A Rational Krylov Toolbox for MATLAB

Berljafa, Mario and Elsworth, Steven 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 Steven Elsworth Stefan Güttel*

November 15, 2017

Contents

1 Overview

Thank you for your interest in the Rational Krylov Toolbox (RKToolbox). The RKToolbox is a collection of scientific computing tools based on rational Krylov techniques. The development started in 2013 and the current version 2.7 provides

- an implementation of Ruhe's (block) rational Krylov sequence method [8, 9], allowing to control various options, including user-defined inner products, exploitation of complex-conjugate shifts, orthogonalization, rerunning [3], and parallelism [4],
- algorithms for the implicit and explicit relocation of the poles of a rational Krylov space [2],
- a collection of utility functions and a gallery of special rational functions (e.g., Zolotarev approximants),
- an implementation of RKFIT [2, 3], a robust algorithm for approximate rational least squares approximation, including automated degree reduction,

*School of Mathematics, The University of Manchester, Alan Turing Building, Oxford Road, M13 9PL Manchester, United Kingdom, Correspondence: stefan.guettel@manchester.ac.uk

- the RKFUN class [3] for numerical computations with rational functions, including support for MATLAB Variable Precision Arithmetic and the Advanpix Multiple Precision toolbox [1], and
- the RKFUNM class, a matrix-valued generalization of RKFUNs, together with the ability to sample and solve nonlinear eigenvalue problems using the NLEIGS [6] and AAA algorithms [7].

This guide explains the main functionalities of the toolbox. To run the embedded MATLAB codes the RKToolbox needs to be in MATLAB's search path. For details about the installation we refer to the **Download** section on http://rktoolbox.org/.

2 Rational Krylov spaces

A rational Krylov space is a linear vector space of rational functions in a matrix times a vector. Let A be a square matrix of size $N \times N$, $\boldsymbol{b}$ an $N \times 1$ starting vector, 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 $A, \boldsymbol{b}, \xi_j$ is defined as

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

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 method by Ruhe [5, 6] computes an orthonormal basis V_{m+1} of $\mathcal{Q}_{m+1}(A, \boldsymbol{b}, q_m)$. 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 $(m+1) \times m$.

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 [8, 9]. Other applications include matrix function approximation and rational quadrature, model order reduction, matrix equations, nonlinear eigenproblems, and rational least squares fitting (see below).

3 Computing rational Krylov bases

Relevant functions: `rat_krylov`, `util_cplxpair`

Let us compute V_{m+1}, $\underline{K_m}$, and $\underline{H_m}$ using the `rat_krylov` function, and verify that the outputs satisfy the rational Arnoldi decomposition by computing the relative residual norm $\|AV_{m+1}\underline{K_m} - V_{m+1}\underline{H_m}\|_2/\|\underline{H_m}\|_2$. For A we take the `tridiag` matrix of size 100 from MATLAB's `gallery`, and $\boldsymbol{b} = [1, 0, \ldots, 0]^T$. The $m = 5$ poles ξ_j are, in order, $-1, \infty, -\mathrm{i}, 0, \mathrm{i}$.

```
N   = 100;                                  % matrix size
A   = gallery('tridiag', N);
b   = eye(N, 1);                            % starting vector
xi  = [-1, inf, -1i, 0, 1i];                % m = 5 poles
[V, K, H] = rat_krylov(A, b, xi);
resnorm = norm(A*V*K - V*H)/norm(H)         % residual check
```

```
resnorm =
   4.5714e-15
```

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

```
disp([isreal(V), isreal(K), isreal(H)])
```

```
     0     0     0
```

However, the poles ξ_j can be reordered so that complex-conjugate pairs appear next to each other using the function `util_cplxpair`. After reordering the poles, we can call the function `rat_krylov` with the `'real'` option, thereby computing a real-valued rational Arnoldi decomposition [9].

```
% Group together poles appearing in complex-conjugate pairs.
xi = util_cplxpair(xi);
[V, K, H] = rat_krylov(A, b, xi, 'real');
resnorm = norm(A*V*K - V*H)/norm(H)
disp([isreal(V), isreal(K), isreal(H)])
```

```
resnorm =
   6.4123e-15
     1     1     1
```

Our implementation `rat_krylov` supports many features not shown in the basic description above.

