Midterm Project-
THE AAA ALGORITHM FOR RATIONAL APPROXIMATION

314652013 陳勇捷

1 Introduction and Problem Statement

1.1 Research Problem

The paper addresses the challenge of constructing accurate and numerically stable rational approximations

$$r(z) = \frac{p(z)}{q(z)} \approx f(z)$$

for given real or complex functions, **without prior tuning** (e.g., specifying the number or location of poles) and independently of domain geometry.

1.2 Significance and Motivation

In practical computation, rational functions are essential tools in numerical analysis and scientific computing because they can represent **poles, singularities, or unbounded behaviors** much more efficiently than polynomials. They also enable for **analytic continuation and extrapolation**, allowing researchers to infer behavior of a function function beyond its sampled region.
Moreover, many practical problems involve irregular or disconnected domains, where a general-purpose, domain-independent approximation method is highly valuable.

1.3 Challenges in Existing Methods

Although rational approximations have been studied for decades, traditional approaches such as **Padé approximation** and **vector fitting** still face several fundamental limitations.

- **Numerical instability:** These methods often produce spurious poles (Froissart doublets) and lead to ill-conditioned p/q representations.
- **Ill-posedness:** Analytic continuation is inherently ill-posed, and small perturbations in the data can cause large changes in the approximation.
- **Dependence on user and geometry:** Classical algorithms require preselecting model degrees or pole positions and rely on domain-specific polynomial bases (e.g., intervals or disks), reducing generality and automation.

1.4 The AAA Algorithm Framework

The paper introduces **the AAA algorithm (Adaptive Antoulas–Anderson)** as a robust and flexible alternative. Its design integrates two core innovations that jointly ensure numerical stability and adaptability:

- **Barycentric rational representation:**
  The algorithm expresses the approximation in a barycentric form, which provides a well-conditioned numerical framework.
  By using localized basis functions $1/(z-z_j)$, this representation avoids the ill-conditioning and coefficient growth associated with traditional polynomial-based formulations.
- **Adaptive greedy selection of support points:**
  At each iteration, the algorithm identifies the sample point with the largest current error and adds it as a new support point.
  This error-driven refinement adaptively distributes the support points across the domain, capturing local complexity and preventing exponential instabilities.

The AAA algorithm operates directly on arbitrary sets of sample points, requires no user-specified parameters, and typically suppresses numerical Froissart doublets.
Overall, this makes it a domain-independent, automatic, and highly stable approach to rational approximation.

2 Method Explanation: The AAA Algorithm

The core idea of the AAA algorithm is to build a rational function that automatically fits a given function $f(z)$ as accurately and stably as possible, without any manual tuning or domain assumptions.
Using the barycentric representation makes the computation much more numerically stable, and the algorithm further applies a greedy strategy to update the support points at each step. The whole process can be imagined as the algorithm "learning" which points of the function matter the most and adding them one by one.

2.1 Barycentric Representation

$$r(z) = \frac{n(z)}{d(z)} = \frac{\sum_{j=1}^{m} \frac{w_j f_j}{z-z_j}}{\sum_{j=1}^{m} \frac{w_j}{z-z_j}}$$

,where z_j are support points chosen from the sample set and w_j are weights determined by a small least-squares problem.
This form automatically satisfies $r(z_j) = f_j$ and avoids the ill-conditioning of polynomial bases since the functions $1/(z-z_j)$ are local and nearly independent.The resulting rational function is of type $(m-1, m-1)$.

2.2 Greedy Iterative Procedure

The AAA algorithm proceeds iteratively, combining greedy selection with least-squares fitting:

1. **Initialization:** Start with a discrete set of sample points $Z = \{z_1, z_2, \ldots, z_M\}$ and their corresponding data $f(Z) = \{f_1, f_2, \ldots, f_M\}$.Initially, no support points are selected $S_0 = \emptyset$ and set the initial rational approximation as $r_0(z) = 0$.

2. **Support point selection:**

   - At step m, compute the current rational approximation $r_{m-1}(z)$.
   - Identify the point $z_m \in Z \setminus S_{m-1}$, where the approximation error is largest:

     $$z_m = \arg \max_{x \in Z \setminus S_{m-1}} |f(z) - r_{m-1}(z)|$$

     This ensures that each new support point is added exactly where the current model deviates most from the target function.
   - Update the support set, add z_m to the support set $S_m = S_{m-1} \cup \{z_m\}$.

