

Method for verifying the accuracy of computational implementations of dynamic system models

Sergii Polozhaenko^{1,*†}, Liudmyla Prokofieva^{2,†}, Andrii Prokofiev^{3,†} and Artem Savieliev^{4,†}

^{1,2,3,5} Odesa Polytechnic National University, 1, Shevchenko ave., Odesa, 65044, Ukraine

Abstract

The relevance of this research stems from the growing need to improve the accuracy of dynamic system models by ensuring reliable computations and real-time error detection. The aim of this work is to improve control over the computation process for numerical solution methods of dynamic equations via algorithmic control. To achieve this, the following tasks are set: to propose criteria for detecting algorithmic errors and to develop a decision-making method to guide the computational process. The research methods involve numerical analysis, interpolation, extrapolation, and the application of quadrature formulas. The results of the work introduce a method with increasing computational accuracy for sequentially determining state functions with higher local error accuracy. The obtained results demonstrate that the interpolation-based method significantly enhances reference value accuracy and reduces noise compared to extrapolation, ensuring reliable model verification.

Keywords

dynamic systems; computational accuracy; numerical solution; mathematical models verification

1. Introduction

A characteristic feature of modern control systems is the widespread use of various types of computing systems that implement modeling and control methods and algorithms. Therefore, the quality of operation of modeling and control systems is largely determined by the characteristics of the computing resources used. The problem of ensuring the quality of operation—in particular, of control systems with a computer in the control loop—is very complex from both a technical and a mathematical standpoint. In addressing this problem, practical experience has brought to the forefront a field of research focused on developing methods and tools that ensure the reliability of computational and information processing operations [1]. In real-world modeling or control systems, computational processes are inevitably accompanied by various types of disturbances and errors. For example, when performing numerical analysis of equations (or other mathematical relationships) describing the dynamics (which constitute the state equations of a dynamic system) on a digital computer, errors in solving these equations (or other relationships) may, among other things, be caused by hardware malfunctions and failures. In this context, disturbances are defined [2] as distortions in the processed and control information caused by self-correcting root causes (incorrect operation of binary hardware elements, interference, power supply voltage fluctuations, etc.). Moreover, most disturbances are random in nature. Therefore, effective real-time monitoring is required to detect and eliminate errors before they propagate through the control loop, making responsiveness one of the key characteristics of error control systems. This paper proposes a mathematical framework that ensures increasing computational accuracy; unlike methods from the

ADT-ISMDDC 2026: Advanced Digital Technologies: Intelligent Systems, Modeling, and Data-Driven Control, September 22–24, 2026, Odesa, Ukraine

* Corresponding author.

† These authors contributed equally.

✉ polozhaenko@op.edu.ua (S. Polozhaenko); prokofieva@op.edu.ua (L. Prokofieva); a.yu.prokofiev@op.edu.ua (A. Prokofiev); savieliev.a.a@op.edu.ua (A. Savieliev)

ORCID 0000-0002-4082-8270 (S. Polozhaenko); 0000-0002-4045-2402 (L. Prokofieva); 0000-0002-4520-4248 (A. Prokofiev); 0000-0002-6949-5959 (A. Savieliev)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

prediction-correction family, it allows for the sequential determination of the values of the sought-after state function of dynamic systems with an increasing order of accuracy of the local error.