- It is possible the use matrix pencils (A, B) instead of a single matrix A. This leads to decompositions of the form $AV_{m+1}\underline{K_m} = BV_{m+1}\underline{H_m}$.
- Both the matrix A and the pencil (A, B) can be passed either explicitly, or implicitly by providing function handles to perform matrix-vector products and to solve shifted linear systems.
- Non-standard inner products for constructing the orthonormal bases are supported.
- One can choose between CGS and MGS with or without reorthogonalization.
- Iterative refinement for the linear system solves is supported.
- Block rational Krylov spaces.

For more details type `help rat_krylov`.

4 Moving poles of a rational Krylov space

Relevant functions: `move_poles_expl`, `move_poles_impl`

There is a direct link between the starting vector $\boldsymbol{b}$ 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 $\boldsymbol{b}$ to $\breve{\boldsymbol{b}}$, 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 $-1, \infty, -\mathrm{i}, 0$, and i into $\breve{\xi}_j = -j$, $j = 1, 2, \ldots, 5$.

```
N  = 100;
A  = gallery('tridiag', N);
b  = eye(N, 1);
xi = [-1, inf, -1i, 0, 1i];
[V, K, H] = rat_krylov(A, b, xi);
xi_new = -1:-1:-5;
[KT, HT, QT, ZT] = move_poles_expl(K, H, xi_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(HT)
moved_poles = util_pencil_poles(KT, HT).'
```

```
resnorm =
   5.8986e-15
moved_poles =
  -1.0000e+00 + 2.0951e-16i
  -2.0000e+00 - 2.9568e-15i
  -3.0000e+00 - 4.4556e-16i
  -4.0000e+00 - 9.8632e-15i
  -5.0000e+00 - 9.4451e-15i
```

5 Rational Krylov fitting (RKFIT)

Relevant function: `rkfit`

RKFIT [2, 3] is an iterative Krylov-based algorithm for nonlinear rational approximation. Given two families of $N \times N$ matrices $\{F^{[j]}\}_{j=1}^{\ell}$ and $\{D^{[j]}\}_{j=1}^{\ell}$, an $N \times n$ block of vectors B, and an $N \times N$ matrix A, the algorithm seeks a family of rational functions $\{r^{[j]}\}_{j=1}^{\ell}$ of type $(m+k, m)$, all sharing a common denominator q_m, such that the *relative misfit*

$$\text{misfit} = \sqrt{\frac{\sum_{j=1}^{\ell} \|D^{[j]}[F^{[j]}B - r^{[j]}(A)B]\|_F^2}{\sum_{j=1}^{\ell} \|D^{[j]}F^{[j]}B\|_F^2}} \to \min$$

is minimal. The matrices $\{D^{[j]}\}_{j=1}^{\ell}$ are optional, and if not provided $D^{[j]} = I_N$ is assumed. The algorithm takes an initial guess for q_m and iteratively tries to improve it by relocating the poles of a rational Krylov space.

