The Fast and Accurate Method of Tuning of SISO Controllers Using Seven Performance Indexes

[1]Stopakevych A.O., [2]Stopakevych O.A.
[1] State University of Intelligent Technologies and Telecommunications
[2] National Odesa Polytechnic University
Odesa, Ukraine

Annotation. The goal of the work is the development of an algorithm and corresponding software, which allows the execution of multi-criteria optimization of a control system with PI controller and FOPDT plant. We use the following seven performance indicators to find the optimal tunings: gain margin, phase margin, cutoff frequency, critical frequency, relative delay margin, relative overshoot, maximum control action magnitude. Optimization criteria can include constraints on any number of indicators. The goal of the work has been achieved by solving the following tasks. The first task is to create formulas and procedures for accurate calculation of performance indicators of control systems. The second task is to develop a procedure for the fastest possible simulation of a control system with the orientation on using a model with internal delays structure and a special solver for it. The third task is to develop an algorithm for fast calculation of performance indicators in the range of all possible rational tuning of the controller for a given FOPDT plant model. The fourth task is to develop a software application with a graphical interface in MATLAB language, which allows convenient optimization for an arbitrary rational FOPDT model. The most significant result was that the pointed-out performance indicators can be accurately calculated for all possible rational tunings of a PI controller with an arbitrary FOPDT model in a second. The significance of the results was that it allows to reduce an optimization procedure to a table search and to achieve any multi-term performance criteria. The effectiveness of the procedure has been demonstrated on a set of different FOPDT models. It is shown that there is no dependence in accuracy of calculation from model or controller coefficients.
Keywords: control system, PI controller, FOPDT, delay, dead time, optimization, performance indicators, simulation speed, formulas, accuracy.

DOI: https://doi.org/10.52254/1857-0070.2025.1-65.12
UDC: 681.51

O metodă rapidă și precisă pentru reglarea controlerelor SISO utilizând șapte criterii de performanță

Stopakevâci A.O.[1], Stopakevâci O.A.[2]
[1] Universitatea de Stat de Tehnologii și Comunicații Inteligente, Odesa, Ucraina
[2] Universitatea Națională Politehnică Odesa, Odesa, Ucraina

Rezumat. Scopul lucrării este dezvoltarea unui algoritm și a software-ului corespunzător, care permite executarea optimizării multicriteriale a unui sistem de control cu controler PI și instalație FOPDT. Utilizăm următorii șapte indicatori de performanță pentru a găsi reglajele optime: marja de câștig, marja de fază, frecvența de tăiere, frecvența critică, marja de întârziere relativă, depășirea relativă, mărimea maximă a acțiunii de control. Criteriile de optimizare pot include constrângeri asupra oricărui număr de indicatori. Scopul lucrării este atins prin rezolvarea următoarelor sarcini. Prima sarcină este de a crea formule și proceduri pentru calculul precis al indicatorilor de performanță ai sistemelor de control. A doua sarcină este de a dezvolta o procedură pentru cea mai rapidă simulare posibilă a unui sistem de control, cu orientarea spre utilizarea unui model cu structură de întârzieri interne și a unui solver special pentru acesta. A treia sarcină este de a dezvolta un algoritm pentru calcularea rapidă a indicatorilor de performanță în intervalul tuturor reglajelor raționale posibile ale regulatorului pentru un anumit model de instalație FOPDT. A patra sarcină este de a dezvolta o aplicație software cu o interfață grafică în limbajul MATLAB, care permite optimizarea convenabilă pentru un model FOPDT rațional arbitrar. Cel mai semnificativ rezultat a fost că indicatorii de performanță evidențiați pot fi calculați cu exactitate pentru toate reglajele raționale posibile ale unui controler PI cu un model FOPDT arbitrar într-o secundă. Semnificația rezultatelor a fost că aceasta permite reducerea unei proceduri de optimizare la o căutare în tabel și realizarea oricărui criteriu de performanță pe mai multe termene. Eficacitatea procedurii a fost demonstrată pe un set de modele FOPDT diferite. S-a demonstrat că nu există nicio dependență în precizia calculului de coeficienții modelului sau ai regulatorului.
Cuvinte-cheie: sistem de control, controler PI, FOPDT, întârziere, timp mort, optimizare, indicatori de performanță, viteză de simulare, formule, precizie.

Быстрый и точный метод настройки SISO регуляторов с использованием семи критериев качества

Стопакевич А. Ал.[1], Стопакевич Ал. А.[2]
[1] Государственный университет интеллектуальных технологий и связи
[2] Национальный университет «Одесская политехника», Одесса, Украина

Аннотация. Целью работы является разработка алгоритма и соответствующего программного обеспечения, позволяющего проводить многокритериальную оптимизацию системы управления с ПИ-регулятором и объектом, заданным FOPDT моделью. Для поиска оптимальных настроек используются следующие семь показателей качества: запас по коэффициенту усиления, запас по фазе, частота среза, критическая частота, относительный запас по запаздыванию, относительное перерегулирование, максимальная величина управляющего воздействия. Критерии оптимизации могут включать ограничения на границы изменения любого количества показателей. Цель работы достигается путем решения следующих задач. Первая задача - создание формул и процедур для точного расчета показателей качества систем управления. Вторая задача - разработка процедуры максимально быстрого моделирования системы управления с ориентацией на использование модели со структурой внутренних задержек и специального решателя для нее. Третья задача - разработка алгоритма быстрого расчета показателей качества в диапазоне всех возможных рациональных настроек регулятора для заданной FOPDT модели. Четвертая задача - разработка программного приложения с графическим интерфейсом на языке MATLAB, позволяющего удобно проводить оптимизацию для произвольной рациональной модели FOPDT. Наиболее важным результатом стало то, что указанные показатели качества могут быть точно рассчитаны для всех возможных рациональных настроек ПИ-регулятора с произвольной FOPDT моделью. Значимость полученных результатов заключается в том, что они позволяют свести процедуру оптимизации к табличному поиску и достичь любых комплексных критериев оптимизации. Эффективность процедуры была продемонстрирована на множестве различных моделей FOPDT. Показано, что точность расчета не зависит от коэффициентов модели или регулятора.

***Ключевые слова*:** запаздывание, оптимизация, показатели качества, быстрое моделирование, объект первого порядка с запаздыванием.

INTRODUCTION

Usually, the tuning of PID family controllers in the design of control systems with FOPDT plant is found using simplified formulas [1], which more or less guarantee achieving one of the performance criteria of the control system. Software tools for tuning calculation are also developed. For example, MATLAB program *pidtune* [2], by default, accurately achieves a phase margin value and approximately an overshoot value of about 5%.

The possibility of replacing PID family controllers with controllers that take into account the saturation of the control action in the control law is discussed in [3]. The authors conclude that the problem with using more theoretically advanced controllers is that their effectiveness is only achieved in cases where there is an accurate linear model. In the other case, a PID family controller will usually show comparable or better control performance than more theoretically advanced controllers.

PID family controllers are limited by their fixed structure and the presence of delay in the plant model. This means that even within linear systems and without saturation of the control action it is analytically impossible to achieve an arbitrary performance indicator of the control system. Universal formulas for estimation of parameters of a control system without modeling are mostly inaccurate [4], so it is possible to achieve a variety of performance indicators only using multi-criteria optimization. The first difficulty of this approach is the presence of unstable zones, due to which the algorithm can find some local minimum. The second difficulty is that the known performance indicators contradict each other [5] and limit the possible range of changing each other. This problem is attempted to be solved by applying heuristic search methods such as genetic algorithms (GA)[6].

The application of heuristic optimization algorithms and the overwhelming number of other classical algorithms to the problem of optimizing PID family controller tunings requires simulation of control system transients. Many works to solve this problem are investigation integral criteria optimization, while multi-criteria optimization is rarely considered. One of the exceptions is the paper [7], in which the applied optimization problem of a PID controller with fractional derivatives is solved with the help of a PSO algorithm. It used the complex 8 term criterion, including direct performance indicators, integral IAE index, and frequency margins indicators.

The specialized multi-parameter optimization algorithm described in [8]. For a PI controller, it allows two indicators to be achieved simultaneously within a period of up to 1000 iterations.

The analysis of genetic algorithm capabilities for the problem of PID controller tuning optimization based on direct and frequency performance indicators, carried out in [9]. Both direct indicators and integral indices were used as optimization criteria. To simplify the computational complexity of the problem, the transient modeling was performed using the delay approximation and the very fast plant was considered for control. The shortcomings of the genetic algorithm were evaluated. In addition to the fact that it may not converge even within the tuning bounds near found by other tuning methods, significant difficulties arise with optimization by frequency indicators in the time domain. The possibility of achieving other performance indicators also has been analyzed.

Of course, it is a desirable option to obtain a controller that satisfies the given constraints by tunings that fit the performance criteria requirements without applying optimization. This problem has been considered in [10-12]. The papers are based on the analytical approach to limit (or exactly achieve) the performance criteria of the control system with 1-2 indicators. However, within the approach limitations, it is difficult to achieve simple dependencies without imposing restrictions on the controller tunings.

