Construction and Optimization of Stability Conditions of Learning Processes in Mathematical Models of Neurodynamics

Andriy Shatyrko[1], Denys Khusainov[1], Oleksii Bychkov[1], Josef Diblik[2] and Jaromir Baštinec[2]

[1] *Taras Shevchenko University of Kyiv, 64, Volodymyrska str., Kyiv, 01033, Ukraine*
[2] *Brno University of Technology, Technická 3058/10, Brno, 61600, Czech Republic*

Abstract

This article is devoted to dynamic processes in the field of artificial intelligence, namely in the tasks of neurodynamics: the field of knowledge in which neural networks are considered as nonlinear dynamical systems and focuses on the problem of stability. The systems under consideration share four common characteristics: a large number of nodes (neurons), nonlinearity, dissipativity, noise.

The purpose of this work is to build to construct of asymptotic stability conditions for dynamic model of neuronet network, which is described in terms of ODE nonlinear systems. Main method of investigation is Lyapunov direct method. Authors show that solution of pointed problem can be reduced to the task of convex optimization. By realization on Python tools the algorithm of Nelder-Mead method, a number of numerical experiments were conducted to select the optimal parameters of the Lyapunov function.

Keywords [1]

Neuronet model, differential equation system, software, stability, Lyapunov function.

1. Introduction

To date, it is difficult to overestimate the achievements of neural networks and their contribution to various fields of science, to the development of the world as a whole and to the lives of each of us in particular. Deep learning is used in a huge number of fields from image recognition, weather prediction or text translations to the creation of new medicines and unique works of art [1-5]. Nevertheless, very often the apparatus of neural networks is not investigated properly, and the programs themselves are used as "black boxes": data is given as an input, and the desired prediction is obtained as an output. It is clear that it is simply impossible to build a good working architecture without understanding how a neural network works.

One of the urgent tasks of the modern theory of neural networks is learning. The dynamics of processes, and more precisely the process of "learning", comparing with processes in electric circuits, can be described using the apparatus of ordinary differential equations. In particular, by a system of stationary nonlinear differential equations with a selected linear part and nonlinearity of a special kinde. This direction of research is relevant, which is confirmed by many recent scientific works [6-11]. One of the universal apparatuses of the mathematical theory of stability, allowing to study the dynamics of the learning process, i.e. the convergence of the solutions of the system to the established one, is the second Lyapunov method [12,13].

2. Formulation of the problem

Without limiting the generality the authors first consider a dynamic system on a plane. It is a system of two nonlinear differential equations of the form

Information Technology and Implementation (IT&I-2022), November 30 - December 02, 2022, Kyiv, Ukraine
EMAIL: shatyrko.a@knu.ua (A. 1); d.y.khusainov@gmail.com (A. 2); oleksiibychkov@knu.ua (A. 3); diblik@vut.cz (A. 4); bastinec@vut.cz (A. 5)
ORCID: 0000-0002-5648-2999 (A. 1); 0000-0001-5855-029X (A. 2); 0000-0002-9378-9535 (A. 3); 0000-0001-5009-316X (A. 4); 0000-0002-6228-3857 (A. 5)

$$\dot{y}_1 = -a_{11}y_1 + b_{11}f_{11}(y_1) + b_{12}f_{12}(y_2) + c_1,$$
$$\dot{y}_2 = -a_{22}y_2 + b_{21}f_{21}(y_1) + b_{22}f_{22}(y_2) + c_2. \qquad (1)$$

We assume that $a_{11} > 0,\ a_{12} > 0$, functions $f_{ij}(y),\ i,j = \overline{1,2}$ are monotonic and continuously differentiable, and the system of differential equations (1) has a single singular point $M_0(y_1^0,\ y_2^0)$, which is a solution of the system of equations

$$-a_{11}y_1 + b_{11}f_{11}(y_1) + b_{12}f_{12}(y_2) + c_1 = 0,$$
$$-a_{22}y_2 + b_{21}f_{21}(y_1) + b_{22}f_{22}(y_2) + c_2 = 0. \qquad (2)$$

Let's make a replacement, a parallel transfer type, a fixed point $M_0(y_1^0,\ y_2^0)$ to the origin of the coordinates

$$y_1 = x_1 + y_1^0,$$
$$y_2 = x_2 + y_2^0. \qquad (3)$$

Then, taking into account (2), (3), system (1) will be reduced to the form

$$\dot{x}_1 = -a_{11}x_1 + b_{11}F_{11}(x_1) + b_{12}F_{12}(x_2),$$
$$\dot{x}_2 = -a_{22}x_2 + b_{21}F_{21}(x_1) + b_{22}F_{22}(x_2), \qquad (4)$$

where

$$F_{11}(x_1) = f_{11}(x_1 + y_1^0) - f_{11}(y_1^0),$$
$$F_{12}(x_2) = f_{12}(x_2 + y_2^0) - f_{12}(y_2^0),$$
$$F_{21}(x_1) = f_{21}(x_1 + y_1^0) - f_{21}(y_1^0),$$
$$F_{22}(x_2) = f_{22}(x_2 + y_2^0) - f_{22}(y_2^0). \qquad (5)$$

Since

$$F_{11}(0) = 0, F_{12}(0) = 0, F_{21}(0) = 0, F_{22}(0) = 0,$$

then the study of the equilibrium position $M_0(y_1^0,\ y_2^0)$ and the convergence of solutions to this point is reduced to the study of the stability of the zero equilibrium position $O(0,0)$ of the system (4).

