The Journal of Open Source Software

RationalFunctionApproximation.jl: A Julia package for approximation by rational functions

Tobin A. Driscoll [1]

1 Department of Mathematical Sciences, University of Delaware, Newark, DE, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

In scientific and engineering problems, it is often necessary or useful to replace an exact but lengthy, or even unknown, expression for a varying value by an efficient approximation. Classically, this is known as the problem of *function approximation* in mathematics. In function approximation, the ideal is to find a constructive procedure that achieves high accuracy with as succinct a representation as possible. One class of approximations that can do an outstanding job in this regard are the *rational functions*. A rational function is the ratio of two polynomials, which themselves are the combination of repeated multiplication and addition. For theoretical reasons, it is known that rational functions are excellent for representing many commonly encountered functions. Algorithmic developments within the last five years have opened a way to rapidly find rational approximations that are customized to a given problem. This package provides that capability to the Julia language.

Statement of need

The `RationalFunctionApproximation.jl` package provides the automatic approximation of a function of a single variable by means of a rational function. Function approximation is a useful tool in scientific computation, as evidenced by the success of Chebfun in Matlab (Tobin A. Driscoll et al., 2014) and ApproxFun in Julia (Olver & Townsend, 2014), both of which are based on Chebyshev and trigonometric polynomial approximations.

Rational functions are a generalization of polynomials that are well-known to improve on polynomials' approximation power for functions that have singularities located near the approximation domain. The AAA algorithm (Nakatsukasa et al., 2018) opened the door to fast adaptive construction of rational interpolants of a function. A number of interesting applications rapidly followed, e. g., Budisa et al. (2022), Costa & Trefethen (2023), Deng & Lustri (2023), Derevianko et al. (2023), Gopal & Trefethen (2019). The `BaryRational` package implements the original AAA algorithm in Julia.

More recently, a new variation of AAA (Tobin A. Driscoll et al., 2023) treats the domain of the given function as a continuum, rather than requiring it to be discretized. This can have significant advantages when singularities are very close to the domain, since exponential spacing of nodes is required. `RationalFunctionApproximation.jl` implements this new algorithm in Julia, incorporating the `ComplexRegions.jl` package(Tobin A. Driscoll, 2019) to provide general curves and regions as domains in the complex plane.

Usage examples

We approximate the function

DRAFT

$$f(x) = \frac{0.01}{\sin(x^2 + 0.2x + 0.02)},$$

which has an off-center hump over the interval $[-1, 1]$, using the following:

```
julia> using RationalFunctionApproximation
julia> f = x -> 1/10^2 / sin(x^2 + 2x/10 + 2/10^2);
julia> r = approximate(f, unit_interval)
Barycentric rational function of type (10,10) on the domain: Path with 1 curve
```

The result is a rational function of type $(10, 10)$, meaning that the numerator and denominator are polynomials of degree 10. The approximation interpolates the original function at 11 greedily selected points retrievable as `nodes(r)`, and it is accurate to over 13 digits over the domain, as shown in Figure 1.

