Comparison of greedy-type approaches involving the Loewner matrix for rational modeling *of scalar functions*

AAA
Randomized SVD
Subset of singular values (Matlab's `svds`)
CUR decomposition
DEIM-CUR decomposition
adaptive Loewner
recursive Loewner

Sanda Lefteriu

8ECM

IMT Lille Douai
École Mines-Télécom
IMT-Université de Lille

Motivation

The Loewner framework relies on the SVD for computing a reduced-order rational model from frequency-domain measurements.

Remark

For large data sets,

the Loewner matrix pencil cannot be stored

Motivation

The Loewner framework relies on the SVD for computing a reduced-order rational model from frequency-domain measurements.

Remark

For large data sets,

the Loewner matrix pencil cannot be stored
& the SVD is expensive.

Motivation

The Loewner framework relies on the SVD for computing a reduced-order rational model from frequency-domain measurements.

Remark

For large data sets,

the Loewner matrix pencil cannot be stored
& the SVD is expensive.

For noise-free measurements of a rational function, use an arbitrary projection [A.C. Antoulas, S. Lefteriu, A.C. Ionita, "A tutorial introduction to the Loewner framework for model reduction"].

Introduction

Commonalities:

- barycentric representation for rational functions

$$H(s) = \frac{\sum_{k=1}^{n} \frac{\beta_k}{s-\lambda_k}}{\sum_{k=1}^{n} \frac{\alpha_k}{s-\lambda_k}},$$

  with free parameters α_k and β_k and support points λ_k.
- greedy selection of support points & evaluation points.

Introduction

Commonalities:

- barycentric representation for rational functions

$$H(s) = \frac{\sum_{k=1}^{n} \frac{\beta_k}{s-\lambda_k}}{\sum_{k=1}^{n} \frac{\alpha_k}{s-\lambda_k}},$$

  with free parameters α_k and β_k and support points λ_k.
- greedy selection of support points & evaluation points.

Approaches compared:

AAA
Randomized SVD
Subset of singular values (Matlab's `svds`)
CUR decomposition
DEIM-CUR decomposition
adaptive Loewner
recursive Loewner

SISO (scalar) case

Partition the data set $(s_i, H_i),\ i = 1, \dots, N$ into

$$\boxed{(\lambda_k, w_k),\ k = 1, \dots, n} \quad \text{and} \quad \boxed{(\mu_h, v_h),\ h = 1, \dots, N-n}.$$

support points (right data) — evaluation points (left data)

A rational function of order $n-1$ can be expressed in barycentric form

$$H(s) = \frac{\sum_{k=1}^{n} \frac{\beta_k}{s-\lambda_k}}{\sum_{k=1}^{n} \frac{\alpha_k}{s-\lambda_k}}, \quad \alpha_k \text{ and } \beta_k \text{ free} \Rightarrow \sum_{k=1}^{n} \alpha_k \frac{H - w_k}{s - \lambda_k} = 0,\ \alpha_k \neq 0.$$

Fix n constraints $H(\lambda_k) = \frac{\beta_k}{\alpha_k} =: w_k \Rightarrow \beta_k = \alpha_k w_k$.
For the remaining $N-n$ conditions $H(\mu_h) = v_h \Rightarrow$ the condition $\mathbb{L}\boldsymbol{c} = \mathbf{0}$,

$$\mathbb{L} = \begin{bmatrix} \frac{v_1 - w_1}{\mu_1 - \lambda_1} & \cdots & \frac{v_1 - w_n}{\mu_1 - \lambda_n} \\ \vdots & \ddots & \vdots \\ \frac{v_{N-n} - w_1}{\mu_{N-n} - \lambda_1} & \cdots & \frac{v_{N-n} - w_n}{\mu_{N-n} - \lambda_n} \end{bmatrix}, \boldsymbol{c} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{N/2} \end{bmatrix},$$

where $\mathbb{L}$ is the Loewner matrix [A. C. Antoulas and B. D. Q. Anderson, "On the scalar rational interpolation problem"].