We now show on a simple example how to use the `rkfit` function. Consider again the tridiagonal matrix A and the vector $\boldsymbol{b}$ from above and let $F = A^{1/2}$.

```
N  = 100;
A  = gallery('tridiag', N);
```

```
b   = eye(N, 1);
F   = sqrtm(full(A));
exact = F*b;
```

Now let us find a rational function $r_m(z)$ of type (m, m) with $m = 10$ such that $\|F\boldsymbol{b} - r_m(A)\boldsymbol{b}\|_2/\|F\boldsymbol{b}\|_2$ is small. 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 for at most 15 iterations and aim at relative misfit $\|F\boldsymbol{b} - r_m(A)\boldsymbol{b}\|_2/\|F\boldsymbol{b}\|_2$ below 10^{-10}. We display the error after each iteration.

```
[xi, ratfun, misfit] = rkfit(F, A, b, ...
                             repmat(inf, 1, 10), ...
                             15, 1e-10, 'real');

disp(misfit)
```

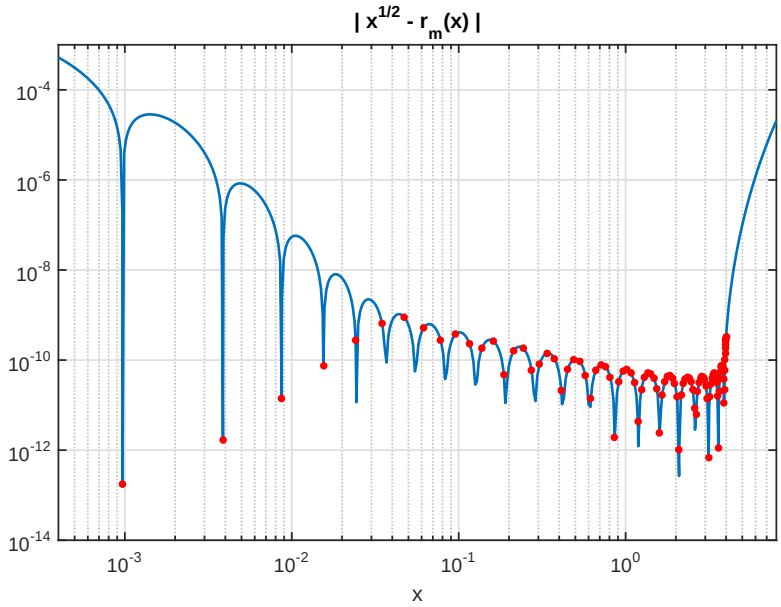
```
   7.0889e-07   1.4498e-10   4.6348e-11
```

The rational function $r_m(A)\boldsymbol{b}$ of type $(10, 10)$ approximates $A^{1/2}\boldsymbol{b}$ to about 10 decimal places. A useful output of `rkfit` is the RKFUN object `ratfun` representing the rational function r_m. It can be used, for example, to evaluate $r_m(z)$:

- `ratfun(A,v)` evaluates $r_m(A)\boldsymbol{v}$ as a matrix function times a vector,
- `ratfun(A,V)` evaluates $r_m(A)V$ as a matrix function times a matrix, e.g., setting $V = I$ as the identity matrix will return the full matrix function $r_m(A)$, or
- `ratfun(z)` evaluates $r_m(z)$ as a scalar function in the complex plane.

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.', 'markers', 15)
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 approximately minimizing the 2-norm error on the eigenvalues of A, and moreover we are implicitly using a weight function given by the components of $\boldsymbol{b}$ in A's eigenvector basis.

Additional features of RKFIT are listed below.

- An automated degree reduction procedure [3, Section 4] is implemented; it takes place if a relative misfit below tolerance is achieved, unless deactivated.
- Nondiagonal rational approximants are supported; they can be specified via an additional `param` structure.
- Utility functions are provided for transforming scalar data appearing in complex-conjugate pairs into real-valued data, as explained in [3, Section 3.5].

For more details type `help rkfit`. Some of the capabilities of `RKFUN` are shown in the following section.

