Modified Kalman Filtering Method for Discrete Signal

Eugene Fedorov[1], Olga Nechyporenko[1], Anait Karapetyan[1], Tetyana Utkina[1]

[1]Cherkasy State Technological University, Shevchenko blvd., 460, Cherkasy, 18006, Ukraine

Abstract

Currently, the development of discrete signal filtering methods that are used in computerized biometric identification systems is an urgent task. In the case of linear filtering, one of the most popular methods is the Kalman filter. A modified Kalman filtering method was proposed to improve the efficiency of digital filtering of a discrete signal. This method provides automation of parameter value determination and improves the speed and accuracy of Kalman filtering by using fewer parameters and identifying them based on immune metaheuristic methods. The proposed metaheuristic methods reduce the probability of convergence to a local extremum by using the Cauchy distribution and make parametric identification more accurate. Algorithms of immune metaheuristic methods for identifying Kalman filter parameters have been developed, which are designed for software implementation on the GPU using CUDA technology, which increases the accuracy of Kalman filtering. Further prospects of the study are to utilize the proposed immune metaheuristic methods for various general and special purpose intelligent systems.

Keywords

continuous optimization, immune metaheuristics, artificial immune network, hybrid immune algorithm, parametric identification of Kalman filter

1. Introduction

Currently, it is an urgent task to create digital filtering methods used for noise suppression, which are used in computerized biometric identification systems as well as speech recognition and understanding systems and computer vision [1-2].

One of the popular methods of digital filtering is the use of recurrent neural networks. Such neural networks include NARMANN, ENN, JNN, GRU, LSTM. The disadvantage of such neural networks is the high computational complexity of identifying their parameters due to the lack of the ability to parallelize the learning algorithm, a large number of connections between neurons and the correct choice of activation functions and the number of neurons in the layers.

Another popular digital filtering methods is smoothing adaptive linear filtering [3-4]. For smoothing adaptive linear filtering, the identification of filter parameters plays an important role.

Approximate methods of determining parameter values based on global search do not guarantee convergence. Approximate methods of determining parameter values based on local search have a high probability of hitting a local extremum. Exact methods of determining parameter values have high computational complexity. Thus, the problem of insufficient quality of parametric identification methods arises.

Modern heuristics (or metaheuristics) are used to increase the speed of filter parameter identification and reduce the probability of hitting a local extremum [5-6]. Metaheuristics expand the capabilities of heuristics by combining heuristic methods based on a high-level strategy [7-8]. Metaheuristics often use the behavior of evolutionary and immune approaches [9-10]. Metaheuristics are approximate and, as a rule, stochastic methods [11-12]. The most effective metaheuristics use experience that accumulates during the search process and is stored in memory [13-14].

Object of the study. The process of Kalman filtering of a discrete signal.

Subject of the study. Kalman filtering method for a discrete signal using parametric identification based on immune metaheuristic methods.

1CMIS-2025: Eighth International Workshop on Computer Modeling and Intelligent Systems, 5 of May 2025, Zaporizhzhia, Ukraine

y.fedorov@chdtu.edu.ua (E. Fedorov); o.nechyporenko@chdtu.edu.ua (O. Nechyporenko); a.karapetian@chdtu.edu.ua (A. Karapetyan); t.utkina@chdtu.edu.ua (T. Utkina)

0000-0003-3841-7373 (E. Fedorov); 0000-0002-3954-3796 (O. Nechyporenko); 0000-0002-7412-3252 (A. Karapetyan); 0000-0002-6614-4133 (T. Utkina)

The aim of the work is to improve the quality of Kalman filtering of a discrete signal using immune metaheuristic methods.

To achieve the stated goal, it is necessary to solve the following tasks:

1. Create a modified Kalman filtering method.
2. Develop a continuous optimization method based on an artificial immune network.
3. Create a continuous optimization method based on a hybrid immune algorithm.
4. Conduct a numerical study of the proposed methods of continuous optimization and Kalman filtering.

2. Literature review

In the pre-processing units of modern computer systems for speaker identification, multimodal interface, medical and technical diagnostics, digital filtering methods are implemented that reduce noise and allow analyzing the spectral features of the signal. Various digital filtering methods, from recurrent neural networks to linear digital filters, require the use of methods for determining parameter values. One of the popular approaches is metaheuristic.

Currently, metaheuristics are divided into evolutionary, biological, non-nature-inspired, immune, mathematical, physical, social, chemical. Most metaheuristics use stochastic search, which reduces the probability of hitting a local extremum. Metaheuristics allow solving continuous optimization problems (calculating the point at which the objective function reaches a maximum or minimum) and discrete optimization problems (e.g., clustering, knapsack problem, traveling salesman problem, assignment problem).

Modern metaheuristics have one or more of the following disadvantages:

- the convergence of the method may not be ensured [15-16];
- the iteration number does not affect the solution finding process [17-18];
- only binary potential solutions are used [19-20];
- the metaheuristic method is associated with solving only one problem or there is only an abstract set of operators of this method available [21-22];
- low accuracy of the method [23-24];
- there is no automation of the procedure of identification of method parameters [25-26];
- the method is not intended for solving problems of conditional optimization [27-28].

Based on this, the problem of constructing high quality metaheuristic optimization methods needs to be addressed. One of the most popular ones are immune metaheuristics.

3. Modified Kalman filtering method

In this paper, it is assumed that for each moment of time n the state matrix $A(n)=A$, the control matrix $B(n)=0$, the control vector $u(n)=0$, the process covariance matrix $Q(n)=Q=I_Q\sigma_Q^2$, the observation matrix $H(n)$ are replaced by the vector $h=1_H\sigma_H^2$, the observed noise covariance matrix $R(n)$ is replaced by the scalar σ_R^2, the observation vector $z(n)$ is replaced by the scalar $z(n)$, the observation vector estimate $z(n)$ is replaced by the scalar $z(n)$, the error vector $e(n)$ is replaced by the scalar $e(n)$, the error vector covariance matrix $D(n)$ is replaced by the scalar $d(n)$, and the optimal Kalman gain matrix $G(n)$ is replaced by the vector $g(n)$. This reduces the amount of computation and the amount of specified data.

