# MATLAB EXAMPLES FOR INTERPOLATION

**Dr Ljubica Dikovic**[1]
[1] College of Applied Science Uzice, Uzice, SERBIA, e-mail (dikoviclj@gmail.com)

***Abstract:*** *In this paper author is going to explore interpolation using MATLAB software as a very interesting topic of numerical analyses, which is applied in many engineering areas. Thanks to the code generated in MATLAB, it is possible to successfully understand this complex mathematical apparatus. Also, Chebyshev approximation and its relation to polynomial interpolation at equidistant nodes have been discussed on the example which is very similar with Runge's function.*

***Key words:*** *interpolation, Lagrangian polynomial, MATLAB, Equidistant network, Chebyshev polynomials*

## 1. INTRODUCTION

Let's assume that some functions are tabulated (see Table 1):

| x | $x_0$ | $x_1$ | $x_2$ | ⋯ | $x_n$ | Interpolation nodes |
|---|---|---|---|---|---|---|
| $y = f(x)$ | $y_0$ | $y_1$ | $y_2$ | ⋯ | $y_n$ | Value of function in nodes |

Table 1.

Interpolation is the process of finding another function that is in some sense close to the first,that is, finding a function that approximates well and which is easy for calculate.
If there is a function with ($n$+1) nodes, for example, $n$ -th polynomial is searched for.
It is required a polynomial by the $n$ –th degree which satisfies the following:

$$P_n(x_i) = y_i, i = 0,1, \cdots n \quad (1)$$

and it is called interpolant. The polynomial and the data function coincide in the nodes of interpolation, while in other nodes it does not have to be the case.
If the distance between the nodes is the same, the nodes are called equidist. Equivalent nodes make up an equidistant network. The distance between the two nodes is called the interpolition step and it is marked with $h$,

$$h = d(x_i, x_{i+1}). \quad (2)$$

The concept of interpolation is based on the idea that there exists a polynomial P (x) such that it is

$$f(x_i) = y_i P_n(x_i) = y_i, i = 0,1, \cdots n. \quad (3)$$

Polynomials can be of the following form:

$$P_n(x_i) = \sum_{k=0}^{n} y_k I_k(x_i), I_k(x_i) = \delta_{ik} = \begin{cases} 1, i = k \\ 0, i \neq k \end{cases} \quad (4)$$

Must be valid for the following:

$$P_n(x_i) = y_i = \sum_{k=0}^{n} y_k I_k(x_i) = \sum_{k=0}^{n} y_k, \quad i = 0,1, \cdots n, \quad (5)$$

where $I_k$ is called basic polynomials.

Are there polynomials $l_k$?

$$l_k = c(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n). \quad (6)$$

Let it be

$$w(x) = (x-x_0)(x-x_1)\cdots(x-x_n) \quad (7)$$

$$w'(x) = (x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n) \quad (8)$$

Then it is

$$l_k(x) = \frac{w(x)}{(x-x_k)w'(x_k)}. \quad (9)$$

The polynomial determined in this way is called Lagrange polynom

$$L_n(x) = \sum_{k=1}^{n} \frac{w_{n+1}(x)}{(x-x_k)w_{n+1}'(x_k)} \cdot f(x_k),\ w_{n+1}(x) = \prod_{k=1}^{n}(x-x_k)\ .\ [5],\ [11] \quad (10)$$

## 2. INTERPOLATION BY LAGRANGIAN POLYNOMIAL

Problem 1: Points $(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ are assigned. A task is to determine the least-degree polynomial, whose graph contains all these points.
For the data given in the table, determine the Lagrange interpolation polynomial.

| xi | 0.5 | 1 | 2.5 |
|---|---|---|---|
| yi | -4.250 | -1.525 | 1.465 |

$$l_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-1)(x-2.5)}{(0.5-1)(0.5-2.5)} \quad (11)$$

$$l_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x-0.5)(x-2.5)}{(1-0.5)(1-2.5)} \quad (12)$$

$$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x-0.5)(x-1)}{(2.5-0.5)(2.5-1)} \quad (13)$$

The requested Lagrange interpolation polynom is

$$L_2(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x) = -1.4288x^2 + 8.0425x - 7.8392. \quad (14)$$

