04 June 2019

Warsaw University of Technology – EiTI faculty

Solving ordinary differential equations

No. 38

Wiktor Bara

Course name: Numerical Methods

Advisor: Andrzej Miękina

**Table of contents\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* Theoretical introduction page 2
* Concise Descriptions of Numerical Algorithms (NAs) page 2
* Methodology of Testing page 3
* The Results and Discussion of Testing page 5
* The Overall Conclusions page 7
* The List of References page 7
* Full Code page 8

**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 Algorithm & Methodology\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The differential equation that is to be solved using algorithms below:

The Algorithms below require matrix **A** that can be obtain in the way:

after derivation

the values included in matrix are the coefficients in the system of equations and are equal:

Buthcer (Gauss-Legendre) implicit method

The method of order 6 that use the table below to obtain the approximated function from differential equation (K=3, p=6). That method use the matrix **A**  that was computed above.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

The description of algorithm of implicit method:

with:

after application to ODE system:

The sixth order Gauss-Legendre method yields:

with:

or:

Euler explicit method

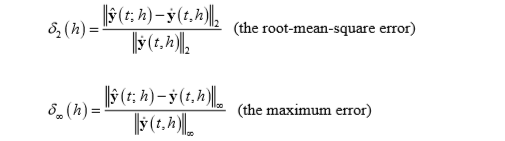
That method also uses the matrix **A**  but the fact that the method is explicit the computation is much easier:

after application to ODE system:

The built in function **ode113** is used as a method to obtain reference values of computed differential equations. That one of the input arguments had to be modified to provide ability to compute differential equations of order 2. The function multiply the input values of y byu matrix **A.**

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

That procedure compares the functions obtained using the Gauss-Legendre method as well as Euler explicit method in comparison to built in method **ode113**.



**Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Obraz zawierający mapa, tekst

Opis wygenerowany automatycznie

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. Obraz zawierający mapa, tekst

Opis wygenerowany automatycznieObraz zawierający mapa

Opis wygenerowany automatycznie

Figure 3) N=30 K=5

Figure 2) N=20 K=5

Figure ) N=10 K=5

Obraz zawierający tekst, mapa

Opis wygenerowany automatycznie

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 4) N=10 K=4

Obraz zawierający mapa, tekst

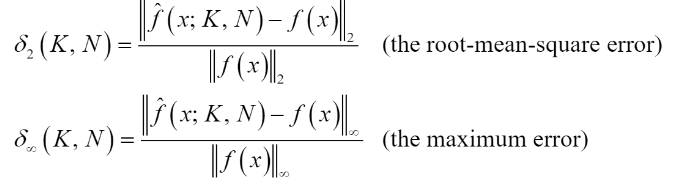
Opis wygenerowany automatycznieObraz zawierający mapa, tekst

Opis wygenerowany automatycznie

Figure 6) N=10 K=9

Figure 5) N=10 K=6

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



The results of investigation are presented on figures below

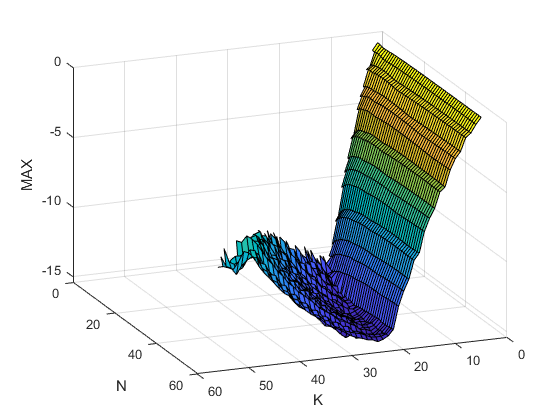
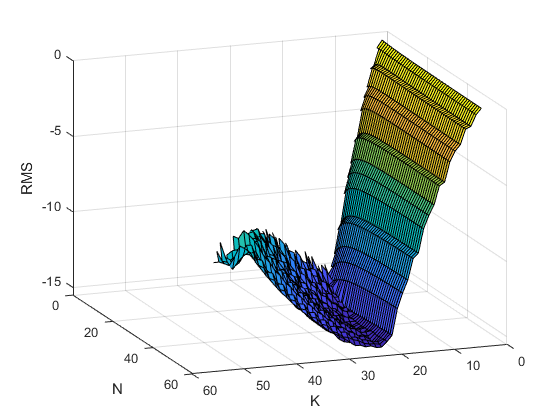


Figure 8) maximum error

Figure 7) root-mean-square 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:



The is the pseudorandom number obtained using the normal distribution with the variance using the build in function randn. For each value of the values of K and N that minimizes norms were determined. The pairs of () were used to approximate the sequence using build in function polyfit. The results are described on figures below.

Obraz zawierający tekst, mapa

Opis wygenerowany automatycznieObraz zawierający tekst, mapa

Opis wygenerowany automatycznie

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