Let the functions $F_{11}(x_1),\ F_{12}(x_2),\ F_{21}(x_1),\ F_{22}(x_2)$ from (5) satisfy the so-called "sector conditions", which can be written in a compact form as follows:

$$\big(L_{11}x_1 - F_{11}(x_1)\big)F_{11}(x_1) > 0,$$
$$\big(L_{12}x_2 - F_{12}(x_2)\big)F_{12}(x_2) > 0,$$
$$\big(L_{21}x_1 - F_{21}(x_1)\big)F_{21}(x_1) > 0,$$
$$\big(L_{22}x_2 - F_{22}(x_2)\big)F_{22}(x_2) > 0. \qquad (6)$$

In fact, this means that these functions are located in the first and third sectors of the coordinate plane. Such functions are, for example,

$$F(x) = \frac{1 - e^{-vx}}{1 + e^{-vx}}, \qquad F(x) = arctgx,$$

which are successfully used in the design of neural networks as activation functions [1].

If the conditions (6) are satisfied, the asymptotic stability conditions and convergence estimates can be obtained using the Lyapunov function of the Lurie-Postnikov type [14]. Since the linear part has a diagonal form, we construct the Lyapunov function in the form of a sum of squares and integral additions with constant coefficients

$$V(x_1, x_2) = h_{11}x_1^2 + h_{22}x_2^2 + \int_0^{x_1}\big(\gamma_{11}F_{11}(s) + \gamma_{21}F_{21}(s)\big)ds +$$
$$+ \int_0^{x_2}\big(\gamma_{12}F_{12}(s) + \gamma_{22}F_{22}(s)\big)ds, \qquad (7)$$
$$h_{11} > 0, h_{22} > 0, \gamma_{11} > 0, \gamma_{21} > 0, \gamma_{12} > 0, \gamma_{22} > 0.$$

Let introduce next notation

$$C = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{12}^T & C_{22} & C_{23} \\ C_{13}^T & C_{23}^T & C_3 \end{pmatrix}, \qquad C_1 = \begin{pmatrix} C_{11}^1 & C_{12}^1 & C_{13}^1 \\ (C_{12}^1)^T & C_{22}^1 & C_{23}^1 \\ (C_{13}^1)^T & (C_{23}^1)^T & C_{33}^1 \end{pmatrix},$$

$$C_{11}=\begin{pmatrix}2h_{11}a_{11} & 0\\ 0 & 2h_{22}a_{22}\end{pmatrix},\qquad C_{12}=\begin{pmatrix}\frac{1}{2}\gamma_{11}a_{11}-h_{11}b_{11} & -h_{11}b_{12}\\ 0 & \frac{1}{2}\gamma_{12}a_{22}\end{pmatrix},$$
$$C_{13}=\begin{pmatrix}\frac{1}{2}\gamma_{21}a_{11} & 0\\ -h_{22}b_{21} & \frac{1}{2}\gamma_{22}a_{22}-h_{22}b_{22}\end{pmatrix},\quad C_{22}=\begin{pmatrix}-\gamma_{11}b_{11} & -\frac{1}{2}\gamma_{11}b_{12}\\ -\frac{1}{2}\gamma_{11}b_{12} & 0\end{pmatrix},$$
$$C_{23}=\begin{pmatrix}-\frac{1}{2}\gamma_{21}b_{11} & 0\\ -\frac{1}{2}\gamma_{21}b_{12}-\frac{1}{2}\gamma_{12}b_{21} & -\frac{1}{2}\gamma_{12}b_{22}\end{pmatrix},\ C_{33}=\begin{pmatrix}0 & -\frac{1}{2}\gamma_{22}b_{21}\\ -\frac{1}{2}\gamma_{22}b_{21} & -\gamma_{22}b_{22}\end{pmatrix},$$
$$C_{11}^{1}=\begin{pmatrix}0&0\\0&0\end{pmatrix},\ C_{22}^{1}=\begin{pmatrix}k_{11}&0\\0&k_{12}\end{pmatrix},\ C_{23}^{1}=\begin{pmatrix}0&0\\0&0\end{pmatrix},\ C_{33}^{1}=\begin{pmatrix}k_{21}&0\\0&k_{22}\end{pmatrix},$$
$$C_{12}^{1}=\begin{pmatrix}-\frac{1}{2}k_{11}L_{11} & -\frac{1}{2}k_{12}L_{12}\\ 0 & 0\end{pmatrix},\ C_{13}^{1}=\begin{pmatrix}0 & 0\\ -\frac{1}{2}k_{21}L_{21} & -\frac{1}{2}k_{22}L_{22}\end{pmatrix}. \tag{8}$$

The following statement holds.

Theorem 1. *Let there exist constants $h_{11}>0, h_{22}>0, \gamma_{11}>0, \gamma_{21}>0, \gamma_{12}>0, \gamma_{22}>0$, such that the matrix $C+C_1$ is positive-definite. Then the zero-equilibrium position of the system (4) will be asymptotically stable.*

Proof. From dependence (7) it follows that the function $V(x_1,x_2)$ satisfies the following two bilateral inequalities

$$h_{11}x_1^2+h_{22}x_2^2\le V(x_1,x_2)\le\left(h_{11}+\frac{1}{2}\gamma_{11}L_{11}+\frac{1}{2}\gamma_{12}L_{21}\right)x_1^2+$$
$$+\left(h_{22}+\frac{1}{2}\gamma_{21}L_{12}+\frac{1}{2}\gamma_{22}L_{22}\right)x_2^2,$$

i.e. is a positive definite function.

