N,1);
F = expm(A); exact = F*v;
poles = inf*ones(1, 16);
for iter = 1:5
    [poles, ratfun, misfit] = rkfit(F, A, v, poles, 'real');
    rel_misfit = misfit/norm(exact);
    disp(sprintf('iter %d: %e',[iter rel_misfit]))
end
```

```
iter 1: 1.814195e-11
iter 2: 6.863362e-13
iter 3: 6.843369e-13
iter 4: 6.843162e-13
iter 5: 6.835966e-13
```

RKFIT seems to have found a minimum after 2 iterations. Here is a contour

plot of the error of $r_m(z)$ as an approximation to $\exp(z)$, together with the poles of r_m and the eigenvalues of A:

```
[X,Y] = meshgrid(linspace(-12,18,300),linspace(-30,30,300));
Z = X + 1i*Y; E = exp(Z) - ratfun(Z);
figure
contourf(X,Y,log10(abs(E)),linspace(-16,8,25)); hold on
plot(poles,'gx'); plot(eig(full(A)),'r.','MarkerSize',12)
grid on; xlabel('real(z)'); ylabel('imag(z)'); colorbar
title('| exp(z) - r_m(z) |','interpreter','tex')
legend('error','poles of r_m','eigenvalues of A','Location','NorthWest')
```



5 Planned features

This Rational Krylov Toolbox is under continuous development and new features will be added over time. Here is our current todo list:

- Make `move_poles_expl` preserve real pencils with complex conjugate eigenpairs.
- Add matrix function codes to the toolbox, like `invsqrtmv` and `logmv` currently available at http://guettel.com/markovfunmv/.
- Add more unit tests for all functionalities.
- Add an optional waitbar to `rat_krylov`.

This guide has been created on

```
disp(date)
```

6 Acknowledgments

This documentation was generated using MATLAB's `publish` command. The convenient 2-line Matlab code for automated download and installation of this toolbox was adopted from a similar code on the Chebfun website.

7 References

[1] M. Berljafa and S. Güttel, *A Rational Krylov Toolbox for MATLAB*, MIMS EPrint 2014.56, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2014. Available at http://eprints.ma.man.ac.uk/2199/.

[2] M. Berljafa and S. Güttel, *Generalized rational Krylov decompositions with an application to rational approximation*, MIMS EPrint 2014.59, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2014. Available at http://eprints.ma.man.ac.uk/2198/.

[3] M. Berljafa and S. Güttel, *RKFIT: A robust algorithm for rational least squares fitting*, in preparation.

[4] S. Güttel and L. Knizhnerman, *A black-box rational Arnoldi variant for Cauchy–Stieltjes matrix functions*, BIT Numer. Math., 53(3):595–616, 2013.

[5] A. Ruhe, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sci. Comput., 19(5):1535–1551, 1998.