Now we will solve the same example using MATLAB. The Lagrange interpolation polynomial calculates the direct command polyfit (x, y, n), where the values of the coordinates x and y are stored in one-dimensional matrices x and y, and n (= number of points-1) is the degree of the required polynomial. We get a polynomial in the form of a one-line matrix of coefficients by potentials. The appropriate MATLAB code is shown below:

```
x=[0.5   1   2.5];
y=[-4.250   -1.525   1.465];
p=polyfit(x,y,2)
p =
-1.7283   8.0425   -7.8392
```

The requested Lagrange interpolation polynom is (14).
Visualization of the Lagrangian polynomial is obtained by entering the following code:

```
x1=0:0.1:3;
plot(x,y,'*',x1,polyval(p,x1))
title('Exploring Lagrange interpolation ','fontsize',12)
ylabel('y','fontsize',12)
xlabel('x','fontsize',12)
legend('Data Points',' Lagrange Interpolation')
hold
```

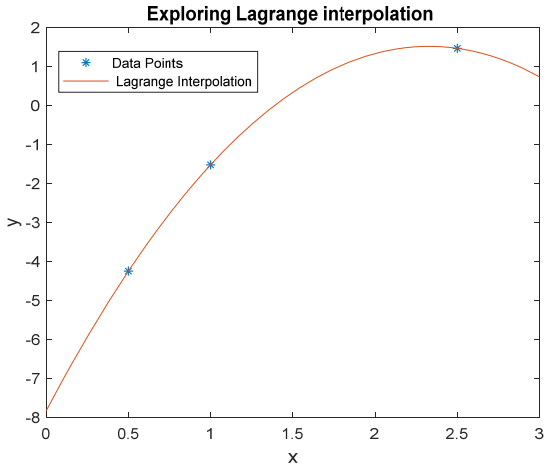


Figure 1. Visualization of the Lagrangian polynomial

Problem 2: The goal is to approximate the function $y = \sin(x)$ by the Lagrange interpolation polynomial at the points with apsis 0, 1, 3, 4, 5. Estimate the absolute error of approximation at the point x = 2 and x = -1.

```
x=[0   1    3    4    5];
y=sin(x);
p=polyfit(x,y,4)
```

The result of this script written in MATLAB is the following polynomial $P(x)$:

$$P(x) = 0.0158x^4 - 0.0727x^3 - 0.8118x^2 + 1.2102x - 0.0000.$$

Determination of the absolute error error of approximation at the point x = 2 and x = -1 is calculated from the following MATLAB commands:

```
>> abs(polyval(p,2)-sin(2))
>>abs(polyval(p,-1)-sin(-1))
```

It is concluded that Lagrange polynomial error in point 2 is approximately tolerable 0.0649 (interpolation data), and at -1 (out of interval) there are not tolerable 0.5920 (see Figure 2).
The folloowing MATLAB script for drawing comparative graphics of the original and interpolated functions is written. Interpolation was carried out using a network of equidistant points on the interval [-1,6].

```
x1=-1:0.05:6;
plot(x,y,'*',x1,sin(x1),x1,polyval(p,x1),'g')
legend('x1','sin(x1)','polyval(p,x1)')
title('Exploring Lagrange interpolation ','fontsize',12)
ylabel('y','fontsize',12)
xlabel('x','fontsize',12)
```

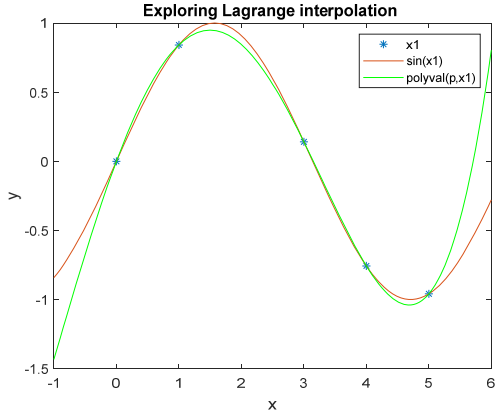


Figure 2. Comparative graphics of the original and interpolated functions

Problem 3. By using MATLAB, perform the interpolation of the function $y = 1./(1 + 20 * x.^2)$ on the interval [-1, 1] by Lagrange's interpolation polynomial of degree 9.
a) Extract the interpolation on an equidistant network.
b) For the interpolation nodes, take the roots of Chebyshev polynomials.

