FUNCTION APPROXIMATION AND THE REMEZ ALGORITHM

ABIY TASISSA

Abstract. The Remez exchange algorithm is explained. An implementation in Python is tested on different test functions. We make a comparison with SLSQP(Sequential Least Squares Programming) optimizer.

Key words. Remez exchange, Minimax polynomial, polynomial interpolation

1. Overview. Given a set of data points and values measured at these points, it is of interest to to determine the function that does a 'good' approximation. This problem belongs to the realm of function approximation or interpolation theory and finds its use in scientific computations[7]. In the past, these problems were handled using complicated tables and the evaluations were done using calculators or just by hand[4]. To increase the number of significant digits, it meant more manual work to be done. With the coming of computers, the computations were very fast and there was no need to store values as things could be computed on the fly.

Approximating functions by rational functions is one class of problems in approximation theory. We are interested in the approximation of a function by a rational function in the L_∞ norm. That is we want to minimize the maximum vertical distance between the function in consideration and a rational function. This is called the Minimax Approximation[7]. The theoretical framework for this was originally done by Chebyshev[16]. He stated what is called equioscillation theorem that gave a necessary and sufficient condition for a minimax polynomial/rational function[7]. By 1915, all the theoretical results regarding Minimax approximations had been established [19]. However, as we will see in later sections, the theoretical tools are useful but don't lead to any feasible way to find the Minimax rational function. In fact, if we are considering polynomials of degree above 1, constructing the minimax rational function is quite complicated[7]. In a series of three papers, a Russian mathematician by the name of Remez introduced an algorithm that computes the minimax approximation[12][13][14]. Remez algorithm became a useful tool in the branches of science and engineering and was used for problems from different fields. One significant application was employing the algorithm for the design of a filter first introduced by Parks and McClellan [11].

Historically, minimax rational approximations were used to representing special functions on a computer[5]. This is because for special functions it might be desirable, from a computational point of view, to do the computation of their polynomial and rational approximations as opposed to dealing with functions themselves. In fact, in the early days of numerical computing, minimax approximations played that role[19]. Minimax polynomial and Rational approximations were used for example in the design of FUNPACK in 1970[5].

The goal of this paper is to give a brief overview of Minimax approximation and Remez algorithm with the focus on the implementation and how it compares with a competing nonlinear algorithm.

2. Introduction and examples. Given a function $f(x)$ on some interval $[a, b]$ where we know its values on some finite set of points, a simple interpolation is a polynomial that passes exactly through these points. There are three important questions to be considered in the process of the interpolation[9]:

1. How smooth is the function f?
2. How many points do we need and where should the points be located at?
3. How do we measure the error between the polynomial and the function f?

Assume we have a reasonably smooth function. Mathematically, the notion of the measure of the error implies a particular choice of norm. Let's define the error as the maximum vertical distance between the polynomial and the function f. Formally this represents the L^∞ norm. The question of where the points should be located is a settle issue which was originally considered by P. L. Chebyshev in 1853[2]. We are interested, given some space, if we can find a polynomial that minimizes the error between the polynomial and the function in consideration in the L^∞ sense. Such a polynomial, if it exists, is called the minimax polynomial and the approximation of $f(x)$ in the the L^∞ sense is known as a Minimax approximation. Chebyshev showed that there exists a unique minimax polynomial. He also put forth a sufficient condition for what it means to be a minimax polynomial. From a theoretical point of view, this concludes Minimax approximation. However the task of constructing a minimax polynomial is not trivial. For a given function f, Remez algorithm is an efficient iterative algorithm that constructs a minimax polynomial

However as simple as they are, polynomials on their own don't capture all the classes of functions we want to approximate[10]. For that, we want to consider rational minimax approximation. In the same sense we discussed above, the Chebyshev condition gives us a criteria for a minimax rational function and the Remez algorithm has a method to deal with rational approximations. In this paper, our focus will be on rational approximation but in explaining the basics of the Remez method, we find it easier to start with the simplest case of polynomial approximation and generalize later on.