Let's calculate its total derivative due to system (4). We will get

$$\frac{d}{dt}V(x_1,x_2)==\left(2h_{11}x_1+\gamma_{11}F_{11}(x_1)+\gamma_{21}F_{21}(x_1)\right)\times\left(-a_{11}x_1+b_{11}F_{11}(x_1)+b_{12}F_{12}(x_2)\right)+$$
$$+(2h_{22}x_2+\gamma_{12}F_{12}(x_2)+\gamma_{22}F_{22}(x_2))\times\left(-a_{22}x_2+b_{21}F_{21}(x_1)+b_{22}F_{22}(x_2)\right)=$$
$$=-2h_{11}a_{11}x_1^2-\gamma_{11}a_{11}F_{11}(x_1)x_1-\gamma_{21}a_{11}F_{21}(x_1)x_1+$$
$$+2h_{11}b_{11}F_{11}(x_1)x_1+\gamma_{11}b_{11}F_{11}^2(x_1)+\gamma_{21}b_{11}F_{21}(x_1)F_{11}(x_1)+$$
$$+2h_{11}b_{12}F_{12}(x_2)x_1+\gamma_{11}b_{12}F_{11}(x_1)F_{12}(x_2)+\gamma_{21}b_{12}F_{21}(x_1)F_{12}(x_2)-$$
$$-2h_{22}a_{22}x_2^2-\gamma_{12}a_{22}F_{12}(x_2)x_2-\gamma_{22}a_{22}F_{22}(x_2)x_2+$$
$$+2h_{22}b_{21}F_{21}(x_1)x_2+\gamma_{12}b_{21}F_{12}(x_2)F_{21}(x_1)+\gamma_{22}b_{21}F_{22}(x_2)F_{21}(x_1)+$$
$$+2h_{22}b_{22}F_{22}(x_2)x_2+\gamma_{12}b_{22}F_{12}(x_2)F_{22}(x_2)+\gamma_{22}b_{22}F_{22}^2(x_2).$$

Let introduce next notation

$$z^T=(x_1,x_2,F_{11}(x_1),F_{12}(x_2),F_{21}(x_1),F_{22}(x_2)).$$

Then the total derivative of the Lyapunov function due to system (4) can be rewritten as a quadratic form

$$\frac{d}{dt}V(x_1,x_2)=-z^TCz.$$

The matrix C in the last expression is not positive definite. Therefore, in order to make the quadratic form negative definite, we use the “sector conditions” (6) and write the total derivative of the Lyapunov function in the form

$$\frac{d}{dt}V(x_1,x_2)=-z^TCz+$$
$$+k_{11}\left(L_{11}x_1-F_{11}(x_1)\right)F_{11}(x_1)+k_{12}\left(L_{12}x_2-F_{12}(x_2)\right)F_{12}(x_2)+$$
$$+k_{21}\left(L_{21}x_1-F_{21}(x_1)\right)F_{21}(x_1)+k_{22}\left(L_{22}x_2-F_{22}(x_2)\right)F_{22}(x_2)-$$
$$-k_{11}\left(L_{11}x_1-F_{11}(x_1)\right)F_{11}(x_1)-k_{12}\left(L_{12}x_2-F_{12}(x_2)\right)F_{12}(x_2)-$$
$$-k_{21}\left(L_{21}x_1-F_{21}(x_1)\right)F_{21}(x_1)-k_{22}\left(L_{22}x_2-F_{22}(x_2)\right)F_{22}(x_2).$$

Let's enter the first terms in the quadratic form, we get

$$\frac{d}{dt}V(x_1,x_2) = -z^T(C + C_1)z -$$
$$-k_{11}\big(L_{11}x_1 - F_{11}(x_1)\big)F_{11}(x_1) - k_{12}\big(L_{12}x_2 - F_{12}(x_2)\big)F_{12}(x_2) -$$
$$-k_{21}\big(L_{21}x_1 - F_{21}(x_1)\big)F_{21}(x_1) - k_{22}\big(L_{22}x_2 - F_{22}(x_2)\big)F_{22}(x_2).$$

Since the inequalities (6) hold, we can discard the remaining terms and write down the estimate of the complete derivative of the Lyapunov function (7)

$$\frac{d}{dt}V(x_1,x_2) \le -z^T(C + C_1)z.$$

The condition for the negative definiteness of the total derivative of Lyapunov function (7) due to the system (4) will be the positive definiteness of the sum of the matrices $C + C_1$. In this way, it was possible to construct a positive definite Lyapunov function, the total derivative of which is negative definite due to the system, and therefore the zero equilibrium position of system (4) will be asymptotically stable. Which had to be proven. The matrices in (8) depend on arbitrary parameters $h_{11} > 0, h_{22} > 0, \gamma_{11} > 0, \gamma_{21} > 0, \gamma_{12} > 0, \gamma_{22} > 0,\ k_{11} > 0, k_{21} > 0, k_{12} > 0, k_{22} > 0$. Therefore, the problem of stability research is reduced to the problem of finding variables for which the matrix $C + C_1$ will be positive definite. In this case, the equilibrium position will be asymptotically stable. Since the minimum eigenvalue of a symmetric positive definite matrix is a convex function, in the general case the problem is reduced to a convex optimization problem with constraints.

2.1. Nelder-Mead method for solving the problem

The use of methods for optimizing functions that require the existence of a gradient is not always practical. This applies especially to those cases when the gradient of the function either does not exist, or it is impractical to calculate it. The Nelder-Mead method or the deformed polyhedron method [15,16] is one of the non-smooth optimization methods used to find the optimum of a function. This method is easy to implement and useful in practice, but, on the other hand, there is no theory of convergence for it - the algorithm can diverge even on smooth functions. One of the main features of the Nelder-Mead algorithm is high efficiency for a possible complex calculation of the function. This is due to the fact that at each step it is necessary to calculate no more than $n + 1$ values of the investigated function for its further analysis, where n is the dimension of the space. The basis of the method is the idea of comparing the values of the function in $n + 1$ vertices of the constructed simplex, and its iterative shift in the direction of the optimal value. The graphic representation of the method, which will be demonstrated later in the work, explains another of its interesting names - the "amoeba method". Let there exist a real function of many variables $f(x), x \in \mathbb{R}^n$. The task of optimizing $f(x) \to \min$ or $f(x) \to \max$ is set, while $f(x)$ does not necessarily have to be smooth, and its noise is allowed. The Nelder-Mead method, which will be used for this task, has mandatory parameters and certain stages of work. The specified parameters $\alpha > 0, \beta > 0, \gamma > 0$ are closely related to the idea of deformation of the $n + 1$ -dimensional simplex, and, accordingly, specify the reflection, compression and stretching of the polyhedron. In practice, standard values are most often chosen for these parameters: $\alpha = 1,\ \beta = 0.5,\ \gamma = 2$.

