Analysis of the degree of polynomials in the interpolation problem

Alicja Winnicka
Institute of Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: Alicja.Lidia.Winnicka@gmail.com

***Abstract*—Interpolation is a tool based on finding a specific function that passes through certain points. This issue is important from the point of view of applications in data processing, and in particular finding a curve showing some initial conditions. In this work, interpolation with multitude of different degrees is analyzed due to their advantages and disadvantages.**

I. Introduction

Mathematics gave the backbone to many fields of science, in particular to computer science. A large number of more and more complex algorithms or applications is based on the determination of specific elements. To this end, a numerical approach is used that the computer is able to perform. However, this approach means approximation, since it is not always possible to accurately determine any values for continuous functions. The reason is an infinite set of values.

The approximation of results is a very important aspect, above all in biometry. In [1], the authors used curve approximation to modify the set of points representing the signature. Again in [2], the idea of using interpolation was used for the purpose of verification of security protocols. Another area where approximation is used is feature extraction [3] and classification [4]. In addition, this type of mathematical operations are used in the process of creating rules for fuzzy systems [5]. In this paper, we analyze different degrees of polynomials in the interpolation process.

II. Polynomial Interpolation

Polynomial interpolation is a method in numerical analysis, which purpose is to find the polynomial approximated to the original function. The interpolation works, when we have a set of points x_0, x_1,..., x_n, which are called nodes and their values y_0, y_1,...,y_n. The searched polynomial must satisfied a few requirements: its degree must be smaller or equal to $(n-1)$ and points (x_i, y_i) must belong to it. It means, that $w(x_0) = y_0$, $w(x_1) = y_1$,..., $w(x_n) = y_n$ and it will be an approximation of searched function.

Interpolation is usually used in the situation, when we have a certain number of measurements, for example in physics experiment, and we have to find an approximated function for them.

Let us assume, that polynomial $w(x)$ has a following formula

$$w(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_2 x^2 + a_1 x + a_0 \quad (1)$$

Rememebring that $w(x_i) = y_i$ we can write the system of equations

$$\begin{cases} a_n x_0^n + a_{n-1}x_0^{n-1} + ... + a_1 x_0 + a_0 = y_0 \\ a_n x_1^n + a_{n-1}x_1^{n-1} + ... + a_1 x_1 + a_0 = y_1 \\ ... \\ a_n x_{n-1}^n + a_{n-1}x_{n-1}^{n-1} + ... + a_1 x_{n-1} + a_0 = y_{n-1} \\ a_n x_n^n + a_{n-1}x_n^{n-1} + ... + a_1 x_n + a_0 = y_n \end{cases} . \quad (2)$$

which can be calculated as the system of matrices

$$\begin{bmatrix} x_0^n & x_0^{n-1} & x_0^{n-2} & ... & x_0 & 1 \\ x_1^n & x_1^{n-1} & x_1^{n-2} & ... & x_1 & 1 \\ ... & ... & ... & ... & ... & ... \\ x_n^n & x_n^{n-1} & x_n^{n-2} & ... & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ ... \\ a_0 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ ... \\ y_n \end{bmatrix} \quad (3)$$

We have to find coefficients a_0, a_1,...,a_n, but the condition number of this matrix may be large. Additionaly number of equations may cause difficulty in calculating it fast enough. In due to it, the simpler method was created – writing the polynomial as Lagrange's polynomial.

A. Lagrange's Polynomial

The Lagrange's polynomial is a easier way to find searched polynomial of the function. Instead of calculating a system of equation or matrices, we must find the polynomial using specific formula

$$w(x) = \sum_{i=0}^{n} y_i \prod_{j=0 \wedge j \neq i}^{n} \frac{x - x_j}{x_i - x_j} \quad (4)$$

Values y_i can be replaced with values of function our $f(x_i)$ and then polynomial will have following formula

$$w(x) = \sum_{i=0}^{n} f(x_i) \prod_{j=0 \wedge j \neq i}^{n} \frac{x - x_j}{x_i - x_j} \quad (5)$$

and $w(x)$ will be an approximation of function $f(x)$ and will have common points in x_0, x_1,..., x_n.

The Alg. 1 shows the pseudocode of finding the Lagrange's polynomial for any number of nodes n.