The structure of the paper is as follows. We discuss Chebychev nodes in section 3. In section 4 we introduce the Minimax approximation for polynomials. In section 5 and 6, we describe the Remez algorithm for polynomials. In section 7, we use the developed ideas and deal with the case of discrete data approximation using rational functions and explain our implementation in python. We then demonstrate the implementation on test functions and make comparisons with SLSQP(Sequential Least Squares Programming) optimizer in section 8. We conclude in section 9.

3. Chebychev nodes. As mentioned in the introduction, the location of the interpolation points is an important consideration in polynomial approximation. Intuitively one might expect that equispaced points in a given interval are ideal for polynomial interpolation. However such nodes are susceptible to what is known as the Runge Phenomenon, which are large oscillations that happen at the end of the intervals[6]. One might consider circumventing this problem by adding more nodes hence a polynomial of higher degree but unfortunately the problem doesn't disappear. In fact, the error grows exponentially at the end intervals[6]. Hence the choice of interpolation points is more complicated than it appears at first sight and a wrong choice can lead to erroneous and meaningless results. Aside from the Runge Phenomenon, another consideration is minimizing the interpolation error. That is, we want inter-

polation points such that the polynomial passing through the points is as close as possible to the function. The solution for these problems are the Chebychev nodes. They avoid the Runge phenomenon and they do minimize the interpolation error in the following way[3]. Given a function f in the interval $[a, b]$, n points $x_0, x_1, \ldots, x_n,$, and the interpolation polynomial P_n, the interpolation error at x is given by

$$E(x) = f(x) - p(x) = \frac{1}{n+1!} f^{(n+1)}(\xi) \prod_{i=0}^{n} (x - x_i)$$

for some ξ in $[a, b]$. If we want to minimize this error, we don't have any control on the $\frac{1}{n+1!} f^{(n+1)}(\xi)$ term since we don't know the function f nor can we specify the value ξ. So in minimizing the error, we consider the minimization of

$$\prod_{i=0}^{n} (x - x_i)$$

The Chebyshev nodes minimize the product above. As we will see in the next section, when one has the flexibility in picking the interpolation points, the Chebyshev nodes are ideal. In an interval $[a, b]$ for an polynomial interpolant of degree $n - 1$, the Chebyshev nodes are given by

$$x_i = \frac{1}{2}(a + b) + \frac{1}{2}(b - a) \cos\left(\frac{2i - 1}{2n}\pi\right) \quad , i = 1, \ldots, n \tag{3.1}$$

4. Minimax Approximation. In approximating functions, polynomials are commonly used. This is because an arbitrary function $f(x)$ can be approximated by polynomials as close as we want. This fact follows from the Stone-Weistrass theorem [15]

THEOREM 4.1. *If f is a continuous complex function on $[a, b]$, there exists a sequence of polynomials P_n such that*

$$\lim_{n \to \infty} P_n(x) = f(x)$$

uniformly on $[a, b]$. If f is real, the P_n may be taken real.

In Minimax approximation, we are trying to find the closest polynomial of degree $\leq n$ to $f(x)$, closest being in the L^∞ sense. Formally the minimax polynomial $P_n^*(x)$ can be defined as follows[8]:

$$E^* = \max_{a \leq x \leq b} \left| P_n^*(x) - f(x) \right| \leq \max_{a \leq x \leq b} \left| P_n(x) - f(x) \right|,$$

where $P_n(x)$ is any polynomial of degree n. That is among polynomials of degree $\leq n$, we are trying to find the polynomial that minimizes the absolute error. We can also define the near-minimiax polynomial[8] $Q_n(x)$ as any polynomial defined as

$$\epsilon = \max_{a \leq x \leq b} \left| P_n^*(x) - Q_n(x) \right|$$

and where ϵ is sufficiently small. To make matters clear, let's try to see the error between the near-minimax polynomial and the function f. We claim that

$$\max_{a \leq x \leq b} \left| Q_n(x) - f(x) \right| \leq E^* + \epsilon$$

The proof is as follows.