In this study, we consider the problem of designing a standard form PI controller which is usually used for FOPDT plant models [1]. FOPDT models with both minimal delay and dominant delay are considered.

FORMULATION OF THE RESEARCH GOAL, A NEW APPROACH TO THE OPTIMIZATION PROBLEM AND RESEARCH TASKS

The goal of the research is the development of an algorithm and corresponding software in MATLAB language, which allows the execution of multi-criteria optimization of a control system with PI controller and FOPDT plant and has the following advantages: speed, guaranteed finding of a solution or certainty that there is no solution, possibility of use for optimization of such criteria which are very difficult to optimize. We use the following seven performance indicators to find the optimal tunings: gain margin (GM), phase margin (PM), cutoff frequency (ω_c), critical frequency (ω_g), relative delay margin (DM) τ_m, relative overshoot M_p, maximum control action magnitude u_{max} when the reference is changed by one.

The basic idea of optimization is to create the space and then to reduce obtaining controller tunings that satisfy the desired criteria to searching in a table for rows with values close to the desired criteria terms values using a distance metric. To form such a space, it is necessary to obtain the entire set of rational PI controller tunings with a small step and to determine the specified seven performance indicators of control systems. The optimal size of tunings set for the problem is 50-100 thousand tunings. It is possible to solve such a problem by simulation of transient processes, but it will be very long. Therefore, another approach is needed here.

The main idea of the proposed approach is to calculate the performance indicators over the whole range of rational controller tuning with the minimum possible use of simulation of transients. If there are calculation formulas, it is no problem to calculate them 100 thousand times for a modern computer. Formulas for obtaining performance indicators for control systems can be derived analytically, but they will not be accurate for all cases. Since accuracy is required for the task under consideration, additional formulas and ways of applying simple optimization methods for accurate determination of performance indicators are proposed. The last two performance indicators are generally not reduced to formulas in which no equations should be solved. For example, extremely complex formulas with inner equations for calculating the last two performance indicators are derived analytically by using the inverse Laplace transform with delay approximation in MATLAB symbolic calculations. On the other hand, use of simplifications seriously decrease accuracy. Therefore, in order to determine the last two indicators, it is necessary to carry out simulations at characteristic points and to relate the obtained data to the calculated indicators in order to obtain their estimation over the whole range.

Thus, the research tasks are as follows

1) To create formulas and procedures for accurate calculation of performance indicators of control systems.

2) To develop a procedure for the fastest possible simulation of the transient processes of the control system with the PI controller and the FOPDT model in MATLAB with the orientation on the application of the model with the structure of internal delays and a special solver for dynamical systems with internal delays.

3) To develop an algorithm for fast calculation of control system performance indicators in the

range of all possible rational tuning of the controller for a given FOPDT plant model.

4) To develop a software application with a graphical interface in MATLAB language, which allows convenient optimization for an arbitrary rational FOPDT model.

DEVELOPMENT OF FORMULAS AND PROCEDURES FOR ACCURATE ANALYTICAL AND SEMI-EMPIRICAL CALCULATION OF PERFORMANCE INDICATORS

The plant model has the following transfer function (TF)

$$P(s)=\frac{k}{T\cdot s+1}\cdot e^{-\tau\cdot s} \qquad (1)$$

The PI controller model has the following TF

$$C(s)=k_p\cdot\left(1+\frac{1}{Ti\cdot s}\right) \qquad (2)$$

The open loop model has the following TF

$$L(s)=C(s)\cdot P(s)=k_p\left(1+\frac{1}{T_i\cdot s}\right)\cdot\frac{k\cdot e^{-s\tau}}{T\cdot s+1}$$

By definition, the stability margin frequency w_g is the frequency at which the amplitude is 0 dB (and, by the same token, the TF modulus is 1).

The TF modulus at frequency ω_g is calculated for open loop model as follows:

$$|L(j\omega_g)|=\left|k_p\cdot\left(1+\frac{1}{j\cdot\omega_g\cdot T_i}\right)\cdot\frac{k\cdot e^{-j\cdot\omega_g\cdot\tau}}{T\cdot j\cdot\omega_g+1}\right|$$

PI controller model module

$$\left|k_p\left(1+\frac{1}{j\omega_g T_i}\right)\right|=k_p\cdot\sqrt{1+\frac{1}{(\omega_g T_i)^2}}$$

Plant model module

$$\left|\frac{k\cdot e^{-j\cdot\omega_g\cdot\tau}}{T\cdot j\cdot\omega_g+1}\right|=\left|\frac{k\cdot e^{-j\cdot\omega_g\cdot\tau}}{\sqrt{(T\cdot\omega_g)^2+1}}\right|=\frac{k}{\sqrt{(T\cdot\omega_g)^2+1}}$$

We rejected the complex exponent due to the fact that such an exponent, regardless of the sign, describes the rotation on the complex plane without changing the length (modulus) of the vector.

Open loop model module

$$|L(j\cdot\omega_g)|=k_p\cdot\left|1+\frac{1}{j\cdot\omega_g\cdot T_i}\right|\cdot\frac{k}{\sqrt{(T\cdot\omega_g)^2+1}}$$

Next, we combine

$$\left|L(j\omega_g)\right|=k_p\cdot\sqrt{1+\frac{1}{(\omega_g\cdot T_i)^2}}\cdot\frac{k}{\sqrt{(T\cdot\omega_g)^2+1}}=1$$

and finally, we get

$$\omega_g=\sqrt{\frac{(k_p^2\cdot k^2\cdot T_i-T_i)+\sqrt{(T_i-k_p^2\cdot k^2\cdot T_i)^2+4\cdot k_p^2\cdot k^2\cdot T^2}}{2\cdot T_i\cdot T^2}} \qquad (3)$$

By finding ω_g we can calculate the PM.

For the phase of the PI controller and the plant, an important parameter is the frequency of the stability margin ω_g .Phase shift of PI controller is

$$\phi_{PI}=-\text{atan}\left(1/(\omega_g\cdot T_i)\right),$$

of aperiodic link is

$$\phi_{FO}=\text{-atan}\left(\omega_g\cdot T\right),$$

of delay link is

$$\phi_D=-\omega_g\cdot\tau\ .$$

Taking the reference point we can get the phase margin (in degrees)

$$PM=\left(\begin{array}{l}\left(\text{-atan}\left(\frac{k_p/T_i}{k_p\cdot\omega_g}\right)\right)-\\ \text{atan}(T\cdot\omega_g)-\tau\cdot\omega_g+\pi\end{array}\right)\cdot\frac{180}{\pi} \qquad (4)$$

If the value exceeds 180°, the phase should be returned.

By definition of the cutoff frequency, the modulus of the TF *L(s)* at the frequency ω_c must equal unity

$$|L(j\cdot\omega_c)|=\left|k_p\cdot\left(1+\frac{1}{j\cdot\omega_c\cdot T_i}\right)\cdot\frac{k\cdot e^{-j\cdot\omega_c\cdot\tau}}{T\cdot j\cdot\omega_c+1}\right|=1$$

$$|L(j\cdot\omega_c)|=\left|k_p\cdot\alpha\cdot\beta\right|=\left|k_p\right|\cdot|\alpha|\cdot|\beta|=1$$

where

$$|\alpha|=\sqrt{1+\frac{1}{(\omega_c T_i)^2}},|\beta|=\frac{k}{\sqrt{(T\omega_c)^2+1}}\ .$$

Next, let's express ω_c from the equation $|L(j\cdot\omega_c)|=1$

$$k_p\cdot k\cdot\sqrt{1+\frac{1}{(\omega_c\cdot T_i)^2}}=\sqrt{(T\cdot\omega_c)^2+1}\rightarrow$$

$$(k_p\cdot k)^2\cdot\left(1+\frac{1}{(\omega_c\cdot T_i)^2}\right)=(T\cdot\omega_c)^2+1\rightarrow$$

$$(k_p\cdot k)^2\cdot(\omega_c^2\cdot T_i^2+1)=\omega_c^2\cdot T_i^2\cdot\left((T\cdot\omega_c)^2+1\right)\rightarrow$$

$$T^2\cdot T_i^2\cdot\omega_c^4-\left((k_p\cdot k)^2\cdot T_i^2-T_i^2\right)\cdot\omega_c^2+(k_p\cdot k)^2=0$$

It's a biquadratic equation

$$a\cdot\omega_c^4+b\cdot\omega_c^2+c=0,\ \ a=T^2\cdot T_i^2,$$

$$b=-\left[(k_p\cdot k)^2\cdot T_i^2-T_i^2\right],\ \ c=(k_p\cdot k)^2\text{'}$$

whose solution reduces to

$$\omega_{ce} = \frac{\pi + \sqrt{\pi^2 - 4 \cdot \pi \cdot \tau \left(\frac{1}{T_i} - \frac{1}{T} \right)}}{4 \cdot \tau} \qquad (5)$$

This formula does not prove to be accurate because the delay has some effect on the dynamics. Since the following performance indicators depend on the cutoff frequency value, high accuracy is desirable. In order to calculate the cutoff frequency accurately, the use of two additional procedures is suggested.