The modified Kalman filtering method for a discrete signal consists of the following stages:

1. Initialization.

The initial estimation of the state vector $s(0)$ of length m (in this work $s(0)=0$) is set. The covariance matrix of the state vector estimate $C(0)$ is set with a size of mxm (in this work, $C(0)$ is filled with uniformly distributed values). In this work, the state matrix A of size mxm,

dispersion σ_R^2, dispersion σ_Q^2 for calculating the matrix Q of size $m \times m$, dispersion σ_H^2 for calculating the vector h of length m are set.

2. Forecast stage.

2.1. Calculating the state vector estimate

$$s(n-1)=A(n)s(n-2)+B(n)u(n). \quad (1)$$

In this work $s(n-1)=A \cdot s(n-2)$.

2.2. Calculating the covariance matrix of the state vector estimate

$$C(n-1)=A(n)C(n-2)A^T(n)+Q(n). \quad (2)$$

In this work $C(n-1)=A \cdot C(n-2) \cdot A^T+Q$, where $Q=I_Q\sigma_Q^2$.

3. Update stage.

3.1. Calculating the estimate of the observation vector

$$z(n)=H(n)s(n-1). \quad (3)$$

In this work $z(n)=h \cdot s(n-1)$, where $h=1_H\sigma_H^2$.

3.2. Calculating the error vector

$$e(n)=z(n)-z(n). \quad (4)$$

In this work $e(n)=z(n)-z(n)$.

3.3. Calculating the error vector covariance matrix

$$D(n)=H(n)C(n-1)H^T(n)+R(n). \quad (5)$$

In this work $d(n)=h \cdot C(n-1) \cdot h^T+\sigma_R^2$, where $h=1_H\sigma_H^2$.

3.4. Calculating the optimal Kalman gain matrix

$$G(n)=\frac{C(n-1)H^T(n)}{D(n)}. \quad (6)$$

In this work $g(n)=\frac{C(n-1)h^T}{d(n)}$, where $h=1_H\sigma_H^2$.

3.5. Updating the state vector estimate

$$s(n)=s(n-1)+G(n)e(n). \quad (7)$$

In this work $s(n)=s(n-1)+g(n)e(n)$.

3.6. Updating the covariance matrix of the state vector estimate

$$C(n)=C(n-1)-G(n)H(n)C(n-1). \quad (8)$$

In this work $C(n)=C(n-1)-g(n) \cdot h \cdot C(n-1)$, where $h=1_H\sigma_H^2$.

4. Selection of criteria for evaluating the effectiveness of the modified Kalman filtering method

In this paper, to evaluate the effectiveness of the modified Kalman filtering method, the accuracy criterion is chosen, which means choosing such values of the parameters A, σ_Q^2, σ_H^2, σ_R^2, that provide a minimum root mean square error

$$F=\sqrt{\frac{1}{n^{\max}}\sum_{n=1}^{n^{mac}}(y(n)-z(n))^2}\rightarrow \min_{A,\sigma_Q^2,\sigma_H^2,\sigma_R^2,} \quad (9)$$

where $y(n)$ – noise-free countdown,

$z(n)$ – observation evaluation,

$n^{\max}$ – number of observations.

In accordance with the selected criterion, immune metaheuristic methods for identifying parameters A, σ_Q^2, σ_H^2, σ_R^2 are proposed in this paper.

5. Modified artificial immune network method

The artificial immune network was proposed by Timmis, Neal, Hunt and later modified by de Castro, von Zuben and is based on the hypothesis of representing the immune system as an idiotypic network. The disadvantage of the clonal selection theory is that it assumes that a set of cells remains unexcited when there is no antigen.

Scientist Erne proposed a hypothesis according to which the immune system is a regulated network of molecules and cells that recognize each other even in the absence of an antigen. Such structures are often called idiotypic networks, they serve as a basis for studying the behavior of the immune system. Erne's theory is interpreted as a system of differential equations describing the dynamics of the concentration of lymphocyte clones and the corresponding immunoglobulin molecules. The theory of idiotypic regulation is based on the assumption that different lymphocyte clones are not isolated from each other, but maintain communication through interactions of their receptors located on the surface of the lymphocyte.

In formulating the foundations of his theory, Jerne introduced the concepts of formal and functional networks. Formal networks serve to study issues of repertoire (recruitment), dualism and suppression. When considering functional networks, a quantitative picture of the theory is presented.

A probabilistic approach to studying idiotypic networks based on the work of Jerne was proposed by the scientist Perelson. This approach is extremely formalized and is mainly associated with the description of phase transitions. Perelson divided the plane of phase variables of the considered system of equations into a subcritical region, a transition region, and a postcritical region. Over the past 20 years, Jerne's proposed immune network theory has received considerable attention, which has led to a detailed study of many computational aspects of the corresponding mathematical models.

The modified artificial immune network method consists of the following steps:

1. Initialization.

1.1. Setting the search area: cell length M, minimum and maximum values of cell components $x_j^{\min}, x_j^{\max}$, $j\in\overline{1,M}$. Setting the maximum number of iterations N, population size K, number of clones L_C.

1.2. Setting the cost function (target function) $F(x)\rightarrow \min_x$, where x – cell (real vector containing A, σ_Q^2, σ_H^2, σ_R^2).

1.3. Setting search parameters: mutation parameter α, compression threshold ε, where $\alpha>0$ (the higher the α, the lower the mutation probability), $\varepsilon>0$. The paper it is proposed to use $\delta\left(x_j^{\max}-x_j^{\min}\right), 0<\delta<1$, instead of parameter α.