Stages of the algorithm [16]:

1) Preparatory stage. Randomly select $n + 1$ -space point $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)}),\ i = 1 \dots n + 1$, which satisfies all the constraints of the problem. These points will form the initial simplex. Values of the function $f_i(x_i), i = 1 \dots n + 1$ are calculated at these $n + 1$ points.

2) Sorting and centering. The vertices of the simplex determined at the previous stage are sorted in order of increasing value of the function $f(x)$ in them. Among these values, three are chosen: x_h, which corresponds to the largest value of the function $f_h(x_h)$, x_g is the value of $f_g(x_g)$, next after f_h, and also the smallest - x_l for f_l. We find the center of the simplex x_c without the point x_h.

$$x_c = \frac{1}{n}\sum_{i=1, i\neq h}^{n} x_i. \qquad (9)$$

3) Stages of deformation: reflection. The point x_h is displayed in x_r relative to the point x_c found by formula (9) with the coefficient α:

$$x_r = (1+\alpha)x_c - \alpha x_h.$$

We find the value of the function $f_r = f(x_r)$ for the point x_r .

4) Checking the direction.

4.1. If $f_r < f_l$, then we try to improve the result obtained in the previous step, get a new point x_e and the value of the function f_e in it, using the stretching coefficient γ

$$x_e = (1-\gamma)x_c - \gamma x_r.$$

If $f_e < f_r$, that is, the solution found at this step is even better than the previous one, then x_h takes the value of x_e , otherwise - the value of x_r . After that, we move on to the next iteration.

4.2. If $f_l < f_r < f_g$, then x_h takes value x_r. After that, we move on to the next iteration.

4.3. If $f_g < f_r < f_h$, then x_h takes value x_r else – x_r takes value x_h. After that go to step 5.

4.4. If $f_h < f_r$, then finding point x_r is not satisfied point, that is why cross to step 5.

5) Stages of deformation: compression. When the attempt to stretch the simplex did not give results, we try to compress it using the given parameter β . We calculate the point x_s and the value of the function in it

$$x_s = \beta x_h + (1-\beta)x_c.$$

6) Checking the direction.

6.1. If $f_s < f_h$, then x_h takes value x_s. We cross to the next iteration.

6.2. If $f_s > f_h$, go to step 7.

7) Stages of deformation: global compression. If the previous steps did not bring improvement in finding the optimum of the function, it means that the initial points turned out to be the best, we make the compression to the point x_i

$$x_i \leftarrow x_l + \frac{x_i - x_l}{2}, i \neq l.$$

8) Verification of convergence and stopping of the algorithm. Conditions for stopping the program are selected. For example, performing a set number of iterations, or achieving a certain accuracy. After checking, we return to step two, or, accordingly, stop the algorithm and get the desired result.

Remarks 1. The previous algorithm is described for the problem of minimizing the function $f(x)$. To find the maximum of this function, all values of $f_i(x_i)$, $i = 1 \dots n+1$ should be compared with the opposite sign.

For a better explanation, the Nelder-Mead algorithm can be visualized for the two-dimensional case. Then the $(n+1)$-dimensional simplex will be a triangle (Fig. 1), and all actions of the algorithm will be easy to represent in the form of ordinary transformations of a simple geometric figure: reflection, expansion and contraction of the triangle relative to its center of gravity without the worst point (marked in blue in Fig. 2). The first step of the algorithm is to display the point with the largest value (marked in red in Fig. 3) of the function relative to the center of gravity. At the new point, the current value of the function is calculated, which determines the next actions.

If the value at the displayed point is even smaller, that is, it is better in the context of the minimization problem, then we perform stretching (Fig. 4). Conversely, if the value at the displayed point turned out to be not too good, we compress the simplex in the direction of the center (Fig. 5). If the previous steps do not contribute to the minimization of the function, then global compression is performed to the point with the smallest value (Fig. 6). In practice, there are very rare cases when it is necessary to apply this last type of polyhedron deformation.

Remarks 2. The deformation parameters α, β, γ , chosen before starting the algorithm, are responsible for how strong the shape will be compressed or stretched at each step. For example, the reflection coefficient $\alpha = 1$ specifies a symmetrical reflection relative to the center. The compression parameter $\beta = 0.5$ moves the examined point to a distance equal to half the distance to the center of gravity, and $\gamma = 2$ – the stretching coefficient moves it to twice the distance, respectively.

This method optimizes the objective function quite quickly and efficiently. On the other hand, due to the lack of convergence theory, in practice the method can lead to incorrect answers even on smooth (continuously differentiable) functions. A situation is also possible when the working simplex is far from the optimal point, and the algorithm performs a large number of iterations, while changing the value of the function little. A heuristic method for solving this problem is to run the algorithm several times and limit the number of iterations. So, having analyzed all the advantages and disadvantages of one of the methods of non-smooth optimization - the Nelder-Mead algorithm, we will try to apply it to find the parameters of the Lyapunov function (7).

