MANCHESTER 1824
The University of Manchester

M\crNA
Manchester Numerical Analysis

The AAA algorithm and its variations to approximate matrix-valued functions

Stefan Güttel, Gian Maria Negri Porzio, Françoise Tisseur

Thursday, October 3rd 2019
NLA Group meeting

The University of Manchester,
Department of Mathematics

Outline of the presentation

Introduction

Leja–Bagby algorithm

The Adapted Antoulas-Anderson algorithm and its variations

The AAA algorithm for matrix-valued functions

Introduction

Notation and settings

- $\Omega \subset \mathbb{C}$ nonempty, bounded open set.
- $f(z) \colon \Omega \to \mathbb{C}$ meromorphic function.

Notation and settings

- $\Omega \subset \mathbb{C}$ nonempty, bounded open set.
- $f(z) \colon \Omega \to \mathbb{C}$ meromorphic function.

Our goal

Return a rational approximation $r(z)$ of $f(z)$, where

- $r(z) = p(z)/q(z)$ is a (d_1, d_2) rational function, i.e., $p(z)$ and $q(z)$ are coprime polynomials of degree d_1 and d_2, respectively.
- $\|r(z) - f(z)\| < \epsilon$.

Why using a rational approximation?

In many applications we cannot handle the original data of the problem (E.g.: NEPs)$\Longrightarrow$ Simplify!

Why using a rational approximation?

In many applications we cannot handle the original data of the problem (E.g.: NEPs)$\Longrightarrow$ Simplify!

Three main reasons:

- Compact representation of a function $f(z)$.
- Better convergence than polynomial approximations.
- Extract information about poles or zeros of a function $f(z)$ inside a region $\Omega \subset \mathbb{C}$.

Leja–Bagby algorithm

The Walsh–Hermite integral formula

- $f(z)$ holomorphic in Ω.
- $\Sigma \subset \Omega$ compact and connected set, $\Gamma := \partial\Omega$.
- $(\sigma_j)_{j=0}^m \subset \Sigma$ and $(\xi_j)_{j=1}^m \subset \mathbb{C} \setminus \overline{\Omega}$.

Theorem (Walsh–Hermite integral formula)

Let $s_m = \prod_{j=0}^m (z - \sigma_j) / \prod_{j=1, \xi_j \neq \infty}^m (z - \xi_j)$. *Then*

$$r_m(z) := \frac{1}{2\pi \mathrm{i}} \int_\Gamma \left(1 - \frac{s_m(z)}{s(\zeta)}\right) \frac{f(\zeta)}{\zeta - z}\, d\zeta$$

is the unique (m, m) *rational function with preassigned poles* ξ_j *that interpolates* $f(z)$ *at the nodes* σ_j *counting multiplicities.*

The idea

A basic error estimate gives:

$$\begin{aligned}\|f(z) - r_m(z)\|_2 &= \frac{1}{2\pi \mathrm{i}} \left\| \int_\Gamma \frac{s_m(z)}{s(\zeta)} \frac{f(\zeta)}{\zeta - z}\, d\zeta \right\|_2 \\ &\leq K \frac{|s_m(z)|}{\min_{\zeta \in \Gamma} |s_m(\zeta)|},\end{aligned}$$

where K depends on f, Σ and Γ.

The idea

A basic error estimate gives:

$$\|f(z) - r_m(z)\|_2 = \frac{1}{2\pi \mathrm{i}} \left\| \int_\Gamma \frac{s_m(z)}{s(\zeta)} \frac{f(\zeta)}{\zeta - z}\, d\zeta \right\|_2 \leq K \frac{|s_m(z)|}{\min_{\zeta \in \Gamma} |s_m(\zeta)|},$$

where K depends on f, Σ and Γ.

The intuition

We need to minimize $s_m(z)$ on Σ and maximise it on Γ. We call the pair (Σ, Γ) a condenser.

The Leja–Bagby algorithm

1. Select a condenser (Σ, Γ).
2. Start with arbitrary $\sigma_0 \in \Sigma$.
3. For $j = 0, 1, \ldots,$ recursively set

$$\begin{cases} \left|s(\sigma_{j+1})\right| = \max\limits_{z\in\Sigma}\left|s_j(z)\right|, \\ \left|s(\xi_{j+1})\right| = \min\limits_{z\in\Gamma}\left|s_j(z)\right|. \end{cases}$$

The Leja–Bagby algorithm

1. Select a condenser (Σ, Γ).
2. Start with arbitrary $\sigma_0 \in \Sigma$.
3. For $j = 0, 1, \ldots,$ recursively set

$$\begin{cases} \left|s(\sigma_{j+1})\right| = \max\limits_{z\in\Sigma}\left|s_j(z)\right|, \\ \left|s(\xi_{j+1})\right| = \min\limits_{z\in\Gamma}\left|s_j(z)\right|. \end{cases}$$

Proposition (Levin, Saff (2006))

Under our assumptions, there exists a positive number $\operatorname{cap}(\Sigma, \Gamma) > 0$, *called the condenser capacity, such that*

$$\limsup_{m\to\infty} \|f - r_m\|_\Sigma^{1/m} \leq e^{-1/\operatorname{cap}(\Sigma,\Gamma)}.$$

All that glitters is not gold

Remarks

- Exponential decay of the error.

All that glitters is not gold

Remarks

- Exponential decay of the error.
- Immediately generalizable to matrix-valued functions.

All that glitters is not gold

Remarks

- Exponential decay of the error.
- Immediately generalizable to matrix-valued functions.
- Necessary to preassign the zeros $(\sigma_j)_{j=0}^{d_1}$ and poles $(\xi_j)_{j=0}^{d_2}$.

All that glitters is not gold

Remarks

- Exponential decay of the error.
- Immediately generalizable to matrix-valued functions.
- Necessary to preassign the zeros $(\sigma_j)_{j=0}^{d_1}$ and poles $(\xi_j)_{j=0}^{d_2}$.
- The capacity depends on Ω; it is known only for specific subset (e.g., circles).

The Adapted Antoulas-Anderson algorithm and its variations

The Adapted Antoulas-Anderson algorithm

The AAA algorithm

- It's a greedy algorithm created by Nakatsukasa, Séte and Trefethen in 2017.

The Adapted Antoulas-Anderson algorithm

The AAA algorithm

- It's a greedy algorithm created by Nakatsukasa, Séte and Trefethen in 2017.
- It does not need preassigned zeros or poles.

The Adapted Antoulas-Anderson algorithm

The AAA algorithm

- It's a greedy algorithm created by Nakatsukasa, Séte and Trefethen in 2017.
- It does not need preassigned zeros or poles.
- It does not depend on the specific set Ω.

The Adapted Antoulas-Anderson algorithm

The AAA algorithm

- It's a greedy algorithm created by Nakatsukasa, Séte and Trefethen in 2017.
- It does not need preassigned zeros or poles.
- It does not depend on the specific set Ω.
- It does not achieve optimality in any particular norm.

The Adapted Antoulas-Anderson algorithm

The AAA algorithm

- It's a greedy algorithm created by Nakatsukasa, Séte and Trefethen in 2017.
- It does not need preassigned zeros or poles.
- It does not depend on the specific set Ω.
- It does not achieve optimality in any particular norm.

The Adapted Antoulas-Anderson algorithm

The AAA algorithm

- It's a greedy algorithm created by Nakatsukasa, Séte and Trefethen in 2017.
- It does not need preassigned zeros or poles.
- It does not depend on the specific set Ω.
- It does not achieve optimality in any particular norm.

The AAA algorithm represents $r(z)$ in a barycentric form.