1.4. Creating an initial population P.

1.4.1. Cell number $k=1$, $P=\varnothing$.

1.4.2. Generating a random cell $x_k=\left(x_{k1},\ldots,x_{kM}\right)$, $x_{kj}=x_j^{\min}+\left(x_j^{\max}-x_j^{\min}\right)U(0,1)$, where $U(0,1)$ – a function that returns a standard uniformly distributed random number.

1.4.3. If $x_k\notin P$, then $P=P\cup\{x_k\}$, $k=k+1$.

1.4.4. If $k\le K$, then move to step 1.4.2.

1.5. Determine the best cell by the target function

$$x^{*}=\arg\min_{x_k}F(x_k),\ k\in\overline{1,K}. \tag{10}$$

2. Iteration number $n=0$.

3. Calculating the affinity of population cells P

$$\Phi(x_k)=1-\frac{F(x_k)-\min\limits_{i\in\overline{1,K}}F(x_i)}{\max\limits_{i\in\overline{1,K}}F(x_i)-\min\limits_{i\in\overline{1,K}}F(x_i)},\ \Phi(x_k)\in[0,1],\ k\in\overline{1,K}. \tag{11}$$

4. Order the population P by the target function, i.e. $F(x_k)<F(x_{k+1})$.

5. Determine the best cell by the target function

$$k^{*}=\arg\min_{k}F(x_k),\ k\in\overline{1,K}. \tag{12}$$

6. Determining the global best cell. If $F\left(x_{k^{*}}\right)<F\left(x^{*}\right)$, then $x^{*}=x_{k^{*}}$.

7. Calculating the average cost value

$$\bar{F}^{source}=\frac{1}{K}\sum_{k=1}^{K}F(x_k). \tag{13}$$

8. Creating the best mutated clones set H.

8.1. Set $k=1$, $H=\varnothing$.

8.2. Creating clones set $\widetilde{P}_k=\{\widetilde{x}_{kl}\}$ for a population cell x_k.

8.3. Creating mutated clones set P_k.

8.3.1. Clone number $l=1$, $P_k=\varnothing$.

8.3.2. Creating a cell (the paper proposes to use the Cauchy distribution)

$$x_{klj}=\widetilde{x}_{klj}+\delta\left(x_j^{\max}-x_j^{\min}\right)e^{-\Phi\left(\widetilde{x}_{kl}\right)}Cauchy(0,1),\ j\in\overline{1,M}, \tag{14}$$

where $Cauchy(0,1)$ – a function that returns a standard Cauchy distributed random number.

8.3.3. Cell correction x_{kl}

$$x_{klj}=\max\left\{x_j^{\min},x_{klj}\right\},\ x_{klj}=\min\left\{x_j^{\max},x_{klj}\right\},\ j\in\overline{1,M}.$$

8.3.4. Calculating $P_k=P_k\cup\{x_{kl}\}$.

8.3.5. If $l<L_C$, then $l=l+1$, go to step 8.3.2.

8.4. Determining the best element of set P_k by the target function $h_k=\arg\min_{x_{kl}} F(x_{kl})$.

8.5. Calculating $H=H\cup\{h_k\}$.

8.6. If $k<K$, then $k=k+1$, go to step 8.2.

9. Calculating the average cost value

$$\bar{F}^{mutate}=\frac{1}{K}\sum_{i=1}^{K}F(h_k). \quad (15)$$

10. If $\bar{F}^{mutate}\geq\bar{F}^{source}$, then move to step 8.

11. Compressing the set H and replacing population cells P with elements of the set H.

11.1. Set $k=1$, $m=1$.

11.2. Forming the ε - neighborhood of the m^{th} element of the set H

$$U_{h_m,\varepsilon}=\{\,\}, \quad (16)$$

where ρ – the distance between h_m and h_l (e.g. Euclidean distance).

11.3. If $|U_{h_m,\varepsilon}|=\varnothing$ or $\max U_{h_m,\varepsilon}=h_m$, then $x_k=h_m$, $k=k+1$.

11.4. If $m<K$, then $m=m+1$, go to step 11.2.

12. If $k=K$, then move to step 14.

13. Initialization of the last population cells P.

13.1. Calculating $x_k=(x_{k1},\ldots,x_{kM})$, $x_{kj}=x_j^{\min}+(x_j^{\max}-x_j^{\min})U(0,1)$.

13.2. If $k<K$, then $k=k+1$, go to step 13.1.

14. If $n<N-1$, then $n=n+1$, go to step 3.

The result is x^{*}.

6. Algorithm of the modified artificial immune network method

The algorithm of the modified artificial immune network method, designed for implementation on GPU using CUDA technology, is shown in Figure 1.

This block diagram functions as follows.

Step 1 – Set the maximum number of iterations N, population size K, number of clones L_C, parameter δ for generating a new solution, compression threshold ε, where $0<\delta<1$, $\varepsilon>0$.

Step 2 – Create an initial population P, using $K\cdot M$ threads that are grouped into K one-dimensional blocks. Each thread calculates $x_{kj}=x_j^{\min}+(x_j^{\max}-x_j^{\min})U(0,1)$.

Step 3 – Determine the best cell by the target function $x^{*}=\arg\min_{x_k} F(x_k)$, $k\in\overline{1,K}$.

Step 4 – Set the iteration number $n=0$.

Step 5 – Calculate the minimum value of the target function a in the current population P, using K threads that are grouped into one one-dimensional block. In this block, the minimum of K elements of the form $F(x_k)$ is calculated based on the reduction.

Step 6 – Calculate the maximum value of the target function b in the current population P, using K threads that are grouped into one one-dimensional block. In this block, the maximum of K elements of the form $F(x_k)$ is calculated based on the reduction.

Step 7 – Calculate the affinity $\Phi(x_k)$ for each cell x_k of population P, using K threads that are grouped into one one-dimensional block $\Phi(x_k)=\frac{b-F(x_k)}{b-a}$.