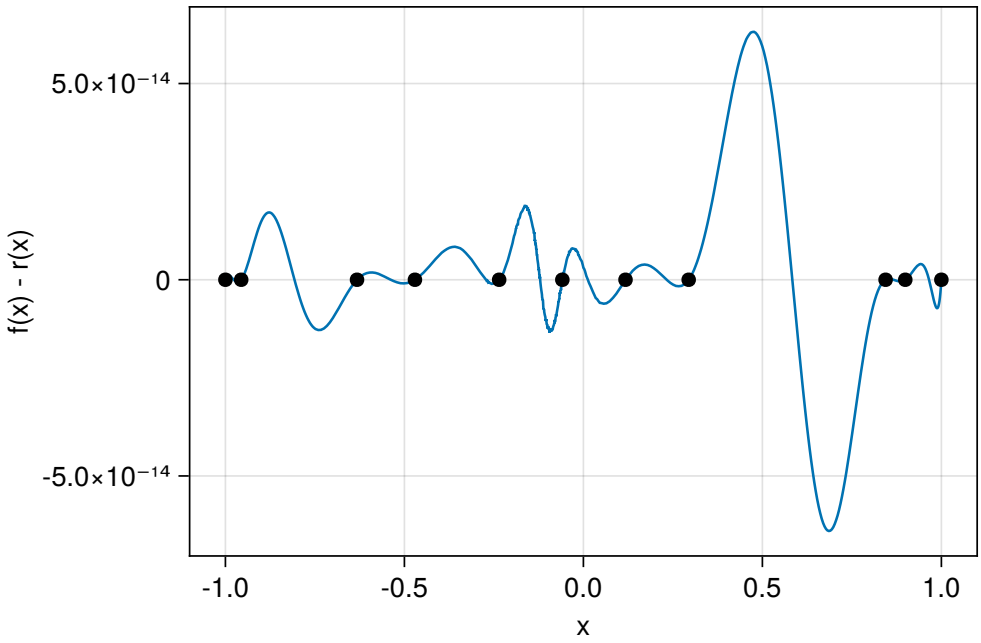

Figure 1: Error in a rational approximation.

The approximation can be visualized in the complex plane using the `DomainColoring` package, which uses hue to indicate phase and brightness to show contours of log magnitude. The result is shown in Figure 2.

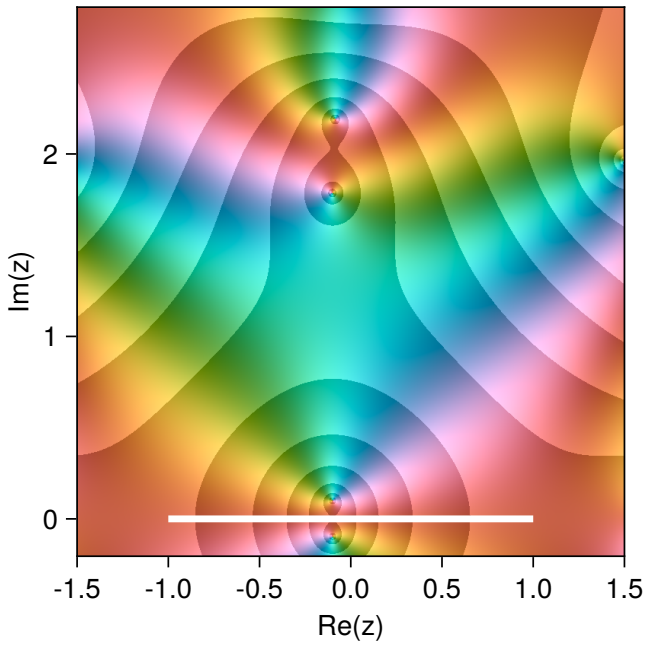

Figure 2: Rational approximation in the complex plane.

As one sees in the figure, there are poles in the approximation that are extremely close to the exact poles of f at $x = -0.1 \pm 0.1i$:

```
julia> p = poles(r)
10-element Vector{ComplexF64}:
   -2.515988566699771 + 0.0im
  -1.8696464689420669 + 0.0im
  -0.1012939445069749 - 1.7844874038085 64im
 -0.10129394450697489 + 1.7844874038085 64im
 -0.09999999999999569 - 0.09999999999998077im
 -0.09999999999999569 + 0.09999999999998077im
 -0.08503139961597209 - 2.1919510617575773im
 -0.08503139961597207 + 2.1919510617575777im
   1.6697943872716796 + 0.0im
   2.3073258267815193 + 0.0im
```

We can also find the residues at those poles:

```
julia> residues(r, p[5:6])
 -4.686750511456871e-15 + 0.049999999999976126im
 -4.686750511456871e-15 - 0.049999999999976126im
```

As seen in Figure 1, the rational approximation is not quite uniformly accurate over the domain, because the AAA algorithm uses a least-squares criterion. However, there is an iteration due to Lawson that is employed by `minimax(r)` to more nearly recover the rational approximation that is optimal in the infinity norm. The resulting error, shown in Figure 3, is roughly equioscillatory.