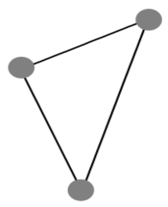
Figure 1: Initial simplex

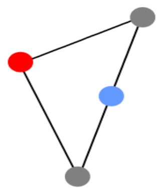
Figure 2: Center of gravity

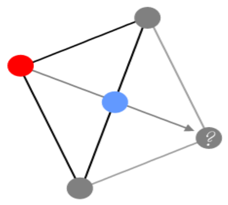
Figure 3: Reflection

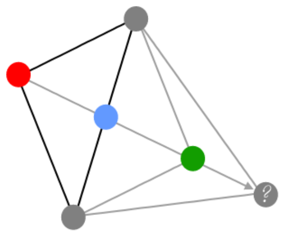
Figure 4: Stretching

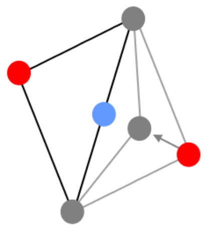
Figure 5: Compression

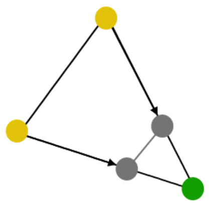
Figure 6: Global compression

2.2. Realization

The Python programming language version 3.6.9 was chosen for the numerical experiment. The Google Colab interactive cloud environment [17] served as the development and implementation environment. The choice of this language is due to its powerful mathematical functionality, which allows solving a sufficiently wide class of various problems, and the ability to graphically display the results of work, which is an important part of any development. The easy-to-learn environment of Google Colab is also convenient for mathematical calculations, as it provides free access to the use of powerful GPU and TPU processors. Thanks to this, all calculations are carried out quickly, and do not load the personal computer. Among the libraries used for the program are NumPy, SciPy for convenient operations with multidimensional arrays, PrettyTable for displaying results in a table, Plotlyb and Matplotlib for creating two-dimensional and three-dimensional visualization, as well as additional tools for generating random numbers, fixing execution time. The program itself consists of several parts: the first is the initialization of parameters and the definition of basic functions for matrices, the second is the implementation of the Nelder-Mead algorithm, the third is a graphical demonstration of the algorithm for the two-dimensional case of Nelder-Mead (simplex - triangle) and numerical comparisons for different initial simplexes and parameters. Testing was carried out for one-dimensional and two-dimensional cases. Accordingly, the coefficients and matrices $C + C_1$ from the formulas found in (8) are specified separately for each of these cases.

3. Results and comparisons

Let's test the operation of the algorithm using examples of finding all three parameters h, k, γ. Let's choose the initial values $a = 10, b = -50, h = 10, k = 10, L = 10, \gamma = 35$. Than the matrix:

$$C + C_1 = \begin{pmatrix} 200 & 625 \\ 625 & 1760 \end{pmatrix}$$

will have a minimum eigenvalue $\lambda_{min} \approx -19{,}51$. Let's run the program for different initial simplexes, which are formed by using five different values of Δ: 0,01; 0,5; 1; 5; 10. This means that to form a polyhedron, a different value of Δ will be added to one of the coordinates in turn for each of the tests. We get the following graph - Fig. 7. The minimum eigenvalue is plotted on the ordinate axis, and the number of steps taken by the algorithm is shown on the abscissa axis. The map legend shows in color the correspondence between the graphs and the choice of the initial simplex, and the final minimum value obtained as a result for each of the cases is signed. It can be seen on the graph that the initial simplex formed with $\Delta = 0{,}5$ has the best result, and the worst with $\Delta = 0.01$, which could not even choose the parameters for which the matrix will be positive definite. It can also be seen that the minimum eigenvalue calculated for each of the selected examples at each step does not grow infinitely. For example, for $\Delta = 0{,}5$ it took about 60 steps for the value to stop changing, while for $\Delta = 1$ it took just over 120 steps.

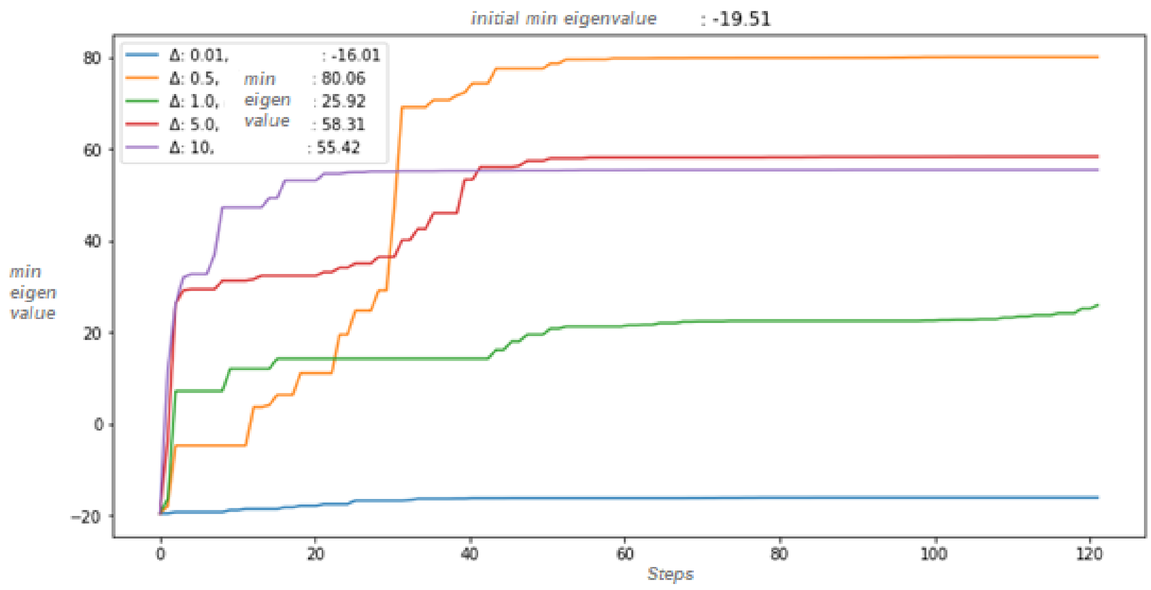

Figure 7: Comparison of the Nelder-Mead algorithm for different initial simplexes without updating the center of mass (one-dimensional system)