The barycentric representation

We write

$$r_m(z) = \frac{n_m(z)}{d_m(z)} = \frac{\sum_{i=1}^{m} \frac{w_i f(z_i)}{z - z_i}}{\sum_{i=1}^{m} \frac{w_i}{z - z_i}},$$

The barycentric representation

We write

$$r_m(z) = \frac{n_m(z)}{d_m(z)} = \frac{\sum_{i=1}^{m} \frac{w_i f(z_i)}{z - z_i}}{\sum_{i=1}^{m} \frac{w_i}{z - z_i}},$$

- z_i are distinct points called sample/support points, while w_i are the weights.
- $r_m(z_i) = f(z_i)$.
- The functions $r_m(z)$ range all over the set of $(m-1, m-1)$ rational functions with no poles at z_i, for $i = 1, \ldots m$.

The AAA cycle

Input

- A set of M points $Z = \{z_i\}_{i=1}^{M}$ (mesh grid or contour of Ω).
- The values $f_i := f(z_i)$ for $i = 1, \ldots, M$.
- Threshold ϵ, max number of points $m_{\max}$.

The AAA cycle

Input

- A set of M points $Z = \{z_i\}_{i=1}^{M}$ (mesh grid or contour of Ω).
- The values $f_i := f(z_i)$ for $i = 1, \ldots, M$.
- Threshold ϵ, max number of points $m_{\max}$.

Goal

Denote with $Z^{(m)} = Z \setminus \{z_1, \ldots, z_m\}$. Our goal is minimizing

$$\sum_{j=1}^{M-m} \left| f(\widetilde{z}_j) d_m(\widetilde{z}_j) - n_m(\widetilde{z}_j) \right|^2 := \| f_m d_m - n_m \|_{Z^{(m)}}$$

with the constraint $\|w\|_2 = 1$.

The AAA cycle

At the beginning of iteration m:

1. Find $z_m = \arg\max |f d_{m-1} - n_{m-1}|_{Z^{(m-1)}}$.

The AAA cycle

At the beginning of iteration m:

1. Find $z_m = \arg\max |f d_{m-1} - n_{m-1}|_{Z^{(m-1)}}$.
2. Remove z_m from $Z^{(m-1)}$ to get $Z^{(m)}$.

The AAA cycle

At the beginning of iteration m:

1. Find $z_m = \arg\max |f d_{m-1} - n_{m-1}|_{Z^{(m-1)}}$.
2. Remove z_m from $Z^{(m-1)}$ to get $Z^{(m)}$.
3. Write $Z^{(m)} = [Z_1^{(m)}, \ldots, Z_{M-m}^{(m)}]^T$. Then

$$\min_{\|w\|_2=1} \|f_m d_m - n_m\|_{Z^{(m)}} = \min_{\|w\|=1} \left\| A^{(m)} w \right\|,$$

with

$$A^{(m)} = \begin{bmatrix} \frac{f(Z_1^{(m)}) - f(z_1)}{Z_1^{(m)} - z_1} & \cdots & \frac{f(Z_1^{(m)}) - f(z_m)}{Z_1^{(m)} - z_m} \\ \vdots & \ddots & \vdots \\ \frac{F_{M-m}^{(m)} - f(z_1)}{Z_{M-m}^{(m)} - z_1} & \cdots & \frac{F_{M-m}^{(m)} - f(z_m)}{Z_{M-m}^{(m)} - z_m} \end{bmatrix} \in \mathbb{C}^{(M-m)\times m}.$$

The AAA cycle

4. Compute the SVD of $A^{(m)} = U\Sigma V^*$: the vector of weights w_m is the final right singular vector.

The AAA cycle

4. Compute the SVD of $A^{(m)} = U\Sigma V^*$: the vector of weights w_m is the final right singular vector.
5. Build $r_m(z)$.

The AAA cycle

4. Compute the SVD of $A^{(m)} = U\Sigma V^*$: the vector of weights w_m is the final right singular vector.
5. Build $r_m(z)$.

The AAA cycle

4. Compute the SVD of $A^{(m)} = U\Sigma V^*$: the vector of weights w_m is the final right singular vector.
5. Build $r_m(z)$.

Computational cost

The expensive part of the algorithm are the $(M - j) \times j$ SVD for $j = 1, \ldots, m$, therefore the cost is $O(m^3 M)$.

The AAA algorithm for matrix-valued functions

What about matrix-valued functions?

From now on, we consider $F(z) \colon \Omega \to \mathbb{C}^{n \times n}$, $F(z) = \sum_{k=1}^{s} f_k(z) A_k$.

Remark

- The Leja–Bagby algorithm can be immediately generalized to build $R(z) \approx F(z)$.
- AAA can be called s times. However, it returns s different set of sample points and weights.

What about matrix-valued functions?

From now on, we consider $F(z) \colon \Omega \to \mathbb{C}^{n \times n}$, $F(z) = \sum_{k=1}^{s} f_k(z) A_k$.

Remark

- The Leja–Bagby algorithm can be immediately generalized to build $R(z) \approx F(z)$.
- AAA can be called s times. However, it returns s different set of sample points and weights.

In 2018, Lietaert, Meerbergen et al. proposed the set-valued AAA algorithm.

The set-valued AAA algorithm

- Assume $F(z) = \sum_{k=1}^{S} f_k(z) A_k$.

The set-valued AAA algorithm

- Assume $F(z) = \sum_{k=1}^{S} f_k(z) A_k$.
- Approximate each scalar function at the same time:

$$f_k(z) \approx \sum_{i=1}^{m} \frac{w_i f_k(z_i)}{z - z_i} \Big/ \sum_{i=1}^{m} \frac{w_i}{z - z_i}, \qquad k = 1, \ldots, s,$$

The set-valued AAA algorithm

- Assume $F(z) = \sum_{k=1}^{S} f_k(z) A_k$.
- Approximate each scalar function at the same time:

$$f_k(z) \approx \sum_{i=1}^{m} \frac{w_i f_k(z_i)}{z - z_i} \Big/ \sum_{i=1}^{m} \frac{w_i}{z - z_i}, \qquad k = 1, \ldots, s,$$

- Greedy strategy: choose $z_m \in Z^{(m-1)}$ such that

$$\max_{i,j} \left| f_i(z_j) - r_{m-1}(z_j) \right| = \max_{i} \left| f_i(z_m) - r_{m-1}(z_m) \right|$$

The set-valued AAA algorithm

- Assume $F(z) = \sum_{k=1}^{S} f_k(z) A_k$.
- Approximate each scalar function at the same time:

$$f_k(z) \approx \sum_{i=1}^{m} \frac{w_i f_k(z_i)}{z - z_i} \Big/ \sum_{i=1}^{m} \frac{w_i}{z - z_i}, \qquad k = 1, \ldots, s,$$

- Greedy strategy: choose $z_m \in Z^{(m-1)}$ such that

$$\max_{i,j} \left| f_i(z_j) - r_{m-1}(z_j) \right| = \max_{i} \left| f_i(z_m) - r_{m-1}(z_m) \right|$$

- Build $R_m(z) = \sum_{k=1}^{S} r_k(z) A_k$.