Proof.

$$\max_{a\leq x\leq b}\left|Q_n(x)-f(x)\right| = \max_{a\leq x\leq b}\left|\left(Q_n(x)-P_n^*(x)\right)+\left(P_n(x)^*-f(x)\right)\right|$$

Using triangle's inequality, we get

$$\max_{a\leq x\leq b}\left|Q_n(x)-f(x)\right| \leq \max_{a\leq x\leq b}\left|Q_n(x)-P_n(x)^*\right| + \max_{a\leq x\leq b}\left|P_n(x)^*-f(x)\right| \leq E^*+\epsilon$$

as desired. □

The implication of the near-minimax polynomial can be described as follows. When we do the Minimax approximation on a computer, we don't get the exact minimax polynomial due to numerical errors. As such for the practical case, for a very small ϵ, $Q_n(x)$ is a very good approximation for the Minimax polynomial. When $\frac{\epsilon}{E^*}\leq\frac{1}{10}$ the near-minimax and the minimax polynomial are almost identical[8].

Chebyshev showed that in approximating a function $f(x)$ on some interval $[a,b]$, there exists a unique minimax polynomial $P_n^*(x)$ of degree n and also put a criteria for the minimax polynomial [2]. This criteria, referred as the oscillation theorem, states:

THEOREM 4.2. *Suppose that $f(x)$ is continuous in $[a,b]$. The polynomial $P_n^*(x)$ is the minimax polynomial of degree n if and only if there exist at least $n+2$ points in this interval at which the error function attains the absolute maximum value with alternating sign. Formally,*

$$\begin{aligned} a\leq x_0,x_1,...,x_{n+1}\leq b \\ f(x_i)-P_n^*(x_i) = (-1)^iE^* \text{ for } i=0,1,...,n+1 \\ E^* = \pm\max_{a\leq x\leq b}\left|f(x)-P_n^*(x)\right| \end{aligned}$$

The above theorem is a sufficient condition in that any polynomial that satisfies the above condition is a minimax polynomial. For a polynomial of degree $n\leq 1$, the construction of the minimax polynomial is analytically tractable as shown in the example below.

Example : Compute the minimax polynomial of degree 1 for $f(x)=e^x$ on the interval $[0,1]$.

Since e^x is a convex function, we know that the extrema of the error function occur at $x=0$, $x=1$ and $x=x_1$ where $0\leq x_1<1$. The equation of the linear polynomial can be written as $P_1(x)=mx+c$. Hence the error function is given by

$$E(x)=e^x-(mx+c)$$

Hence at x_1 set $E'(x)=0$ and we get $m=e^{x_1}$. Now we enforce the oscillation criteria as follows at the three points

$$e^0-(0+c)=E \tag{4.1}$$

$$e^{x_1}-(mx_1+c)=-E \tag{4.2}$$

$$e^1-(m+c)=E \tag{4.3}$$

Using 4.1 and 4.3, we get

$$m=e-1\approx 1.7183$$

and using 4.2 and 4.3, we get

$$c = \frac{1}{2}\left(m - mx_1 + e^1 - m\right) \approx 0.8941$$

Therefore the linear minimax polynomial is given by $P_1(x) = 1.7183x + 0.8941$. Fig. 4.1 shows the function and the minimax polynomial.

However the reader should note that being able to find a minimax polynomial as in the example above is a rare event. When the degree of the interpolating polynomial is ≥ 2, the process above becomes complicated and evaluation of the proper polynomial coefficients doesn't lead to an explicit solution. As such one has to resort to an algorithm of some sort. In our case, we use the Remez algorithm which is an iterative algorithm that computes the minimax polynomial. We discuss the algorithm in the next section.

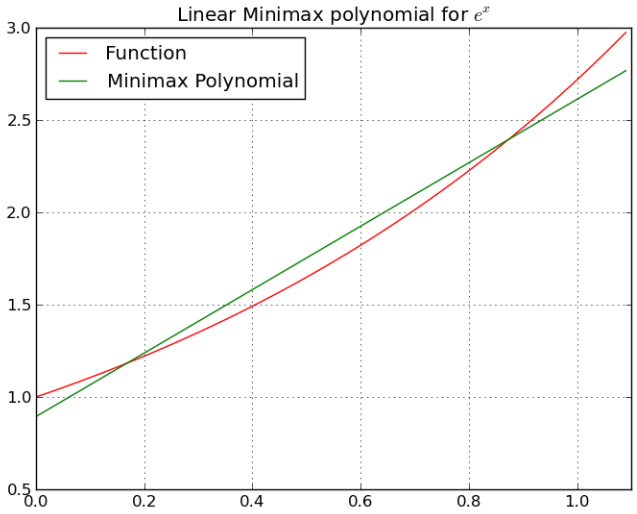

FIG. 4.1. *A linear minimax Polynomial for e^x. This is the only case we can construct the minimax polynomial. For polynomial of degree ≥ 1, one has to resort to an algorithm*