The initial simplex for the best of the obtained cases $\Delta = 0{,}5$, formed on the basis of the initial parameters $h = 10, k = 10, \gamma = 35,$ has $n + 1 = 4$ vertices: $(10{,}5;\ 35; 10), (10;\ 35{,}5; 10), (10;\ 35; 10{,}5), (10;\ 35; 10).$
Accordingly, the optimized parameters are $h \approx 4$, $k \approx 0{,}79$, $\gamma \approx 40{,}82$, and the positive definite matrix $C + C_1$ will have the form

$$C + C_1 = \begin{pmatrix} 80{,}08 & 0{,}04 \\ 0{,}04 & 80{,}17 \end{pmatrix}, \lambda_{min} \approx 80{,}06.$$

Let's try to slightly improve the work results by updating the center of mass after each deformation of the polyhedron. Let's run the program again on the same data, and compare the obtained results (Fig. 8) for 20 steps of the algorithm.

We see that now the minimum eigenvalue of the matrix has become much larger and reaches $\lambda_{min} \approx 1892{,}98$ in just 20 steps. In this case, large value of parameters $\Delta = 10$ and $\Delta = 5$ give the best value for the formation of the initial simplex, on the other hand, small Δ did not increase the minimum eigenvalue too much, although they fulfilled the condition of positive definiteness of the matrix $\lambda_{min} > 0$. It is worth noting that for the experiment in Fig. 8 as the number of steps increases, the λ_{min} grows very quickly. For example, for 100 steps and $\Delta = 5$, the minimum eigenvalue reaches $\lambda_{min} \approx 7{,}07 \times 10^{17}$. Now consider a two-dimensional system of equations for which the matrix $C + C_1$ has a dimension of 6×6. Let the parameters be set as follows:

$a_{11} = 6, a_{22} = 6, b_{11} = -3, b_{12} = -2, b_{21} = -2, b_{22} = -3, L_{11} = 2, L_{12} = 3, L_{21} = 4, \quad L_{22} = 7,\ h_{11} = 10, h_{22} = 92, \gamma_{11} = 57, \gamma_{12} = 2, \gamma_{21} = 1, \gamma_{22} = 71, k_{11} = 99, k_{12} = 74, k_{21} = 59, k_{22} = 71.$

Then the matrix $C + C_1$:

$$C + C_1 = \begin{pmatrix} 120 & 0 & 102 & -91 & 3 & 0 \\ 0 & 1104 & 0 & 6 & 66 & 240{,}5 \\ 102 & 0 & 270 & 57 & 1{,}5 & 0 \\ -91 & 6 & 57 & 74 & 3 & 3 \\ 3 & 66 & 1{,}5 & 3 & 59 & 71 \\ 0 & 240{,}5 & 0 & 3 & 71 & 284 \end{pmatrix}, \lambda_{min} \approx -36{,}03.$$

Let's run the Nelder-Mead algorithm for this case, and again analyze the results that depend on the choice of the initial simplex - Fig. 9.

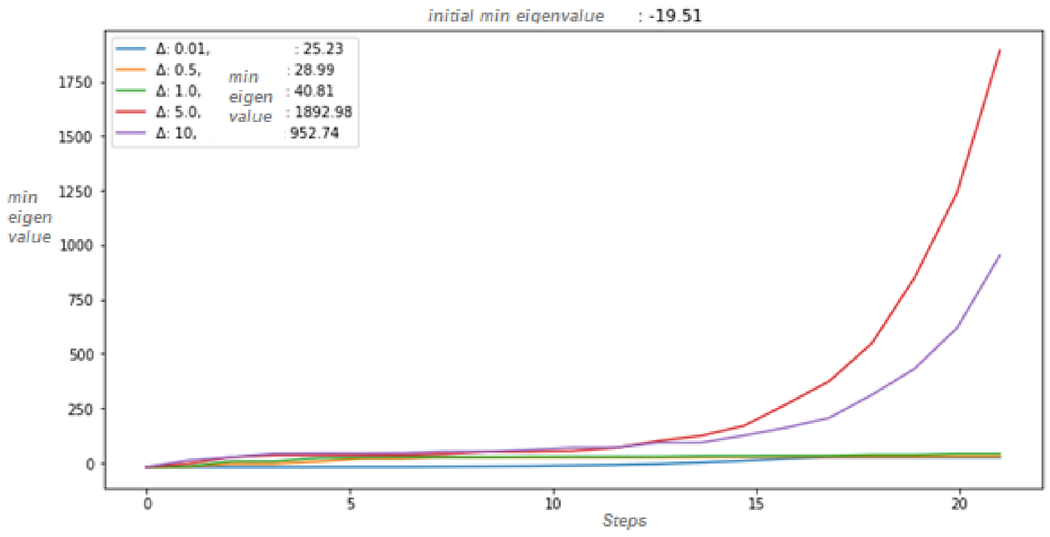

Figure 8: Comparison of the Nelder-Mead algorithm for different initial simplexes with updating the center of mass (one-dimensional system)