The first procedure is the correction, based on the value of ω_g, which is calculated exactly according to the formula (5)

$$\omega_{ce2} = 0.97225 \cdot \omega_{ce} + 0.012757 \cdot \omega_g - 0.00029127 \cdot \omega_g \cdot \omega_{ce} - 0.0016191 + 0.00018348 \cdot {\omega_g}^2 + 0.00019431 \cdot {\omega_{ce}}^2 \qquad (6)$$

The second procedure is a simple optimization, which finds the exact value of ω_c by taking w_{ce2} as the starting point and finding the value of w_c, which corresponds to the change of the sign of the angle w_{cea} by the formula

$$\omega_{cea} = \angle \left(k \cdot k_p \cdot \frac{1 + \frac{1}{T_i \cdot j \cdot \omega_c} e^{-\tau \cdot j \cdot \omega_c}}{T \cdot j \cdot \omega_c + 1} \right) \cdot \frac{180}{\pi} \qquad (7)$$

It takes no more than 10 calculations with bisection method to find a value to the third digit after the point.

We can find the gain margin (abs) by referring to the exact value of the cutoff frequency ω_c. Let's proceed directly from the definition of GM

$$GM = \frac{1}{\left| k_p \left(1 + \frac{1}{T_i \cdot (i \cdot \omega_c)} \right) \frac{k}{T \cdot (i \cdot \omega_c) + 1} e^{-\tau \cdot (i \cdot \omega_c)} \right|} \qquad (8)$$

The delay margin is directly related to PM, ω_g, we define it as

$$\tau_m = \tau_d / \tau = \left(\frac{PM}{\omega_g} \cdot \frac{\pi}{180} \right) / \tau \qquad (9)$$

It remains to define two parameters: the maximum control action u_{max} and the value of relative overshoot M_p. These parameters are related to the damping ζ and, in principle, to all the above performance indicators, and there is a significant and very non-linear relationship with τ / T. These indicators are easily determined when an aperiodic process is considered (analytically, the problem of calculating controller tunings for such a process is given in [16]) or at least the tuning method is known (such a problem is considered in [10,15]). But in the general case, complex polynomial equations are required, which do not provide high accuracy. Therefore, another approach has been adopted: to simulate control systems with a certain set of controllers, to obtain the value for each case of the maximum value of the control action u_{max} and the maximum value of the controlled variable y_{max}. For these performance indicators with the help of some previously calculated ones, we can obtain exact approximation equations related to the calculated indicators for all points. These will be local equations that describe only one case and are not complex. It is recommended to use the spline interpolation algorithm "thinplateinterp", implement by the *fit* function.

Thus, the data for calculating u_{max} and M_p is derived from the results of calculating the performance indicators of the set.

From the values of $PM, \tau_m, M_p = y_{max} - 1$, a spline equation is obtained to calculate $M_p = f(PM, \tau_m)$.

From the values of PM, ω_g, u_{max}, a spline equation is obtained to calculate $u_{max} = f(\omega_g, u_{max})$.

In order to make the simulation of control systems with the desired controllers as fast as possible, we will perform the simulation in a special way. We will focus on the peculiarities of the implementation of the modeling procedure in MATLAB.

DEVELOPMENT OF THE PROCEDURE FOR THE FASTEST POSSIBLE TRANSIENT SIMULATION WITH PI CONTROLLER AND FOPDT PLANT IN MATLAB

In scientific literature, the problem of transients' simulation for application in optimization problems using mathematical programming languages such as MATLAB is usually not considered. This is in our opinion a significant disadvantage, because, as it turns out, the time of model construction and the time of transient simulation within the capabilities of MATLAB language can differ by orders of magnitude depending on the applied procedures. Thus, in the example [17] the step function with sampling time Δt =0.001 is applied for simulation of transient processes. Specification of a Δt value automatically switch the solver into the mode of discrete modeling with transformation of the model into discrete form. This is convenient for the calculation of integral index, but from simu-

lation speed point of view it is not an optimal approach.

For accurate and fast determination of performance indicators it is necessary: not to use delay approximation and to avoid discrete modeling. The delay approximation slightly modifies the performance of control system (see examples of the effect of different approximations on direct control system performance indicators in [19]). And since a small sampling time is critical for the simulation process only at a small interval from the end of the delay until reaching 2/3 of the value of the controlled variable, it is reasonable to simulate the rest of the process dynamics with a large sampling time. For this purpose, a variable step solver is needed to support delayed systems.

Tunings of the PI controller (variables kp, Ti) for the plant model (variables k, T, tau) will be determined by the program *pidtune*. In total, within the standard MATLAB tools (without Simulink) we can distinguish 6 options for the implementation of transient simulation by reference change:

1) P=k*exp(-tau*s)/(T*s+1); C=*pidstd*(kp,Ti); CL=(C*P)/(C*P+1);

2) P=k*exp(-tau*s)/(T*s+1); C=*pidstd*(kp,Ti); CL=*feedback*(C*P,1);

3) P=*tf*(k,[T 1],'ioDelay',tau); C=*pidstd*(kp,Ti); CL=(C*P)/(C*P+1);

4) P=tf(k,[T 1],'ioDelay',tau); C=*pidstd*(kp,Ti); CL=*feedback*(C*P,1);

5) P=*ss*(-1/T,k/T,1,0,'InputDelay',tau); C=*ss*(0,1,kp/Ti,kp); L=P*C; CL=L/(L+1);

6) P=*ss*(-1/T,k/T,1,0,'InputDelay',tau); C=*ss*(0,1,kp/Ti,kp); CL=*feedback*(P*C,1);

The first two options require initialization of the variable s=*tf*('s');

For all cases the transient simulation is performed with the command

[y,t]=*step*(CL).

The functions *tic* and *toc* are used to measure the time to construct the control system model and the time to simulate the transient process using this model.

Let us draw some conclusions about the found features of these options. The *feedback* function closes the loop without creating redundant states, which affects the speed of transient simulation. Writing transfer functions in algebraic form with respect to the operator *s* to create a model in TF slows down the construction of the control system model and slightly, but still increases the simulation time. The state-space (SS) form, as the experiment shows, is the most effective, both in terms of speed of model construction and in terms of speed of transient computation.

This behavior is explained by the fact that MATLAB, when performing operations with TF, converts them to state space and, when possible, translates them back to the original form of the model after execution. Since MATLAB (as well as similar packages) uses linear algebra libraries (LAPACK, Intel oneMKL, etc.) at the lower level, the natural form of model representation for it is the state space. A detailed description of the development history of MATLAB and an overview of its modern architecture can be found in [18]. After R2017b, MATLAB begins to use the descriptive state space (DSS) form [20] with the addition of internal delays to represent delayed systems. In essence, DSS is a SS model (parameters *A,B,C,D*) extended by a fifth parameter, matrix *E*. The rule for calculating the derivative of the state vector in DSS is as follows:

$$E \cdot \dot{x} = A \cdot x + B \cdot u, y = C \cdot x + D \cdot u$$

A model in the DSS form with internal delays is described by a system of equations of the following form

$$\begin{aligned} E \cdot \dot{x} &= A \cdot x(t) + B_1 \cdot u(t) + B_2 \cdot w(t) \\ y(t) &= C_1 \cdot x(t) + D_{11} \cdot u(t) + D_{12} \cdot w(t) \\ \bar{z}(t) &= C_2 \cdot x(t) + D_{21} \cdot u(t) + D_{22} \cdot w(t) \end{aligned} \quad (10)$$

As we can see, matrix *B is* split into two parts, matrix *C* is also split, and matrix *D* is split into four parts. The system model now has two types of inputs and outputs. The external inputs with respect to *u(t)* and the outputs *y(t)* go first, followed by the internal inputs and outputs. The internal outputs of the system *z(t)* pass through the delay block and as a signal *w(t)* enters the internal inputs. The number of internal and external outputs may not be equal.

When constructing a closed-loop control system, a DSS system with two identical delays is created. The first delay reflects the delay in the numerator, the second - in the denominator of the control system TF. We derive analytical model of closed-loop control system in DSS with delays for compactness with respect to zero matrices

$$\begin{aligned} &A = 0_{n\times n}, B_1 = 0_{n\times m_1}, B_2 = 0_{n\times m_2}, \\ &C_1 = 0_{m_1\times n}, C_2 = 0_{m_2\times n}, \\ &D_{11} = 0_{m_1\times m_1}, D_{12} = 0_{m_1\times m_2}, \\ &D_{21} = 0_{m_2\times m_1}, D_{22} = 0_{m_2\times m_2}, E = 0_{n\times n} \end{aligned} \quad (11)$$

The model of closed-loop control system with controlled variable as output variable is described by the following parameters:

$$
\begin{aligned}
&n=5, m_1=1, m_2=2, A(1,1)=A(3,3)=-1/T,\\
&B_1(5,1)=-1, B_2(1,1)=B_2(3,2)=k/T\\
&C_2(1,2)=C_2(2,4)=k_p/T_i,\\
&C_2(5,1)=C_2(5,2)=k_p,\\
&A(2,5)=A(4,5)=A(3,5)=A(5,5)=C_1(1)=\\
&E(1,1)=E(2,2)=E(3,3)=E(4,4)=1
\end{aligned}
\tag{12}
$$

The model of closed-loop control system with manipulated variable as output variable is described by the following parameters:

$$
\begin{aligned}
&n=4, m_1=1, m_2=1, A(2,2)=-1/T,\\
&B_1(4,1)=-1, B_2(2,1)=k/T\\
&C_2(1,1)=C_2(1,3)=k_p/T_i,\\
&C_1(1,4)=C_2(1,4)=k_p\\
&A(1,4)=A(3,4)=A(4,2)=A(4,4)=\\
&E(1,1)=E(2,2)=E(3,3)=1
\end{aligned}
\tag{13}
$$

Now we have all the input data for the software implementation of transient simulation.

