14 May 2019

Warsaw University of Technology – EiTI faculty

Approximation of functions

No. 38

Wiktor Bara

Course name: Numerical Methods

Advisor: Andrzej Miękina

Table of contents

Introduction

The numerical approximation of functions can be obtained in many different ways. Every algorithm that is used has his own accuracy, complicity and is more convenient to specific functions that the other one. The algorithm that is considered in this task is least-squared approximation and is often used to approximate functions with finite number and discreate values. The algorithm is the best when number of arguments (nodes) is bigger.

Description of Numerical Algorithms

Least-squared approximation of a function on the basis on its discreate values

This approximation bases on values of concreate function to obtain the approximate one, the number N is equal to number of nodes, points that we obtain from original function, the K is the number that describes our accuracy in this algorithm.

f(x) - function

$\Phi_k(x) \mid k = 1,2,\ldots,K$ - linearly independent function

$$\mathbf{\Phi} = \begin{bmatrix} \Phi_1(x_1) & \Phi_2(x_1) & \cdots & \Phi_K(x_1) \\ \Phi_1(x_2) & \Phi_2(x_2) & & \Phi_K(x_2) \\ \vdots & & \ddots & \vdots \\ \Phi_1(x_N) & \Phi_2(x_N) & \cdots & \Phi_K(x_N) \end{bmatrix}$$

the matrix created of linearly independent function

$\boldsymbol{y} = [f(x_1), f(x_2), \ldots, f(x_N)]^T$ – the vector of all values of original function

$$\mathbf{\Phi}^T \cdot \mathbf{\Phi} \cdot \mathbf{p} = \mathbf{\Phi}^{\boldsymbol{T}} \cdot \boldsymbol{y} \quad => \quad \mathbf{p} = (\mathbf{\Phi}^{\boldsymbol{T}} \cdot \mathbf{\Phi})^{-\mathbf{1}} \cdot \mathbf{\Phi}^{\boldsymbol{T}} \cdot \boldsymbol{y}$$

$\hat{f}(x; \boldsymbol{p}) = \sum_{k=1}^{K} p_k \phi_k(x)$ – the approximated function

Chebyshev polynomials

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \qquad \text{for k=2,3,…,K}$$

Methodology

The procedure of obtaining original function

The actual function is equal to

$f(x) = \sin(\pi x)e^{x-\frac{1}{3}}$

The domain of the function is $x \in [-1,1]$ and the vector of x is uniformly distributed points in that domain. The x is obtained using build in function.

The calculation of **x**:

$$\boldsymbol{x} = linspace[-1,1,N]'$$

The calculation of **y**:

$$\boldsymbol{y} = \sin(\boldsymbol{x}.* \, pi) .* \exp(\boldsymbol{x} - \frac{1}{3})$$

The procedure of obtaining approximation of function

The independent function that is used to obtained matrix is the Chebyshev polynomials. The matrix is obtained using those polynomial and the vector of x.

The vector of p is obtained using the operator of pseudoinversion “\” that is build in function

$$\boldsymbol{A}\backslash\boldsymbol{B} = \boldsymbol{A}^{-1} \cdot \boldsymbol{B}$$

The vector **p** calculation:

$$\boldsymbol{p} = (\boldsymbol{\Phi^T} \cdot \boldsymbol{\Phi})\backslash\boldsymbol{\Phi^T} \cdot \boldsymbol{y}$$

After obtaining all data that is needed to compute approximated function the actual algorithm contains the multiplication of matrix $\boldsymbol{\Phi^T}$ with the vector **p**. The multiplication is the array kind so it returns no vector but the matrix that contains all multiplications $p_k\phi_k(x)$, the actual value of approximated function is the sum of those points that are in the same row of that matrix. The $\hat{\boldsymbol{f}}$ is the vector that contains all values of approximated function, it is analogical to **y** vector.

The vector $\hat{\boldsymbol{f}}$ calculation:

$$\hat{\boldsymbol{f}} = sum(\boldsymbol{\Phi^T} .* \boldsymbol{p})$$

The procedure to obtain the root-mean-squared error and the maximum error

The errors that are to compute include input data N and K, those data define the amount of computation that have to be don and the comparison of those errors with respect to concreate N and K, where $N \in [5,55]$ and K<N.