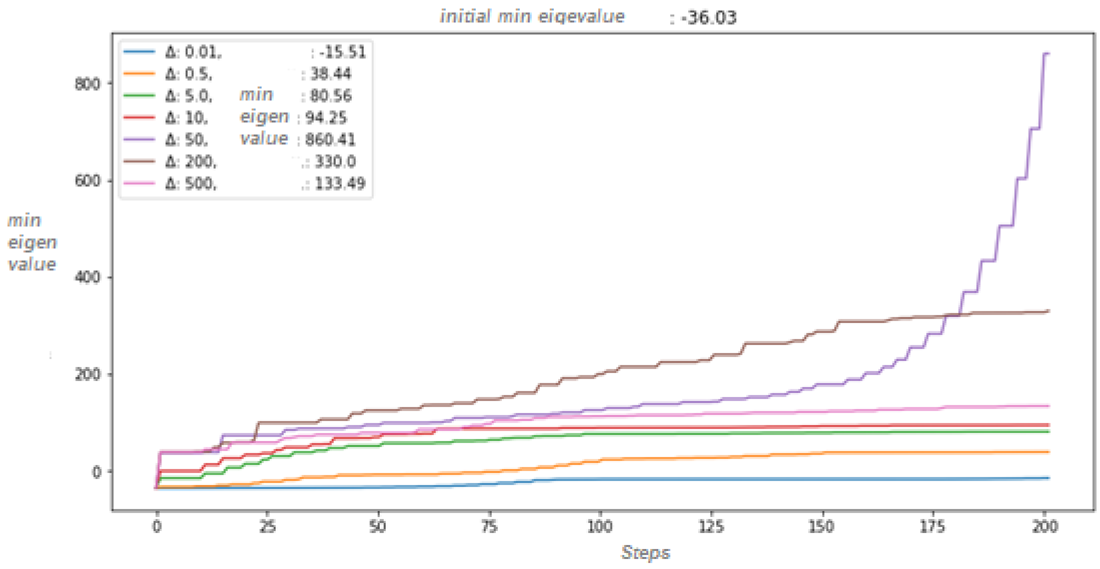

Figure 9: Comparison of the Nelder-Mead algorithm for different initial simplexes with updating the center of mass (two-dimensional system)

This graph shows that, unlike the previous results for the one-dimensional case, not every parameters set for different steps was able to maximize the minimum eigenvalue of the matrix. For example, for a small $\Delta = 0{,}01$, the minimum eigenvalue remained negative. We see that the value $\Delta = 50$ performed best. For it, the result of 200 steps of the algorithm is the optimized matrix

$$C + C_1 \approx\approx \begin{pmatrix} 6{,}2\times10^4 & 0 & -5{,}8\times10^3 & -1{,}3\times10^3 & -1{,}8\times10^2 & 0 \\ 0 & 3{,}2\times10^4 & 0 & 2{,}3\times10^2 & -5{,}3\times10^3 & -1{,}2\times10^4 \\ -5{,}8\times10^3 & 0 & 6{,}1\times10^3 & 49 & 88{,}9 & 0 \\ -1{,}3\times10^3 & 2{,}3\times10^2 & 49 & 9\times10^2 & 1{,}4\times10^2 & 1{,}2\times10^2 \\ -1{,}8\times10^2 & -5{,}3\times10^3 & 88{,}9 & 1{,}4\times10^2 & 5{,}4\times10^3 & 3{,}2\times10^3 \\ 0 & -1{,}2\times10^4 & 0 & 1{,}2\times10^2 & 3{,}2\times10^3 & 1{,}8\times10^4 \end{pmatrix},$$

and for it $\lambda_{min} \approx 34{,}67$ with the parameter values found by the algorithm $h_{11} \approx 5164{,}74$; $h_{22} \approx 2693{,}52$; $\gamma_{11} \approx 48{,}96$; $\gamma_{12} \approx 77{,}44$; $\gamma_{21} \approx 59{,}25, \gamma_{22} \approx 3248{,}9; k_{11} \approx 5983{,}56$; $k_{12} \approx 903{,}42; k_{21} \approx 5359{,}02$; $k_{22} \approx 8521{,}24$.

Let's run the program for the same initial parameters, but without updating the center of mass for each operation. We will get the following results (Fig. 10):

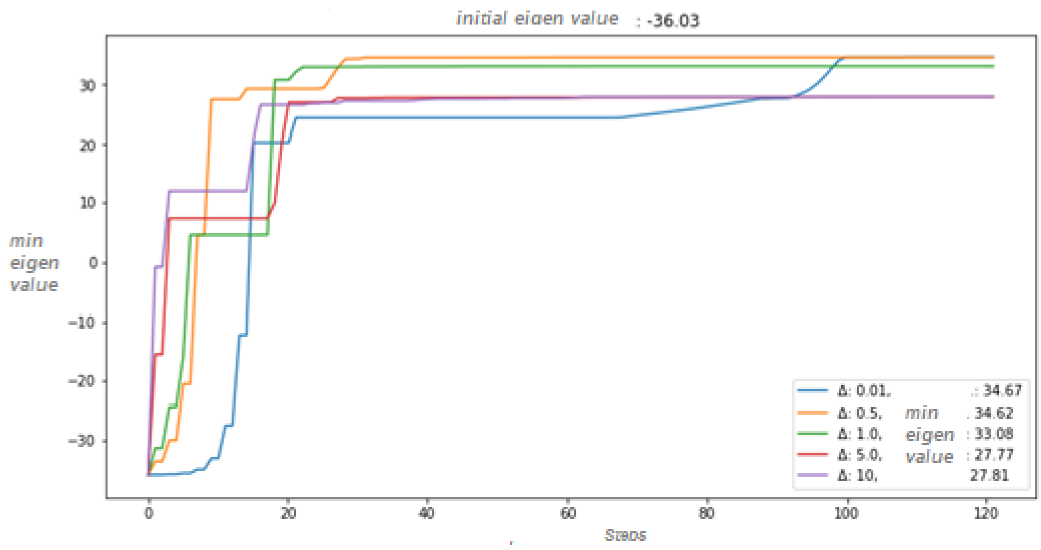

Figure 10: Comparison of the Nelder-Mead algorithm for different initial simplexes without updating the center of mass (two-dimensional system)

Comparing with the previous result (Fig. 9) for the two-dimensional case, the experiment gave much worse results. In addition, all the obtained values are approximately the same. We see that $\Delta = 0{,}5$ and $\Delta = 0{,}01$ performed best. Although the first simplex converged in about 30 steps, while the second only in 100. We conclude that for this case small values of Δ are better suited for determining the initial simplex.