5. Remez Algorithm. The Remez algorithm is due to a Russian mathematician Evgeny Remez who published the result in 1934[18]. It is an efficient iterative algorithm that computes the minimax polynomial. To initialize the algorithm, we need a set of $n + 2$ points in the interval $[a, b]$. One could choose different initial points but a common choice is the Chebyshev nodes. This is because, as we discussed earlier, the Chebyshev nodes are not prone to the Runge phenomenon and minimize the interpolation error. Hence the polynomial that passes through the Chebyshev nodes is a good initial estimate for the minimax polynomial. Let the polynomial $P_n(x)$ passing through the Chebyshev nodes be:

$$P_n(x) = b_0 + b_1x^1 + ... + b_nx^n$$

where $b_0, b_1, \ldots, b_n$ are the coefficients. Now we want to force the oscillation criteria on this polynomial. That is we want the error between the polynomial and the function

f to oscillate alternatively at the Chebyshev nodes. For that, we we write the system of equations below:

$$b_0 + b_1 x_i^1 + ... + b_n x_i^n + (-1)^i E = f(x_i) \text{ for } i = 0, 1, 2, ..., n+1$$

We can write this in a matrix form as follows

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n & E \\ 1 & x_1 & x_1^2 & \cdots & x_1^n & -E \\ 1 & x_2 & x_2^2 & \cdots & x_1^n & E \\ \vdots & & & & & \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n & (-1)^i E \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \\ E \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n+2}) \end{pmatrix}$$

We can solve this $(n+2)(n+2)$ system to find the coefficients $b_0, b_1, ..., b_n, E$. We have now enforced the oscillation criteria but note that the error E is not necessarily the extrema of the error function. For that reason, we need to move to a new set of points. This leads us into the second step of the algorithm called the exchange step.

What we have so far is an error function that alternates in sign at the $n+2$ points. From the intermediate value theorem, it follows that the error function has $n+1$ roots. We compute the roots using any numerical method and consider the $n+2$ intervals

$$[a, z_0], [z_0, z_1], [z_1, z_2] ..., [z_{n-1}, z_n], [z_n, b]$$

where $z_0, z_1, \ldots, z_n$ are the $n+1$ roots. For each interval above, we find the point at which the error functions attains its maximum or minimum value. We can do this by differentiating the error function and locating the minimum or maximum in each interval. If it happens that the minimum or maximum doesn't exist, we compute the value of the error at the two end points and take the one with the largest absolute value. This provides us with a new set of points

$$x_0^*, x_1^*, \ldots x_{n+1}^*$$

This new set of points will be used in the second step of iteration. We continue the iteration until a stopping criteria is met.

At the end of each iteration, we obtain a new set of control points. We evaluate the error at these control points. Let $E_m = \min_i |E_i|$ and $E_M = \max_i |E_i|$. As we converge and approach the minimax polynomial, the difference between the old and new set of control points is minimized. Hence a reasonable stopping criteria is to stop the iteration when

$$E_M = \alpha \, E_m \tag{5.1}$$

where α is some constant but not arbitrary. We choose $\alpha = 1.05$ but anything closer to 1 is a reasonable choice.

Before we discuss the implementation of the Remez algorithm in Python, we summarize the main steps.

1. For a given function $f(x)$ on an interval $[a, b]$, specify the degree of interpolating polynomial

2. Compute the $n+2$ Chebyshev points using 3.1 i.e $x_0, x_1, \ldots x_{n+1}$.
3. Enforce the oscillation criteria by solving the $(n+2)(n+2)$ systems of equations below.