2. Literature Review

Currently, both software-based and hardware-based methods for controlling modeling (or control) processes, as well as various combinations thereof, are used [1–3]. Software-based control methods, in turn, can be divided into test-based and software-logic-based methods [4]. Test-based control is designed to detect equipment or software failures [5, 6] at a time when no operational task is being solved on the computing and control device or when test signals are allowed to be input during the task-solving process. In this case, it is necessary to monitor the computational process while the modeling or control system is operating. The primary focus of this monitoring is equipment failures. Software-logic monitoring methods are considered more efficient, as they allow for monitoring during the problem-solving process. A fairly comprehensive overview of existing software-logic control (SLC) methods is provided in a series of papers [7–9]. The main limitation to the application of existing control methods for numerical solutions—in particular, of complex nonlinear differential equations (as the primary mathematical framework describing the dynamic behavior of systems) in modeling and control systems—is the time required to perform the control. In this work, using differential equations as an example, methods are proposed for controlling errors [8, 10–12] caused by disruptions during the numerical solution of these equations (in terms of control theory—dynamic equations) over time, that is, at the same rate as the implementation of the control action in the control loop. In this context, differential equations were chosen for further discussion because they are predominantly used as model elements in the synthesis of control for dynamic systems.

3. Purpose and Objectives of the Study

The purpose of this work is to improve control over the computation process for any methods of numerical solution of dynamic equations by developing algorithmic control methods based on the application of a control algorithm that is executed during the numerical solution of the given (main) problem, but which is simpler than the main problem itself. To achieve this purpose, the following tasks have been set:

- to propose criteria that allow one to conclude whether an error is present (or absent) in the main algorithm for the numerical implementation of the dynamic system's mathematical models, thereby enabling the detection of failures in the operation of the main computational algorithm;
- to develop a decision-making method that determines the presence or absence of a failure and makes a decision regarding the further course of the computational process for performing practical calculations when organizing the control of the computational process.

4. Development of a method for verifying the reliability of computational implementations of dynamic system models

4.1 Determining the possibilities for monitoring algorithms for the numerical solution of problems in the computational implementation of dynamic system models

Numerical methods for solving dynamic equations (or other mathematical relations that form the system of equations of dynamic systems) are a *step-by-step* procedure for obtaining the desired function. In this process, the computational process is naturally divided into time-invariant segments corresponding to the determination of the desired function at a specific integration step.

Furthermore, the computational process is divided within each integration step into time-invariant segments that correspond to the determination of the values of the right-hand side of the system of equations being solved (using the example of a dynamic system modeled as a differential equation)

$$\dot{\mathbf{Y}} = \mathbf{f}(\mathbf{Y}, \mathbf{U}, t), \mathbf{Y}(t_0) = \mathbf{Y}_0 \quad (1)$$

where $\mathbf{Y} = (y_1, y_2, \dots, y_n)^T$ is the vector of state variables; $\mathbf{U} = (u_1, u_2, \dots, u_n)^T$ is the vector of control signals; $\mathbf{f} = (f_1, f_2, \dots, f_n)^T$ is the vector-function; t is the independent time parameter.

Let us consider the capabilities of the numerical solution procedures (1) regarding the organization of control (self-control) of computational processes. To organize control (self-control), reference values are required against which the calculation results are compared. Verified results from previous calculations or results obtained with a low probability of failure can be used as reference values.

For any methods of numerically solving dynamics equations of the form (1), failure control can be organized by comparing the results obtained at the current and subsequent integration steps, respectively. The presence or absence of failures is determined by checking the condition

$$\xi = \rho(y_i, y_{i+1}) \in D, \quad (2)$$

where ξ is a measure of the proximity of the values y_i and y_{i+1} ; D is the domain of valid values for this measure.

To control errors caused by divergences and failures, methods for controlling the systematic error of the numerical solution at the integration stage or specific features of the design of computational algorithms can be used.

In the prediction and correction methods [13–16], based on the difference $\xi = |y_f - y_c|$ between the predicted y_f and the corrected y_c values of the function of interest, conclusions can be drawn regarding the presence of a malfunction in the equipment. However, the existing correlation between the values y_f and y_c undermines the reliability of the control.