Those errors are computed with build in function “norm”. The error for every K and N is calculated using previous algorithms and the result is showed on diagram.

$$\delta_2(K,N) = \frac{\left\|\hat{f}(x;K,N) - f(x)\right\|_2}{\left\|f(x)\right\|_2} \quad \text{(the root-mean-square error)}$$

$$\delta_\infty(K,N) = \frac{\left\|\hat{f}(x;K,N) - f(x)\right\|_\infty}{\left\|f(x)\right\|_\infty} \quad \text{(the maximum error)}$$

\

The procedure to carry out an investigation of $\delta_2(K,N)\ \ and\ \delta_\infty(K,N)$ on standard deviation.

The corruopted data is obtained by multiplying σ that is obtained by build in function logspace and the random number from normal distribution, the output of that multiplication is added to values of original function.

For every corrupted data the root-mean-square error and the maximum error are computed, for each σ the smollest value of those errors, and the variables N and K are saved. The pairs of $(\sigma, norm(K,N))$ were used to approximate the sequence using build in function polyfit.

Results

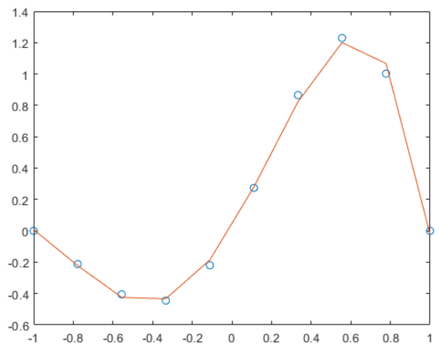

Figure 1) N=10 K=5

The results of three examples of approximation are visible on the three figures on the left. The approximation differs only in the number of nodes of origin function and the K variable is the same in every one of those. As we can see the number of nodes is significant the approximation of function strongly differs if the number of nodes is twice or triple bigger.

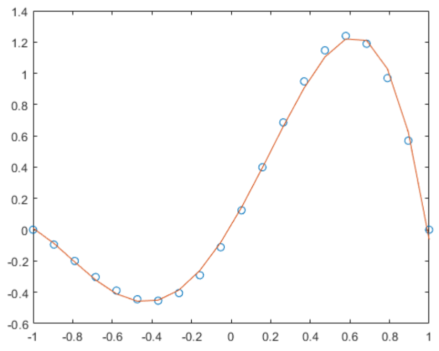

Figure 2) N=20 K=5

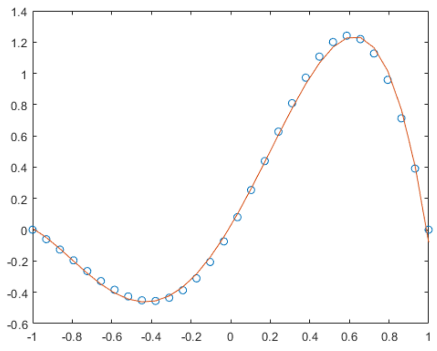

Figure 3) N=30 K=5

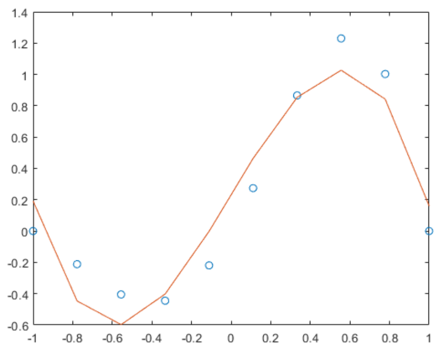

Figure 4) N=10 K=4

The results of the next three examples are a little bit different, in this case the only things that differs is K number. As we can see, the K is also significant to approximation of this type but with higher K those differences almost disappear

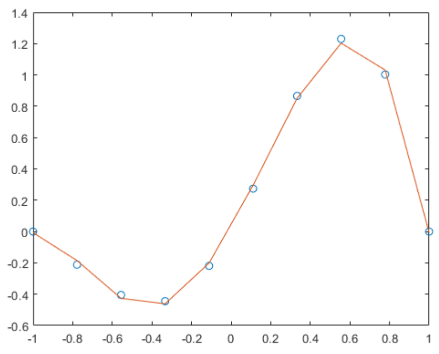

Figure 5) N=10 K=6

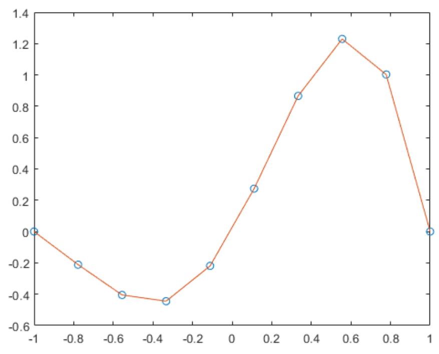

Figure 6) N=10 K=9

The accuracy of obtained approximation was checked, the dependence on K and N investigation was carried, using two following accuracy indicators:

$$\delta_2(K,N) = \frac{\left\|\hat{f}(x;K,N) - f(x)\right\|_2}{\left\|f(x)\right\|_2} \quad \text{(the root-mean-square error)}$$

$$\delta_\infty(K,N) = \frac{\left\|\hat{f}(x;K,N) - f(x)\right\|_\infty}{\left\|f(x)\right\|_\infty} \quad \text{(the maximum error)}$$