The matrix $A^{(m)}$ in the set-valued is given by

$$A^{(m)}w = \begin{bmatrix} \frac{F^{(m)}_{1,1}-f(z_1)}{Z^{(m)}_1-z_1} & \cdots & \frac{F^{(m)}_{1,1}-f(z_m)}{Z^{(m)}_1-z_m} \\ \vdots & \ddots & \vdots \\ \frac{F^{(m)}_{1,M-m}-f(z_1)}{Z^{(m)}_{M-m}-z_1} & \cdots & \frac{F^{(m)}_{1,M-m}-f(z_m)}{Z^{(m)}_{M-m}-z_m} \\ \frac{F^{(m)}_{2,1}-f(z_1)}{Z^{(m)}_1-z_1} & \cdots & \frac{F^{(m)}_{2,1}-f(z_m)}{Z^{(m)}_1-z_m} \\ \vdots & \ddots & \vdots \\ \frac{F^{(m)}_{2,M-m}-f(z_1)}{Z^{(m)}_{M-m}-z_1} & \cdots & \frac{F^{(m)}_{2,M-m}-f(z_m)}{Z^{(m)}_{M-m}-z_m} \\ \vdots & \vdots & \vdots \\ \frac{F^{(m)}_{s,1}-f(z_1)}{Z^{(m)}_1-z_1} & \cdots & \frac{F^{(m)}_{s,1}-f(z_m)}{Z^{(m)}_1-z_m} \\ \vdots & \ddots & \vdots \\ \frac{F^{(m)}_{s,M-m}-f(z_1)}{Z^{(m)}_{M-m}-z_1} & \cdots & \frac{F^{(m)}_{s,M-m}-f(z_m)}{Z^{(m)}_{M-m}-z_m} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

A simple error bound

Assume $\|f_k(z) - r_k(z)\|_Z < \epsilon$ for every k. Then

$$\begin{aligned}\|F(z) - R(z)\|_Z^2 &\leq \sum_{i=1}^{M} \left\| \sum_{k=1}^{s} (f_k(z_i) - r_k(z_i)) A_k \right\|_2^2 \\ &\leq \sum_{k=1}^{s} \sum_{i=1}^{M} |f_k(z_i) - r_k(z_i)|^2 \|A_k\|_2^2 \\ &\leq s\epsilon \cdot \max_{1 \leq k \leq s} \|A_k\|_2 .\end{aligned}$$

A simple error bound

Assume $\|f_k(z) - r_k(z)\|_Z < \epsilon$ for every k. Then

$$\begin{aligned}\|F(z) - R(z)\|_Z^2 &\leq \sum_{i=1}^{M} \left\| \sum_{k=1}^{s} (f_k(z_i) - r_k(z_i)) A_k \right\|_2^2 \\ &\leq \sum_{k=1}^{s} \sum_{i=1}^{M} |f_k(z_i) - r_k(z_i)|^2 \|A_k\|_2^2 \\ &\leq s\epsilon \cdot \max_{1 \leq k \leq s} \|A_k\|_2 .\end{aligned}$$

Issue

The set-valued AAA does not consider $\|A_k\|_2$!

Weighted AAA adaptation

Instead of approximate $f_k(z)$, we can consider $\widetilde{f}_k(z) = f_k(z)\|A_k\|_2$ and $\widetilde{A}_k = A_k/\|A_k\|_2$. Under the previous assumptions, we get

$$\begin{aligned}\|F(z) - R(z)\|_Z^2 &\leq \sum_{i=1}^{M} \left\| \sum_{k=1}^{s} (\widetilde{f}_k(z_i) - \widetilde{r}_k(z_i))\widetilde{A}_k \right\|_2^2 \\ &\leq s\epsilon.\end{aligned}$$

Remark

If s is small, then we obtain an error of the same order of magnitude as AAA for scalar functions.

A toy problem

A small example

We want to approximate

$$F(z) = zI + \mathrm{e}^{2\mathrm{i}z}A_1 + (z+4)^{1/3}A_2$$

on the unit disk, where $A_i \in \mathbb{C}^{20\times 20}$ and $\|A_1\|_2 = 1, \|A_2\|_2 = 10^9$.

- We consider the error returned by set-valued AAA and weighted AAA for the scalar functions $\mathrm{e}^{2\mathrm{i}z}$ and $(z+4)^{1/3}$
- We consider the approximation error on 300 random points inside the unit circle.

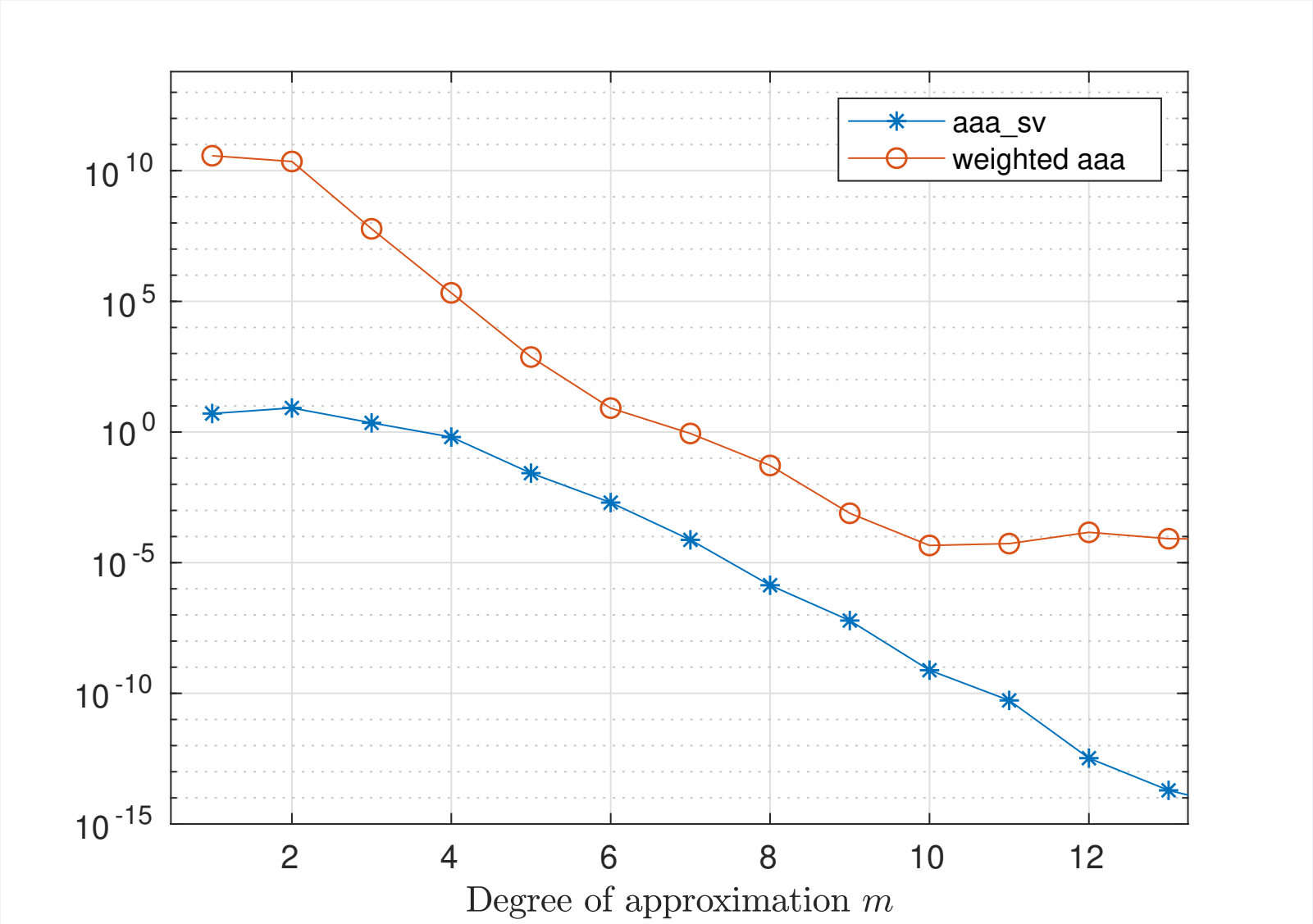

Figure 1: Errors $\max_j \left\| f_j(z) - r_j(z) \right\|_Z$ returned by the two implementation of the AAA algorithm for different degrees of approximation.

