Pade Approximants and the Remez Exchange Algorithm

Ryan Maguire

January 24, 2023

Outline

- Taylor / Maclaurin Series
- Chebyshev Polynomials
- The Remez Exchange Algorithm
- Pade Approximants

Polynomial Approximations

If you have an analytic real-valued function you could, in principle, compute the function to arbitrary precision using the Taylor series.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \tag{1}$$

where $f^{(n)}$ denotes the n^{th} derivative of f, and $f^{(0)}(x_0)$ is simply $f(x_0)$. This is likely familiar to any student of Calculus.

Polynomial Approximations

Given a fixed $N \in \mathbb{N}$ one might ask what is the *best* polynomial approximation of degree N of a function f on some interval $[a, b]$? There are a few ways to phrase what *best* should mean.

- Is easy to understand and implement.
- Minimizes the root-mean-square error.
- Minimizes the sup norm of f on $[a, b]$.

Polynomial Approximations

The anwer to 1.) is probably the Taylor / Maclaurin series. It is rather easy to understand and implement, and does a good job approximating a function near the point x_0. There are several theorems, like the alternating series test, that also give bounds on the error.

Least-squares methods give the answer to 2.), and statisticians, applied mathematicians, and physics make frequent use of this.

We want to deal with the third problem. The issue with only caring about the RMS error is that there may be parts of $[a, b]$ where the approximation is horrible. Minimizing the sup norm means your approximation is good for all numbers under consideration.

Chebyshev Polynomials

To solve this we use the Remez exchange algorithm. This involves Chebyshev polynomials, so it'd be useful to discuss these briefly. The defining characteristic is:

$$T_n\big(\cos(\theta)\big) = \cos(n\theta) \tag{2}$$

but it is easier to work with their recurrence relation:

$$T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x) \tag{3}$$

where $T_0(x) = 1$ and $T_1(x) = x$.

Chebyshev Polynomials

These polynomials come from a Sturm-Liouville system:

$$(1 - x^2)\ddot{y} - x\dot{y} + n^2 y = 0 \tag{4}$$

On the interval $[-1, 1]$ the weight is $w(x) = (1 - x^2)^{-\frac{1}{2}}$ meaning we can approximate smooth functions on $[-1, 1]$ via:

$$f(x) = \sum_{n=0}^{\infty} a_n T_n(x) = \sum_{n=0}^{\infty} T_n(x) \int_{-1}^{1} \frac{f(x) T_n(x)}{\sqrt{1 - x^2}} \, \mathrm{d}x \tag{5}$$

and stopping this sum at some appropriate integer $N \in \mathbb{N}$.

Chebyshev Polynomials

Chebyshev polynomials have been well-studied and their use is widespread. Evaluation of special functions, like Bessel functions and Fresnel integrals, often involves Chebyshev expansions at some point.

The Remez algorithm that we'll be discussing uses the extrema of these polynomials. From $T_n(\cos(\theta)) = \cos(n\theta)$ we can see that the $|T_n(x)|$ attains it's maximums at:

$$x_k = \cos\left(\frac{k\pi}{n}\right) \tag{6}$$

for $0 \leq k \leq n$.

The Remez Exchange Algorithm

Now we want to find the best polynomial approximation of a smooth function f on some interval $[a, b]$. We can linearly translate this back to $[-1, 1]$, the domain of the Chebyshev polynomials, and then translate back at the end, so for ease we suppose $a = -1$ and $b = 1$. We start with $N + 2$ samples $x_0, \ldots, x_{N+1}$, where N is the desired degree of the polynomial. These are taken to be the extrema of $T_{N+1}(x)$. We then set up the following $(N + 2) \times (N + 2)$ system of linear equations:

$$\left(\sum_{n=0}^{N} b_k x_k^n \right) + (-1)^k \varepsilon = f(x_k) \tag{7}$$

for each $0 \leq k \leq N + 1$. The variable ε is the approximate supremum error for the polynomial $p(x)$ defined by:

$$p(x) = \sum_{k=0}^{N} b_k x^k \tag{8}$$

The Remez Exchange Algorithm

Solving this system of equations gives us a guess $p_0(x) = p(x)$ to the minimax polynomial, the polynomial which minimizes the sup norm. We compute p_1 as follows.