Review of AAA

Y. Nakatsukasa, O. Sète, L. N. Trefethen, "The AAA algorithm for rational approximation".

```
function [r,pol,res,zer,z,f,w,errvec] = aaa(F,Z,tol,mmax)
M = length(Z);                              % number of sample points
if nargin<3, tol = 1e-13; end               % default relative tol 1e-13
if nargin<4, mmax = 100; end                % default max type (99,99)
if ~isfloat(F), F = F(Z); end               % convert function handle to vector
Z = Z(:); F = F(:);                         % work with column vectors
SF = spdiags(F,0,M,M);                      % left scaling matrix
J = 1:M; z = []; f = []; C = [];            % initializations
errvec = []; R = mean(F);
for m = 1:mmax                              % main loop
  [~,j] = max(abs(F-R));                    % select next support point
  z = [z; Z(j)]; f = [f; F(j)];             % update support points, data values
  J(J==j) = [];                             % update index vector
  C = [C 1./(Z-Z(j))];                      % next column of Cauchy matrix
  Sf = diag(f);                             % right scaling matrix
  A = SF*C - C*Sf;                          % Loewner matrix
  [~,~,V] = svd(A(J,:),0);                  % SVD
  w = V(:,m);                               % weight vector = min sing vector
  N = C*(w.*f); D = C*w;                    % numerator and denominator
  R = F; R(J) = N(J)./D(J);                 % rational approximation
  err = norm(F-R,inf);
  errvec = [errvec; err];                   % max error at sample points
  if err <= tol*norm(F,inf), break, end     % stop if converged
end
r = @(zz) feval(@rhandle,zz,z,f,w);         % AAA approximant as function handle
[pol,res,zer] = prz(r,z,f,w);               % poles, residues, and zeros
[r,pol,res,zer,z,f,w] = ...
   cleanup(r,pol,res,zer,z,f,w,Z,F);        % remove Frois. doublets (optional)
```

The missing piece: the shifted Loewner matrix

Define the shifted Loewner matrix [A. J. Mayo and A. C. Antoulas, "A framework for the solution of the generalized realization problem"]:

$$\underbrace{\sigma\mathbb{L}}_{\in\mathbb{C}^{N/2\times N/2}} = \begin{bmatrix} \frac{\mu_1 v_1 - \lambda_1 w_1}{\mu_1 - \lambda_1} & \cdots & \frac{\mu_1 v_1 - \lambda_{N/2} w_{N/2}}{\mu_1 - \lambda_{N/2}} \\ \vdots & \ddots & \vdots \\ \frac{\mu_{N/2} v_{N/2} - \lambda_1 w_1}{\mu_{N/2} - \lambda_1} & \cdots & \frac{v_{N/2}\mu_{N/2} - \lambda_{N/2} w_{N/2}}{\mu_{N/2} - \lambda_{N/2}} \end{bmatrix}$$

The missing piece: the shifted Loewner matrix

Define the shifted Loewner matrix [A. J. Mayo and A. C. Antoulas, "A framework for the solution of the generalized realization problem"]:

$$\underbrace{\sigma\mathbb{L}}_{\in\mathbb{C}^{N/2\times N/2}} = \begin{bmatrix} \frac{\mu_1 v_1 - \lambda_1 w_1}{\mu_1 - \lambda_1} & \cdots & \frac{\mu_1 v_1 - \lambda_{N/2} w_{N/2}}{\mu_1 - \lambda_{N/2}} \\ \vdots & \ddots & \vdots \\ \frac{\mu_{N/2} v_{N/2} - \lambda_1 w_1}{\mu_{N/2} - \lambda_1} & \cdots & \frac{v_{N/2}\mu_{N/2} - \lambda_{N/2} w_{N/2}}{\mu_{N/2} - \lambda_{N/2}} \end{bmatrix}$$