Step 8 – Order population P by the target function, i.e. $F(x_k)<F(x_{k+1})$.

Step 9 – Determine the best cell by the target function $k^{¿}=\arg\min_k F(x_k)$, $k\in\overline{1,K}$.

Step 10 – Determine the global best cell. If $F(x_{k^{¿}})<F(x^{¿})$, then $x^{¿}=x_{k^{¿}}$.

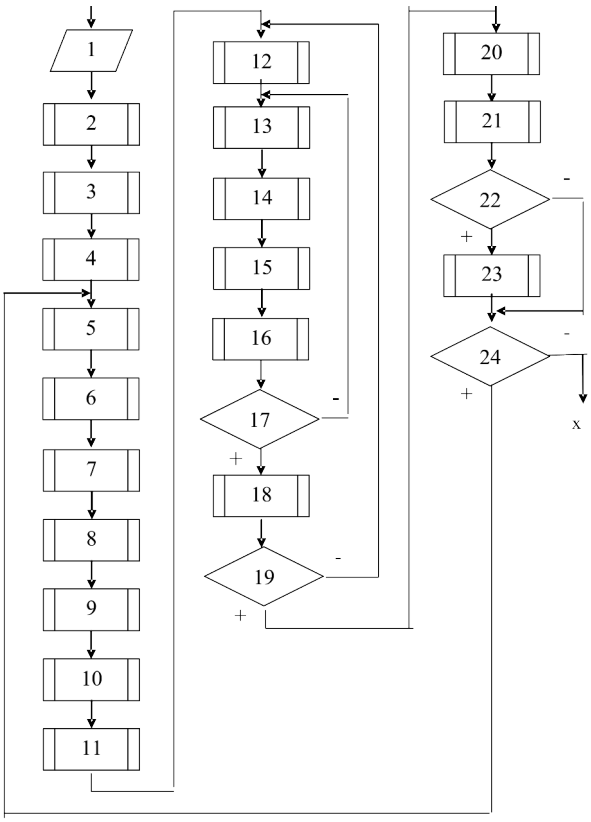

Figure 1: Block diagram of the algorithm of the modified artificial immune network method

Step 11 – Calculate the average cost value $\bar{F}^{source}$, using K threads that are grouped into one one-dimensional block. In this block, the sum of K elements of type $\frac{F(x_k)}{K}$ is calculated based on the reduction.

Step 12 – Set the cell number $k=1$.

Step 13 – Create clones set $\widetilde{P}_k=\{\widetilde{x}_{kl}\}$ for population cell x_k.

Step 14 – Generate a mutant clones set P_k for a population cell x_k, using $L_C\cdot M$ threads that are grouped into L_C one-dimensional blocks. Each thread calculates $x_{klj}=\widetilde{x}_{klj}+\frac{1}{\alpha}e^{-\Phi(\widetilde{x}_{kl})}Cauchy(0,1)$.

Step 15 – Perform the correction of the mutant clones set P_k for the population cell x_k, using $L_C\cdot M$ threads, that are grouped into L_C one-dimensional blocks. Each thread calculates $x_{klj}=\max\{x_j^{\min},x_{klj}\}$, $x_{klj}=\min\{x_j^{\max},x_{klj}\}$.

Step 16 – Determine the best mutated clone by the target function for each population cell x_k $h_k = \arg\min_{x_{kl}} F(x_{kl})$, $l \in \overline{1, L_C}$.

Step 17 – If $k < K$, then $k = k + 1$, go to step 13.

Step 18 – Calculate the average cost value $\bar{F}^{mutate}$, using K threads that are grouped into one one-dimensional block. In this block, the sum of K elements of the form $\frac{F(h_k)}{K}$ is calculated based on the reduction.

Step 19 – If $\bar{F}^{mutate} \geq \bar{F}^{source}$, then go to step 12.

Step 20 – Compute a distances set $\{\rho(h_m, h_l)\}$, using $K \cdot K$ threads that are grouped into K one-dimensional blocks. Each thread computes $\rho(h_m, h_l)$.

Step 21 – Compress set H and replace population cells P with elements of set H.

Step 22 – If $k = K$, then go to step 24.

Step 23 – Initialize the last $K - k$ population cells P, using $(K-k) \cdot M$ threads that are grouped into $K - k$ one-dimensional blocks. Each thread calculates $x_{kj} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) U(0,1)$.

Step 24 – If $n < N - 1$, then $n = n + 1$, go to step 5.

7. Modified hybrid immune algorithm method

The hybrid immune algorithm was proposed by scientists Lucinska and Wierzchon and is a modification of the artificial immune network. Its distinctive feature is the use of two types of mutations.

The modified hybrid immune algorithm method consists of the following steps:

1. Initialization.

1.1. Setting the search area: cell length M, minimum and maximum values of cell components $x_j^{\min}, x_j^{\max}$, $j \in \overline{1, M}$. Setting the maximum number of iterations N, population size K, number of clones L_C, memory size L_M.

1.2. Setting the cost function (target function) $F(x) \to \min_x$, where x – cell (real vector).

11.3. Setting the search parameters: compression threshold ε, maximum cell age $a^{\max}$, memory size excess coefficient α, where $\varepsilon > 0$, $a^{\max}$ – is a natural number, $\alpha > 1$.

1.4. Creating an initial population P.

1.4.1. Cell number $k = 1$, $P = \varnothing$.

1.4.2. Generating a random cell $x_k = (x_{k1}, \ldots, x_{kM})$, $x_{kj} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) U(0,1)$, where $U(0,1)$ – a function that returns a standard uniformly distributed random number.