Let's consider two additional options of implementation of such a structure.

Option #7. The model of control system is realized as a DSS system and transferred to the step function

CL = *setDelayModel*(A,B1,B2,C1,C2, D11,D12,D21,D22,[tau;tau]); CL.E=E;

[y,t]=*step*(CL_my);

Option #8. The model is constructed using the *isproper* function and simulated by the undocumented function *ddaeresp*. This function, after resource consuming check of several conditions, is called in the step function to simulate systems with internal delays.

First, let's define the structure containing the DSS model.

DI=ltipack.*ssdata*();DI.a=A; DI.b=[B1 B2];

DI.c=[C1; C2]; DI.d=[D11 D12; D21 D22];

DI.e=E;

DI.Delay=*struct*('Input',0,'Output',0,…
'Internal', [tau;tau]); DI.Ts=0;

The systems (12), (13) are not proper This means that the descriptor system has impulse (algebraic) modes. The indicator of an improper system is the incomplete rank of the matrix *E*. A proper system is elementarily translated into a regular state space by multiplying both parts of the equation by E^{-1}. An improper system requires special algorithms to recalculate it into a standard state space form [21], the main idea of which is to remove the algebraic part by finding such a transformation of the system that the matrix E becomes diagonal. The *isproper* function transforms the system into the proper form, i.e., a regular state space system, and applies scaling to emphasize the dominant dynamics in the frequency domain.

[is_ok, D_CL] = *isproper*(DI,2);

The D_CL system can now be modeled using a delayed differential equation solver. The basic description of the solver's algorithm is given in [22]. The algorithm is designed for stable linear systems, is limited to a few possible input signals (in the MATLAB implementation it seems to be only step signals), and is oriented on the fact that the value of the rational sample time of the solver increases with the simulation time. The smallest sample time is reasonable at points between delays.

SimInfo = *struct* ('FinalValue', Inf, 'IC',[], ...
'MaxSample', 50000, 'DivThreshold', 10000, ...
'ComputeX', 0, 'XMap', {[]}, 'uinit', 0,...
'du', 0, 'xinit', []); [y,t,tFocus] = ...
ddaeresp(D_CL,[],[],SimInfo);

The solver simulates systems on a channel-by-channel basis. Therefore, construction of MIMO systems does not improve speed. The *ddaeresp* function is not described in the MATLAB documentation, the list of possible parameters depends on the language version and can be changed without warning. The operability of the structure has been verified in R2021a and R2024a. For R2021a parameters after XMap are not required, but parameter overflow is not a problem, the problem is the lack of required parameters. The list of parameters for specific MATLAB version can be obtained from the *step* function code or by using a debugger.

To correctly set up an experiment to determine the fastest option, it is necessary to take into account certain peculiarities of the MATLAB environment. In MATLAB, operations that do the same thing can be unequal in terms of speed and resources used. In addition, MATLAB uses optimization to perform repetitive operations, but this also has its own peculiarities, and the efficiency can be significantly different [23].

The following techniques were used for the clean experiment: 1) before executing the script, the workspace was cleared with the clear all command; 2) transients were simulated when executing the script with only one of the options, since sequential execution with different options distorts the results; 3) only model construction time and simulation time were measured separately, auxiliary operations were not measured.

The results of experiment to find the fastest option of simulation implementation are shown in Table 1. The main experiment was conducted

on a 4-core Intel processor in the MATLAB R2021a environment.

The following conclusions can be drawn from Table 1:

1) the execution time of one iteration does not depend or almost does not depend on the ratio of FOPDT parameters of the model;

2) in cyclic simulation execution, the average time is less than the time of one iteration due to MATLAB optimization algorithms;

3) the acceleration factor of cyclic operations is very uneven and is for #1 - 32, #2 - 62, #3 - 51, #4 - 138, #5 - 147, #6 - 308, #7 - 256, #8 - 999;

4) *ss*, *feedback*, *step* is the most effective combination among the standard approaches;

5) application of the *setDelayModel* function with passing the model in the DSS with delay and its subsequent modeling with step function is less effective than option #6;

6) option #8 allows achieving significant acceleration, especially when applied in cyclic algorithms.

Table 1.

Experimental results on determining the speed of different options for obtaining transients of control system by reference with controlled variable as the output in MATLAB R2021a

Option #	Single experiment				320 experiments performed in a loop without clearing the workspace between loops				
	k=1 T=10 τ=5	k=5 T=10 τ=5	k=1 T=50 τ=5	k=1 T=10 τ=50	k=1, 21, 41, 61, 81 T=1, 26, 51, 76 τ= 1, 26, 51, 76, 101, 126, 151, 176, 201, 226, 251, 276				Simulations per second (average of 2 attempts)
					Attempt 1		Attempt 2		
1	0.4*	0.4	0.38	0.38	7.43	0.023219	7.36	0.023	21.63624
	0.2**	0.2	0.17	0.17	1.78	0.005563	1.77	0.005531	90.14085
	0.54***	**0.55**	**0.54**	**0.54**	**9.21**	0.028781	**9:12**	0.0285	**17.45772**
2	0.3	0.3	0.31	0.31	5.82	0.018188	5.8	0.018125	27.53873
	0.1	0.1	0.08	0.08	0.8	0.0025	0.79	0.002469	201.2579
	0.39	**0.38**	**0.39**	**0.38**	**6.62**	0.020688	**6.59**	0.020594	**24.22407**
3	0.3	0.3	0.3	0.3	4.85	0.015156	4.94	0.015438	32.68641
	0.2	0.2	0.17	0.16	1.76	0.0055	1.79	0.005594	90.14085
	0.47	**0.47**	**0.47**	**0.46**	**6.61**	0.020656	**6.73**	0.021031	**23.98801**
4	0.2	0.2	0.22	0.22	3.18	0.009938	3.24	0.010125	49.84424
	0.1	0.1	0.08	0.08	0.8	0.0025	0.81	0.002531	198.7578
	0.29	**0.29**	**0.3**	**0.3**	**3.98**	0.012438	**4.04**	0.012625	**39.90025**
5	0.2	0.2	0.17	0.17	1.62	0.005063	1.63	0.005094	98.46154
	0.2	0.2	0.16	0.16	1.67	0.005219	1.68	0.00525	95.52239
	0.33	**0.33**	**0.33**	**0.33**	**3.29**	0.010281	**3.31**	0.010344	**48.48485**
6	0.2	0.2	0.15	0.15	1.49	0.004656	1.51	0.004719	106.6667
	0.1	0.1	0.08	0.08	0.75	0.002344	0.76	0.002375	211.9205
	0.23	**0.23**	**0.23**	**0.23**	**2.24**	0.007	**2.28**	0.007125	**70.79646**
7	0.1	0.1	0.08	0.08	0.8	0.0025	0.8	0.0025	200
	0.2	0.2	0.17	0.17	1.69	0.005281	1.71	0.005344	94.11765
	0.25	**0.25**	**0.25**	**0.25**	**2.49**	0.007781	**2.51**	0.007844	**64**
8	0.1	0.1	0.09	0.08	0.82	0.002563	0.82	0.002563	195.122
	0	0	0.04	0.04	0.52	0.001625	0.52	0.001625	307.6923
	0.12	**0.12**	**0.12**	**0.12**	**1.34**	0.004188	**1.33**	0.004156	**119.8502**

Note: * – time of control system model construction, ** – simulation time, *** – sum.

The experiment was also repeated in R2024a version. The results are slightly different, but tendency is the same.

AN ALGORITHM FOR GENERATING A SET OF RATIONAL PI CONTROLLER TUNINGS WITH THE EVALUATION OF SEVEN PERFORMANCE INDICATORS

In order to be able to easily select desired values of performance indicators and estimate their variation, it is necessary to calculate the set of all possible controller tunings. If the step is small enough, this is a large number - tens of thousands, so a special approach is needed to solve the problem in seconds.