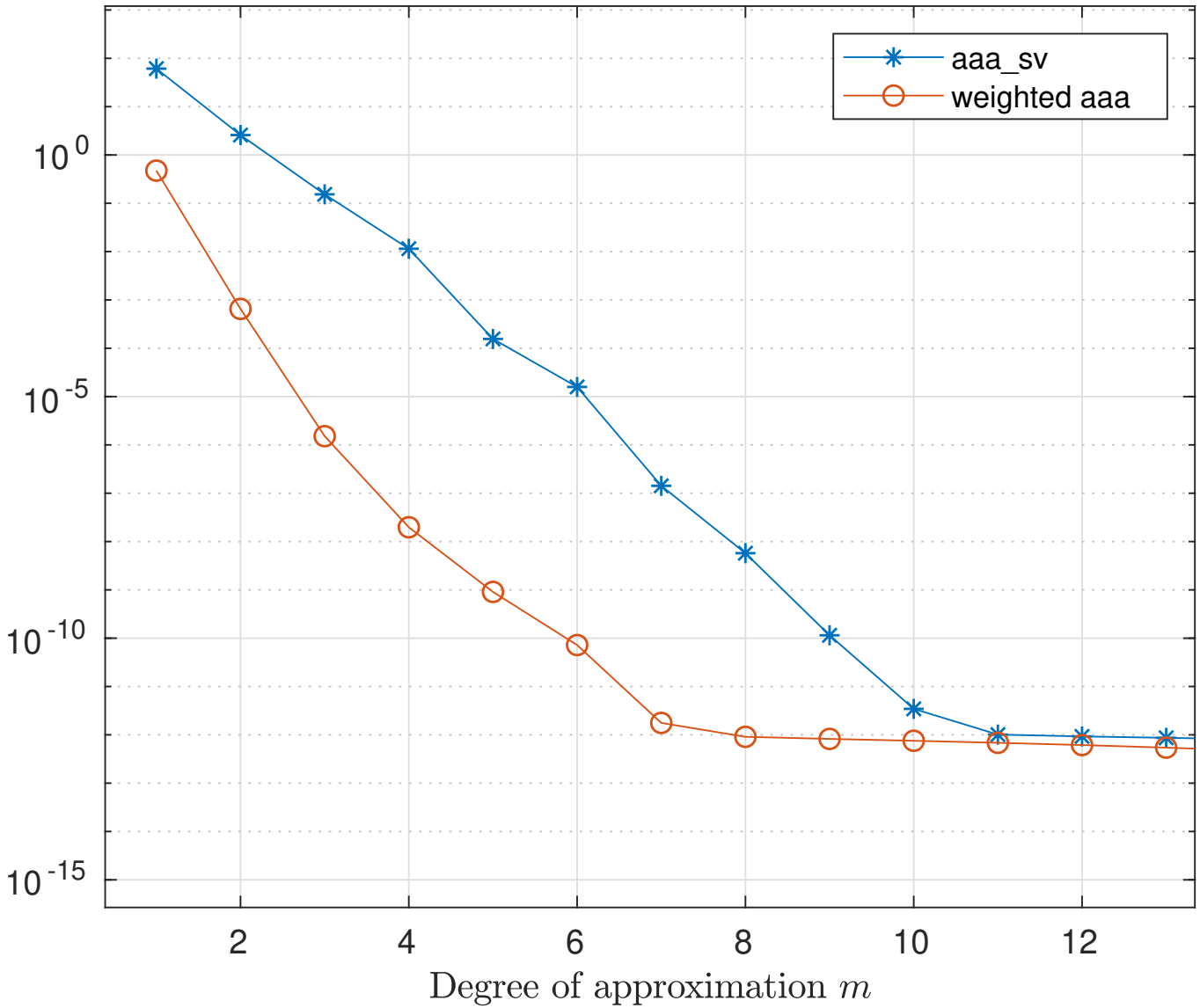

Figure 2: Plot for $\sum_{i=1}^{300} \left\| F(\widetilde{z}_i) - R(\widetilde{z}_i) \right\|_\infty / 300$.

The bent_beam problem

We want to approximate the **bent_beam** problem from the NLEVP library:

$$F(\lambda) = \begin{bmatrix} f_-(\lambda) & g_-(\lambda) & -f_+(\lambda) & -g_+(\lambda) & 0 & 0 \\ 0 & 0 & \alpha(\lambda)g_-(\lambda) & \alpha(\lambda)f_+(\lambda) & \sin\beta(\lambda)\ell & 0 \\ g_-(\lambda) & f_+(\lambda) & g_+(\lambda) & f_-(\lambda) & 0 & 0 \\ 0 & 0 & \alpha(\lambda)^2 EIf_-(\lambda) & \alpha(\lambda)^2 EIg_-(\lambda) & -\beta(\lambda)\tau\cos\beta(\lambda)\ell & 0 \\ \alpha(\lambda)^2 EIf_+(\lambda) & \alpha(\lambda)^2 EIg_+(\lambda) & 0 & 0 & 0 & \beta(\lambda)\tau\sin\beta(\lambda)\ell \\ \alpha(\lambda)g_+(\lambda) & \alpha(\lambda)f_-(\lambda) & 0 & 0 & 0 & -\cos\beta(\lambda)\ell \end{bmatrix},$$

where

- $f_+(\lambda) = \cosh\alpha(\lambda)\ell + \cos\alpha(\lambda)\ell \quad f_-(\lambda) = \cosh\alpha(\lambda)\ell - \cos\alpha(\lambda)\ell;$
- $g_+(\lambda) = \sinh\alpha(\lambda)\ell + \sin\alpha(\lambda)\ell, \quad g_-(\lambda) = \sinh\alpha(\lambda)\ell - \sin\alpha(\lambda)\ell;$
- $\alpha(\lambda) = \sqrt{\lambda}(m/EI)^{1/4}, \quad \beta(\lambda) = \lambda\sqrt{m/\tau};$
- EI, ℓ, τ and m are physical constants.