1.4.3. Setting the age of cell $a_k = 1$.

1.4.4. If $(x_k, a_k) \notin P$, then $P = P \cup \{(x_k, a_k)\}$, $k = k + 1$.

1.4.5. If $k \leq K$, then move to step 1.4.2.

1.5. Creating the initial set of sets of mutated clones $\{P_1, \ldots, P_K\}$.

1.5.1. Cell number $k = 1$.

1.5.2. Mutated clone number $l = 1$, $P_k = \varnothing$.

1.5.3. Creating a randomly mutated clone $x_{kl} = (x_{kl1}, \ldots, x_{klM})$, $x_{klj} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) U(0,1)$.

1.5.4. Generating a random vector of standard deviations $\sigma_{klj}=(\sigma_{kl1},\ldots,\sigma_{klM})$, $\sigma_{klj}=U(0,1)$.

1.5.5. If $x_{kl}\notin P_k \wedge x_{kl}\neq x_k$, then $P_k=P_k\cup\{(x_{kl},\sigma_{kl})\}$, $l=l+1$.

1.5.6. If $l\leq L_C$, then move to step 1.5.3.

1.5.7. If $k<K$, то $k=k+1$, go to step 1.5.2.

1.6 Initializing a set of memory cells $Q=\varnothing$.

2. Iteration number $n=0$.

3. Creating the best mutated clones set H.

3.1. Cell number $k=1$, $H=\varnothing$.

3.2. Creating a set of clones $\widetilde{P}_k=\{(\widetilde{x}_{kl},\widetilde{a}_{kl})\}$ for a population cell (x_k,a_k).

3.3. Modification of a set of mutated clones P_k.

3.3.1. Clone number $l=1$.

3.3.2. Set $\lambda=U(0,1)$.

3.3.3. If $\lambda>0.2$, then calculation of the vector of standard deviations

$$\sigma_{klj}=\begin{cases}2(\widetilde{x}_{lkj}-x_{klj}), & F(x_{klj})<F(\widetilde{x}_{klj})\\ \sigma_{klj}, & F(x_{klj})\geq F(\widetilde{x}_{klj})\end{cases},\ j\in\overline{1,M}, \tag{17}$$

creation of a cell (the work suggests using the Cauchy distribution)

$$x_{klj}=\sigma_{klj}Cauchy(0,1)+\widetilde{x}_{klj},\ j\in\overline{1,M}, \tag{18}$$

where $Cauchy(0,1)$ – a function that returns a standard Cauchy distributed random number.

3.3.4. Cell correction x_{kl}

$$x_{klj}=\max\{x_j^{\min},x_{klj}\},\ x_{klj}=\min\{x_j^{\max},x_{klj}\},\ j\in\overline{1,M}.$$

3.3.5. If $\lambda\leq 0.2$, then $x_{kl}=\widetilde{x}_{kl}$, $j=round(1+(M-1)U(0,1))$, $x_{klj}=x_j^{\min}+(x_j^{\max}-x_j^{\min})U(0,1)$.

3.3.6. If $l<L_C$, then $l=l+1$, go to step 3.3.2.

3.4. Determining the best mutated clone of set P_k by the target function

$$h_k=\arg\min_{x_{kl}}F(x_{kl}). \tag{19}$$

3.5. Calculating $H=H\cup\{h_k\}$.

3.6. If $k<K$, then $k=k+1$, go to step 3.2.

4. Modification of population P.

4.1. Cell number $k=1$.

4.2. If $F(h_k)<F(x_k)$, then $x_k=h_k$, $a_k=1$.

4.3. If $F(h_k)\geq F(x_k)$, then $a_k=a_k+1$.

4.4. If $k<K$, then $k=k+1$, go to step 4.2.

5. Adding cells from population P, that have reached age $a^{\max}$, to the set of memory cells Q.

5.1. Set $k=1$, $i=|Q|+1$.

5.2. If $a_k \geq a^{\max}$, then $q_i = x_k$, $Q = Q \cup \{q_i\}$, $i = i+1$, and in population P pair (x_k, a_k) is initialized, i.e. $x_{kj} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) U(0,1)$, $j \in \overline{1,M}$, $a_k = 1$.

5.3. If $k < K$, then $k = k+1$, go to step 5.2.

6. If $|Q| < \alpha \cdot L_M$, then move to step 8.

7. Set compression Q.

7.1. Set $i = 1$, $m = 1$, $\breve{Q} = \varnothing$

7.2. Formation of the ε-neighborhood of the i th element of the set Q

$$U_{q_i,\varepsilon} = \{ q_j \mid \rho(q_i, q_j) \leq \varepsilon \}, \quad (20)$$

where ρ – the distance between q_i and q_j (e.g. Euclidean distance).

7.3. If $|U_{q_i,\varepsilon}| = \varnothing$ or $\max U_{q_i,\varepsilon} = q_i$, then $\breve{q}_m = q_i$, $m = m+1$, $\breve{Q} = \breve{Q} \cup \{\breve{q}_m\}$.

7.4. If $i < L_M$, then $i = i+1$, go to step 7.2.

7.5. Set $Q = \breve{Q}$.

7.6. Order the set Q by the target function, i.e. $F(q_i) < F(q_{i+1})$.

7.7. If $|Q| > L_M$, then remove from the ordered set Q the last worst $|Q| - L_M$ cells by the target function.

8. Determining the best memory cell of the set Q by the target function

$$x^{\text{¿}} = \arg\min_{q_i} F(q_i). \quad (21)$$

9. If $n < N-1$, then $n = n+1$, go to step 3.

The result is $x^{\text{¿}}$.

8. Algorithm of the modified hybrid immune algorithm method

The algorithm of the modified hybrid immune algorithm method, designed for implementation on GPU using CUDA technology, is shown in Figure 2.

This block diagram functions as follows.

Step 1 – Set the maximum number of iterations N, population size K, number of clones L_C, memory size L_M, compression threshold ε, maximum cell age $a^{\max}$, memory oversize factor α, where $\varepsilon > 0$, $a^{\max}$ is a natural number, $\alpha > 1$.