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n & E \\ 1 & x_1 & x_1^2 & \cdots & x_1^n & -E \\ 1 & x_2 & x_2^2 & \cdots & x_1^n & E \\ \vdots & & & & & \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n & (-1)^i E \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \\ E \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n+2}) \end{pmatrix}$$

   We get $b_0, b_1, \ldots, b_{n+1}, E$.
4. Form a new polynomial P_n with the above coefficients.

$$P_n(x) = b_0 + b_1 x^1 + \ldots b_n x^n$$

5. Compute the extremes of the error function

$$P_n(x) - f(x)$$

   This will give a new set of control points $x_0^*, x_1^*, \ldots x_{n+2}^*$.
6. If the stopping criteria is met 5.1, then stop the iteration. If not use the new control points and proceed to step 3.

Despite the choice of initial points, it has been shown that the Remez algorithm has has a quadratic convergence [20, 8]. This summarizes the Remez algorithm. In the next section, we discuss how the ideas discussed in this and previous sections could be applied to rational minimax approximations.

6. Rational Minimax Approximation for Discrete Data. In this paper, our interest is to find a rational approximation given a discrete number of points and their corresponding values. This problem is natural since we usually don't have a priori the function we want to approximate. All we have is data points and measurements and using this we want to find the best approximation. In rational minimax approximation, we are trying to find

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

that is we want to find polynomial $p(x)$ and $q(x)$ of degree m and n respectively. We can always normalize and choose $b_0 = 1$ and this leads us to write the minimax problem using Chebyshev condition as follows:

$$f(x) - \frac{\sum_{i=0}^{m} a_i x^i}{\sum_{i=0}^{n} b_i x^i} = (-1)^E \quad (i = 0, 1, ..., m+n+1) \tag{6.1}$$

Note that since we chose $b_0 = 1$, we have $m+n+1$ unknowns as opposed to $n+m+2$ unknowns. The ideas developed earlier could be easily extended to this case as will see below. This is mainly due to the fact that the equioscillation theorem is not only restricted for polynomials. For a rational minimax approximation we have the following theorem [10]

THEOREM 6.1. *Suppose that $f(x)$ is continuous in $[a, b]$. The rational function $R^*_{n+m}(x)$ is the minimax rational if and only if there exist at least $m+n+2$ points*

in this interval at which the error function attains the absolute maximum value with alternating sign. Formally,

$$\begin{aligned} a \leq x_0, x_1, ..., x_{n+1} \leq b \\ f(x) - \frac{\sum_{i=0}^{m} a_i x^i}{\sum_{i=0}^{n} b_i x^i} = (-1)^E \quad (i = 0, 1, ..., m+n+1) \\ E^* = \pm \max_{a \leq x \leq b} \left| f(x) - R^*_{n+m}(x) \right| \end{aligned}$$

With this, we are ready to describe the Remez algorithm for rational functions with discrete number of data. Assume we have $x_0, x_1, ..., x_r$ data points and the corresponding values $y_0, y_1, ..., y_r$. Given this data, we want to find the rational minimax approximation. Remez's algorithm could be modified and we can summarize it as follows:

1. Specify the degree of interpolating rational function i.e m and n.
2. Pick $m+n+2$ points from the data points $x_0, x_1, ..., x_r$. Note that these are neither Chebyshev nodes nor have any particular property.
3. Enforce the oscillation criteria by considering Equation. 6.1. However notice that unlike the case of polynomials, this is no longer a linear system of equations since we have the error term multiplying the polynomial in the denominator. However, we can solve these equations iteratively by linearizing the equation 6.1 and obtaining the following iteration formula[1]

   $$\left((-1)^k E_0 - f(x)\right) \sum_{i=1}^{n} b_i x^i + (-1)^k E + \sum_{i=0}^{m} a_i x^i = y_i \quad (i = 0, 1, ..., m+n+1)$$

   Note that E_0 is the initial guess. An initial guess $E_0 = 0$ is a good starting point. We do the iteration till E converges to a stable value and finish the step by solving for $a_0, a_1, ..., b_1, b_2, ...b_n$ and E. Here is the matrix we solve at every step of the iteration.