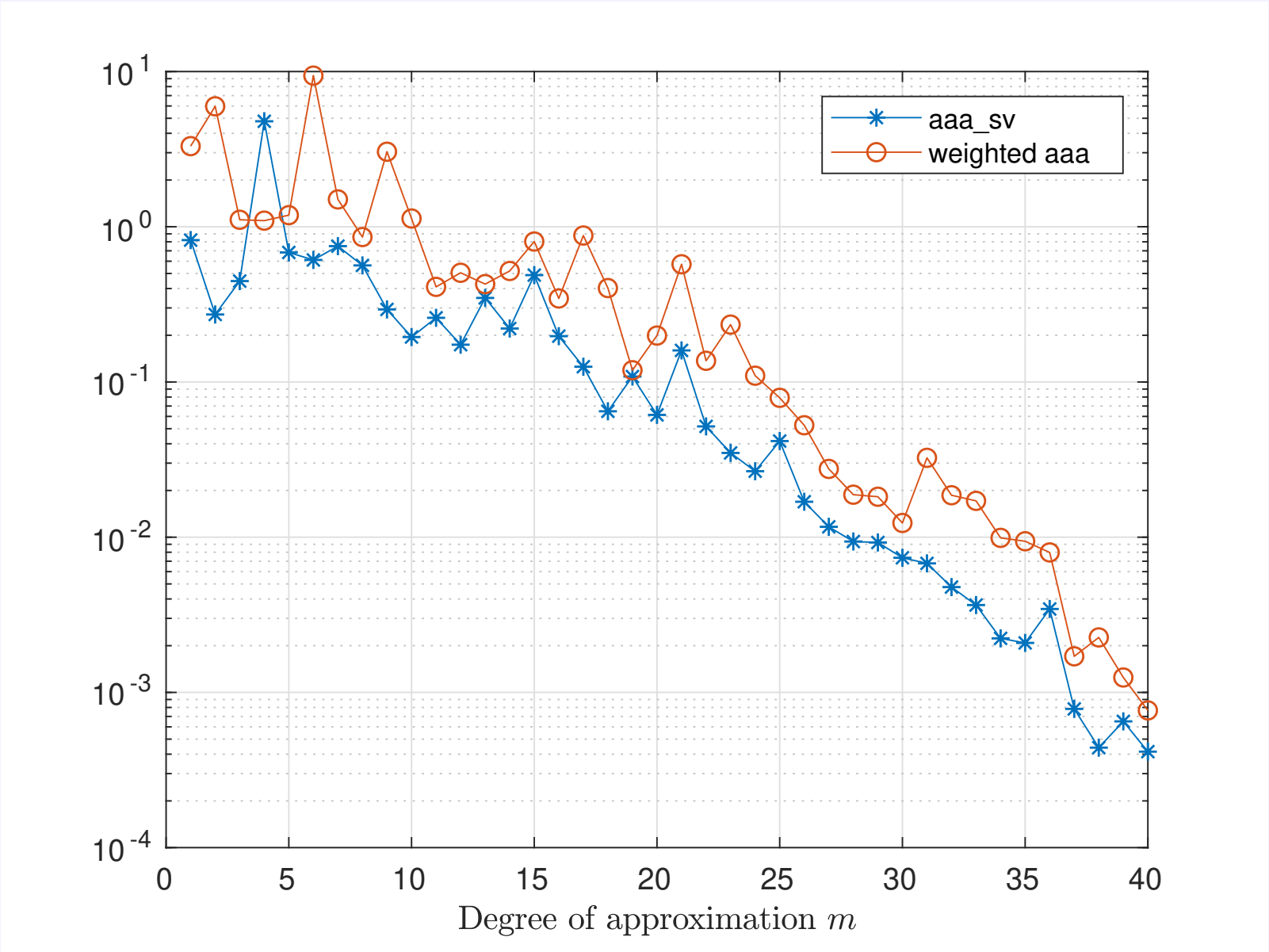

Figure 3: Errors $\max_j \left\| f_j(z) - r_j(z) \right\|_Z$ returned by the two implementation of the AAA algorithm for different degrees of approximation.

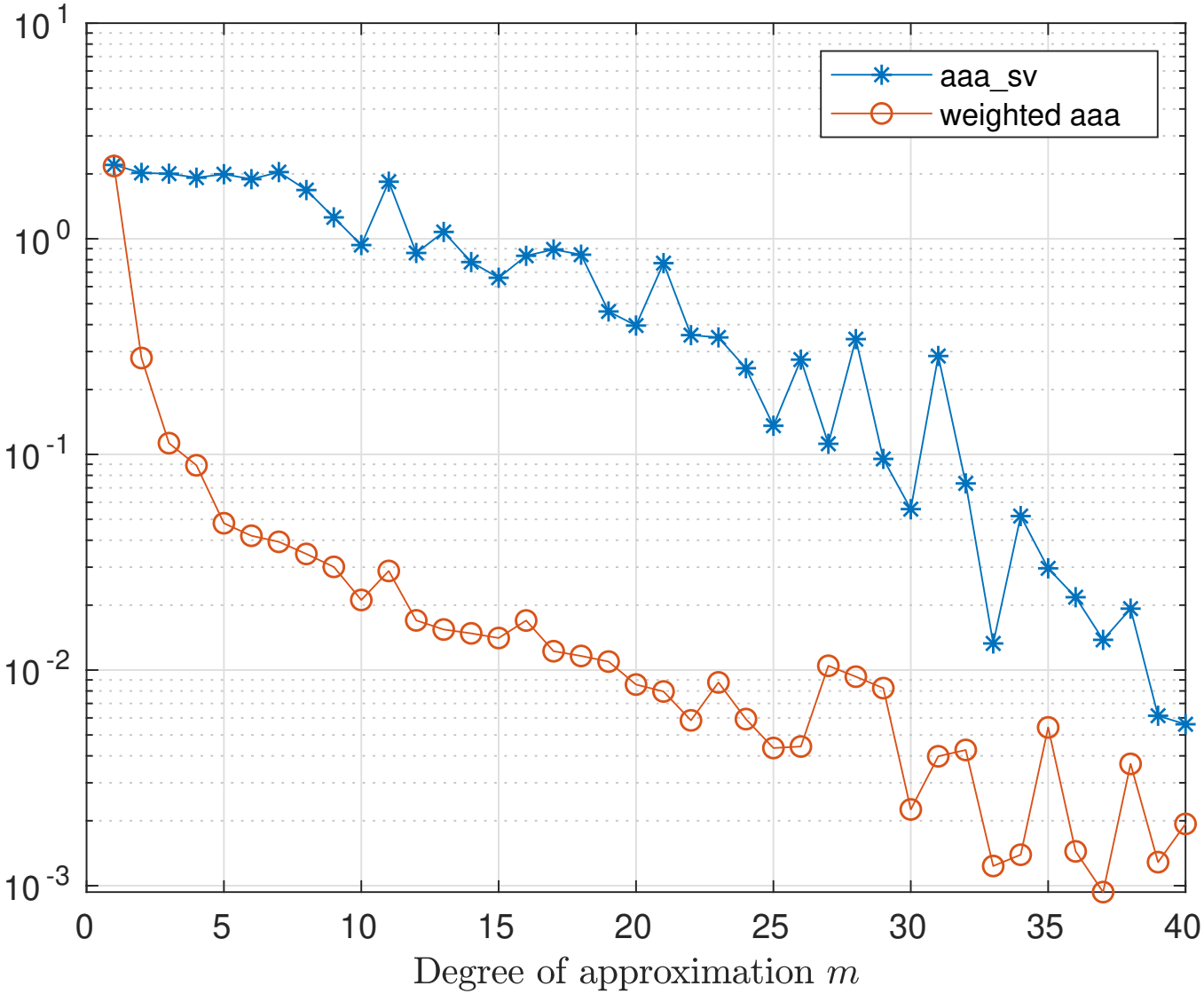

Figure 4: Plot for $\sum_{i=1}^{300} \left\| F(\widetilde{z}_i) - R(\widetilde{z}_i) \right\|_\infty / 300$.

$F(z)$ as a black-box function

What if we do not have $f_k(z)$, but only $F(z)$?

AAA for black-box matrix-valued functions

Güttel and Elsworth [2018] suggested the following strategy:

1. Build $f(z) := v_1^* F(z) v_2$, with v_i Gaussian vectors.
2. Approximate $f(z)$ with AAA algorithm and extract w_j, z_j and $f(z_j)$.
3. Use w_j, z_j and $f(z_j)$ to build an approximation $R(z) \approx F(z)$.

$F(z)$ as a black-box function

What if we do not have $f_k(z)$, but only $F(z)$?

AAA for black-box matrix-valued functions

Güttel and Elsworth [2018] suggested the following strategy:

1. Build $f(z) := v_1^* F(z) v_2$, with v_i Gaussian vectors.
2. Approximate $f(z)$ with AAA algorithm and extract w_j, z_j and $f(z_j)$.
3. Use w_j, z_j and $f(z_j)$ to build an approximation $R(z) \approx F(z)$.

Issue

No robust result about the decay of the approximation error.

