Midterm project

October 31, 2025

1 Introduction and Problem statement

Traditional rational approximation excels in handling issues involving poles or singularities better than polynomial approximation. However, it faces challenges, such as the appearance of spurious poles, also known as Froissart doublets. Therefore, the author aims to develop an algorithm that is fast and flexible, while effectively avoiding Froissart doublets.

1.1 What is the Froissart doublets

The Froissart doublets refer to poles with very small residues, or to pole–zero pairs that are so close together that they nearly cancel each other. For example, consider

$$r(x) = \frac{x - 1.0000001}{x - 1}$$

In fact $r(x) \approx 1$ and mathematically the pole and zero can almost cancel. However, in computer arithmetic

$$r(x) = 1 + \frac{0.000001}{x - 1}$$

this will be represented as a pole at $x = 1$ and a zero at $x = 1.0000001$. This artificial pole–zero pair may cause numerical instability.

2 Method Overview

In traditional rational approximation, we aim to find a rational function

$$r(x) = \frac{p(x)}{q(x)} = \frac{\sum_{i=0}^{m} a_i x^i}{\sum_{i=0}^{n} bix^i}$$

However, solving for the coefficients a_i and b_i typically involves ill-conditioned systems, and the resulting approximation may introduce spurious artifacts such as Froissart doublets.

2.1 Barycentric formula

Here, the AAA algorithm base on the barycentric formula, given by

$$r(x) = \frac{n(x)}{d(x)} = \frac{\sum_{j=1}^{m} \frac{w_j f(x_j)}{x - x_j}}{\sum_{j=1}^{m} \frac{w_j}{x - x_j}}.$$

where $m > 1$ is an integer and $x_1, \ldots, x_m$ are a set of real or complex distinct support points, $f(x_1), \ldots, f(x_m)$ are a set of real or complex data values, and $w_1, \ldots, w_m$ are a set of real or complex weights. The AAA procedure incrementally adds support points one by one, and at each step determines the weights w_j. This process continues until the approximation satisfies the error criterion

$$\|f(x) - r(x)\| < \epsilon$$

where ϵ is a prescribed tolerance, and $\|.\|$ is the discrete 2-norm. Notice that, with the nodal polynomial, the Barycentric formula give a rational function of type $(m-1, m-1)$, and also according to the Theorem 2.1 in the paper, this kind of rational function range over the set of rational function of type $(m-1, m-1)$ that has no pole at point x_i.

2.2 The process of AAA algorithm

Let $X = \{x_1, x_2, \ldots, x_m\}$ be the set of sample point, with corresponding function value $\{f_1, f_2, \ldots, f_m\}$. We begin with the empty set $S = \phi$. At each iteration, we select the new support point from X where the approximation error is largest, that is,

$$\arg\max_{x \in X} \|f(x) - r(x)\|.$$

The selected point is then added to the support set S. Using the point in S, we construct the rational approximation in barycentric form:

$$r(x) = \frac{n(x)}{d(x)} = \frac{\sum_{j=1}^{m} \frac{w_j f(x_j)}{x - x_j}}{\sum_{j=1}^{m} \frac{w_j}{x - x_j}},$$

where $x_j \in S$, then, the weights w_j are determined by solving a least-squares problem:

$$\min_{w_j} \|f(x)d(x) - n(x)\|, \text{subject to } \|w_j\| = 1,$$

where $n(x)$ and $d(x)$ are the numerator and denominator of the barycentric form. Finally, we check whether the error between $r(x)$ and $f(x)$ is less than the prescribed tolerance ϵ. If the error criterion is not satisfied, a new support point is added from X, and the process is repeated.

2.3 Solving the Least Squares Problem for w_j

We want the rational approximation $r(x)$ to be as close as possible to $f(x)$. This leads to the following constrained least squares problem:

$$\min_{w_j} \|f(x)d(x) - n(x)\|, \quad \text{subject to } \|w_j\| = 1.$$

Expanding the numerator and denominator in barycentric form, the least squares problem can be rewritten as

$$\min_{w_j} \left\| f(x) \sum_{j=1}^{m} \frac{w_j}{x - x_j} - \sum_{j=1}^{m} \frac{w_j f(x_j)}{x - x_j} \right\| = \min_{w_j} \left\| \sum_{j=1}^{m} \frac{w_j \left[f(x) - f(x_j) \right]}{x - x_j} \right\|. \tag{1}$$

Here,

$$w^{(m)} = \begin{bmatrix} w_1 & w_2 & \dots & w_m \end{bmatrix}^\top, \qquad \|w^{(m)}\| = 1$$

is a normalized column vector. Now define the *Loewner matrix* $A^{(m)}$, whose entries are given by

$$A^{(m)}_{k,j} = \frac{f(x_k) - f(x_j)}{x_k - x_j}, \quad k \neq j,$$

then the least squares problem (1) can be compactly expressed as

$$\min_{w^{(m)}} \|A^{(m)} w^{(m)}\|. \tag{2}$$

Let the singular value decomposition of $A^{(m)}$ be

$$A^{(m)} = U \Sigma V^*.$$

By the orthogonality of U and V, setting $y = V^* w^{(m)}$ transforms (2) into

$$\min_{y} \|\Sigma y\|, \quad \|y\| = 1.$$

Since Σ is the diagonal matrix of singular values of $A^{(m)}$, the optimal choice is

$$y = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \end{bmatrix}^\top,$$

which implies that $w^{(m)}$ is the right singular vector corresponding to the smallest singular value of $A^{(m)}$.