   $$\begin{pmatrix} 1 & x_0 & x_0^2 & .. & x_0^m & .. & (E_0 - y_0)x_0 & (E_0 - y_0)x_0^n & E \\ 1 & x_1 & x_1^2 & .. & x_1^m & .. & (E_0 - y_1)x_1 & -1(E_0 - y_1)x_1^n & E \\ 1 & x_2 & x_2^2 & .. & x_2^m & .. & (E_0 - y_2)x_2 & (E_0 - y_2)x_2^n & E \\ \vdots & & & & & & & & \\ 1 & x_d & x_d^2 & .. & x_d^n & .. & (E_0 - y_2)x_d & (-1)^d(E_0 - y_2)x_d^n & E \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ b_1 \\ b_2 \vdots \\ b_n \\ E \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_d \end{pmatrix}$$

   where $d = n + m + 2$. After solving this, we get $a_0, a_1, ..., b_1, b_2, ...b_n$ and E.
4. Form a new rational function R_n with the above coefficients.

   $$R_n(x) = f(x) - \frac{\sum_{i=0}^{m} a_i x^i}{\sum_{i=0}^{n} b_i x^i} = (-1)^E$$

5. Compute the error function

   $$R_n(x) - f(x)$$

Now to find the a new set of control points, we do something similar to the Remez exchange and end with new points $x_0^*, x_1^*, \ldots x_{m+n+2}^*$ as explained in step 6.

6. If no residual is numerically greater than $|E|$, we are done. If not, find the local extreme of the residuals. If we find a local extrema r_i at x_i that is not one of the nodes in the set of original nodes, then replace it the nearest x in the original set of nodes with the condition that the residual is of the same sign.
7. Once we have all the new control points, proceed(go back) to step 3.

7. Implementation. The Remez algorithm was implemented in Python in about 100 lines of code. To get the coefficients of the polynomial and the error in the third step of the algorithm, we use the solve method in the linear algebra package in SciPy.

```
b=scipy.linalg.solve(P,y)
```

where we solve

$$\begin{pmatrix} 1 & x_0 & x_0^2 & .. & x_0^m & .. & (E_0-y_0)x_0 & (E_0-y_0)x_0^n & E \\ 1 & x_1 & x_1^2 & .. & x_1^m & .. & (E_0-y_1)x_1 & -1(E_0-y_1)x_1^n & E \\ 1 & x_2 & x_2^2 & .. & x_2^m & .. & (E_0-y_2)x_2 & (E_0-y_2)x_2^n & E \\ \vdots & & & & & & & & \\ 1 & x_d & x_d^2 & .. & x_d^n & .. & (E_0-y_2)x_d & (-1)^d(E_0-y_2)x_d^n & E \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ b_1 \\ b_2 \vdots \\ b_n \\ E \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_d \end{pmatrix}$$

Veidinger proved that the matrix above is always invertible i.e $(n+2)$ linear equations are independent[20]. Hence we will always get a solution i.e a b vector $b = b_0, b_1, \ldots, b_n, E$.

8. Results and Comparison with SLSQP. We compare the Remez implementation for the discrete rational minimax approximation with SLSQP(Sequential Least Squares Programming). SLSQP solves constrained optimization problems and is available in the SciPy optimize package. We rewrite the minimax approximation problem in terms of constrained optimization as follows:

$$\min_{e\in\mathbb{R}, a\in\mathbb{R}^{n+1}, b\in\mathbb{R}^n} e$$
$$e \geq \frac{\sum_{i=0}^{m} a_i x^i}{(1+\sum_{i=1}^{n} b_i x^i) - f(x)}$$
$$e \leq \frac{\sum_{i=0}^{m} a_i x^i}{(1+\sum_{i=1}^{n} b_i x^i) - f(x)}$$

Before going into comparison with SLSQP, we first do a validation of the code.

8.1. Validation. If provided a polynomial or a rational function, our implementation should return the exact polynomial or rational within a single iteration. Here we show that this is the case with two simple examples.

Example 1: Consider the polynomial defined in the following way.

$$f(x) = x^2 \text{ and } x \in (0, 1)$$

We use 100 equally spaced points between $(0, 1)$. When we run the code, it returns the following coefficients:

$$\begin{aligned} a_0 &= -1.04083409e - 17 \\ a_1 &= 0.00000000e + 00 \\ a_2 &= 1.00000000e + 00 \end{aligned}$$

and we know that $b_0 = 1.0$. Since $a_2 = 1.0$, we see that the code returns the polynomial exactly in a single iteration.