Here's the MATLAB code that produced the plot.

```
% Function for interpolation
f = inline('1./(1+20*x.^2)');
% Number of nodes
n = 10;
% Grid
X = linspace(-1,1);
 % Equidistant network
xeqd = linspace(-1, 1, n);
% Roots of Chebyshev polynomials
xceb = cos((2*(n-1:-1:0)+1) / (2*n)*pi);
% We calculate the values of the function in the points of both
networks
yeqd = feval(f, xeqd);
yceb = feval(f, xceb);
% interpolation
Leqd = polyfit(xeqd, yeqd, n-1);
Lceb = polyfit(xceb, yceb, n-1);
% Equidistant network
subplot(1,2,1); grid on; plot(X, f(X), 'b', X, polyval(Leqd, X),
'r', xeqd, yeqd, 'ro')
title('f(x)- Equidistant network'); legend('f(x)', 'Leqd(x)');
 % Roots of Chebyshev polynomials
subplot(1, 2, 2); grid on; plot(X, f(X), 'b', X, polyval(Lceb, X),
'r', xceb, yceb, 'ro')
title('f(x)–      Roots      of      Chebyshev      polynomials');
legend('f(x)','Lceb(x)');
```

The result of the script is displayed in the Figure 3.
Note that this example shows that the oscillation of the interpolation polynomial and the error can be significantly reduced if they are used roots of Chebyshev polynomials'for interpolation.
The Chebyshev Polynomials (of the first kind) are defined by

$$T_n(x) = \cos[n \cdot \arccos(x)]. \quad (15)$$

Also, the Chebyshev polynomials are defined by the triple recurrence relation

$$T_0(x) = 1, T_1(x) = x, T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad (16)$$

Chebyshev polynomials are important in approximation theory because the roots of the Chebyshev polynomials of the first kind, which are also called Chebyshev nodes, are used as nodes in polynomial interpolation. The resulting interpolation polynomial minimizes the problem of Runge's phenomenon and provides an approximation that is close to the polynomial of best approximation to a continuous function under the maximum norm. A Chebyshev polynomial of either kind with degree n has n different simple roots, called Chebyshev roots, in the interval [−1,1]. The roots of the Chebyshev polynomial of the first kind are sometimes called Chebyshev nodes because they are used as nodes in polynomial interpolation. Using the trigonometric definition and the fact that

$$\cos\left((2k+1)\frac{\pi}{2}\right) = 0 \quad (17)$$

one can easily prove that the roots of $T_n$ are

$$x_k = \cos\left(\frac{\pi\left(k+\frac{1}{2}\right)}{n}\right), k = 0,1,\cdots n. \text{ [12], [1],[2],[3],[4][5]} \quad (18)$$

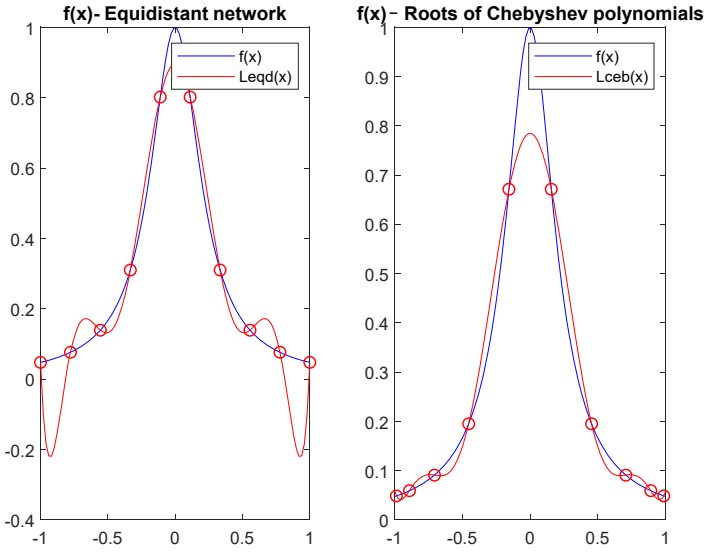


Figure 3. Comparative visualization of the the function $y = 1./(1 + 20 * x.^2)$ for n=10