The results of investigation are presented on figures below

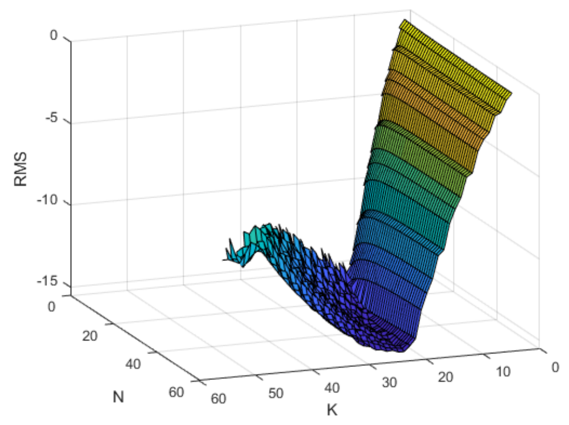

Figure 7) root-mean-square error

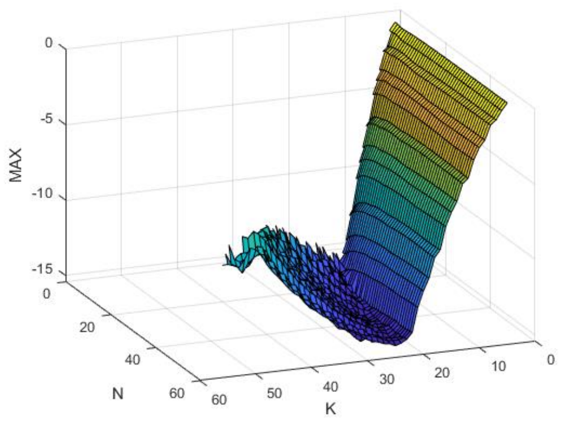

Figure 8) maximum error

From those figures is obviously visible that the best accuracy of approximation appears around K=20, the value of N variable has much lesser influence on the output but the small decrease can be observe on those axis when N increases. After breakpoint in K=20 the error increase and the approximation is much less accurate.

The simulation of approximation method on real data was carried out using indicators (norms) and investigation of them. The data was corrupted as follows:

$$\tilde{y}_n = y_n + \Delta\tilde{y}_n \text{ for } n = 1, \ldots, N$$

The $\Delta\widetilde{y_n}$ is the pseudorandom number obtained using the normal distribution with the variance $\sigma^2 \in [10^{-5}, 10^{-1}]$ using the build in function randn. For each value of σ the values of K and N that minimizes norms were determined. The pairs of $(\sigma, norm(K,N))$ were used to approximate the sequence using build in function polyfit. The results are described on figures below.

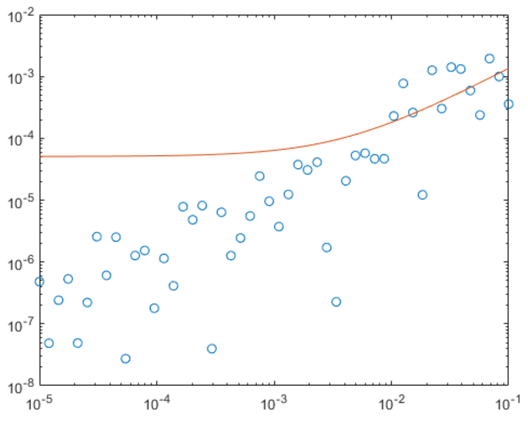

Figure 9) root-mean-square error

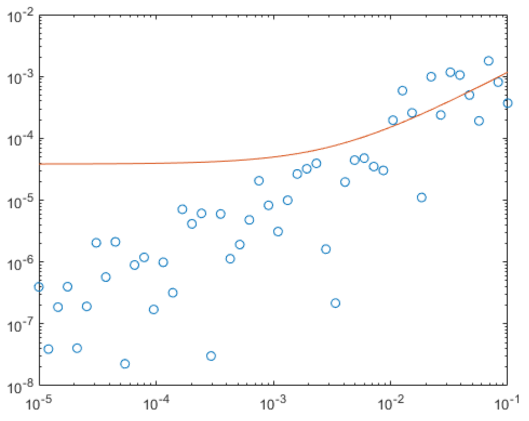

Figure 10) maximum error

Conclusions

The algorithm of least-square approximation is one of the most commonly used types of approximation. The procedure has a lot of advantages:

-the application of this is commonly used because it’s hard to find application where this algorithm is useless

-the complexity of this algorithm is not overwhelming and is easy to explain how it works

But as every algorithm the least-squared approximation has also disadvantages, as:

-it’s sensitive to outlines

-tendency to overfit data

List of References

[1] https://www.mathworks.com,

[2] Lecture notes ENUME 2019, Roman Morawski

SOURCE CODE

```
clear all
close all

[x10, y10] = seq(10);
[x20, y20] = seq(20);
[x30, y30] = seq(30);

F10 = lsmatrix(x10, 5);
F20 = lsmatrix(x20, 5);
F30 = lsmatrix(x30, 5);

p10 = paramiter(F10, y10);
p20 = paramiter(F20, y20);
p30 = paramiter(F30, y30);

f10 = sum(F10'.*p10);
f20 = sum(F20'.*p20);
f30 = sum(F30'.*p30);

figure('name', 'sin10')
%set(gca, 'YScale', 'log')
plot(x10,y10, 'o')
hold on
plot(x10,f10)

figure('name', 'sin20')
%set(gca, 'YScale', 'log')
plot(x20,y20, 'o')
hold on
plot(x20,f20)

figure('name', 'sin30')
%set(gca, 'YScale', 'log')
plot(x30,y30, 'o')
hold on
plot(x30,f30)

%--------------------------------------------




i=1;
sigma = logspace(-5,-1,50);
for c=sigma
    rmsc(i,1)=1000;
    mrsc(i,1)=1000;
    for n=5:55
        for k=3:n-1
            [x,y] = seq(n);
            F = lsmatrix(x,k);
            p = paramiter(F,y);
            f = (sum(F'.*p))';
            rms(n,k) = (norm((f-
y),2)/norm(y,2));
            mrs(n,k) = (norm((f-
y),inf)/norm(y,inf));

            yc=y+randn(n,1)*c;
            pc=paramiter(F,yc);
            fc=(sum(F'.*pc))';

            a=norm((fc-yc),2)/norm(yc,2);
            if(a<rmsc(i,1))
                rmsc(i,1) = a;
                rmsc(i,2) = n;
                rmsc(i,3) = k;
            end
                a=norm((fc-yc),inf)/norm(yc,inf);
                if(a<mrsc(i,1))
                    mrsc(i,1) = a;
                    mrsc(i,2) = n;
                    mrsc(i,3) = k;
                end
            end
        end
        i=i+1;
    end

    figure('name','rms')
    surf(log10(rms));
    xlabel('K');
    ylabel('N');
    zlabel('RMS');

    figure('name','mrs')
    surf(log10(mrs));
    xlabel('K');
    ylabel('N');
    zlabel('MAX');


    a = polyfit(sigma,rmsc(:,1)',1);
    b = logspace(-5,-1);
    c = (polyval(a,b));

    figure
    set(gca, 'XScale', 'log')
    loglog(sigma,rmsc(:,1),'o');
    hold on
    loglog(b,c);


    a = polyfit(sigma,mrsc(:,1)',1);
    b = logspace(-5,-1);
    c = (polyval(a,b));

    figure
    set(gca, 'XScale', 'log')
    loglog(sigma,mrsc(:,1),'o');
    hold on
    loglog(b,c);



    %---------------------------------------------
    ---------

    function [x,y] = seq(N)
    %UNTITLED5 Summary of this function goes here
    %   Detailed explanation goes here

    x = linspace(-1,1,N)';
    y = sin(x.*pi).*exp(x- 1/3);

    end

    function [p] = paramiter(F, y)
    %UNTITLED10 Summary of this function goes
    here
    %   Detailed explanation goes here

    p = (transpose(F)*F)\transpose(F) * y;

    end
```

```
function [F] = lsmatrix(x,K)
%UNTITLED12 Summary of this function goes
here
%   Detailed explanation goes here
F(:,1) = ones(size(x,1),1);
F(:,2) = x;
for k=3:K
    F(:,k) = 2*x.*F(:,k-1) - F(:,k-2);
end
end
```