DRAFT

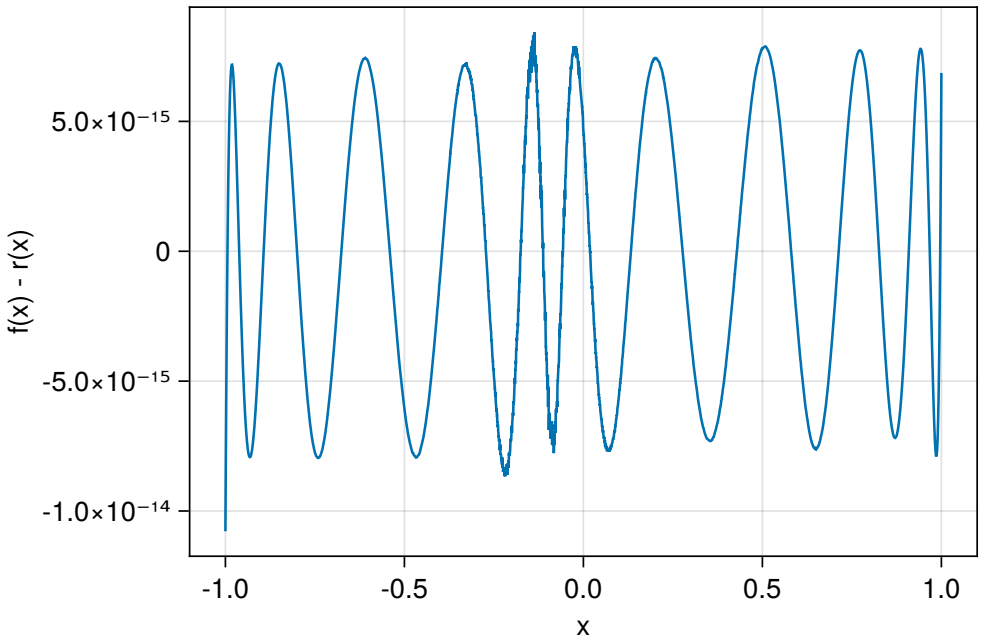

Figure 3: Error in a nearly-minimax rational approximation.

We can see the usefulness of the improved, continuous AAA algorithm (Tobin A. Driscoll et al., 2023) by approximating a function that has a singularity very close to its domain, such as

$$g(x) = \tanh(5000(x - 0.2)).$$

If the proximity of the pole is comparable to or less than the sample spacing, the approximation by the original, discrete AAA method (which is implemented in the package) will be poor:

```
julia> g = x -> tanh(5000 * (x - 0.2));
julia> x = -1:0.001:1;
julia> r = aaa(x, g.(x));
julia> xx = -1:.0001:1;
julia> println("error = $(maximum(abs, @. g(xx) - r(xx) ))");
error = 0.1401166557529342
```

The error could be much reduced by using a finer grid of samples. But the continuous AAA algorithm finds those samples automatically:

```
julia> r = approximate(g, unit_interval);
julia> println("error = $(maximum(abs, @. g(xx) - r(xx) ))");
error = 4.0453862482081604e-11
```

The domain of an approximation can be any region defined using the `ComplexRegions` package. Here, we approximate the function $\tan(1/z^4)$ by a rational function that is analytic in the exterior of the unit circle, resulting in Figure 4.

```
r = approximate(z -> tanh(1/z^4), exterior(unit_circle));
```

Approximation of $\tan(1/z^4)$.

Figure 4: Approximation of $\tan(1/z^4)$.

DRAFT

Finally, here is the approximation of a function with a branch cut over the interior of a cross-shaped region:

```
julia> import ComplexRegions.Shapes, ComplexPlots
julia> r = approximate(z -> log(0.35 + 0.4im - z), interior(Shapes.cross))
Barycentric rational function of type (5,5) on the domain: Region interior to Polygon wi
```

The result is shown in Figure 5.