Let us illustrate this using the example of the improved Euler–Cauchy method [17, 18]. Suppose the prediction is performed according to the formula:

$$y_{i+1} = y_i + hf_i, \quad (3)$$

where h is the integration step; $f_i = f(\mathbf{Y}, \mathbf{U}, t_i)$, and the correction is performed according to the expression

$$y_{i+1} = y_i + \frac{h}{2}(f_i + \tilde{f}_{i+1}), \quad (4)$$

where $\tilde{f}_{i+1} = f(\tilde{\mathbf{Y}}, \mathbf{U}, t_{i+1})$.

As can be seen from (3) and (4), the function $f(\mathbf{Y}, \mathbf{U}, t_i)$ is computed twice per integration step at the points t_i and t_{i+1} . Since this process typically accounts for the bulk of the computational work, let us assume that a failure occurred at the moment of calculating $f(\mathbf{Y}, \mathbf{U}, t_i)$ and the value of the function f_i was obtained with an error of Δf_i .

Then the value of the function that can be obtained according to formula (3) will be

$$\tilde{y}_{i+1} = y_i + h(f_i + \Delta f_i) = y_i + h f_i + h \Delta f_i, \quad (5)$$

Here, $\Delta y_{i+1} = h \Delta f_i$ is the error in the forecast value.

Let's calculate the value of the function y_{i+1} according to formula (4)

$$y_{i+1} = y_i + \frac{h}{2} \left[(f_i + \Delta f_i) + f_i + \frac{\partial f_i}{\partial y} h_i (f_i + \Delta f_i) + \frac{\partial f_i}{\partial t} h \right], \quad (6)$$

The difference between the predicted (5) and corrected (6) values will be equal to

$$\xi = |\tilde{y}_{i+1} - y_{i+1}| = \frac{h}{2} \left| \Delta f_i \left(1 - \frac{\partial f_i}{\partial y} h \right) - h \left(\frac{\partial f_i}{\partial y} f_i + \frac{\partial f_i}{\partial t} \right) \right|, \quad (7)$$

from which it follows that the value ξ depends on the values of the derivatives $\partial f_i / \partial y$ and $\partial f_i / \partial t$.

In contrast to modified prediction and correction methods, we propose a method *with increasing computational accuracy*, the essence of which lies in sequentially obtaining the value of the function at the integration step with an increasing order of accuracy of the local error (the essence being the error at the integration step). It is clear that for an arbitrary order of accuracy, r , we can write:

$$y_{i+1}^{(r)} = y_i^{(r)} + \frac{h}{2} [f(y_i^{(r)}, u_i, t_i) + f(y_{i+1}, u_{i+1}, t_{i+1})], \quad (8)$$

$$y_{i+1}^{(r-1)} = y_i^{(r)} + h f(y_i^{(r)}, u_i, t_i), \quad (9)$$

where the superscript indicates the order of the local error of the approximate value.

By comparing the values of $y_{i+1}^{(r-1)}$ and $y_{i+1}^{(r)}$, one can draw conclusions about the local error in the solution. This same technique can also be used to detect errors caused by failures. Thus, if there was no failure in the previous step, an error in determining $y_{i+1}^{(r-1)}$ can occur only if expression (9) holds.

Given the computational complexity—and, consequently, the probability of failure—for procedures (8) and (9), this probability is lower than for other *single-step* methods of the same order of accuracy with respect to the step size h . Additionally, since the probability of failure when computing $y_{i+1}^{(r-1)}$ is significantly lower than the probability of failure when computing the function $f(y, u, t)$, when the latter has a complex form, it is advisable (from the perspective of error control) to organize the control process using a method *with increasing second-order computational accuracy*, i.e., $(r-1)=2$. The described process can be easily implemented, for example, in the second-order accuracy method $(r-1)=2$ using expressions (8) and (9). In this case, when calculating $y_{i+1}^{(2)}$, the function $f(y, u, t)$ is not computed; instead, the value $f(y_{i+1}^{(2)}, u_{i+1}, t_{i+1})$ obtained in the previous integration step is used, which significantly reduces the probability of an error occurring when calculating the value $y_{i+1}^{(2)}$. An algorithm that implements the method *with increasing computational accuracy* must meet the following requirements:

1. The calculation process at each integration step must begin with determining the function f at the point t_{i+1} . In this case, the function $f(y, u, t_{i+1})$ is not computed; instead, the value of the function f obtained and verified during the calculation of the most accurate solution to

system (1) at the previous integration step is used. The calculation formula for obtaining the first approximate value $y(t_i + h)$ with a local error of $O(h^2)$ is given by

$$y_{i+1}^{(2)} = y_i^{(r)} + h f(y_i^{(r-1)}, u_i, t_i), \quad (10)$$

Where r is the order of the local error of the approximate solution; $f(y_i^{(r-1)}, u_i, t_i)$ is the estimate of the derivative used in determining $y_i^{(r)}$.

If we disregard the single-step condition and incorporate information about the problem being solved over an interval longer than the integration step h , we can increase the order of accuracy of the first approximate value of the function $y(t_i + h)$. When constructing a *multi-step* procedure to obtain the first approximation of the function at the point t_{i+1} , one can use estimates of the derivative of the function $\frac{\partial f(y + \Delta y, u, t_{i+1})}{\partial y}$ obtained within the step $[t_{i-1}, t_i]$.

The correctness of the calculations according to (8) is verified by checking the condition

$$\xi \in D, \quad (11)$$

where $\xi = \xi(y_{i+1}^{(2)}, y_{i+1}^{(3)})$ – the measure of computational accuracy; D – the permissible range of values for this measure.

If condition (10) is not satisfied, the incorrect solution is corrected by recalculating the values of $y_{i+1}^{(3)}$, or by replacing $y_{i+1}^{(3)}$ with $y_{i+1}^{(2)}$. In the latter case, an error will be introduced, which is determined by the difference between the residual terms in the corresponding formulas and is a small quantity of the order of $O(h^2)$. In this case, the correction time will be a random variable with a mathematical expectation of

$$M[T_{\text{кор}}] = (t_{\text{наб}} + t_{\text{оп.к}}) \sum_{m=0}^H C_H^m p^m q^{H-m}, \quad (12)$$

where t_{comp} – the time required to compute the next approximation of the function $y(t)$ at point t_{i+1} ; t_{check} – the time required for the check operation; $H = kN$ (k is the number of divisions required to obtain a solution with increasing accuracy, N is the number of integration steps); p is the probability of failure when computing the next approximation; $q = 1 - p$.

The expected value of the total solution time for N calculation steps is

$$M[T] = H(t_{\text{наб}} + t_{\text{оп.к}}) + M[T_{\text{кор}}], \quad (13)$$

If the correction is performed by replacing the incorrect value of $y_{i+1}^{(r)}$ with the value of this function at the point $t_i + 1$ that is closest in accuracy, then an error will be introduced into the solution

$$\xi = y_{i+1}^{(l+1)} - y_{i+1}^{(l)} = \eta_{i+1}^{(l+1)} - \eta_{i+1}^{(l)}, \quad (14)$$

where $y_{i+1}^{(l)}$ and $y_{i+1}^{(l+1)}$ are successive approximations of the function $y(t)$ with local error orders l and $(l+1)$, respectively; $\eta_{i+1}^{(l)}$ and $\eta_{i+1}^{(l+1)}$ are small quantities with smallness orders h^l and h^{l+1} respectively.

2. The number of partitions k (i.e., the number of values $y_{i+1}^{(r)}$ obtained with increasing accuracy on the interval $[t_i, t_{i+1}]$) can be chosen by minimizing (11) subject to the constraint on the value of $\xi = (y_{i+1}^{(l+1)} - y_{i+1}^{(l)})$, or conversely, by minimizing ξ subject to the constraint on $M[T]$. However, in practice, $t_{\text{HAG}} \gg t_{\text{OH.K}}$, and therefore, when designing an algorithm for the method with increasing accuracy, it is advisable to choose the maximum number of partitions k that is possible for the method of the selected accuracy order r .