4. Conclusion

The following results were achieved in this work:

- Conditions for the stability of learning processes in neurodynamics models were obtained on the example of the Hopfield network using the second Lyapunov method.
- The computer program was written that solves the problem of parameter selection in the Lyapunov function using the Nelder-Mead method with constraints for one-dimensional and two-dimensional dynamical systems.

As a result of the analysis of the work of the computer program, the following conclusions were made. The Nelder-Mead algorithm or the deformed polyhedron method, used for cases of optimization of a function without a gradient, coped well with the task of maximizing the minimum eigenvalue of a symmetric positive definite matrix for one-dimensional and two-dimensional dynamical systems. In particular, the parameters for obtaining a positive minimum eigenvalue were selected in just a few steps. It was also analyzed how the selection of the initial simplex for the Nelder-Mead method affects the obtained results for both cases. As a result, such Δ was obtained for the simplex, for which the minimum eigenvalue constantly increases with increasing steps. This means that, depending on the mathematical problem, in further studies, with the help of this program, it will be possible to select the necessary values to obtain the maximum positive definite matrix of any dimension. All results are constructive from the point of view of performing computational experiments. In the future, they can be extended to the case of multidimensional systems. It is also known that Hopfield networks are more adequately described in terms of functional-differential equations with a time-delay. And all the results presented in this work can be used to continue the original author's research, started in the their works [18,19].

5. Acknowledgements

Work is conducted under the Agreement on scientific cooperation between Faculty of Computer Science and Cybernetics Taras Shevchenko National University of Kyiv, Ukraine and the Faculty of Electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic. This research was supported by the project of specific university research at Brno University of Technology FEKT-S-20-6225 and research project #BF015-04 at Taras Shevchenko National University of Kyiv.

6. References

[1] Simon Haykin, Neural Networks and Learning Machines, 3rd ed., 2009. – 937 c.

[2] H Akca, R Alassar, V Covachev, Z Covacheva, EA Al-Zahrani (2004) Continuous- time additive Hopfield-type neural networks with impulses. J Math Anal Appl 290(2): 436-451.

[3] Y Li, L Lu (2004) Global exponential stability and existence of periodic solutions of Hopfield-type neural networks with impulses. Physics Letters A 333: 62-71.

[4] Castro, Fidelis Zanetti de and Marcos Eduardo Valle. “Continuous-Valued Quaternionic Hopfield Neural Network for Image Retrieval: A Color Space Study.” 2017 Brazilian Conference on Intelligent Systems (BRACIS) (2017): 186-191.

[5] Perry, Stuart W. and Ron J. Wyber. “A Hopfield neural network approach for the reconstruction of wide-bandwidth sonar data.” Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No.00TH8501) 2 (2000): 876-885 vol.2.

[6] Kuroe, Yasuaki and Hitoshi Iima. “A model of Hopfield-type octonion neural networks and existing conditions of energy functions.” 2016 International Joint Conference on Neural Networks (IJCNN) (2016): 4426-4430.

[7] Li, Yongkun et al. “Almost automorphic synchronization of quaternion-valued high-order Hopfield neural networks with time-varying and distributed delays.” IMA J. Math. Control. Inf. 36 (2019): 983-1013.

[8] Valle, Marcos Eduardo and Fidelis Zanetti de Castro. “On the Dynamics of Hopfield Neural Networks on Unit Quaternions.” IEEE Transactions on Neural Networks and Learning Systems 29 (2018): 2464-2471.

[9] Kobayashi, Masaki. “Stability Conditions of Bicomplex-Valued Hopfield Neural Networks.” Neural Computation 33 (2021): 552-562.

[10] Li, Q., Yang, X. (2005). Complex Dynamics in a Simple Hopfield-Type Neural Network. In: Wang, J., Liao, X., Yi, Z. (eds) Advances in Neural Networks – ISNN 2005. ISNN 2005. Lecture Notes in Computer Science, vol 3496. Springer, Berlin, pp 357–362

[11] M. E. Valle and F. Z. de Castro, "On the Dynamics of Hopfield Neural Networks on Unit Quaternions," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 6, pp. 2464-2471, June 2018, doi: 10.1109/TNNLS.2017.2691462.

[12] Lakshmikantham V., Leela S., Martinyuk A.A. Stability Analysis of Nonlinear Systems, - Birkhauser, 2015. – 329p.

[13] Gao, Jin and Lihua Dai. “Anti-periodic synchronization of quaternion-valued high-order Hopfield neural networks with delays.” AIMS Mathematics (2022): DOI:10.3934/math.2022775

[14] A.I. Lur’e, Some Non-Linear Problems in the Theory of Automatic Control, Her Majesty’s Stationery O_ce, London, 1957 (a translation from the Russian original: Nekotorye nelineinye zadachi teorii avtomaticheskogo regulirovaniya, Gos. Isdat. Tekh., Moscow, 1957).

[15] Nelder, J.A. and Mead, R. (1965), “A simplex method for function minimization”, Comput. J., **7**, pp. 308–*313*.

[16] Nelder-Mead algorithm [Електронний *ресурс*]. – 2022. – *Режим доступу:* http://www.scholarpedia.org/article/Nelder-Mead_algorithm.

[17] Welcome To Colaboratory [Electron]. – *2022.* – https://colab.research.google.com

[18] Khusainov D.Ya., Diblik J., Bastinec Ja., Shatyrko A.V. Investigating Dynamics of One Weakly Nonlinear System with Delay Argument // Journal of Automation and Information Sciences, 50(1) 2018. – pp.20-38.

[19] Khusainov D.Ya., Diblik J., Bastinec Ja., Shatyrko A.V. Estimates of Solution Convergence Dynamical Processes in Neuronet with Time Delay / Conference Proceedings “IEEE ATIT 2019”, p.411-414