A Hybrid strategy: AAA + Leja–Bagby

A summary

- AAA: Doesn't need preassigned poles, but no result on the decay
- Leja–Bagby: Exponential decay, but needs a condenser.

A Hybrid strategy: AAA + Leja–Bagby

A summary

- AAA: Doesn't need preassigned poles, but no result on the decay
- Leja–Bagby: Exponential decay, but needs a condenser.

The best of two worlds

1. Approximate $f(z) = v_1^* F(z) v_2$ with the AAA algorithm.
2. Extract the zeros σ_j and the poles ξ_j from $r(z) \approx f(z)$.
3. Use the zeros and poles as the condenser $\mathrm{cap}(\Sigma, \Gamma)$ for the Leja–Bagby algorithm on $F(z)$.

The Hadeler problem

We approximate the **hadeler** problem

$$F(z) = (e^z - 1)B + z^2 A_2 - 100I,$$

from the NLEVP library, where $B, A_2 \in \mathbb{R}^{600\times 600}$.

- We assume we only have its black-box form, so we consider $f(z) = v_1^T F(z) v_2$.
- We compare the errors of AAA and of AAA plus Leja–Bagby.

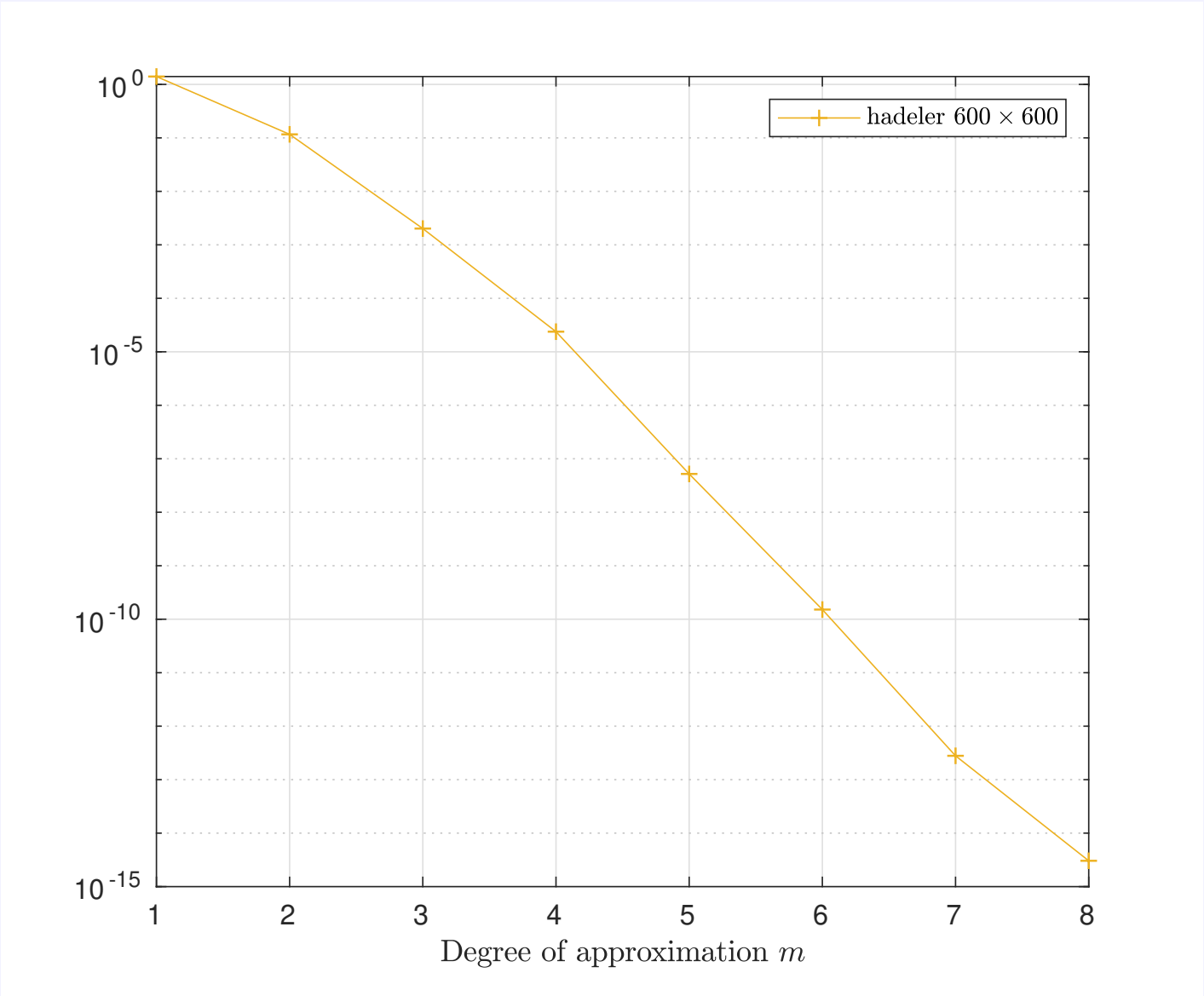

Figure 5: The approximation of $f(z) = v_1^T F(z) v_2$ returned by AAA is very good.