3 Simple Implementation / Experiment

In this section, we present a small-scale implementation of the AAA algorithm in Python in order to demonstrate the core idea of adaptive rational approximation. Our goal is not to reproduce the full experimental results of the paper, but rather to validate the method on representative test functions and to analyze its numerical behavior.

3.1 Gamma function

We first applied our simplified AAA implementation to approximate the Gamma function, we want to reduce Figure 4.2 in the original paper. The algorithm adaptively selects support points on the boundary of the domain, and the resulting approximation closely matches the original figure. This demonstrates that the Python implementation successfully replicates the behavior reported in the paper.

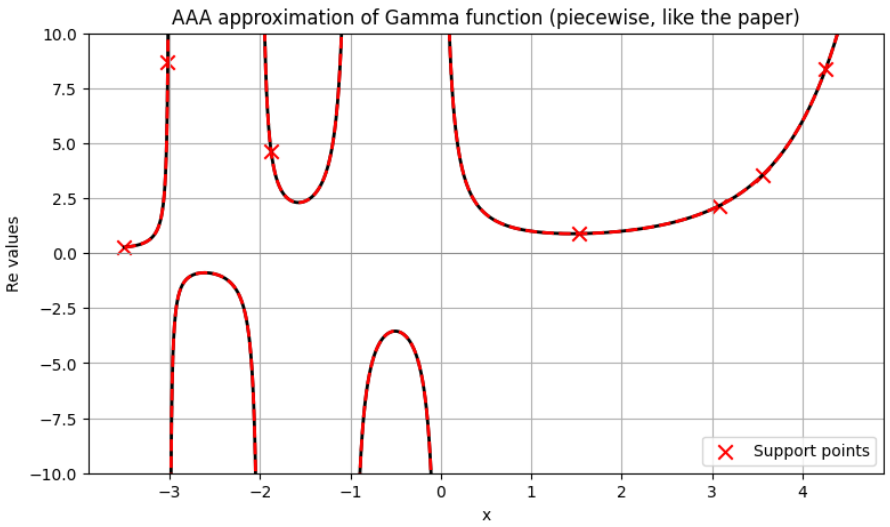

Figure 1: Using the *AAA* algorithm to approximate the gamma function.

3.2 Runge function

Next, we tested the AAA algorithm on the Runge function

$$f(x) = \frac{1}{1+25x^2}, \qquad x \in [-1, 1],$$

The AAA algorithm was applied with two different tolerance levels.

Case 1: tol $= 10^{-13}$**.** The algorithm terminated after selecting only three support points, yet the approximation error reached about 10^{-15}, which is essentially machine precision. This shows the effectiveness of the barycentric rational form in avoiding large oscillations.

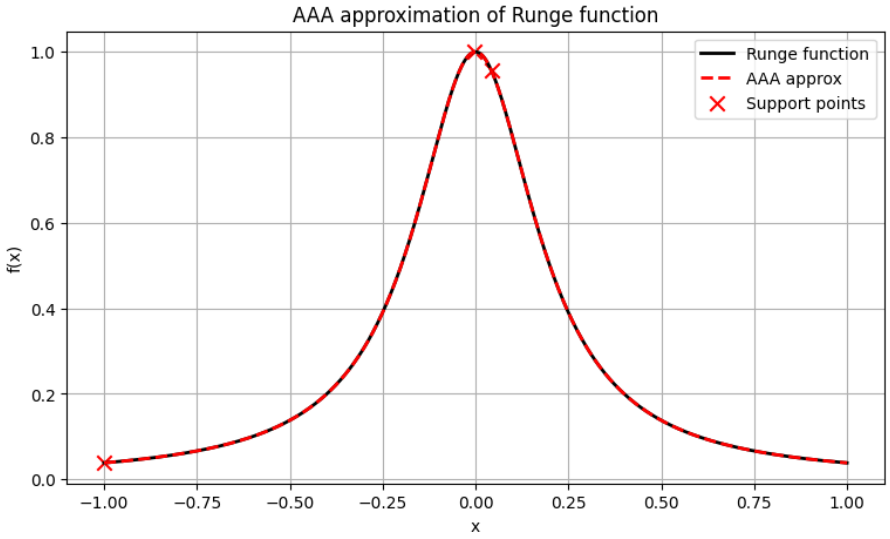

Figure 2: The approximation of Runge function using *AAA* algorithm with tol=10^{-13}.

```
Weights w:
[-0.55340644+0.j  0.6224845 -0.j  0.55340253-0.j]
support point = [-0.00250627 -1.          0.04260652]
the number of support points = 3
final max error ≈ 1.1102230246251565e-15
```

Figure 3: The approximation of Runge function using AAA algorithm with tol=10^{-13}.