Step 2 – Create an initial population $P = \{(x_k, a_k)\}$, using $K \cdot M$ threads that are grouped into K one-dimensional blocks. Each thread of each k th block calculates $x_{kj} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) U(0,1)$, $a_k = 1$.

Step 3 – Set the cell number $k = 1$.

Step 4 – Generate a set of mutated clones $P_k = \{(x_{kl}, \sigma_{kl})\}$, using $L_C \cdot M$ threads that are grouped into L_C one-dimensional blocks. Each thread calculates $x_{klj} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) U(0,1)$, $\sigma_{klj} = U(0,1)$.

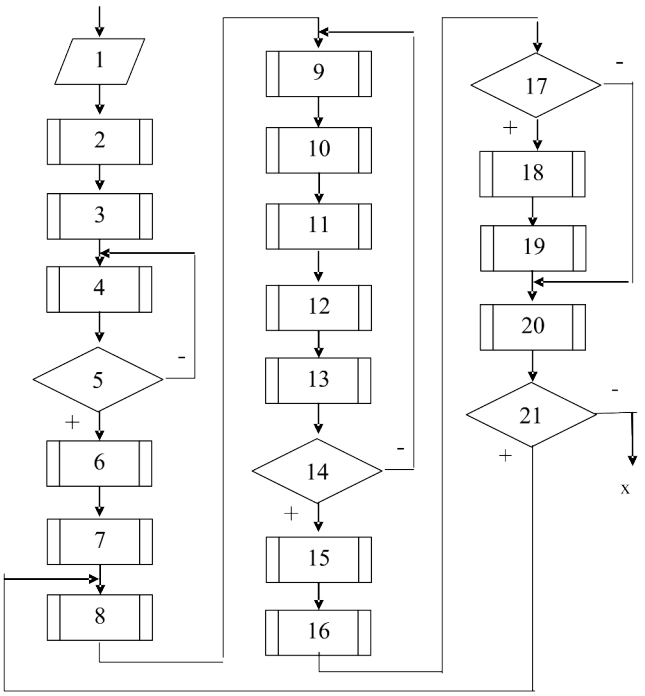

Figure 2: Block diagram of the algorithm of the modified hybrid immune algorithm method

Step 5 – If $k<K$, then $k=k+1$, go to step 4.

Step 6 – Initialize a set of memory cells $Q=\varnothing$.

Step 7 – Set the iteration number $n=0$.

Step 8 – Set the cell number $k=1$.

Step 9 – Create clone set $\widetilde{P}_k=\{\widetilde{x}_{kl}\}$ for population cell (x_k,a_k).

Step 10 – Calculate $\lambda_l=U(0,1)$, $l\in\overline{1,L_C}$.

Step 11 – Modify the set of mutated clones P_k for population cell (x_k,a_k), using $L_C\cdot M$ strands that are grouped into L_C one-dimensional blocks. Each strand of each l th block calculates:

If $\lambda_l>0.2$, then $\sigma_{klj}=\begin{cases}2(\widetilde{x}_{lkj}-x_{klj}), & F(x_{klj})<F(\widetilde{x}_{klj})\\ \sigma_{klj}, & F(x_{klj})\geq F(\widetilde{x}_{klj})\end{cases}$, $x_{klj}=\sigma_{klj}Cauchy(0,1)+\widetilde{x}_{klj}$,

If $\lambda_l\leq 0.2$, then $x_{kl}=\widetilde{x}_{kl}$, $j=round(1+(M-1)U(0,1))$, $x_{klj}=x_j^{\min}+(x_j^{\max}-x_j^{\min})U(0,1)$.

Step 12 – Perform the correction of the modified set of mutated clones P_k for the population cell (x_k,a_k) using $L_C\cdot M$ strands, which are grouped into L_C one-dimensional blocks. Each thread of each l th block calculates:

If $\lambda_l>0.2$, then $x_{klj}=\max\{x_j^{\min},x_{klj}\}$, $x_{klj}=\min\{x_j^{\max},x_{klj}\}$.

Step 13 – Determine the best mutated clone by the target function for each population cell (x_k,a_k)

$$h_k=\arg\min_{x_{kl}}F(x_{kl}),\quad l\in\overline{1,L_C}.$$

Step 14 – If $k<K$, then $k=k+1$, go to step 9.

Step 15 – Modify population P using K threads that are grouped into one one-dimensional block. Each thread calculates:

If $F(h_k)<F(x_k)$, then $x_k=h_k$, $a_k=1$, If $F(h_k)\geq F(x_k)$, then $a_k=a_k+1$.

Step 16 – Add population P cells that have reached the age $a^{\max}$ to memory cell set Q.

Step 17 – If $|Q|<\alpha\cdot L_M$, then move to step 20.

Step 18 – Compute distance set $\{\rho(q_i,q_l)\}$ using $Q\cdot Q$ threads that are grouped into Q one-dimensional blocks. Each thread calculates $\rho(q_i,q_l)$.

Step 19 – Compress set Q.

Step 20 – Determine the best memory cell of the set Q by the target function $x^{\text{¿}}=\arg\min_{q_i} F(q_i)$, $k\in\overline{1,K}$.

Step 21 – If $n<N-1$, then $n=n+1$, go to step 8.

9. Experiments and results

The numerical study of the proposed metaheuristic methods was carried out using the Python package in the Google Colab environment. Numerical experiments were carried out using the CUDA parallel information processing technology on a GeForce 920M video card with 1025 threads in a one-dimensional block.

In this work, the following parameters were used for the modified artificial immune network method: the maximum number of iterations $N=100$, population size $K=20$, number of clones $L_C=10$, parameter for generating a new solution $\delta=0.1$, and compression threshold $\varepsilon=0.1$.

In this work, the following parameters were used for the modified hybrid immune algorithm method: maximum number of iterations $N=100$, population size $K=20$, number of clones $L_C=10$, memory size $L_M=10$, compression threshold $\varepsilon=0.1$, maximum cell age $a^{\max}=10$, and memory oversize factor $\alpha=1.5$.