Example 2: Consider the rational function defined in the following way.

$$f(x) = 1/(1 + x^2) \text{ and } x \in (0, 1)$$

We use 100 equally spaced points between $(0, 1)$. When we run the code, it returns the following coefficients:

$$\begin{aligned} a_0 &= 1.00000000e + 00 \\ b_1 &= 4.44089210e - 16 \\ b_2 &= 1.00000000e + 00 \end{aligned}$$

and we know that $b_0 = 1.0$. Since $a_0 = 1.0$ and $b_2 = 1.0$, we see that the code returns the rational function exactly in a single iteration.

8.2. Comparison with SLSQP. In this section, we compare the discrete minimax approximation employing Remez algorithm with the SLSQP algorithm. To make a fair comparison between the two algorithms, we use the same number of points in a given interval. We then specify the degree of the numerator and denominator of the rational function. These two values will be the same for both algorithms. The two algorithms then iterate through the discrete number of points to minimize the error. As such, we take the number of iterations to be a good criteria in comparing these two algorithms. We test this for different classes of functions and we also note how well the function is approximated in each of these two methods by looking at the plots.

We first compare the number of iterations for different classes of functions. Table. 8.1 shows the comparison. All but two of the functions are evaluated at 100 points in the interval $(0, 1)$. For the $\sin(x)$ function, we use the interval $(0, \pi)$ and for the $\log(x)$ function, we use the interval $(1, 2)$. We do this since $\log(0)$ is undefined if we choose $(0, 1)$ as the interval.

To see how well Remez algorithm compares with the SLSQP algorithm, we make the plots of the functions above using the coefficients we obtain numerically. Figures 8.1, 8.2, 8.3 and 8.4 show the difference between the analytical plot and the numerical plots. From the plots, we see that the Remez algorithm approximates the functions better than the SLSQP algorithm and it does so with no need for multiple iterations. In fact for the test functions considered here, it is hard to look the difference between the analytic plot and the numerical plot based on Remez since they are on top of each other.

Function	Degree of Rational Function(m,n)	Iterations: Remez	Iterations: SLSQP
x^2	$(2,0)$	1	5
$\frac{1}{1+x^2}$	$(0,2)$	1	8
$\sqrt{x}$	$(4,2)$	1	26
e^x	$(1,1)$	3	10
$\sin(x)$	$(3,2)$	2	25
$\log(x)$	$(2,2)$	1	8

TABLE 8.1
Number of Iterations to converge for Remez and SLSQP algorithms for different functions

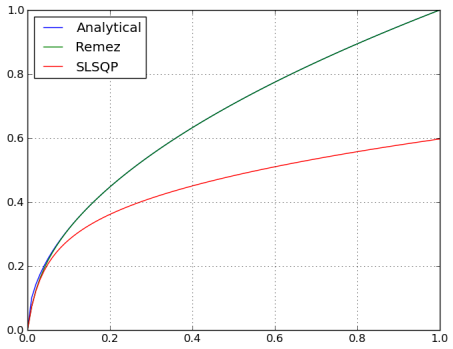

FIG. 8.1. $f(x) = \sqrt{x}$

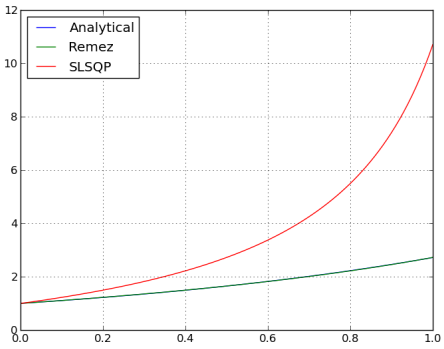

FIG. 8.2. $f(x) = e^x$

9. Conclusion. We see that Remez algorithm is an efficient method to construct the minimax rational function in the approximation of a discrete set of data. It takes few iterations and does comparably better than a nonlinear optimizer like SLSQP. However it is not always the case that, unlike the case for polynomials, the process of finding rational functions doesn't always converge[1] but for most common functions the Remez algorithm leads to a good approximation.

Acknowledgments. The author thanks Professor Steven Johnson for suggesting this topic and giving directions during office hours.