Case 2: tol = 10^{-15}**.** When the tolerance was tightened to 10^{-15}, the algorithm continued to select up to twenty support points. However, the maximum error did not decrease beyond 10^{-15}, due to the limitation of double-precision arithmetic. This indicates that additional support points beyond a certain threshold do not improve accuracy, but instead reflect the numerical attempt of the algorithm to overfit within machine precision.

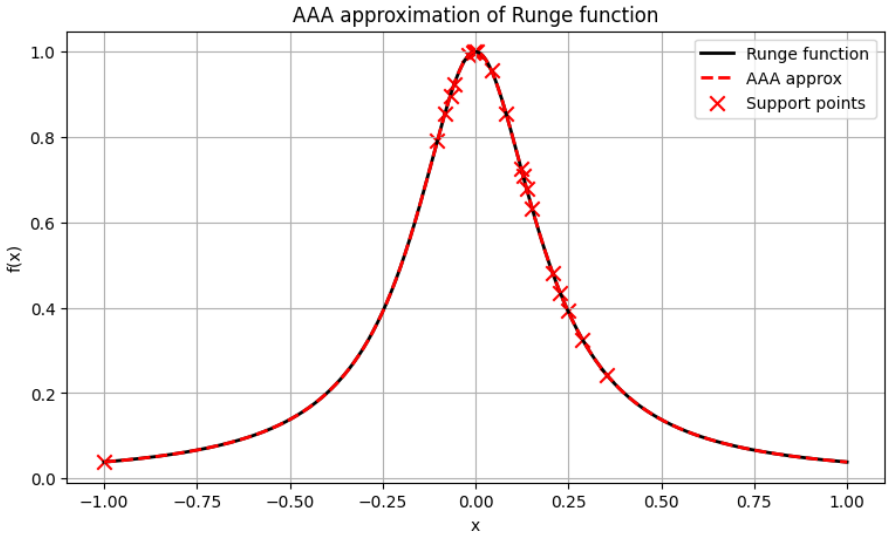

Figure 4: The approximation of Runge function using AAA algorithm with tol=10^{-15}.

```
Weights w:
[ 0.1969162 -0.j -0.93117084+0.j -0.00153044+0.j -0.01042243+0.j
 -0.09407221+0.j -0.04348978+0.j -0.05302547+0.j -0.00460712+0.j
  0.06266929-0.j -0.00644867+0.j  0.16408539-0.j -0.02153427+0.j
 -0.1040122 +0.j  0.06078668-0.j -0.04486053+0.j -0.15752857+0.j
  0.02717933-0.j -0.02356315+0.j -0.07948117+0.j  0.01227996-0.j]
support point = [-0.00250627 -1.          0.04260652 -0.10275689  0.00250627  0.35338346
 -0.0075188   0.08270677  0.12280702  0.20802005 -0.01754386  0.28822055
  0.12781955  0.2481203   0.15288221  0.22807018 -0.08270677 -0.05764411
  0.13784461 -0.06766917]
the number of support points = 20
final max error ≈ 2.7755575615628914e-15
```

Figure 5: The approximation of Runge function using AAA algorithm with tol=10^{-15}.

3.3 Discussion

From these experiments we observe that:

- The AAA algorithm can achieve near machine-precision accuracy with only a few support points.
- Increasing support points beyond necessity may lead to spurious pole-zero pairs (Froissart doublets), which highlights both the strength and the limitation of the AAA approach.

Overall, these results confirm that the AAA algorithm provides an efficient and adaptive method for rational approximation.

4 Summary and Conclusion

The AAA algorithm provides an efficient and adaptive framework for rational approximation. It performs particularly well for smooth or moderately analytic functions, such as the Runge function and the Gamma function examined in this report. Unlike traditional polynomial interpolation, which often suffers from oscillations near the boundaries (the Runge phenomenon), the AAA algorithm constructs rational approximation in barycentric form. This formulation allows it to approximate functions with poles or rapid variations accurately and stably.

One of the key strengths of the AAA algorithm lies in its automatic and adaptive support-point selection. By iteratively identifying regions with the largest approximation error, it can achieve high accuracy with only a few support points. Our experiments demonstrate that, for smooth functions, only three support points are sufficient to reach an accuracy close to machine precision. When the tolerance is set tighter than the achievable precision of floating-point arithmetic, the algorithm continues to add more support points, increasing computational cost without any further improvement in accuracy. This behavior reflects the numerical limits imposed by machine precision rather than an inherent flaw in the algorithm itself.

In summary, the AAA algorithm is highly effective for rational approximation of well-behaved functions and requires minimal user intervention. Its main limitations arise from numerical precision and the conditioning of the Loewner matrix when very high accuracy is requested. Future improvements could focus on more robust stopping criteria to avoid unnecessary iterations and on extending the algorithm to handle multi-dimensional data, noisy measurements, or time-varying systems.