Moving the poles of a rational Krylov space

Mario Berljafa        Stefan Güttel

June 2015

Contents

1 Rational Krylov spaces 1
2 The poles of a rational Krylov space 1
3 Moving the poles explicitly 2
4 Moving the poles implicitly 3
5 Some fun with moving poles 3
6 References 6

1 Rational Krylov spaces

A rational Krylov space is a linear vector space of rational functions in a matrix times a vector [5]. Let $A$ be a square matrix of size $N \times N$, $b$ an $N \times 1$ nonzero 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 $A$, $b$, $\xi_j$ is defined as

$$Q_{m+1} \equiv Q_{m+1}(A, b, q_m) = q_m(A)^{-1}\text{span}\{b, Ab, \ldots, A^m b\},$$

where $q_m(z) = \prod_{j=1}^{m} (z - \xi_j)$ is the common denominator of the rational functions associated with $Q_{m+1}$. The rational Krylov sequence method by Ruhe [5] computes an orthonormal basis $V_{m+1}$ of $Q_{m+1}$. The first column of $V_{m+1}$ can be chosen as $V_{m+1} e_1 = b/\|b\|_2$. The basis matrix $V_{m+1}$ satisfies a rational Arnoldi decomposition of the form

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

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

2 The poles of a rational Krylov space

Given a rational Arnoldi decomposition of the above form, it can be shown [1] that the poles $\xi_j$ of the associated rational Krylov space are the generalized eigenvalues of the lower $m \times m$ subpencil of $(H_m, K_m)$. Let us verify this at a simple example by first constructing a rational Krylov space associated with the $m = 5$ poles $-1, \infty, -i, 0, i$. The matrix $A$
is of size $N = 100$ and chosen as the `tridiag` matrix from MATLAB’s `gallery`, and $b$ is the first canonical unit vector. The `rat_krylov` command is used to compute the quantities in the rational Arnoldi decomposition:

```matlab
N = 100;
A = gallery('tridiag', N);
b = eye(N, 1);
m = 5;
xi = [-1, inf, -1i, 0, 1i];
[V, K, H] = rat_krylov(A, b, xi);
```

Indeed, the rational Arnoldi decomposition is satisfied with a residual norm close to machine precision:

```matlab
format shorte
disp(norm(A*V*K - V*H) / norm(H))
```

```
3.5143e-16
```

And the chosen poles $\xi_j$ are the eigenvalues of the lower $m \times m$ subpencil:

```matlab
disp(eig(H(2:m+1,1:m),K(2:m+1,1:m)))
```

```
-1.0000e+00 + 0.0000e+00i
Inf + 0.0000e+00i
0.0000e+00 - 1.0000e+00i
0.0000e+00 + 0.0000e+00i
0.0000e+00 + 1.0000e+00i
```

### 3 Moving the poles explicitly

There is a direct link between the starting vector $b$ and the poles $\xi_j$ of a rational Krylov space $Q_{m+1}$. A change of the poles $\xi_j$ to $\tilde{\xi}_j$ can be interpreted as a change of the starting vector from $b$ to $\tilde{b}$, and vice versa. Algorithms for moving the poles of a rational Krylov space are described in [1] and implemented in the functions `move_poles_expl` and `move_poles_impl`.

For example, let us move the poles of the above rational Krylov space $Q_{m+1}$ to the points $-1, -2, \ldots, -5$:

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

The output of `move_poles_expl` are unitary matrices $Q$ and $Z$, and transformed upper Hessenberg matrices $\tilde{K}_m = Q K_m Z$ and $\tilde{H}_m = Q H_m Z$, so that the lower $m \times m$ part of the pencil $(\tilde{H}_m, \tilde{K}_m)$ has as generalized eigenvalues the new poles $\tilde{\xi}_j$:

```matlab
disp(eig(HT(2:m+1,1:m),KT(2:m+1,1:m)))
```

```
-1.0000e+00 + 0.0000e+00i
Inf + 0.0000e+00i
0.0000e+00 - 1.0000e+00i
0.0000e+00 + 0.0000e+00i
0.0000e+00 + 1.0000e+00i
```
Defining $\tilde{V}_{m+1} = V_{m+1}Q^*$, the transformed rational Arnoldi decomposition is

$$AV_{m+1}\tilde{K}_m = \tilde{V}_{m+1}\tilde{H}_m.$$  

This can be verified numerically by looking at the residual norm:

```matlab
VT = V*QT';
disp(norm(A*VT*KT - VT*HT) / norm(HT))
```

$6.8004e-16$

It should be noted that the function `move_poles_expl` can be used to move the $m$ poles to arbitrary locations, including to infinity, and even to the eigenvalues of $A$. In latter case, the transformed space $\tilde{V}_{m+1}$ does not correspond to a rational Krylov space generated with starting vector $\tilde{V}_{m+1}e_1$ and poles $\tilde{\xi}_j$, but must be interpreted as a filtered rational Krylov space. Indeed, the pole relocation problem is very similar to that of applying an implicit filter to the rational Krylov space [3,4]. See also [1] for more details.