Compute $|f(x) - p_0(x)|$ and find approximations to the local extrema. This is done by applying Newton's method to the samples x_k. These new values are our new samples y_k. We replace the x_k with these y_k and repeat the process from the previous slide.

The Remez Exchange Algorithm

As we apply more iterations, the value ε starts to stabilize to some constant. Once this happens we are done, and we have found the minimax polynomial. The error in this approximation is given by ε.

To be precise, we are done when the local extrema of $f(x) - p_n(x)$ are all of the same magnitude and oscillate. It is a well-known theorem that the minimax polynomial of f on $[a, b]$ of degree N is the unique polynomial that achieves this property. This is the equioscillation theorem.

The Remez Exchange Algorithm

As an example, the coefficients for $\arctan(x^2)$ on a small interval.

$$1.00000000000000000000 \tag{9}$$
$$0.33333333333329318027 \tag{10}$$
$$-0.19999999998764832476 \tag{11}$$
$$0.14285714272503466371 \tag{12}$$
$$-0.11111110405462355788 \tag{13}$$
$$0.09090887133436506561 \tag{14}$$
$$-0.07691876205044829994 \tag{15}$$
$$0.06661073137387531206 \tag{16}$$
$$-0.05833570133790573486 \tag{17}$$
$$0.04976877994615932360 \tag{18}$$
$$-0.03653157274421691552 \tag{19}$$
$$0.06285820115365782362 \tag{20}$$

Note these approximate the Taylor coefficients very well.

Pade Approximants

Now, for some history. In the 7th century, Indian mathematician Bhaskara I. writes:

> *I briefly state the rule (for finding the bhujaphala and the kotiphala, etc.) without making use of the Rsine-differences 225, etc. Subtract the degrees of a bhuja (or koti) from the degrees of a half circle (that is, 180 degrees). Then multiply the remainder by the degrees of the bhuja or koti and put down the result at two places. At one place subtract the result from 40500. By one-fourth of the remainder (thus obtained), divide the result at the other place as multiplied by the 'anthyaphala (that is, the epicyclic radius). Thus is obtained the entire bahuphala (or, kotiphala) for the sun, moon or the star-planets. So also are obtained the direct and inverse Rsines.*

Pade Approximants

Let's be thankful for modern notation, and write:

$$\sin(x) \approx \frac{4x(180 - x)}{40500 - x(180 - x)} \tag{21}$$

where x is in degrees. This approximation beats the Taylor polynomial of similar degree by a good margin. Bhaskara did not write where this formula came from, but it is possible to reverse engineer it.

Pade Approximants

The Taylor polynomial of degree N can be described as the unique polynomial p that satisfies:

$$\frac{\mathrm{d}^n}{\mathrm{d}\,x^n} p(x_0) = \frac{\mathrm{d}^n}{\mathrm{d}\,x^n} f(x_0) \tag{22}$$

for $0 \leq n \leq N$. The (N, M) Pade approximant is computed via:

$$R(x) = \frac{\sum_{n=0}^{N} a_n (x - x_0)^n}{1 + \sum_{m=1}^{M} b_m (x - x_0)^m} \tag{23}$$

and requiring that:

$$\frac{\mathrm{d}^n}{\mathrm{d}\,x^n} R(x_0) = \frac{\mathrm{d}^n}{\mathrm{d}\,x^n} f(x_0) \tag{24}$$

for $0 \leq n \leq N + M$.

Pade Approximants

This requirement leads to an $(N+M+1) \times (N+M+1)$ system of equations (with the $N+1$ unknowns $a_0, \ldots, a_N$ and the M unknowns $b_1, \ldots, b_M$). Solving this gives us the coefficients of the Pade approximant, which can be efficiently evaluated using Horner's method twice and one division.

The Pade approximant is often more accurate than the Taylor polynomial of similar degree. The method is used frequent in software libraries when quadruple precision ($2^{-112} \approx 10^{-34}$ relative error) is required.