6 The RKFUN class

RKFUN is the fundamental data type to represent and work with rational functions. It has already been described above how to evaluate an RKFUN object `ratfun` for scalar or matrix arguments by calling `ratfun(z)` or `ratfun(A,v)`, respectively. There are more than 30 RKFUN methods implemented, and a list of these can be obtained by typing `methods rkfun`:

```
basis          - Orthonormal rational basis functions of an RKFUN.
coeffs         - Expansion coefficients of an RKFUN.
contfrac       - Convert an RKFUN into continued fraction form.
diff           - Differentiate an RKFUN.
disp           - Display information about an RKFUN.
double         - Convert an RKFUN into double precision (undo vpa or mp).
ezplot         - Easy-to-use function plotter for RKFUNs.
```

```
feval          - Evaluate an RKFUN at scalar or matrix arguments.
hess           - Convert an RKFUN pencil to (strict) upper-Hessenberg form.
inv            - Invert an RKFUN corresponding to a Moebius transform.
isreal         - Returns true if an RKFUN is real-valued.
minus          - Scalar subtraction.
mp             - Convert an RKFUN into Advanpix Multiple Precision format.
mrdivide       - Scalar division.
mtimes         - Scalar multiplication.
plus           - Scalar addition.
poles          - Return the poles of an RKFUN.
poly           - Convert an RKFUN into a quotient of two polynomials.
power          - Integer exponentiation of an RKFUN.
rdivide        - Division of two RKFUN.
residue        - Convert an RKFUN into partial fraction form.
rkfun          - The RKFUN constructor.
roots          - Compute the roots of an RKFUN.
size           - Returns the size of an RKFUN.
subsref        - Evaluate an RKFUN (calls feval).
times          - Multiplication of two RKFUNs.
type           - Return the type (m+k,m) of an RKFUN.
uminus         - Unary minus.
uplus          - Unary plus.
vpa            - Convert RKFUN into MATLAB's variable precision format.
```

The names of these methods should be self-explanatory. For example, `roots(ratfun)` will return the roots of `ratfun`, and `residue(ratfun)` will compute its partial fraction form. Most methods support the use of MATLAB's Variable Precision Arithmetic (VPA) and, preferably, the Advanpix Multiple Precision toolbox (MP). So, for example, `contfrac(mp(ratfun))` will compute a continued fraction expansion of `ratfun` using multiple precision arithmetic. For more details on each of the methods, type `help rkfun.<method name>`. The RKFUN gallery provides some predefined rational functions that may be useful. A list of the options can be accessed as follows:

```
help rkfun.gallery
```

```
  GALLERY    Collection of rational functions.

  obj = rkfun.gallery(funname, param1, param2, ...) takes
  funname, a case-insensitive string that is the name of
  a rational function family, and the family's input
  parameters.

  See the listing below for available function families.

  constant   Constant function of value param1.
  cheby      Chebyshev polynomial (first kind) of degree param1.
  cayley     Cayley transformation (1-z)/(1+z).
  moebius    Moebius transformation (az+b)/(cz+d) with
             param1 = [a,b,c,d].
  sqrt       Zolotarev sqrt approximation of degree param1 on
```

```
                the positive interval [1,param2].
  invsqrt       Zolotarev invsqrt approximation of degree param1 on
                the positive interval [1,param2].
  sqrt0h        balanced Remez approximation to sqrt(x+(h*x/2)^2)
                of degree param3 on [param1,param2],
                where param1 <= 0 <= param2 and h = param4.
  sqrt2h        balanced Zolotarev approximation to sqrt(x+(hx/2)^2)
                of degree param5 on [param1,param2]U[param3,param4],
                param1 < param2 < 0 < param3 < param4, h = param6.
  invsqrt2h     balanced Zolotarev approximation to 1/sqrt(x+(hx/2)^2)
                of degree param5 on [param1,param2]U[param3,param4],
                param1 < param2 < 0 < param3 < param4, h = param6.
  sign          Zolotarev sign approximation of degree 2*param1 on
                the union of [1,param2] and [-param2,-1].
  step          Unit step function approximation for [-1,1] of
                degree 2*param1 with steepness param2.
```

Another way to create an RKFUN is to make use of MATLAB's symbolic engine. For example, `r = rkfun('(x+1)*(x-2)/(x-3)^2')` returns a rational function as expected. Alternatively, one can specify a rational function by its roots and poles (and an optional scaling factor) using the `rkfun.nodes2rkfun` function. For example, `r = rkfun.nodes2rkfun([-1,2],[3,3])` will create the same rational function as above. One can also specify a rational interpolant by its barycentric representation; see the function `util_bary2rkfun` and reference [5].