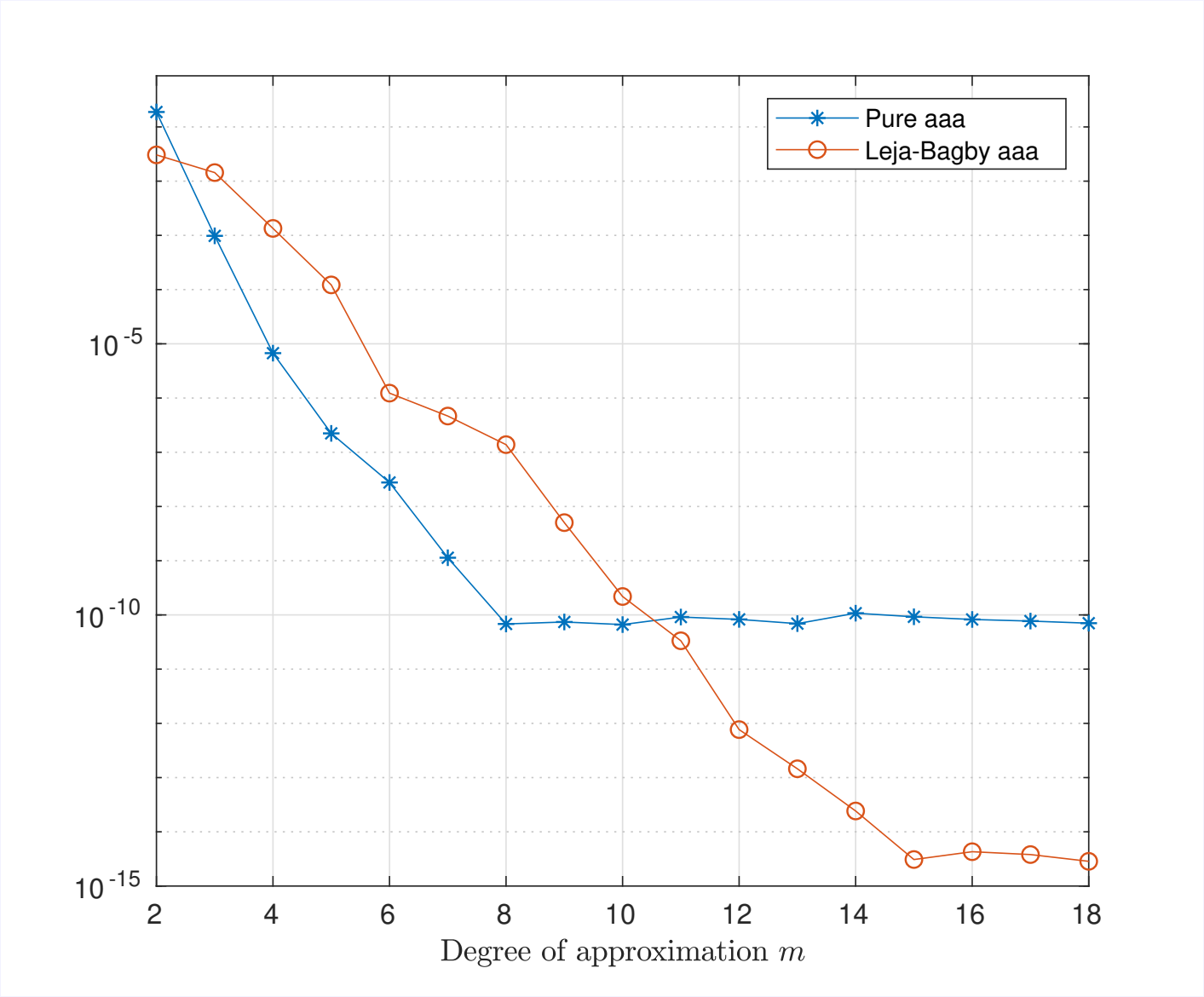

Figure 6: Plot for $\sum_{i=1}^{300} \left\| F(\widetilde{z}_i) - R(\widetilde{z}_i) \right\|_\infty / 300$.

The gun problem

We consider the **gun** problem

$$F(z) = K - zM + \mathrm{i}(z - \sigma_1^2)^{\frac{1}{2}} W_1 \mathrm{i}(z - \sigma_2^2)^{\frac{1}{2}} W_2,$$

where the matrices M, K, W_1, W_2 are real and symmetric of size 9956×9956 and $\sigma_1 = 0$, $\sigma_2 = 108.8774$. We approximate it inside the disk centered in 62500 with radius 50000.

- We assume we only have its black-box form, so we consider $f(z) = v_1^T F(z) v_2$.
- We compare the errors of AAA and of AAA plus Leja–Bagby.

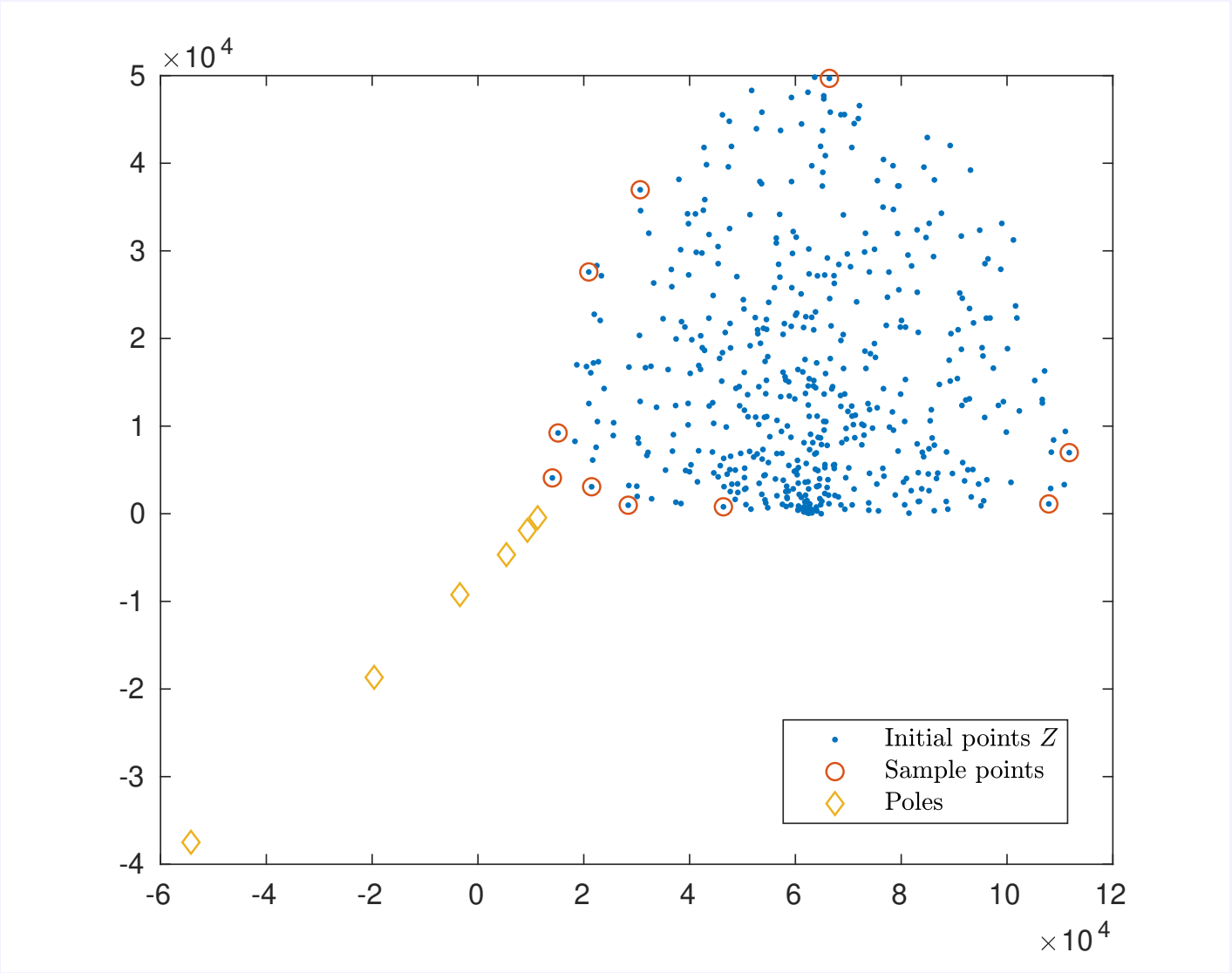

Figure 7: The points used in the AAA algorithm and in Leja–Bagby refinement.