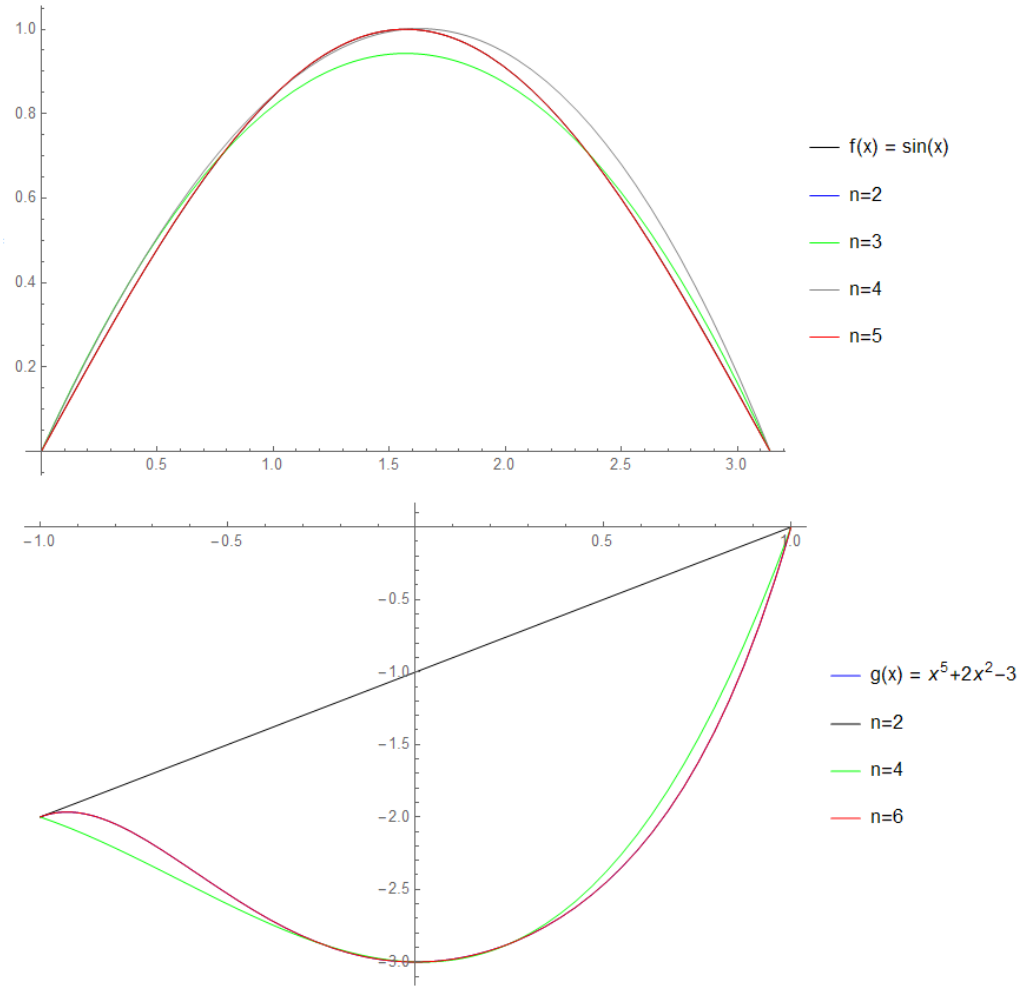

Figure 1: Two functions with their polynomial interpolations for various n.