Let us estimate the leading term of the error using Runge's rule [18, 19]. In this case, the application of this rule boils down to the following. For a given point t on the interval $[t_0, t_N]$, using the k th-order accuracy method, we can sequentially obtain the approximate values of the function y_{h_1} and y_{h_2} at steps $h_1 \neq h_2$, respectively. Let us define the quantity:

$$\xi(t) = \frac{y_{h_1} - y_{h_2}}{h_2^k - h_1^k}. \quad (15)$$

Then, taking into account the magnitude of the leading term of the error for each of the values $h=h_1$ and $h=h_2$

$$\xi_1 = h_1^k \frac{y_{h_1} - y_{h_2}}{h_2^k - h_1^k} \text{ and } \xi_2 = h_2^k \frac{y_{h_1} - y_{h_2}}{h_2^k - h_1^k}, \quad (16)$$

we can draw conclusions regarding the accuracy of the calculations.

With a uniform step size for the interval $[t_0, t_N]$, that is, if $h_1 = h$ and $h_2 = 2h$, the expression for the leading term of the error takes the following form:

$$\xi = y - y_h \approx \frac{y_h - y_{2h}}{2^k - 1}, \quad (17)$$

Depending on the calculated value of ξ , to achieve the specified accuracy, the step size h must either be decreased or increased (although in the latter case, the calculated accuracy will exceed the specified accuracy, the computation time may increase significantly, and the convergence of the solution may deteriorate).

An error caused by a failure and exceeding the permissible value of the leading term of the error estimate ξ_{perm} will be detected and eliminated by recalculating with a reduced step size h . The Runge rule [15] requires additional computations, which significantly increases the time required and the probability of *first-kind* errors (also known as α -errors, i.e., a false-rejection conclusion is drawn, leading to the rejection of a true hypothesis), and therefore the application of this rule in control systems is problematic (or even unacceptable). It should also be noted that the use of variable step size in calculations – that is, $h \neq \text{const}$ – in control systems is further complicated by the presence of control signals, which must be either pre-stored or calculated using interpolation formulas when the step size h changes, introducing an additional error into the solution.

Therefore, when evaluating computational errors for control systems, we will demonstrate the possibility of applying *quadrature formulas*. We will use the values y_i and $i = \overline{1, N}$ obtained using the main algorithm, applying the following quadrature formula

$$y(t_i) = y_0^* + \int_{t_0}^{t_N} f(y^*, u, t) dt, \quad (18)$$

where y^* is the exact value of the function, and then we can obtain

$$\bar{y}_N = y_0^* + \sum_{i=0}^N A_i f(y_i, u_i, t_i), \quad (19)$$

where A_i are constant coefficients.

The conclusion regarding the presence of an error is made by checking the condition:

$$\xi = |y_N - \bar{y}_N| \leq \xi_{\text{доп}}, \quad (20)$$

where $\xi_{\text{доп}}$ is the permissible value of the error ξ .

Below, we propose algorithmic control methods [17] that can be applied to control the computation process for any method of numerical solution of dynamic equations. In addition, we will consider control based on the application of an additional (verification) algorithm at each step of the numerical solution of the dynamic equation.

4.2 Extrapolation-Based Control Method

We propose a simplified verification method, the problem of constructing which we formulate as follows: for a given class of problems Q , obtain a procedure and a solution algorithm A_k such that, when applied to solve problem, the result of their execution coincides Y , with an accuracy of ε , with the result obtained using the reference algorithm A_b . In this case, the time and computational costs (in particular, machine memory) required to implement the algorithm A_k must remain within specified limits.