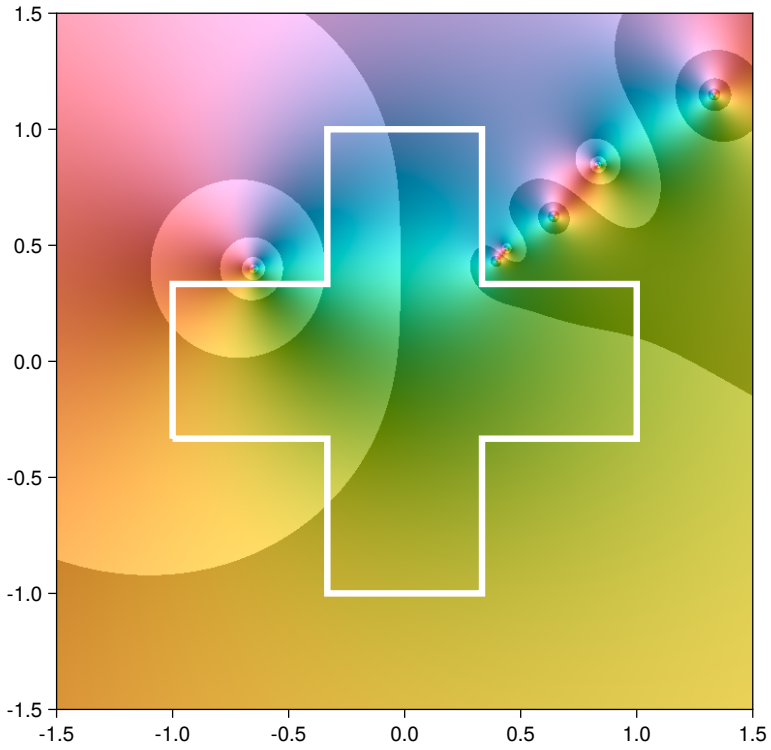

Figure 5: Approximation of a log function.

Acknowledgements

I acknowledge with gratitude the help and support of my paper coauthors, Yuji Nakatsukasa and Nick Trefethen. I also thank Daan Huybrechs for a valuable Julia-specific suggestion.

References

Budisa, A., Hu, X., Kuchta, M., Mardal, K.-A., & Zikatanov, L. (2022). *Rational approximation preconditioners for multiphysics problems* (No. arXiv:2209.11659). arXiv. https://arxiv.org/abs/2209.11659

Costa, S., & Trefethen, L. N. (2023). AAA-least squares rational approximation and solution of Laplace problems. In A. Hujdurović, K. Kutnar, D. Marušič, Š. Miklavič, T. Pisanski, & P. Šparl (Eds.), *European Congress of Mathematics* (1st ed., pp. 511–534). EMS Press. https://doi.org/10.4171/8ecm/16

Deng, G., & Lustri, C. J. (2023). Exponential asymptotics of woodpile chain nanoptera using numerical analytic continuation. *Studies in Applied Mathematics*, *150*(2), 520–557. https://doi.org/10.1111/sapm.12548

Derevianko, N., Plonka, G., & Petz, M. (2023). From ESPRIT to ESPIRA: Estimation of signal parameters by iterative rational approximation. *IMA Journal of Numerical Analysis*, *43*(2), 789–827. https://doi.org/10.1093/imanum/drab108

Driscoll, Tobin A. (2019). *ComplexRegions.jl: A Julia package for regions in the complex plane*.

Driscoll, Tobin A., Hale, N., & Trefethen, L. N. (2014). *Chebfun guide*. Pafnuty Publications, Oxford.

Driscoll, Tobin A., Nakatsukasa, Y., & Trefethen, L. N. (2023). AAA rational approximation on a continuum. *SIAM Journal on Scientific Computing*, to appear.

Gopal, A., & Trefethen, L. N. (2019). Solving Laplace Problems with Corner Singularities via Rational Functions. *SIAM Journal on Numerical Analysis*, *57*(5), 2074–2094. https://doi.org/10.1137/19m125947x

Nakatsukasa, Y., Sète, O., & Trefethen, L. N. (2018). The AAA Algorithm for Rational Approximation. *SIAM Journal on Scientific Computing*, *40*(3), A1494–A1522. https://doi.org/10.1137/16m1106122

Olver, S., & Townsend, A. (2014). A practical framework for infinite-dimensional linear algebra. *Roceedings of the 1st Workshop for High Performance Technical Computing in Dynamic Languages – HPTCDL '14*. https://doi.org/10.1109/HPTCDL.2014.10

DRAFT