7 The RKFUNM class

The RKFUNM class is the matrix-valued generalization of RKFUN. RKFUNM objects are mainly generated via the `util_nleigs` and `util_aaa` sampling routines for nonlinear eigenvalue problems. The class provides a method called `linearize`, which returns a matrix pencil structure corresponding to a linearization of the RKFUNM. This pencil can be used in combination with the `rat_krylov` function for finding eigenvalues of the linearization. We illustrate the capabilities of the RKFUNM class with the help of a simple example. Assume we want to find numbers z in the interval $[-\pi, \pi]$ where the 2×2 matrix $F(z)$ defined below becomes singular:

```
F = @(z) [ sin(5*z) , 1 ; 1 , 1 ];
```

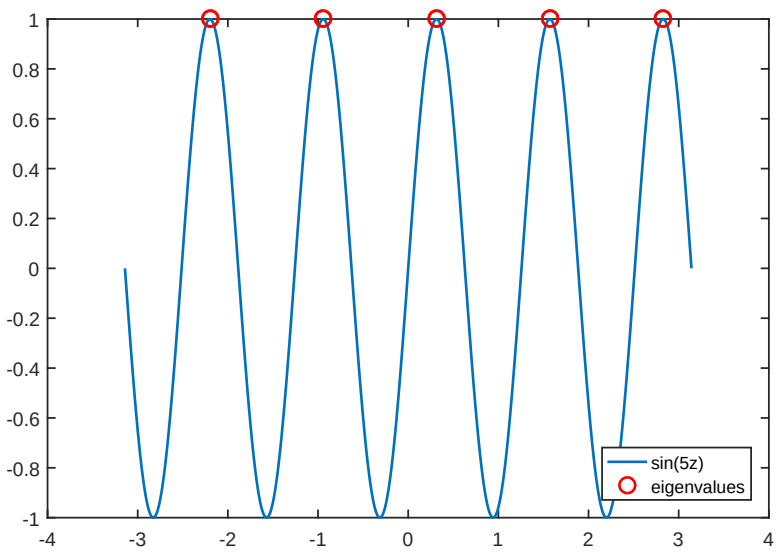
With the help of the AAA algorithm [7] we sample this matrix at sufficiently many points in the search interval and construct an accurate rational interpolant, which is then converted into RKFUNM format; see [7] for details on the conversion:

```
Z = linspace(-pi,pi,500);
ratm = util_aaa(F,Z)
```

```
ratm =
        RKFUNM object of size 2-by-2 and type (21, 21).
        Real dense coefficient matrices of size 2-by-2.
        Real-valued Hessenberg pencil (H, K) of size 22-by-21.
```