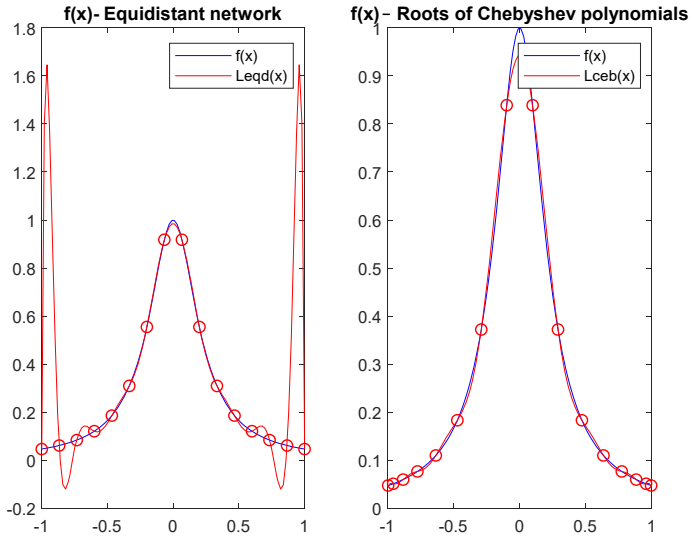


Figure 4. Comparative visualization of the the function $y = 1./(1 + 20 * x.^2)$ for n=16

## CONCLUSION

It is well known that fitting a polynomial to a function at more points might not produce a better approximation. If the function you're interpolating is smooth, then interpolating at more points may or may not improve the fit of the interpolation, depending on where you put the points. The famous example of Runge shows that interpolating Runge's function at more points can make the fit worse i.e. when interpolating at 16 evenly spaced points, the behavior is wild at the ends of the interval. In this paper, Chebyshev approximation and its relation to polynomial interpolation at equidistant nodes has been discussed on the example which is very similar with Runge's function.

## LITERATURE

[1] H. E. Salzer, Lagrangian Interpolation at the Chebyshev Points xn, ν ≡ cos (νπ/n), ν = 0(1)n; some Unnoted Advantages, The Computer Journal, Volume 15, Issue 2, 1972, P.156–159, https://doi.org/10.1093/comjnl/15.2.156

[2] W.S. Don, D. Gottlieb, The Chebyshev–Legendre Method: Implementing Legendre Methods on Chebyshev Points, SIAM Journal on Numerical Analysis, 1994, 31:6, 1519-1534
[3] K. Xu, The Chebyshev points of the first kind, Journal Applied Numerical Mathematics,Volume 102, Issue C, 2016, P. 17-30, Elsevier Science Publishers doi>10.1016/j.apnum.2015.12.002
[4] J. Boyd, Finding the Zeros of a Univariate Equation: Proxy Rootfinders, Chebyshev Interpolation, and the Companion Matrix. SIAM Review. (2013). 55. 10.1137/110838297.
[5] D. Radunovic , A. Samardzic, F.Maric, Numericke metode Zbirka zadataka kroz C, Fortran i Matlab, Akademska misao, Beograd, 2005
[6]www.halvorsen.blog/documents/teaching/courses/matlab/powerpoint/MATLAB%20Examples%20-%20Interpolation%20and%20Curve%20Fitting.pdf
[7] http://www.eas.uccs.edu/~mwickert/ece1010/lecture_notes/1010n6a.PDF
[8] http://www.matrixlab-examples.com/linear-regression.html
[9] http://matlab.izmiran.ru/help/techdoc/ref/interp1.html
[10] https://csivc.csi.cuny.edu/supernova7/files/robbins/CSC270/mat/LESSON7.pdf
[11] http://www.math.rs/p/files/16-02V_INTERPOLACIJA.pdf
[12]https://www.etf.ues.rs.ba/~pnata/Numeri%C4%8Dka%20matematika-skenirane%20vje%C5%BEbe/NUMET_zbirka%20Desanka%20R.pdf