Theorem

- *If the matrix pencil* $(\sigma\mathbb{L}, \mathbb{L})$ *is regular and* $\mu_h, \lambda_k \notin eig(\sigma\mathbb{L}, \mathbb{L})$, *then*

$$\boldsymbol{E} = -\mathbb{L}, \boldsymbol{A} = -\sigma\mathbb{L}, \boldsymbol{B} = \boldsymbol{V}, \boldsymbol{C} = \boldsymbol{W} \text{ and } \boldsymbol{D} = \boldsymbol{0}$$

  is a minimal realization of an interpolant (i.e., the transfer function $\boldsymbol{H}(s) = \boldsymbol{W}(\sigma\mathbb{L} - s\mathbb{L})^{-1}\boldsymbol{V}$ *satisfies* $\boldsymbol{H}(\lambda_k) = \boldsymbol{w}_k$ *and* $\boldsymbol{H}(\mu_h) = \boldsymbol{v}_h$*.)*

- *If singular, perform SVD of* $x\mathbb{L} - \sigma\mathbb{L} = \boldsymbol{Y}\Sigma\boldsymbol{X}^*$, $x = \lambda_k$ *or* μ_h, *to determine* n = *dimension of regular part. Using* $\boldsymbol{Y} = \boldsymbol{Y}_{(:,1:n)}$, $\boldsymbol{X} = \boldsymbol{X}_{(:,1:n)}$,

$$\begin{array}{llllll} \boldsymbol{E} & := & -\boldsymbol{Y}^*\mathbb{L}\boldsymbol{X} & \boldsymbol{A} & := & -\boldsymbol{Y}^*\sigma\mathbb{L}\boldsymbol{X} \\ \boldsymbol{B} & := & \boldsymbol{Y}^*\boldsymbol{V} & \boldsymbol{C} & := & \boldsymbol{W}\boldsymbol{X}. \end{array}$$

Review of randomized SVD

```
function [U,S,V]=rsvd (X,r)
  N=size(X,2);
  P=randn(N,2r);
  Z=X*P;
  W1 = orth(Z);
  B = W1'*X;
  [W2,S,V] = svd(B,'econ');
  U = W1*W2;
  U = U(:,1:r);
  S = S(1:r,1:r);
  V=V(:,1:r);
end
```

Review of Matlab's `svds`

Dominant singular values are found via a Krylov procedure.

Review of the CUR decomposition

Low rank approximate decomposition $\boldsymbol{A} \approx \boldsymbol{CUR}$ [B. Kramer, A. A. Gorodetsky, "System identification via CUR-factored Hankel approximation"]

- $\boldsymbol{C}$, $\boldsymbol{R}$ = subsets of the columns and rows of $\boldsymbol{A}$
- $\boldsymbol{U}$ constructed to make $\boldsymbol{CUR}$ a good approximation (maximum-volume–based cross-approximation algorithm).

Algorithm 1. `cur-cross`: Cross-approximation of a matrix

Input: Matrix $H \in \mathbb{R}^{n_c \times n_r}$;
Rank upper bound estimate r;
Initial column indices $\mathcal{J} = [j_1, j_2, \ldots, j_r]$;
Stopping tolerance $\delta > 0$;
`maxvol` tolerance ϵ

Output: $\mathcal{I}, \mathcal{J}$ such that $H(\mathcal{I}, \mathcal{J})$ has "large" volume

$k = 1$
$Q, R = \mathtt{qr}(H(:, \mathcal{J}))$
$\mathcal{I} = \mathtt{maxvol}(Q, \epsilon)$
$Q, R = \mathtt{qr}((H(\mathcal{I}, :))^T)$
$\mathcal{J} = \mathtt{maxvol}(Q, \epsilon)$
$\hat{Q} = Q(\mathcal{J}, :)$
$H_1 = H(:, \mathcal{J}) \left(Q\hat{Q}^{-1}\right)^T$
repeat
  $Q, R = \mathtt{qr}(H(:, \mathcal{J}))$
  $\mathcal{I} = \mathtt{maxvol}(Q, \epsilon)$
  $Q, R = \mathtt{qr}((H(\mathcal{I}, :))^T)$
  $\mathcal{J} = \mathtt{maxvol}(Q, \epsilon)$
  $\hat{Q} = Q(\mathcal{J}, :)$
  $H_{k+1} = H(:, \mathcal{J}) \left(Q\hat{Q}^{-1}\right)^T$
  $k = k + 1$
until $\|H_{k+1} - H_k\|_F / \|H_k\|_F \leq \delta$

Algorithm 2. `maxvol`: Find dominant submatrix [8].