To find the starting tunings, we apply the SIMC method [24], since it gives satisfactory tunings for any value of τ /T and is simple

$$k_{pn} = \frac{1}{k} \cdot \left(\frac{T}{\lambda + \tau} \right),\ T_{in} = min(T, 4 \cdot (\tau + \lambda)),\ \lambda = \tau \quad (14)$$

The value of the controller gain coefficient as in Ziegler-Nichols based methods is determined by the parameter $a = k \cdot \tau / T$. The value of integration time is determined by a nonlinear function. The cutoff frequency and critical frequency are completely determined by the value of τ . Despite the strong correlation of the control system performance indicators, the method ensures that all frequency margins and oscillation indicators are close to a constant value. The magnitude of GM is not high (~2) and the relative DM is low compared to some other methods.

The proposed algorithm starts by checking the input data – FOPDT model parameters *k*, *T*, τ. The parameters must be real positive numbers. The model with $\tau / T > 6$ is rejected as irrational, since it is appropriate to use an I-controller, a Dahlin controller or a Smith predictor with a PI-controller for such plants. The model with $\tau / T < 0.03$ is defined as reasonable for using analytical tuning methods for plant models without delay, depending on the peculiarities of the dynamics of the real plant. A model with at least one parameter less than 10^{-2} or more than 1000 is defined as unsuitable because it will lead to problems with calculation accuracy. Such a model should be rescaled in time and/or in gain units.

An array with possible controller tunings is initialized using the *meshgrid* function and the indicators of the control system, the array is sorted by the column k_p . By performing element-by-element operations on the functions with respect to the whole columns of the array, the indicators of the control systems are calculated and inappropriate tunings are rejected in this sequence.

1) Calculation $\omega_g = f_{\omega_g}(k, T, k_p, T_i)$

2) Calculation $PM = f_{PM}(T, \tau, k_p, T_i, \omega_g)$

3) Deleting tunings with PM<5° or PM>90°

4) Calculation $\omega_g = f_{\omega_g}(k, T, k_p, T_i)$

5) Calculation $\omega_c = f_{\omega_c}(k, T, \tau, k_p, T_i, \omega_g)$

6) Deleting tunings with imaginary values ω_c or ω_g (occurs in some extreme cases)

7) Calculation $GM = f_{GM}(k, T, \tau, k_p, T_i, \omega_c)$

8) Deleting tunings with GM<1

9) Calculation $\tau_m = f_{\tau_m}(\tau, PM, \omega_g)$

10) With an interval of 300 a set of tunings is created for simulation of control systems.

11) Simulation is carried out using option #8 for closed-loop control system according to the formed set (with respect to the controlled variable and the manipulated variable), the maximum values of the controlled variables are fixed in. y_{max} and of manipulated variables in u_{max} .

12) Based on the simulation results, functions are generated using spline approximation (*fit* function with "thinplateinterp" method) $f_{M_p}(PM, \tau_m)$, $f_{u_{max}}(PM, \omega_g)$.

13) The values of M_p, u_{max} for all tunings for which they were not obtained by simulation are calculated using the functions.

14) Tunings with $M_p > 2$ are deleted.

15) The negative values of performance indicator M_p are equated to zero.

Having a set of all suitable tunings according to the formulas (3)-(9) we form a table of the following form: $k_p, T_i, GM, PM, \omega_c, \omega_g, \tau_m, u_{max}, M_p$.

By setting one or more constraints on the minimum and maximum $\omega_c, \omega_g, \tau_m, u_{max}, M_p$ we select a sub-table with allowable performance indicator values. Preferably, the user should enable the constraints sequentially, so that the tolerance of other indicators that can be included in the constraints can be calculated. There should be a reasonable difference between the minimum and maximum of the allowable indicator value (at least 5%). Focusing on the average values of the minimum and maximum of each indicator using the Euclidean metric, we find the most appropriate row in the subtable. Then, if the appropriate tuning is found, we demonstrate to the user transients and accurately calculated performance indicators.

CHECKING THE EFFICIENCY OF THE CALCULATION OF THE FIRST FIVE INDICATORS

The first four indicators are frequency parameters, and τ_m in principle, is calculated by the formula from two frequency parameters. In MATLAB, the first four indicators can be obtained using the *margin* function, and all five indicators using the *allmargin* function.

As input data for the experiment, we will use the set of admissible controller tunings filtered in step 8 of the algorithm for the plant model $2 \cdot e^{-5s} / (10 \cdot s + 1)$. Let's save this set into a mat file, clear the workspace and check the calculation speed using *margin*, *allmargin* and proposed algorithm functions. We will use for model construction the option #6 (without feedback) as the fastest of the standard MATLAB approaches.

```
k=2;T=10;tau=5;
P=ss(-1/T,k/T,1,0,'InputDelay',tau);

In loop
#1 C=ss(0,1,kp/Ti,kp); C=ss(0,1,kp/Ti,kp);
[GM,PM,wc,wg]=margin (C*P);
#2 C=ss(0,1,kp/Ti,kp); fc=allmargin (C*P);
```

After discarding irrational tunings, the set of 90,000 tunings was reduced to a set of 59,652 tunings.

The computation time of constructing P*C systems without changing P to compute function arguments is 39.64 s, i.e., 1502.65 computations per second.

The calculation time of P*C systems construction together with the calculation of frequency response by the margin function was 274.09 s, i.e. 217.64 calculations per second. The disadvantage is that the function does not give the DM value τ_m.

The calculation time of P*C systems together with the calculation of frequency characteristics by the *allmargin* function was 324.768334 c, i.e. 183.68 calculations per second. The disadvantage is that it is necessary to take minima from the obtained vectors, as well as to convert the DM from absolute to relative.

Now let us conduct an experiment using the functions developed for calculating the performance indicators, which can take as arguments the vectors of parameters. The results of the experiment are summarized in Table 2.

Table 2.

Results of the experiment for speed measurement of software realization of the functions based on the formulas (3-9) in MATLAB R2021a

Function	Performance indicators calculation time for 59,625 control system models	Number of calculations per second	Normalized
$\omega_g = f_{\omega_g}(k,T,k_p,T_i)$	0.00150740	39554862.7	0.0020
$PM = f_{PM}(T,\tau,k_p,T_i,\omega_g)$	0.00521730	11428325.0	0.0070
$\omega_c = f_{\omega_c}(k,T,\tau,k_p,T_i,\omega_g)$	0.69884850	85318.9	0.9424
$GM = f_{GM}(k,T,\tau,k_p,T_i,\omega_c)$	0.03476820	1714929.2	0.0469
$\tau_m = f_{\tau_m}(\tau,PM,\omega_g)$	0.00122290	48757052.9	0.0016
Total of 5 parameters	0.74156430	80404.4	1

As we can see, we achieve acceleration compared to *margin*/*allmargin* function by hundreds of times. 94.2% of the time is taken by the calculation of w_c, which optimally refines the solution having a certain error and searches for the exact solution with an accuracy of 3 digits after the point. If we lower the requirement, the speedup will be even greater, but a value less than a second is quite satisfactory.

VERIFICATION OF THE ALGORITHM CALCULATIONS ACCURACY

We will check the accuracy using the following methodology. We will select 6 representative FOPDT models and use the proposed algorithm to find the PI controller tunings corresponding to the phase margin PM=40, 50, 60°. Let's compare the estimation of the algorithm and the results of direct transient modeling and frequency response estimation in MATLAB. More precisely, we compare the algorithm results with the results obtained by *allmargin* and *stepinfo* functions from the step function. The comparison of the results is presented in Table 3.

The developed algorithm searched among a set of tens of thousands of pre-calculated performance indicators of the control system for a row with controller tunings that is closer to the de-

sired PM. We see that the tunings suitable for the desired phase margin are found with high accuracy, which actually depends on the step with which we consider the controller tunings. The results for models #1 and 2 show that the algorithm is independent of the scale of the coefficients. The results for the other models demonstrate that the algorithm calculate indicators equally accurately for both dominant time constant (#6) and dominant delay (#4) and in the intermediate cases (#3, #5).

In general, differences occur at the third digit after the point. A definite exception is only the parameter M_p , which in some cases has an error at the second digit. But here it is a matter of a minor difference between the transients given by the step and *ddaeresp* functions.

THE PROBLEM OF MULTI-PARAMETER OPTIMIZATION OF PID FAMILY CONTROLLERS

Among the scientific works devoted to the optimization of PID family controllers for delayed plants, clearly a smaller part considers multi-parameter optimization. In general, optimization by integral criteria is considered to a greater extent, since it is convenient for optimization algorithms. Both classical algorithms and heuristic algorithms cannot be efficient in such a problem and cannot always find the optimum.

Table 3.