### 4 Moving the poles implicitly

Assume we are given a nonzero vector $\tilde{b} \in \mathbb{Q}_{m+1}$ with coefficient representation $\tilde{b} = V_{m+1}c$, where $c$ is a vector with $m+1$ entries. The function `move_poles_impl` can be used to obtain a transformed rational Arnoldi decomposition with starting vector $\tilde{b}$.

As an example, let us take $c = [0, \ldots, 0, 1]^T$ and hence transform the rational Arnoldi decomposition so that $\tilde{V}_{m+1}e_1 = v_{m+1}$, the last basis vector in $V_{m+1}$:

```matlab
c = zeros(m+1,1); c(m+1) = 1;
[KT, HT, QT, ZT] = move_poles_impl(K, H, c);
VT = V*QT';
```

The poles of the rational Krylov space with the modified starting vector can again be read off as the generalized eigenvalues of the lower $m \times m$ part of $(\tilde{H}_m, \tilde{K}_m)$:

```matlab
disp(eig(HT(2:m+1,1:m),KT(2:m+1,1:m)))
```

$$
\begin{align*}
3.2914e+00 & - 5.5756e-02i \\
1.8705e+00 & - 1.2100e-01i \\
7.7852e-01 & - 9.2093e-02i \\
1.9752e-01 & - 3.0824e-02i \\
4.4392e-03 & - 3.5884e-04i
\end{align*}
$$

This implicit pole relocation procedure is key element of the RKFIT algorithm described in [1,2].
5 Some fun with moving poles

To conclude this example, let us consider a $10 \times 10$ random matrix $A$, a random vector $b$, and the corresponding 6-dimensional rational Krylov space with poles at $-2, -1, 0, 1, 2$:

$$A = (\text{randn}(10) + 1i*\text{randn}(10))*0.5;$$
$$b = \text{randn}(10,1) + 1i*\text{randn}(10,1);$$
$$m = 5;$$
$$\xi = -2:2;$$
$$[V, K, H] = \text{rat_krylov}(A, b, \xi);$$

Here are the eigenvalues of $A$:

$$\text{figure}$$
$$\text{plot}(\text{eig}(A), 'k0', 'MarkerFaceColor', 'y')$$
$$\text{axis}([-2.5, 2.5, -2.5, 2.5]), \text{grid on}, \text{hold on}$$

We now consider a $t$-dependent coefficient vector $c(t)$ such that $V_{m+1}c(t)$ is continuously "morphed" from $v_1$ to $v_2$. The poles of the rational Krylov space with the transformed starting vector $V_{m+1}c(t)$ are then plotted as a function of $t$.

$$\text{for } t = \text{linspace}(1, 2, 51),$$
$$c = \text{zeros}(m+1,1);$$
$$c(\text{floor}(t)) = \cos(\pi*(t-\text{floor}(t))/2);$$
$$c(\text{floor}(t)+1) = \sin(\pi*(t-\text{floor}(t))/2);$$
$$[K_T, H_T, Q_T] = \text{move_poles_impl}(K, H, c);% \text{transformed pencil}$$
$$\xi_{new} = \text{sort}(\text{eig}(H_T(2:m+1,1:m),K_T(2:m+1,1:m)));% \text{new poles}$$
$$\text{plot}(\text{real}(\xi_{new}), \text{imag}(\xi_{new}), 'b+')$$
$$\text{end}$$
As one can see, only one of the five poles starts moving away from $-2$, with the remaining four poles staying at their positions. This is because "morphing" the starting vector from $\mathbf{v}_1$ to $\mathbf{v}_2$ only affects a two-dimensional subspace of $Q_{m+1}$ which includes the vector $\mathbf{b}$ and is itself a rational Krylov space, and this space is parameterized by one pole only.

As we now continue morphing from $\mathbf{v}_2$ to $\mathbf{v}_3$, another pole starts moving:

```matlab
for t = linspace(2,3,51),
    c = zeros(m+1,1);
    c(floor(t)) = cos(pi*(t-floor(t))/2);
    c(floor(t)+1) = sin(pi*(t-floor(t))/2);
    [KT, HT, QT, ZT] = move_poles_impl(K, H, c);
    xi_new = sort(eig(HT(2:m+1,1:m),KT(2:m+1,1:m)));
    plot(xi_new, 'r+')
end
```

Morphing from $\mathbf{v}_3$ to $\mathbf{v}_4$, then to $\mathbf{v}_5$, and finally to $\mathbf{v}_6$ will eventually affect all five poles of the rational Krylov space:
for t = linspace(3, 5.99, 150)
c = zeros(m+1,1);
c(floor(t)) = cos(pi*(t-floor(t))/2);
c(floor(t)+1) = sin(pi*(t-floor(t))/2);
[KT, HT, QT, ZT] = move_poles_impl(K, H, c);
xi_new = sort(eig(HT(2:m+1, 1:m), KT(2:m+1, 1:m)));
switch floor(t)
case 3, plot(xi_new', 'g+')
case 4, plot(xi_new', 'm+')
case 5, plot(xi_new', 'c+')
end
end

6 References