Input: Matrix $Q \in \mathbb{R}^{n_c \times r}$;
Convergence tolerance ϵ

Output: Row indices $\mathcal{I} = (i_1, i_2, \ldots, i_r)$ such that $Q(\mathcal{I}, :)$ is approximately dominant;

$L, U, \mathcal{P} = \mathtt{lu}(Q)$ {pivoted LU decomposition; $\mathcal{P}$ is a permutation vector}
$Q_{\mathcal{P}} = Q(\mathcal{P}, :)$
$G = Q_{\mathcal{P}} \left[Q_{\mathcal{P}}(1{:}r, :)\right]^{\dagger}$
$i^*, j^* = \arg\max_{i,j} |G(i, j)|$
$g = G(i^*, j^*)$
while $|g| > 1 + \epsilon$ **do**
  $\mathcal{P}(j^*) = i^*$
  $\mathcal{P}(i^*) = j^*$
  $Q_{\mathcal{P}} = Q(\mathcal{P}, :)$
  $G = Q_{\mathcal{P}} \left[Q_{\mathcal{P}}(1{:}r, :)\right]^{\dagger}$
  $i^*, j^* = \arg\max_{i,j} |G(i, j)|$
  $g = G(i^*, j^*)$
end while
$\mathcal{I} = \mathcal{P}(1{:}r)$

Review of the DEIM-CUR decomposition

Rank r SVD $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{W}^*$

- use $\boldsymbol{V}$ & $\boldsymbol{W}$ to select columns & rows of $\boldsymbol{A}$ to form $\boldsymbol{C}$ & $\boldsymbol{R}$
- interpolatory projection with $\mathcal{O}(Nn)$ vs typical projection with $\mathcal{O}(N^2n)$.

Input: $\mathbf{V}$, an $m \times k$ matrix ($m \geq k$)

Output: $\mathbf{p}$, an integer vector with k distinct entries in $\{1, \ldots, m\}$

$\mathbf{v} = \mathbf{V}(:, 1)$
$[\sim, p_1] = \max(|\mathbf{v}|)$
$\mathbf{p} = [p_1]$
for $j = 2, 3, \ldots, k$
  $\mathbf{v} = \mathbf{V}(:, j)$
  $\mathbf{c} = \mathbf{V}(\mathbf{p}, 1 : j-1)^{-1}\mathbf{v}(\mathbf{p})$
  $\mathbf{r} = \mathbf{v} - \mathbf{V}(:, 1 : j-1)\mathbf{c}$
  $[\sim, p_j] = \max(|\mathbf{r}|)$
  $\mathbf{p} = [\mathbf{p};\ p_j]$
end

[D. C. Sorensen, M. Embree, "A DEIM induced CUR factorization"]

Review of adaptive Loewner

Algorithm 2 $(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C})$ = ADAPTIVEapproach$(\mathbf{S}, \omega)$

Given: $\mathbf{S}, \omega$ where $\mathbf{S}^{(i)}$ are the measured S-parameters at $f_i = \frac{\omega_i}{2\pi}$.
Output: $\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}$ such that $\mathbf{H}(j\omega_i) \approx \mathbf{S}^{(i)}$, $i = 1, \ldots, k$.