   This greedy rule allows the algorithm to focus its attention on difficult regions, such as near singularities or rapid variations in $f(z)$.

3. **Weight Determination (By Least-Squares):** Once the new support point is added, the algorithm recomputes the barycentric weights $w_1, \ldots, w_m$ so that the new rational function best fits the data.
   The weights are determined by minimizing the residual

   $$\min_{\|w\|_m=1} \|fd - n\|_{Z^{(m)}}$$

   which is equivalent to solving the matrix problem

   $$\min_{\|w\|_m=1} \|A^{(m)}w\|_{M-m}, \quad A_{ij}^{(m)} = \frac{F_i^{(m)} - f_j}{Z_i^{(m)} - z_j}$$

   The optimal w is obtained from the right singular vector corresponding to the smallest singular value in the SVD

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

4. **Updating the Approximation:**After the new weights are found, the barycentric numerator and denominator are updated using

   $$N = C(wf), \quad D = Cw$$

   , where $C_{ij} = \frac{1}{Z_i^{(m)} - z_j}$.
   These correspond to sampled values of the numerator $n(z)$ and denominator $d(z)$.
   The new rational approximant is then updated as $r_m(z) = n(z)/d(z)$. This process repeats, each time using the latest r_m to look for the next weak spot.

5. **Stopping and Refinement:**The iteration repeats until the nonlinear residual is sufficiently small, typically below a default tolerance of 10^{-13}.
   In most cases, it has few or no Froissart doublets (spurious poles and zeros that nearly cancel each other).If any are detected, they can be removed by performing one final least-squares step.

3 Simple Implementation / Experiment

3.1 Gamma function

We next applied our MATLAB implementation of the AAA algorithm to approximate the Gamma function $f(x) = \Gamma(x)$ on the interval $[-3.5, 4.5]$.

The goal is to reproduce the characteristic behavior shown in Figure 4.2 of the original paper.
The algorithm automatically selects a small set of support points, shown as red dots in the figure, by iteratively locating the positions where the current residual is largest. The resulting rational approximation $r(x)$, plotted in blue, accurately captures the overall structure of the Gamma function, including its poles at $x = 0, -1, -2, -3$. We observe that most support points are concentrated near these singularities, indicating that the algorithm effectively adapts its node distribution to the regions where the function varies most rapidly.

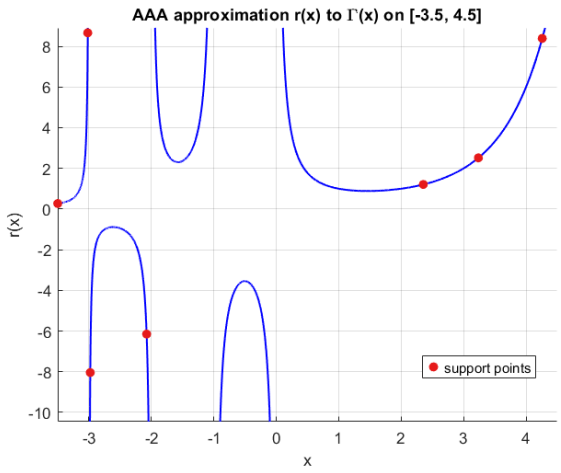

Figure 1: Using the AAA algorithm to approximate the gamma function

4 Summary or conclusion

In summary, the AAA algorithm is a clear and practical method for adaptive rational approximation.
It works by writing a function in barycentric form and then adding support points step by step at the places where the current error is largest. This simple idea allows the algorithm to approximate functions that are hard for polynomial methods, especially those with poles or sharp changes. With only a small number of support points, the AAA method can reach high accuracy and remain numerically stable, making it useful when data points are uneven or expensive to obtain.

Still, the method also has some limits.
Its performance can drop when the data include noise or sudden jumps, or when the function values vary by many orders of magnitude, since the least-squares and SVD steps may become unstable. For very large datasets, the cost of repeatedly building and decomposing the Loewner matrix can also be high.
In addition, the choice of tolerance and stopping rule depends on the problem; a poor choice may cause underfitting or overfitting.

Possible improvements include adding regularization to handle noisy data, using faster or randomized SVD routines for large-scale cases, and extending the idea to higher-dimensional or matrix-valued functions.
Overall, the AAA algorithm shows a good balance between simplicity, adaptivity, and accuracy, and it remains a powerful and stable approach for rational approximation in many numerical problems.