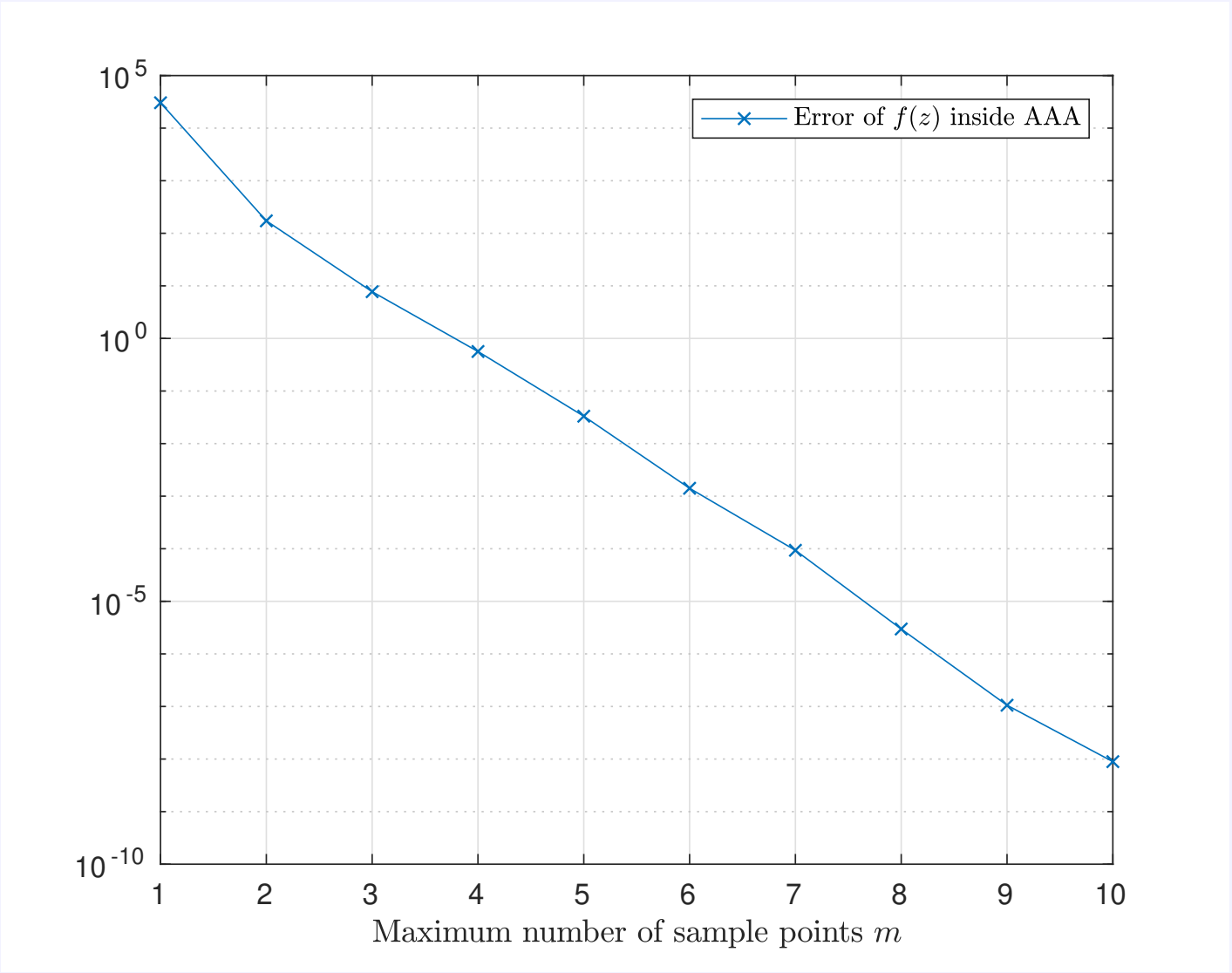

Figure 8: The approximation of $f(z) = v_1^T F(z) v_2$ returned by AAA is good.

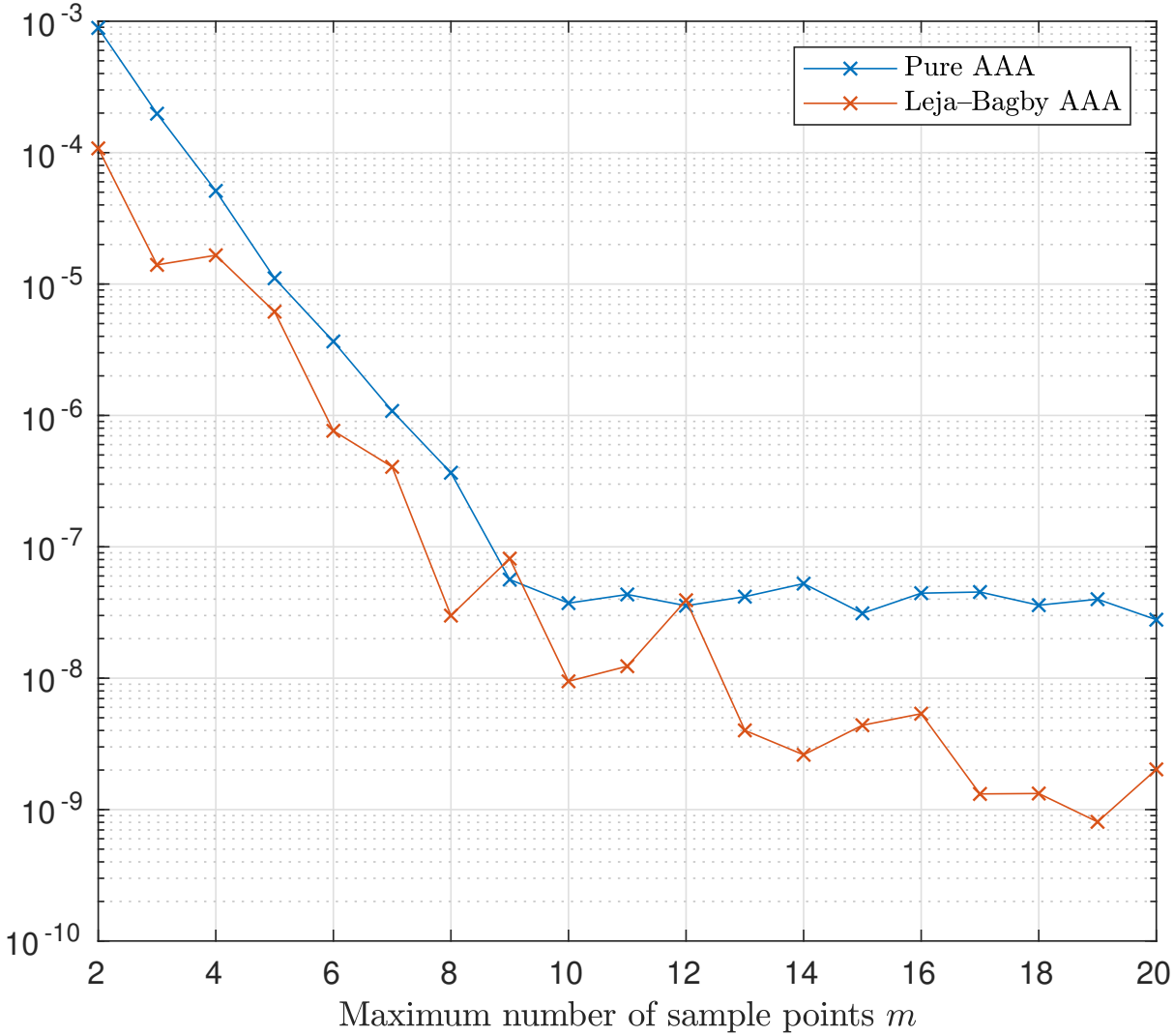

Figure 9: Plot for $\sum_{i=1}^{300} \left\| F(\widetilde{z}_i) - R(\widetilde{z}_i) \right\|_\infty / 300$.

Thanks for your attention.
Questions?

Retrieve zeros and poles from barycentric form

In order to find the zeros of a rational function in barycentric form, we need to solve the linear eigenvalue problem

$$\begin{bmatrix} 0 & w_1 & w_2 & \dots & w_m \\ 1 & z_1 & & & \\ 1 & & z_2 & & \\ \vdots & & & \ddots & \\ 1 & & & & z_m \end{bmatrix} = \lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}.$$

If we want the poles, substitute w_j with $f(z_j)w_j$.

The AAA Cycle

Algorithm 1: Pseudocode for the AAA algorithm

Input: $f, Z, m_{\max}, \epsilon$

Output: $r(z)$.

$Z^{(0)} \leftarrow Z$

$m \leftarrow 0$

while $\|f(z) - r_m(z)\|_{Z^{(m)}} > \epsilon$ *and* $m < m_{max}$ **do**

- Compute $z_m = \arg\max |f d_{m-1} - n_{m-1}|_{Z^{(m-1)}}$
- Remove z_m from $Z^{(m-1)}$ and get $Z^{(m)}$
- Compute the SVD of matrix $A^{(m)}$
- Retrieve the vector of weights w
- Build $r_m(z)$
- $m \leftarrow m + 1$