1: Generate N, a set of p linearly distributed numbers between 1 and k, set $m \leftarrow p$.
2: Construct Λ, M, $\mathbf{L}$, $\mathbf{R}$, $\mathbf{V}$, $\mathbf{W}$, $\mathbb{L}$, $\sigma\mathbb{L}$ as described in Section V-A using $\omega_{N(1)}, \ldots, \omega_{N(p)}$ and $\mathbf{S}^{(N(1))}, \ldots, \mathbf{S}^{(N(p))}$, set $\mathbf{E} \leftarrow -\mathbb{L}$, $\mathbf{A} \leftarrow -\sigma\mathbb{L}$, $\mathbf{B} \leftarrow \mathbf{V}$, $\mathbf{C} \leftarrow \mathbf{W}$.
3: **for** $i = 1, \ldots, k$ **do**
4: $\quad\mathbf{H}_i \leftarrow \mathbf{C}(j \cdot \omega_i \mathbf{E} - \mathbf{A})^{-1}\mathbf{B}$.
5: $\quad[\mathbf{Y}, \Sigma, \mathbf{X}] = \text{svd}(\mathbf{H}_i - \mathbf{S}^{(i)})$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_p)$, $err_i \leftarrow \sigma_1$.
6: **end for**
7: Sort err descendingly: $[vmax, imax] = \text{sort}(err)$, where $vmax$ are the sorted values and $imax$ are the sorted indices.
8: **while** $vmax(1) > th$, where $th = Noise \cdot \sqrt{k}$ **do**
9: $\quad$Update Λ, M using $\omega_{imax(1)}, \ldots, \omega_{imax(p)}$.
10: $\quad$**for** $i = 1, \ldots, p$ **do**
11: $\quad\quad[\mathbf{Y}, \Sigma, \mathbf{X}] = \text{svd}(\mathbf{H}_{imax(i)} - \mathbf{S}^{(imax(i))})$, $\mathbf{r}_{i+m} = \mathbf{X}(:, i)$, $\mathbf{w}_{i+m} = \mathbf{S}^{(imax(i))}\mathbf{r}_{i+m}$, $\ell_{i+m} = \mathbf{Y}(:, i)^T$, $\mathbf{v}_{i+p} = \ell_{i+p}\overline{\mathbf{S}^{(imax(i))}}$.
12: $\quad$**end for**
13: $\quad$Update $\mathbb{L}$, $\sigma\mathbb{L}$ using (18) and (22), set $\mathbf{E} \leftarrow -\mathbb{L}$, $\mathbf{A} \leftarrow -\sigma\mathbb{L}$, $\mathbf{B} \leftarrow \mathbf{V}$, $\mathbf{C} \leftarrow \mathbf{W}$, $m \leftarrow m + p$.
14: $\quad$Repeat Steps 3-7.
15: **end while**

[S. Lefteriu, A. C. Antoulas, "A New Approach to Modeling Multiport Systems From Frequency-Domain Data"]

Review of recursive Loewner

Define the generating matrices as the 2×2 rational matrices

$$\boldsymbol{\Theta}(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} \boldsymbol{W} \\ -\boldsymbol{R} \end{bmatrix} (s\mathbb{L} - \mathbb{L}\Lambda)^{-1} \begin{bmatrix} \boldsymbol{L} & \boldsymbol{V} \end{bmatrix},$$

and its inverse

$$\bar{\boldsymbol{\Theta}}(s) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} -\boldsymbol{W} \\ \boldsymbol{R} \end{bmatrix} (s\mathbb{L} - M\mathbb{L})^{-1} \begin{bmatrix} \boldsymbol{L} & \boldsymbol{V} \end{bmatrix}.$$

Interpolation conditions are satisfied:

$$\begin{bmatrix} \ell_j & \boldsymbol{v}_j \end{bmatrix} \boldsymbol{\Theta}(\mu_j) = \mathbf{0}_{1\times 2},$$

$$\bar{\boldsymbol{\Theta}}(\lambda_i) \begin{bmatrix} -\boldsymbol{w}_i \\ \boldsymbol{r}_i \end{bmatrix} = \mathbf{0}_{2\times 1}.$$