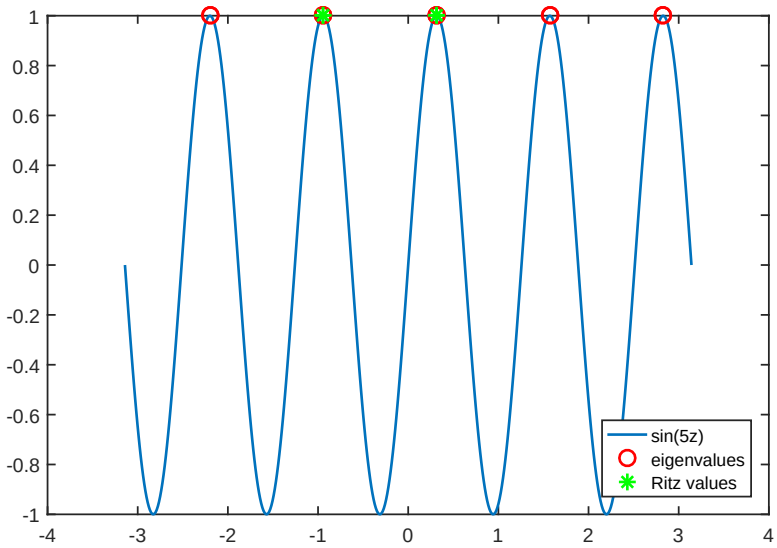
We see that `ratm` is indeed an RKFUNM object representing a matrix-valued rational function of degree 21. In order to solve the nonlinear eigenvalue problem $F(z)v = 0$, $v \neq 0$, we have to linearize `ratm` and find the eigenvalues of the linearization near the search interval:

```
AB = linearize(ratm);
[A,B] = AB.get_matrices();
evs = eig(full(A), full(B));
evs = evs(abs(imag(evs))< 1e-7 & abs(evs)<pi);
figure; plot(Z,sin(5*Z));
hold on; h2 = plot(real(evs),0*evs + 1,'ro');
legend('sin(5z)','eigenvalues','Location','SouthEast')
```



Indeed, the solutions of $\sin(5z) = 1$ are the points where $F(z)$ becomes singular. For linearizations of larger dimension the pencil `AB` should not be converted into matrix form using `AB.get_matrices()`. Instead, `AB` should be used as input to `rat_krylov` for solving the linear eigenvalue problem iteratively:

```
xi = zeros(1,25); % shifts for the Krylov space
[m,n] = type(ratm);
dimlin = m*size(ratm,1); % dimension of linearization
v = randn(dimlin, 1);    % starting vector of Krylov space
[V, K, H] = rat_krylov(AB, v, xi);
[X, D] = eig(H(1:end-1, :), K(1:end-1, :));
ritzval = diag(D); % Ritz values
ritzval = ritzval(abs(imag(ritzval))< 1e-7 & abs(ritzval)<pi);
plot(ritzval, 0*ritzval + 1, 'g*');
legend('sin(5z)','eigenvalues',...
    'Ritz values','Location','SouthEast')
```



8 References

[1] Advanpix LLC., *Multiprecision Computing Toolbox for MATLAB,* version 4.3.3.12213, Tokyo, Japan, 2017. http://www.advanpix.com/.

[2] M. Berljafa and S. Güttel. *Generalized rational Krylov decompositions with an application to rational approximation,* SIAM J. Matrix Anal. Appl., 36(2):894–916, 2015.

[3] M. Berljafa and S. Güttel. *The RKFIT algorithm for nonlinear rational approximation,* SIAM J. Sci. Comput., 39(5):A2049–A2071, 2017.

[4] M. Berljafa and S. Güttel, *Parallelization of the rational Arnoldi algorithm,* SIAM J. Sci. Comput., 39(5):S197–S221, 2017.

[5] S. Elsworth and S. Güttel, *Conversions between barycentric, RKFUN, and Newton representations of rational interpolants,* MIMS Eprint 2017.40 (http://eprints.maths.manchester.ac.uk/2593/), Manchester Institute for Mathematical Sciences, The University of Manchester, UK, 2017.

[6] S. Güttel, R. Van Beeumen, K. Meerbergen, and W. Michiels, *NLEIGS: A class of fully rational Krylov methods for nonlinear eigenvalue problems,* SIAM J. Sci. Comput., 36(6):A2842–A2864, 2014.

[7] Y. Nakatsukasa, O. Sète, and L. N. Trefethen, *The AAA algorithm for rational approximation,* arXiv preprint arXiv:1612.00337, 2016.

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

[9] A. Ruhe. *The rational Krylov algorithm for nonsymmetric eigenvalue problems. III: Complex shifts for real matrices,* BIT, 34:165–176, 1994.