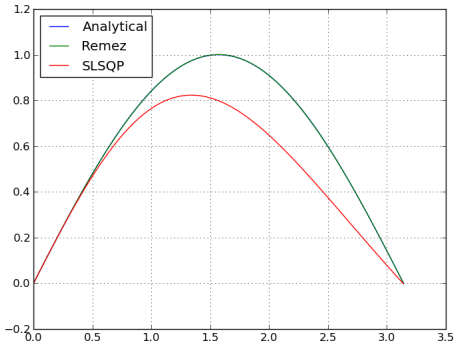

FIG. 8.3. $f(x) = \sqrt{x}$

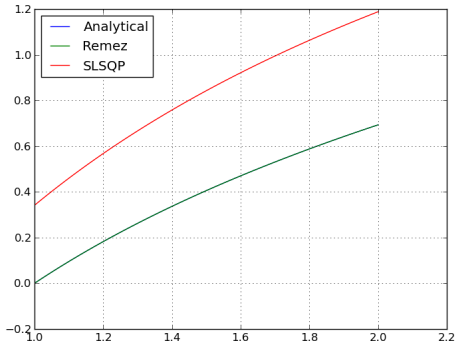

FIG. 8.4. $f(x) = e^x$

REFERENCES

[1] H.M ANTIA *Numerical Methods for Scientists and Engineers*, Birkhauser 2002

[2] NEAL L. CAROTHERS *A Short Course on Approximation Theory*, Lecture Notes at `http://personal.bgsu.edu/~carother/Approx.html`

[3] E. WARD CHENEY, AND DAVID R. KINCAID *Numerical Mathematics and Computing*, Cengage Learning, 2012

[4] W.J CODY *A survey of practical rational and polynomial approximation of functions*, SIAM Review, July 1970

[5] W.J CODY *The FUNPACK package of special function subroutines*, ACM Trans. Math. Softw, 9, 1975

[6] GERMUND DAHLQUIST, AND AKE BJORCK *Numerical Methods*, Dover Books, 2003

[7] P.J DAVIS *Interpolation and approximation*, Dover Publications, New York, 1975

[8] W. FRASER, *A Survey of methods of Computing Minimax and Near-Minimax Polynomial Approximations for functions of a Single Independent Variable*, , Journal of the Association for Computing Machinery,Vol 12, No. 3 (July, 1965), pp. 295-314

[9] F. W. LUTTMANNAND AND T. J. RIVLIN, *Some Numerical Experiments in the Theory of Polynomial Interpolation*, IBM Journal, May 1965, pp. 187-191

[10] JEAN-MICHEL MULLER *Elementary Functions: Algorithms and Implementation*

[11] T.W PARKS AND J.H MCCLELLAN *Chebyshev approximation for nonrecursive digital filters with linear phase*,IEEE Trans. Circuit Theory, 1972

[12] E REMES *Sur la d etermination des polynomes dapprox imation de degr e donn ee*, Comm. Soc. Math. Kharkov 1934

[13] E REMES *ur le calcul effectif des polynomes dapproxi mation de Tchebychef*, Compt. Rend. Acad. Sci. 1934

[14] E REMES *ur un proc ed e convergent dapproximation s successives pour d eterminer les polynomes dapproximation*, Compt. Rend. A cad. Sci.1934

[15] WALTER RUDIN, *Principles of Mathematical Analysis* , McGraw-Hill, 1976

[16] K.G STEFENS, *The history of approximation theory: From Euler to Bernstein*, Birkhauser, Boston 2006

[17] SHERIF A. TAWFIK, *Minimax Approximation and Remez Algorithm*, Lecture Notes at `http://www.math.unipd.it/~alvise/CS_2008/APPROSSIMAZIONE_2009/MFILES/Remez.pdf`

[18] LLOYD N. TREFETHEN, *Approximation Theory and Approximation Practice*, SIAM, 2012

[19] LLOYD N. TREFETHEN, *Barycentric Remez algorithms for best polynomial approximation in the CHEBFUN system*, Numerical Mathematics 2008

[20] L. VEIDINGER , *On the numerical determination of the best approximations in the Chebyshev sense*, Numer. Math, 1960, 99-105