Recursive Greedy Approach

1. Set $\boldsymbol{L}_o$, $\boldsymbol{R}_o$, $\boldsymbol{V}_o$ and $\boldsymbol{W}_o$.
2. Construct an order 1 model: $\mathbb{L}_1$, $\sigma\mathbb{L}_1$, $\boldsymbol{V}_1$, $\boldsymbol{W}_1$.
3. Evaluate the generating system for this model at all frequency samples $i = 1, \ldots, N$ and compute errors:
$$\begin{bmatrix} \ell_{e,i} & \boldsymbol{v}_{e,i} \end{bmatrix} = \begin{bmatrix} \ell_{o,i} & \boldsymbol{v}_{o,i} \end{bmatrix} \boldsymbol{\Theta}(\mu_i)$$
$$\begin{bmatrix} -\boldsymbol{w}_{e,i} \\ \boldsymbol{r}_{e,i} \end{bmatrix} = \bar{\boldsymbol{\Theta}}(\lambda_i) \begin{bmatrix} -\boldsymbol{w}_{o,i} \\ \boldsymbol{r}_{o,i} \end{bmatrix}$$
4. Select the 2 measurements which give the largest errors in $\ell_e \boldsymbol{w}_e$ and $\boldsymbol{v}_e \boldsymbol{r}_e$, respectively. Use ℓ_e, $\boldsymbol{v}_e$, $\boldsymbol{r}_e$, $\boldsymbol{w}_e$ to compute the associated $\mathbb{L}_2$ and $\sigma\mathbb{L}_2$.
5. While norm$(x\mathbb{L}_2 - \sigma\mathbb{L}_2)$ is not small
   - Update the model by
$$\mathbb{L} = \text{diag}\,[\mathbb{L}_1, \;\; \mathbb{L}_2], \;\; \sigma\mathbb{L} = \begin{bmatrix} \sigma\mathbb{L}_1 & \boldsymbol{L}_1\boldsymbol{W}_2 \\ \boldsymbol{V}_2\boldsymbol{R}_1 & \sigma\mathbb{L}_2 \end{bmatrix},$$
$$\boldsymbol{W} = [\boldsymbol{W}_1 \quad \boldsymbol{W}_2], \;\; \boldsymbol{V} = \begin{bmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \end{bmatrix}$$
   - Repeat steps **3** and **4**.

[S. Lefteriu, A. C. Antoulas, "A New Approach to Modeling Multiport Systems From Frequency-Domain Data"]

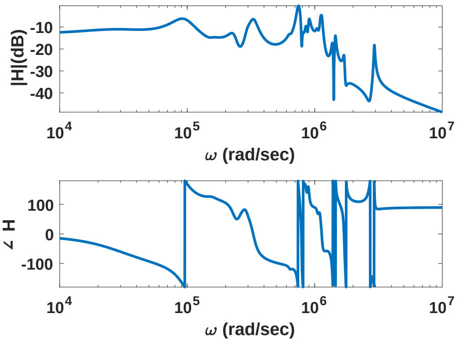
Summary of the approaches

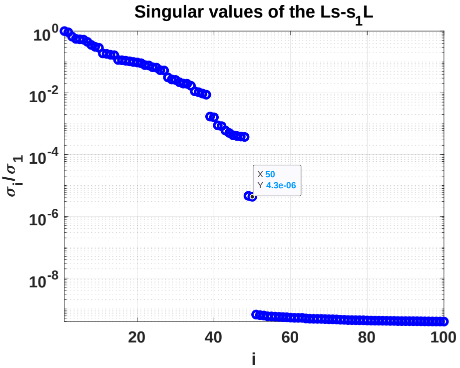
Table: N = number of measurements, n = order of the model