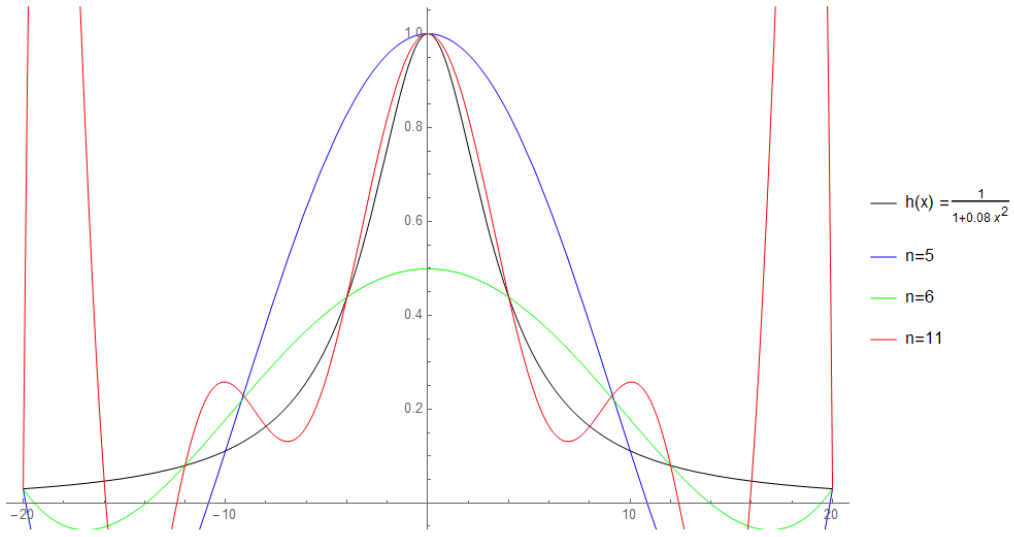

Figure 2: The function, which shows us the problem called Runge's phenomenon.

Data: Given set of points $x_0, x_1, ..., x_n$ and values in these points $y_0, y_1, ..., y_n$
Result: The polynomial w approximated to the original function

```
Start;
w(x) = 0;
i = 0;
j = 0;
for i ≤ n do
    Φ = 1;
    for j ≤ i − 1 do
        Φ = Φ (x − x_j)/(x_i − x_j);
        j++;
    end
    for i + 1 ≤ j ≤ n do
        Φ = Φ (x − x_j)/(x_i − x_j);
        j++;
    end
    w = w + y_i Φ;
    i++;
end
Return w as found polynomial;
Stop;
```

Algorithm 1: Polynomial interpolation algorithm.

B. Error of polynomial interpolation

All of approximation methods are determined by an error dependent on the features of the method. For the polynomial interpolation method, when we have function $f : [a, b] \to \mathbb{R}$ and its approximated polynomial calculated on set od nodes x_0, x_1,...,x_n, the error have the following formula

$$f(x) - w(x) = \frac{M_n}{n!}(x - x_0)(x - x_1)...(x - x_n) \quad (6)$$

where M_n is determined by the formula:

$$M_n = sup|f^{(n)}(x)| \quad (7)$$

and $f^{(n)}(x)$ means n-th derivative of $f(x)$.

III. Experiments

To find the difference for various n and x_i, we compared polynomials for three functions:

- $f(x) = sin(x) \in [0, \pi]$,
- $g(x) = x^5 + 2x^2 - 3 \in [-1, 1]$,
- $h(x) = \frac{1}{1+0.08x^2} \in [-20, 20]$.

For each of them we used another interval and nodes to show additional features of Lagrange's interpolation. For the trigonometric function $f(x) = sin(x)$ we found the second, third, fourth and fifth degree polynomial. The effect was showed on Fig.1. We can see on the graph, that the second degree polynomial, linear function, is not visible. This is due to our interval and nodes ($x_0 = 0$ and $x_1 = \pi$) and their values, which are 0. Because it is linear function, it found the polynomial $w(x) = 0$. For a quite accurate approximation of $sin(x)$ five nodes was enough. The found polynomial has a following formula

$$w(x) = 0.99x + 0.05x^2 - 0.23x^3 + 0.04x^4 \quad (8)$$

and covers the original function on the graph.

Second analyzed function is $g(x)$, which is third degree polynomial. In this way we can say, that for $n = 4$ it will find precisely the same polynomial as $g(x)$. However it finds a little different polynomial showed on Eq.9.

$$w(x) = -3 - 6.710^{-16z} + 2x^2 + 3.610^{-15}x^3 + 8.810^{-16}x^4 + x^5 \quad (9)$$

We can see, that coefficients for x^3 and x^4 are very close to be 0. In this case if indeed they would equal zero, we would get the function $g(x)$. Similar to previous function $f(x)$, the third degree polynomial $w(x)$ covers original function on the graph.

The above results suggest us, that for the bigger number of the nodes we will get polynomial, which is more approximated to original function. However with bigger n, the calculating last longer and becomes less effective.

The third function $h(x) = \frac{1}{1+0.08x^2}$ is moe complicated, because it shows us the problem called Runge's phenomenon. It is happening for the functions, where our nodes are equidistant. Then with increasing the degree of polynomial, quality of interpolation increases, but after a while rapidly deacreases. It is the most visible at the end of the interval, in this case $[-20, 20]$. The Fig. 2 shows the effect of eqidistant nodes. We can see, that each one graph for increased degree is less effective atthe ends of interval. It is the most visible for tenth degree (red color), where difference between polynomial and original function $h(x)$ is huge.