In the work, a one-dimensional signal was generated, to which additive Gaussian noise with zero mathematical expectation and dispersion of 135 was added.

In the work, the following root mean square errors were calculated based on the formulas:

$$RMS_{yz}=\sqrt{\frac{1}{200}\sum_{n=1}^{200}\left(y(n)-z(n)\right)^2}, \quad (22)$$

$$RMS_{xz}=\sqrt{\frac{1}{200}\sum_{n=1}^{200}\left(x(n)-z(n)\right)^2}, \quad (23)$$

where $y(n)$ – noise-free countdown,

$z(n)$ – observation (noise-free countdown),

$z(n)$ – observation evaluation.

Table 1 presents the root mean square errors for immune metaheuristic methods. For all three methods RMS_{yz} =9.25.

Table 1

Root mean square errors

Immune metaheuristic methods	RMS_{xz}
Clonal selection method	4.9
Modified artificial immune network method	4.5
Modified hybrid immune algorithm method	3.8

For example, for the modified hybrid immune algorithm method, the following parameter values were obtained:

$$A=\begin{bmatrix}1.6 & -0.8\\ 1 & -0.3\end{bmatrix},\ \sigma_Q^2=0.35,\ \sigma_R^2=4.16,\ \sigma_H^2=0.38.$$

Figure 3 shows the original signal without noise and the signal with noise (observation).
Figure 4 shows the original signal without noise and the observation estimate signal.

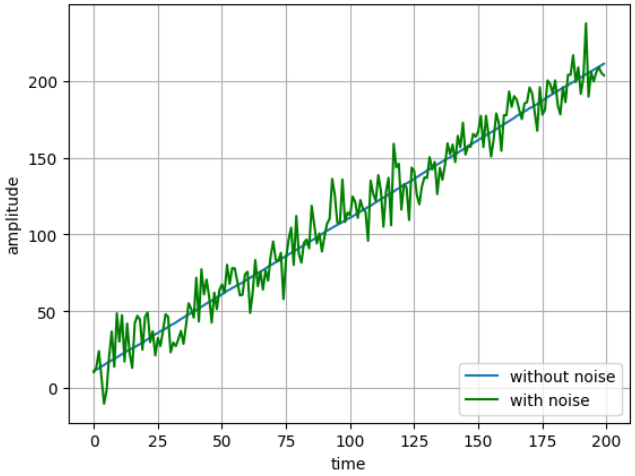

Figure 3: Original signal without noise and signal with noise (observation)

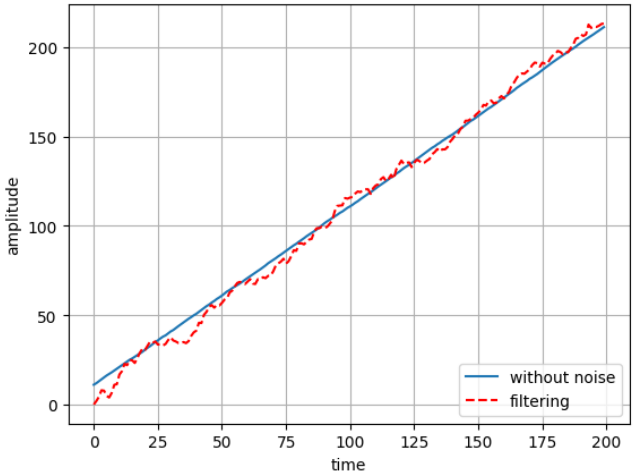

Figure 4: The original signal without noise and the observation estimate signal

10.Discussion

The advantage of using the proposed immune metaheuristic methods:

1. Automation of determination of Kalman filtering parameters A, σ_Q^2, σ_H^2, σ_R^2.
2. Immune metaheuristic methods, due to their stochastic nature, reduce the probability of convergence to a local extremum.
3. For immune metaheuristic methods, it is proposed to replace the Gaussian distribution with the Cauchy distribution, which is long-tailed, i.e. to reduce the probability of convergence to a local extremum.
4. According to Table 1, the modified hybrid immune algorithm method gives the best results in terms of the root mean square error.
5. According to Figure 3 and Figure 4, the original signal without noise and the filtered signal differ insignificantly, while there is a significant difference between the original signal without noise and the signal with noise.

Conclusions

1. A modified Kalman filtering method was developed that provides automation of parameter value determination and increases the speed and accuracy of Kalman filtering by using fewer parameters and identifying them based on immune metaheuristic methods.
2. A modified artificial immune network method was created that reduces the probability of convergence to a local extremum by using the Cauchy distribution and makes the proposed method more accurate than the existing one.
3. A modified hybrid immune algorithm method was developed, which, by using the Cauchy distribution, reduces the probability of convergence to a local extremum and makes the proposed method more accurate than the existing one.
4. Algorithms of immune metaheuristic methods for identifying Kalman filter parameters have been developed, which are intended for software implementation on GPU using CUDA technology, which increases the accuracy of Kalman filtering. The numerical studies conducted have confirmed the operability of the developed software and allow us to recommend it for practical use.
5. Further research prospects include the use of the proposed immune metaheuristic methods for various general-purpose and special-purpose intelligent systems, for example, for training neural networks.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Guo, W. Li, B. Zhang, Ya. Liu, Differential privacy Kalman filtering for graphical dynamic systems: Performance improvement and privacy calibration, Digital Signal Processing, vol. 152, 104589 (2024). doi: https://doi.org/10.1016/j.dsp.2024.104589.

[2] Sh. Zhong, B. Peng, J. He, Zh. Feng, M. Li, G. Wang, Kalman filtering based on dynamic perception of measurement noise, Mechanical Systems and Signal Processing, vol. 213, 111343 (2024). doi: https://doi.org/10.1016/j.ymssp.2024.111343.