Approach	Computational complexity	Storage
SVD	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
AAA	$\mathcal{O}(Nn^3)$	$\mathcal{O}(Nn)$
Randomized SVD	$\mathcal{O}(N^2\log(n) + Nn^2)$	$\mathcal{O}(N^2)$
`svds`	$\mathcal{O}(N^2 n)$	$\mathcal{O}(N^2)$
CUR	$\mathcal{O}(Nn^2 + n^3)$	$\mathcal{O}(N^2)$
DEIM-CUR	$\mathcal{O}(Nn) + \mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
Adaptive	$\mathcal{O}(Nn^4 + N\log(N))$	$\mathcal{O}(N + n^2)$
Recursive	$\mathcal{O}(Nn^4 + N\log(N))$	$\mathcal{O}(N + n^2)$

Random dynamical system + noise

$n = 50$, 5 000 meas + complex conjugates, SNR $= 100$





Random dynamical system + noise

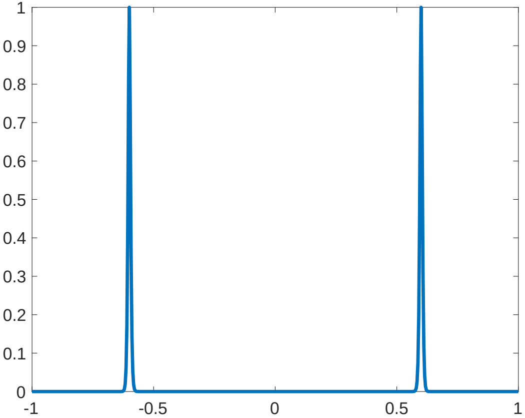
Table: $N = 10\ 000$, $n = 50$

Approach	**$\mathcal{H}_2$ error**	**CPU time**
SVD	5.5130e-10	4.8e+02
AAA	1.0423e-09	4.2e+00
Randomized SVD	5.5130e-10	1.8e+00
`svds`	5.5130e-10	5.7e+01
CUR	2.5472e-05	9.1e+00
DEIM-CUR	9.1053e-08	8.5e-01
Adaptive	6.4280e-07	1.6e+02
Recursive	9.0368e-08	1.0e+02

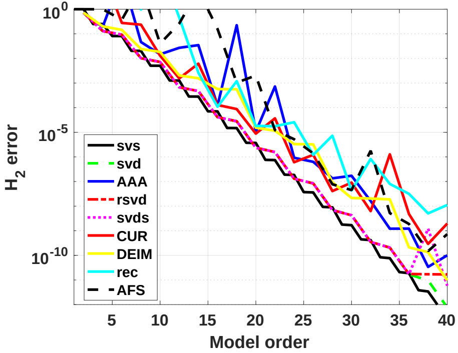
Two-peak function

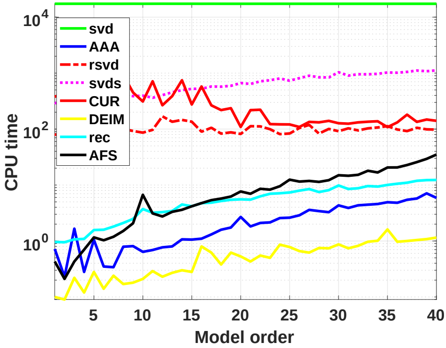
[D. S. Karachalios, I. V. Gosea, A. C. Antoulas. "Data-driven approximation methods applied to non-rational functions"], [S. I. Filip, Y. Nakatsukasa, L. N. Trefethen, B. Beckermann, "Rational minimax approximation via adaptive barycentric representations"]

$$f(x) = \frac{100\pi(x^2 - 0.36)}{\sinh(100\pi(x^2 - 0.36))}, \quad x \in [-1, 1], \quad x_i \leftarrow \text{linspace}(-1, 1, N = 50\ 000)$$



Two-peak function - results





Conclusion

Summary

- There is no clear winner ☺
- AAA chooses the support points from the available measurements in a greedy fashion, while the remaining measurements are evaluation points.
- For the other methods, one decides a-priori on the split of measurements between equal sets of left and right data from which support and evaluation points are chosen in a greedy fashion.

To do

- Further numerical tests.

Thank you!