Comparison of obtained performance indicators by the proposed algorithm (alg.) and by standard MATLAB (ML) functions *allmargin* and *step*+*stepinfo*

Model	~PM	kp	Ti	Method	*GM*	*PM*	ω_c	ω_g	τ_m	$u_{\max}$	M_p
$\frac{1\cdot e^{-2\cdot s}}{3\cdot s+1}$	40	1.8	5.6	Alg.	1.485	40.002	0.867	0.551	0.633	2.496	0.375
				ML	1.485	40.006	0.867	0.551	0.634	2.496	0.374
	50	1.5	4.8	Alg.	1.748	50.000	0.851	0.450	0.970	2.161	0.253
				ML	1.749	50.005	0.851	0.450	0.970	2.161	0.254
	60	1.3	4.6	Alg.	2.086	60.001	0.847	0.363	1.443	1.820	0.116
				ML	2.086	60.001	0.847	0.363	1.443	1.820	0.119
$\frac{100\cdot e^{-2\cdot s}}{3\cdot s+1}$	40	1.8e-2	5.6	Alg.	1.485	40.002	0.867	0.551	0.633	0.025	0.375
				ML	1.485	40.006	0.867	0.551	0.634	2.496	0.374
	50	1.5e-2	4.8	Alg.	1.748	50.000	0.851	0.450	0.970	0.022	0.253
				ML	1.749	50.005	0.851	0.450	0.970	0.022	0.254
	60	1.3e-2	4.6	Alg.	2.086	60.001	0.847	0.363	1.443	0.018	0.116
				ML	2.086	60.001	0.847	0.363	1.443	0.018	0.119
$\frac{2\cdot e^{-1\cdot s}}{3\cdot s+1}$	40	7.5e-1	1.7	Alg.	2.675	39.999	1.409	0.608	1.148	1.200	0.373
				ML	2.676	39.999	1.410	0.608	1.148	1.195	0.311
	50	1.3	5.0	Alg.	1.896	50.004	1.647	0.841	1.038	1.585	0.214
				ML	1.898	50.002	1.649	0.841	1.038	1.585	0.215
	60	1.2	7.8	Alg.	2.081	60.001	1.689	0.767	1.365	1.397	0.082
				ML	2.081	60.002	1.690	0.767	1.365	1.397	0.078
$\frac{2\cdot e^{-15\cdot s}}{3\cdot s+1}$	40	1.2e-1	3.9	Alg.	1.753	40.000	0.111	0.062	0.754	0.701	0.392
				ML	1.752	40.000	0.111	0.062	0.754	0.700	0.384
	50	1.2e-1	4.7	Alg.	2.098	50.002	0.116	0.052	1.116	0.614	0.217
				ML	2.097	50.002	0.116	0.052	1.116	0.614	0.217
	60	1.2e-1	5.6	Alg.	2.594	60.001	0.121	0.042	1.663	0.529	0.058
				ML	2.591	60.001	0.121	0.042	1.663	0.529	0.055
$\frac{1\cdot e^{-2\cdot s}}{10\cdot s+1}$	40	5	23	Alg.	1.648	40.001	0.819	0.492	0.710	05.439	0.351
				ML	1.648	40.001	0.819	0.492	0.710	5.441	0.341
	50	4.3	26	Alg.	1.924	50.002	0.822	0.420	1.039	4.632	0.199
				ML	1.924	50.002	0.822	0.420	1.039	4.632	0.193
	60	3.4	19	Alg.	2.406	60.000	0.813	0.330	1.589	3.760	0.061
				ML	2.407	60.000	0.814	0.330	1.589	3.760	0.061
$\frac{1\cdot e^{-2\cdot s}}{40\cdot s+1}$	40	17	29	Alg.	1.832	40.000	0.779	0.426	0.820	18.156	0.382
				ML	1.833	39.985	0.780	0.426	0.820	18.161	0.378
	50	14	38	Alg.	2.257	50.000	0.784	0.348	1.255	14.622	0.203
				ML	2.257	50.000	0.784	0.348	1.255	14.624	0.205
	60	9.5	28	Alg.	3.273	59.999	0.778	0.239	2.192	10.183	0.067
				ML	3.275	59.999	0.778	0.239	2.192	10.186	0.067

Some combinations of parameters in the optimization criterion, regardless of weighting, can quickly lead the optimization algorithm into the zone of instability. At the same time, determining from time domain simulations "how unstable the system is" in the form of a number is not a simple task.

In general, dynamic optimization, both building a search route according to a certain rule and moving to random points of space (as GA), has a low probability of getting into the zone of global optimum in problems with more than one performance indicator in the search criterion. This is explained by the fact that the achievement of one performance indicator imposes significant restrictions on the possibility of achieving other performance indicators. And the nature of change in the zone of admissible parameters is not analytically predictable.

Let us consider, for example, model #5 from Table 3 and demonstrate in Table 4 the feasibility of applying the search procedure by sequential refinement of constraints. The main requirement of this method is the need to rank the performance indicators in a certain order, i.e., to establish their priority. Further, the introduction of each additional constraint on a new performance indicator is realized taking into account the constraint zone imposed on the introduction of a constraint on an indicator with a higher priority. In this example, first the phase margin constraint, then the maximum control action, and finally the overshoot constraint. The specific values of the constraints need to be approximated for the procedure, but the specific constraints on the second parameter should continue to be introduced with an eye towards possible values. If the range at any step is not satisfactory, the requirements for higher priority indicators should be relaxed.

Table 4.

Demonstration of finding optimal tuning by entering sequential constraints on performance indicators.

Constraints			Possible values of performance indicators						
PM	u_{max}	M_p	*GM*	*PM*	ω_c	ω_g	τ_m	M_p	u_{max}
-	-	-	1.06-215.13	5-90	0.5-0.83	0.01-0.75	0.06-88.45	0-2	0.95-9.1
50-70			1.91-44.72	50-70	0.5-0.83	0.04-0.42	1.03-15.57	0-0.21	1-4.65
50-70	1.5-2	-	4.72-9.15	50.01-69.96	0.68-0.78	0.12-0.18	2.48-4.65	0.01-0.18	1.5-2.0
50-70	1.5-2	0.01-0.05	4.72-6.69	63.53-69.96	0.76-0.78	0.13-0.18	3.45-4.65	0.01-0.05	1.5-2.0

This approach is similar to the one used in [8], in which it is emphasized that the optimization problem of PID family controllers should be solved by a hierarchy of constraints. This avoids a highly rapid change of the function during the route search process. The route itself should be conducted on a selected rectangular zone, which defines using the Routh or Nyquist stability criteria (for delayed systems).

Under the above constraints, the algorithm based on the Euclidean metric offers the following satisfying settings:

$$k_p = 1.57, T_i = 7.7 \rightarrow$$

$$GM = 4.84, PM = 63.53°, \omega_c = 0.77, \omega_g = 0.17,$$

$$\tau_m = 3.25, M_p = 0.05, u_{max} = 1.95$$

The narrowing of the area of possible settings when restrictions are imposed is shown in Fig. 1.

Thus, we see that the zone of admissible indicators is distinguished from the zone of admissible indicators imposed by the prior constraints. The fact that classical and, for example, genetic optimization algorithms can get, say, into zone 4 under a multi-parameter criterion is possible.

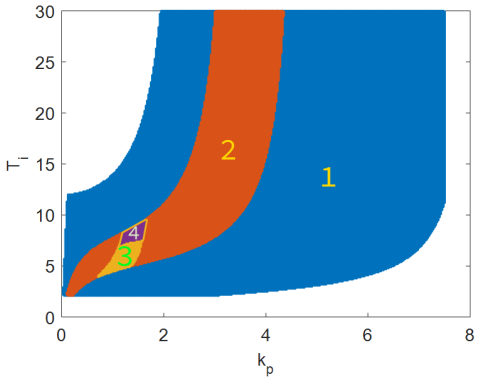

Fig. 1 - Visualization of the change in the zone of possible tunings depending on the search area: 1 - rational constraints only, 2 – constraints on *PM*, 3- constraints on *PM* & u_{max}, 4 – constraints on *PM* & u_{max} & M_p

However, the absence of the zone of possible indicators in the optimization problem may lead to the fact that the search route deviates, for example, to the zone of optimum for 2-3 indicators and will not find the optimum for 4th. The proposed approach does not have such a disadvantage, because it is focused on the definition of possible indicator zones.

CONCLUSIONS

The problem of finding PI controller tunings for the FOPDT model under user-specified constraints on the time domain and frequency performance indicators is solved. The solution of the problem does not require the use of resource-consuming optimization algorithms and allows the user to control the zone of achievable indicators when introducing new constraints. Application of theoretical and empirical formulas allowed to make the software realization of calculation of control systems performance indicators for tens of thousands of combinations of parameters k_p, T_i almost instantaneous. This was also helped by MATLAB architecture, simple calculations in which in terms of speed approach such languages as FORTRAN and C [25] Attention to the procedure of transient simulation in MATLAB allowed us to create a procedure of derivation of empirical spline equations on the magnitude of overshoot and maximum control action on a relatively small set of transient modeling results, which is carried out significantly faster than with the use of standard approaches to transient modeling.

The approach taken can potentially be used for other PID family controllers, for integral delayed control systems, for SOPDT control systems (at least aperiodic) and for control systems with known disturbance models.

APPENDIX 1.