[3] A. Kh. Roonizi, Kalman filter/smoother-based design and implementation of digital IIR filters, Signal Processing, vol. 208 (2023) 1-10. doi: https://doi.org/10.1016/j.sigpro.2023.108958.

[4] M. Cheng, F. Fang, I.M. Navon, Ch. Pain, Ensemble Kalman filter for GAN-ConvLSTM based long lead-time forecasting, Journal of Computational Science, vol. 69 (2023) 1-16. doi: https://doi.org/10.1016/j.jocs.2023.102024.

[5] X.-S. Yang, Nature-inspired Algorithms and Applied Optimization, Charm: Springer, 2018. doi: 10.1007/978-3-642-29694-9.

[6] A. Nakib, El-G. Talbi, Metaheuristics for Medicine and Biology, Berlin: Springer-Verlag, 2017. doi: 10.1007/978-3-662-54428-0.

[7] S. Subbotin, A. Oliinyk, V. Levashenko, E. Zaitseva, Diagnostic rule mining based on artificial immune system for a case of uneven distribution of classes in sample, Communications, volume 3 (2016) 3-11.

[8] X.-S. Yang, Optimization Techniques and Applications with Examples, Hoboken, New Jersey: Wiley & Sons, 2018. doi: 10.1002/9781119490616.

[9] T. Neskorodieva, E. Fedorov, M. Chychuzhko, V. Chychuzhko, Metaheuristic Method for Searching Quasi-Optimal Route Based On the Ant Algorithm and Annealing Simulation, Radioelectronic and Computer Systems 1(2022) 92–102. doi: 10.32620/reks.2022.1.07.

[10] A. Vasuki, Nature Inspired Optimization Algorithms, 1st ed., Boca Raton, FL: Chapman and Hall/CRC, New York, 2020. doi: 10.1201/9780429289071.

[11] H. Wang, M. Huang, J. Wang, An effective metaheuristic algorithm for flowshop scheduling with deteriorating jobs, Journal of Intelligent Manufacturing, vol. 30 (2019) 2733–2742. doi: 10.1007/s10845-018-1425-8.

[12] C. Blum, G. R. Raidl, Hybrid Metaheuristics. Powerful Tools for Optimization, Charm: Springer, 2016. doi: 10.1007/978-3-319-30883-8.

[13] R. Martí, P. M. Pardalos, M. G. C. Resende, Handbook of Heuristics, Charm: Springer, 2018. doi: 10.1007/978-3-319-07124-4.
[14] B. Chopard, M. Tomassini, An Introduction to Metaheuristics for Optimization, Springer, New York, 2018. doi: 10.1007/978-3-319-93073-2.
[15] J. Radosavljević, Metaheuristic Optimization in Power Engineering, New York: Institution of Engineering and Technology, 2018. doi: 10.1049/PBPO131E.
[16] N. S. Jaddi, J. Alvankarian, S. Abdullah, Kidney-inspired algorithm for optimization problems, Communications in Nonlinear Science and Numerical Simulation 42(2017) 358–369. doi: 10.1016/j.cnsns.2016.06.006.
[17] Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: A MATLAB platform for evolutionary multi-objective optimization, IEEE Computational Intelligence Magazine 12(2017) 73–87. doi: 10.1109/MCI.2017.2742868.
[18] E. Fedorov, M. Chychuzhko, V. Chychuzhko, Approaches to the creation of a software agent based on meta-heuristic and artificial neural networks, Radioelectronic and Computer Systems, vol. 1 (2019) 58-65. doi: https://doi.org/10.32620/reks.2019.1.06.
[19] E. Fedorov, O. Nechyporenko, Methods for Solving the Traveling Salesman Problem Based on Reinforcement Learning and Metaheuristics, in: CEUR Workshop Proceedings, 2022, vol. 3309, pp. 94–103.
[20] A. Shukla, R. Tiwari, Discrete Problems in Nature Inspired Algorithms, 1st ed., Boca Raton, FL: Chapman and Hall/CRC, New York, 2019. doi: 10.1201/9781351260886.
[21] A. Slowik, Swarm Intelligence Algorithms, A Tutorial, 1st ed., Boca Raton, FL: Chapman and Hall/CRC, New York, 2021. doi: 10.1201/9780429422614.
[22] O. Bozorg Haddad, M. Solgi, H. Loaiciga, Meta-heuristic and Evolutionary Algorithms for Engineering Optimization, Hoboken, New Jersey: Wiley & Sons, 2017. doi: 10.1002/9781119387053.
[23] K.-L. Du, M. N. S. Swamy, Search and Optimization by Metaheuristics. Techniques and Algorithms Inspired by Nature, Charm: Springer, 2016. doi: 10.1007/978-3-319-41192-7.
[24] A. Kaveh, T. Bakhshpoori, Metaheuristics Outlines, MATLAB Codes and Examples, Cham: Springer, 2019. doi: 10.1007/978-3-030-04067-3.
[25] A. Nayyar, D.-N. Le, N. G. Nguyen, Advances in Swarm Intelligence for Optimizing Problems in Computer Science, 1st ed., Boca Raton, FL: Chapman and Hall/CRC, New York, 2018. doi: 10.1201/9780429445927.
[26] H. Emami, Anti-coronavirus optimization algorithm, Soft Computing, vol. 26 (2022) 4991-5023. doi:10.1007/s00500-022-06903-5.
[27] A. M. Khalid, K. M. Hosny, S. Mirjalili, COVIDOA: a novel evolutionary optimization algorithm based on coronavirus disease replication lifecycle, Neural Computing and Applications, vol. 34 (2022) 22465-22492. doi:10.1007/s00521-022-07639-x.
[28] A. Al-Betar, Z. A. A. Alyasseri, M. A. Awadallah, I. A. Doush, Coronavirus herd immunity optimizer (CHIO), Neural Computing and Applications, vol.33 (2021) 5011-5042. doi:10.1007/s00521-020-05296-6.