To prevent increasing of this error, we should find another nodes, which will be concetrated at the ends of interval. For this function we used two set of nodes to find tenth degree polynomial:

$$x_i = -20, -16, -12, -8, -4, 0, 4, 8, 12, 16, 20 \quad (10)$$

and

$$x_1 = -20, -19, -17, -15, -10, -5, 0, 5, 10, 15, 17, 19, 20 \quad (11)$$

where the first one shows us the Runge's phenomenon and the other is to reduce difference between function and its polynomial. Comparision of both polynomials is shown on Fig.3.

Increasing density of nodes at the ends of the interval causes aproaching the polynomial to the original function, but it also caused less effectivness for values about $[-10, -5]$ and $[5, 10]$. To reduce these errors we should find more nodes in this interval and calculate a polynomial with the bigger degree. However, the calculations will last longer because of bigger set of data, which will decrease effectiveness of the interpolation.

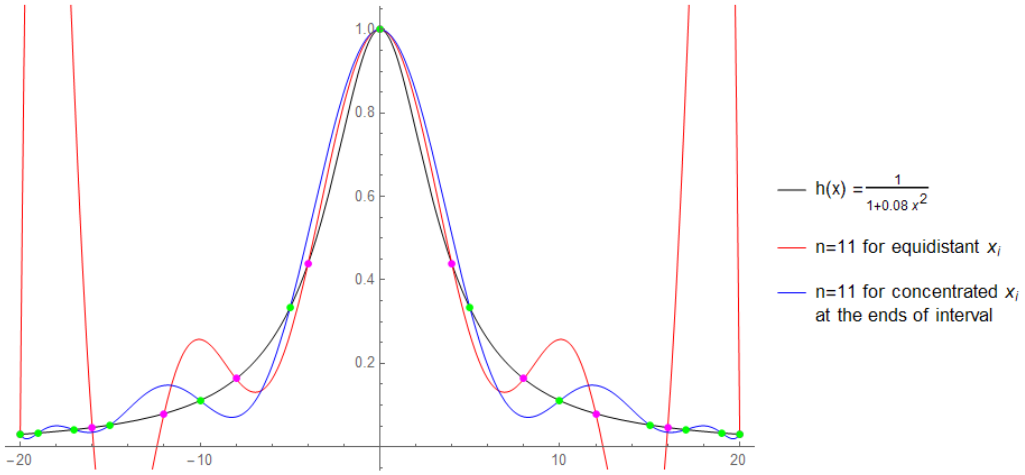

Figure 3: Two different set of nodes for the same function h(x).

IV. Conclusion

In this paper we compared the degree of polynomials in polynomial interpolation. This is accurate method for function, which are similar to polynomials and usually the better result is given for the bigger number of nodes. Howevere with increasing number of nodes and degree of polynomial, the time for calculations increases too.

References

[1] D. Połap and M. Woźniak, "Flexible neural network architecture for handwritten signatures recognition," *International Journal of Electronics and Telecommunications*, vol. 62, no. 2, pp. 197–202, 2016.

[2] M. Rocchetto, L. Viganò, and M. Volpe, "An interpolation-based method for the verification of security protocols," *Journal of Computer Security*, vol. 25, no. 6, pp. 463–510, 2017.

[3] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Weinberger, "Deep feature interpolation for image content changes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7064–7073.

[4] S. Niklaus, L. Mai, and F. Liu, "Video frame interpolation via adaptive convolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 670–679.

[5] L. Yang, F. Chao, and Q. Shen, "Generalized adaptive fuzzy rule interpolation," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 4, pp. 839–853, 2016.

[6] R. Damaševičius, C. Napoli, T. Sidekerskienė, and M. Woźniak, "Imf mode demixing in emd for jitter analysis," *Journal of Computational Science*, vol. 22, pp. 240–252, 2017.

[7] M. Wozniak, C. Napoli, E. Tramontana, G. Capizzi, G. L. Sciuto, R. K. Nowicki, and J. T. Starczewski, "A multiscale image compressor with rbfnn and discrete wavelet decomposition," in *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–7.