DEVELOPMENT OF THE SOFTWARE APPLICATION WITH A GRAPHICAL INTERFACE

A program with a graphical interface on MATLAB was developed for the convenience of the search. The user, specifying the parameters of the FOPDT model, gets a set of rational tunings displayed on the graph. Further, by including the desired constraints and specifying their value, the user requests to find tunings. The step response of the plant and the transients of the control system by reference (controlled variable and control action) are visualized so that the user can evaluate the processes. The actual values of the achieved quality parameters are also calculated so that the user can evaluate their deviations.

The application screenshots at different cases are shown in Fig. 2-4

The user graphical interface consists of 5 zones (frames) arranged in 2 columns.

Zone #1 of the first column allows setting the parameters of the FOPDT model. Changing the parameters clears all other zones. When the user presses the button for calculation, the other zones are activated.

Zone No. 2 of the first column visualizes a set of acceptable and ensuring control system stability values of controller tunings, which will be used for further search of optimal values corresponding to the specified criteria. On the horizontal scale of the graph – k_p values on the vertical scale – T_i values.

Zone No. 3 of the first column allows setting a performance indicators constraint. To enable the constraint, the user must activate the corresponding checkbox. The left input field in the row means the minimum value, the right field means the maximum value. A certain difference (5%) between the minimum and maximum is mandatory. The search procedure tries to find the average value. Along with the input fields, the zones of permissible performance indicator deviations are marked. Values are refreshed when a constraint on any parameter is disabled or enabled. When the user has set the desired constraints, the user presses the button to search for the zone of acceptable indicators. Such controller tunings are found that correspond to the center by Euclidean metric for the given number of performance indicators.

Zone No. 4 of the second column contains three graphs: the step response of the plant model, the transient of the controlled variable at set point and the transient of the control action at set point in the control system by with the found PI controller tunings.

Zone No. 5 of the second column contains the achieved indicators for all parameters of the control system. The first column in the zone displays the theoretical estimation, on the basis of which the tunings in the table were searched. The second column displays the actual parameters of the control system obtained by simulation and frequency calculations. Ideally, the estimate and the actual value should be very close.

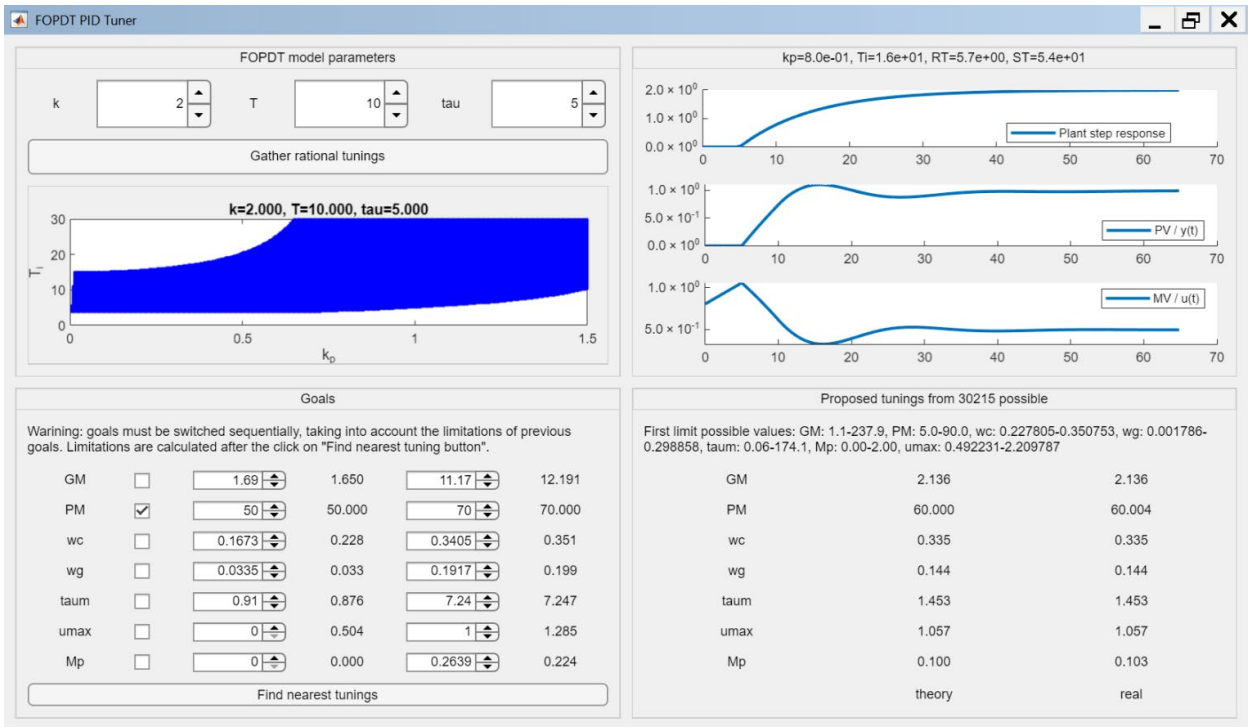

Fig. 2 – Searching for the controller tunings with $PM = 50..70°$ for the plant with $\tau / T = 0.5$

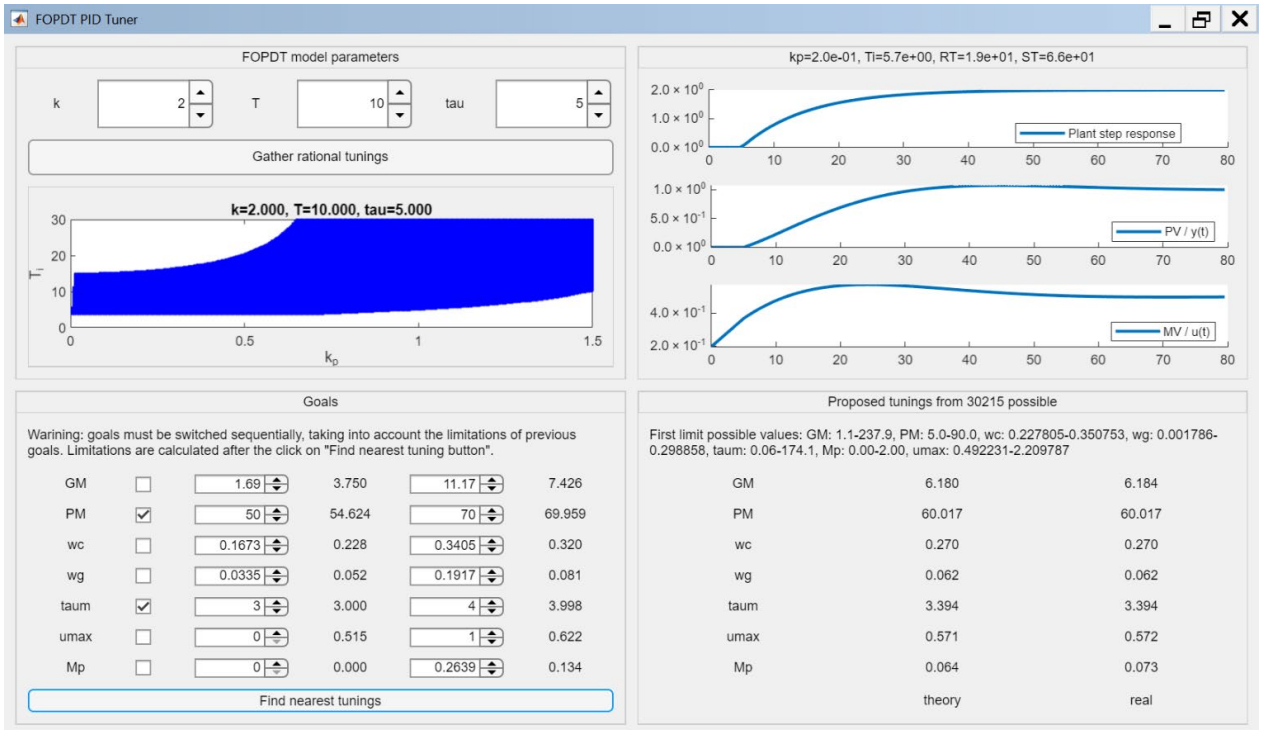

Fig. 3 - Multi-parameter search of controller tuning for plant with $\tau / T = 0.5$

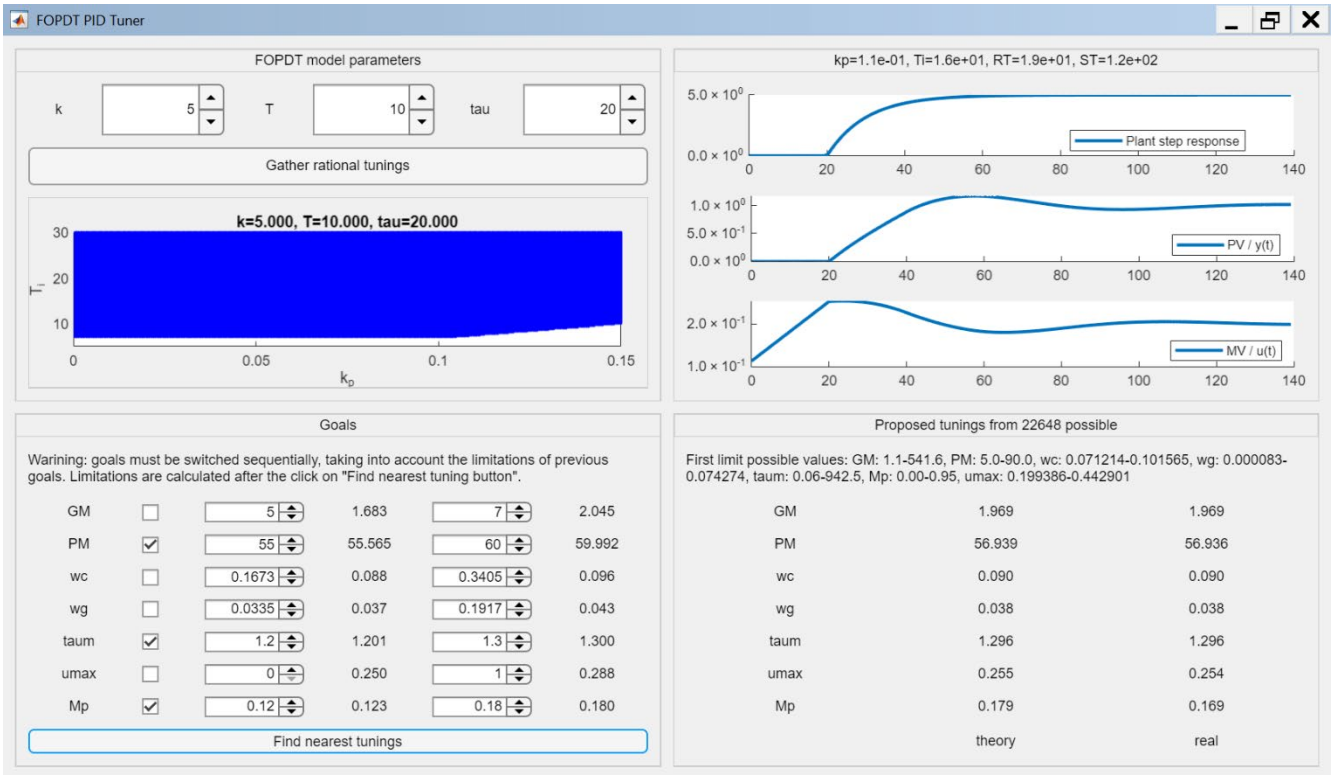

Fig. 4 – Achievement of PM, τ_m, M_p for the plant with $\tau / T = 2$

Analyzing the figures, we see that, in general, the theoretical indicators in all cases do not deviate much from those found by simulation and frequency calculations. The performance indicators requested by the user are fully achieved.

References

[1] O'Dwyer A. Handbook of PI and PID Controller Tuning Rules (3rd Edition). London, Imperial College Press, 2009.

[2] Gumussoy S., Gahinet P. Computer Aided Control System Design for Time Delay Systems Using MATLAB®. *Advances in Delays and Dynamics*, 2014, pp. 257–270. doi:10.1007/978-3-319-01695-5_19.

[3] Da Silva L.R., Flesch R.C.C., Normey-Rico J.E. Controlling industrial dead-time systems: When to use a PID or an advanced controller. *ISA Transactions*, 2020, vol. 99, pp. 339–350. doi:10.1016/j.isatra.2019.09.008

[4] Ho W.K., Hang C.C. Zhou J. H. Performance and gain and phase margins of well-known PI tuning formulas. *IEEE Transactions on Control Systems Technology*, vol. 3, no. 2, pp. 245-248. doi: 10.1109/87.388135.

[5] Garpinger O., Hägglund T., Åström K.J. Performance and robustness trade-offs in PID control. *Journal of Process Control*, 2014, vol 24, no. 5, 2014, pp 568-577. doi: 10.1016/j.jprocont.2014.02.020.

[6] Patil R., Jadhav S., Patil M. Review of Intelligent and Nature-Inspired Algorithms-Based Methods for Tuning PID Controllers in Industrial Applications. *Journal of Robotics and Control*, 2024, vol 5, no. 2. doi: 10.18196/jrc.v5i2.20850

[7] Zamani M., Karimi M., Sadati N., Parniani M. Design of a fractional order PID controller for an AVR using particle swarm optimization. *Control Engineering Practice*, 2009, vol. 17, no.12, pp. 1380-1387. doi: 10.1016/j.conengprac.2009.07.005.

[8] Carotenuto L., Pugliese P., Sergeyev Y. Maximizing performance and robustness of PI and PID controllers by global optimization. *Control and Intelligent Systems*, 2006, vol. 34. 10.2316/Journal.201.2006.3.201-1558.

[9] Mirzal A., Yoshii S., Furukawa M. PID Parameters Optimization by Using Genetic Algorithm. *ArXiv*, 2012. https://arxiv.org/abs/1204.0885

[10] Xia H., Yu M. A Method of Performance Assessment of PID Controller with Actuator Saturation. *International Conference on Mechatronics, Electronic, Industrial and Control Engineering* (Paris, France), 2015. doi: 10.2991/meic-15.2015.189

[11] Ntogramatzidis L., Ferrante A. Exact Tuning of PID Controllers in Control Feedback Design. *IET Control Theory and Applications*, 2015, vol. 5, no. 4, pp. 565–578, doi:10.1049/iet-cta.2010.0239.

[12] Wu Z., Li D., Xue Y. A new PID controller design with constraints on relative delay margin for first-order plus dead-time systems. *Processes*, 2019, vol. 7, no. 10, p. 713.

doi: 10.3390/pr7100713
[13] Srivastava S., Pandit V.S. A PI/PID controller for time delay systems with desired closed loop time response and guaranteed gain and phase margins. *Journal of Process Control*, 2016, vol. 37, pp. 70-77, doi: 10.1016/j.jprocont.2015.11.001.
[14] Tan K.K., Lee T.H., Wang Q.G. Enhanced Automatic Tuning Procedure for Process Control of PI/ PID Controllers, *AIChE*, 1996, vol.42, no.9. p. 2555-2562 doi: 10.1002/aic.690420916
[15] Chua BL., Tai F.S., Aziz N.A., Choong T.S.Y. PID tuning with input constraint: application on food processing. *International Journal of Engineering and Technology*. 2009. V.6 (2). P. 83–89.
[16] Kula K.S. Tuning a PI/PID Controller with Direct Synthesis to Obtain a Non-Oscillatory Response of Time-Delayed Systems. *Poland Appl. Sci*., 2024, vol.14, no.13. P.5468. doi: 10.3390/app14135468
[17] Sudhakar K. MATLAB code for tuning a PID controller using Genetic Algorithm (GA), 2020. URL https://github.com/SudhakarKuma/GA-PID
[18] Moler C., Little J. A history of MATLAB. *Proc. ACM Program. Lang.*, 2020, vol. 4, pp. 1–67. doi: 10.1145/3386331
[19] Cao M., Yang J. The Effect of the Approximation Method for Large Time Delay Process on the Performance of IMC-PID Controller. *International Conference on Control, Power, Communication and Computing Technologies* (Kannur, Kerala, India), 2018, pp.73-78. doi:10.1109/iccpcct.2018.8574299
[20] Varga A. Descriptor System Tools (DSTOOLS) User's Guide, *ArXiv*, 2018. https://arxiv.org/abs/1707.071402
[21] Galvão R.K.H., Kienitz K.H., Hadjiloucas S. Conversion of descriptor representations to state-space form: an extension of the shuffle algorithm. *International Journal of Control*, 2017, vol. 91, no. 10, pp. 2199–2213. doi: 10.1080/00207179.2017.1336671
[22] Shampine L.F., Gahinet P. Delay-differential-algebraic equations in control theory. *Applied Numerical Mathematics*, 2004, vol. 56, no. 3–4. pp. 574–588.
[23] Altman Y.M. Accelerating MATLAB Performance: 1001 tips to speed up MATLAB programs. Boca Raton, FL, USA, CRC Press, 2014
[24] Skogestad S. Simple Analytic Rules for Model Reduction and PID Controller Tuning, *Journal of Process Control*, vol. 13, no. 4, 2003, pp. 291-309. doi:10.1016/S0959-1524(02)00062-8
[25] Weiss A.J., Elsherbeni A.Z. Performance of MATLAB and Python for Computational Electromagnetic Problems. *ACES Journal*, 2020, vol. 35, no. 7, pp. 770–777.

About authors.

Stopakevych Andrii
PhD, Associate Professor.
State University of Intelligent Technologies and Telecommunications. Area of scientific interests: complex systems for automatic control of technological processes, software technologies of industrial automation. Ukraine, Odesa.
E-mail: stopakevich@gmail.com

Stopakevych Oleksii
PhD, Associate Professor, Senior Scientist. National Odesa Polytechnic University. Area of scientific interests: system analysis and theory of complex control systems, multivariable systems, cybersecurity problems, software development.
Ukraine, Odesa
E-mail: stopakevich@op.edu.ua