When computing using the main algorithm A_b for the class of dynamic object modeling (or control) problems Q , most of the time is spent computing the right-hand side of system (1). The computation time for the A_k algorithm can be significantly reduced compared to the A_b algorithm if it uses the values of the function $f(Y, U, t)$ obtained using the A_b algorithm. Since the computational process according to algorithm A_b is also subject to failures, omitting the computation of the function $f(Y, U, t)$ will significantly reduce the probability of first-order errors – α -errors. In this case, it is possible to improve the reliability of control at the step of the computational process being controlled by using, in the control procedure (control algorithm), information that has already been verified in previous steps – that is, by applying the concept of *extrapolation* for control. Then, the control of each component of the vector equation (1) using the extrapolation method can be performed separately; therefore, we will consider the construction of the control procedure for the method (and the corresponding algorithm) for a specific component y of the vector Y , without losing the generality of the exposition.

Given the above, the control value of the function at the $(i+1)$ -th integration step under control can be determined by the formula

$$\bar{y}_{i+1} = \sum_{p=0}^{p_i} a_p y_{p-1} + h \sum_{s=1}^{s_i} b_s f(Y, U, t_{i-s}), \quad (21)$$

where $h = t_{i+1} - t_i$ – the integration step; a_p, b_s – coefficients.

Equation (21) uses only the values of the function and its derivatives that have been verified in previous integration steps. The upper bounds of the sums p_1 and s_1 are chosen based on the required order of accuracy of the algorithm under consideration. It is advisable to choose the coefficients a_p , b_s based on the condition of minimum local error (i.e., the equality of the residual term $r_i = 0$ between the exact solution and the numerical solution (21)).

It is known that in many methods (and their corresponding algorithms) for solving dynamic equations (this primarily applies to cases where dynamic processes are described by differential equations), estimates of the components of the result for the current steps of the numerical solution are determined (in the cases of differential equations mentioned, these are the derivatives between integration nodes). Since these estimates provide information, verified at the previous step, regarding the behavior of the solution at a point that is closer to the controlled value of the function than the value $f(\mathbf{Y}, \mathbf{U}, t_{i-s})$ in (21) when $s_1=1$, it is advisable to take this information into account during extrapolation. If, during extrapolation, we take into account a single estimate of the specified components of the solution (for example, the derivative for differential equations of dynamics) obtained using the main algorithm, then one term will be added to expression (21):

$$\bar{y}_{i+1} = \sum_{p=0}^{p_1} a_p y_{p-1} + h \left[\sum_{s=1}^{s_1} b_s f(\mathbf{Y}, \mathbf{U}, t_{i-s}) + b_0 f(\mathbf{Y}, \mathbf{U}, t_{i+1}) \right], \quad (22)$$

where $f(\mathbf{Y}, \mathbf{U}, t_{i-s})$ is an estimate of a component (e.g., a derivative), and b_0 is a coefficient.

It is evident that, in the general case, any information regarding the solution obtained using the main algorithm A_b and verified in the previous steps of the numerical procedure can be taken into account for extrapolation. Thus, in the numerical procedure of the “prediction-correction” method, an estimate of the solution components at the point t_{i+1} should be made, which can be used when constructing a verification formula to check the solution in subsequent steps.

In this case, the verification (check) formula will take the form:

$$\bar{y}_{i+1} = \sum_{p=0}^{p_1} a_p y_{p-1} + h \left[\sum_{s=1}^{s_1} b_s f(\mathbf{Y}, \mathbf{U}, t_{i-s}) + b_0 f(\tilde{\mathbf{Y}}, \mathbf{U}, t_i) \right], \quad (23)$$

The value $f(\tilde{\mathbf{Y}}, \mathbf{U}, t_i)$, which is used in the verification formula, represents an estimate of the solution component at the current step of the numerical procedure, obtained using the main algorithm A_b at the previous step. If an iterative process is used in the “prediction-correction” method, then the control formula uses the estimate of the derivative obtained at the last iteration for the previous step.

Let us consider the derivation of the coefficients a_p and b_s for the control formula (22). Since the control formula (22) is essentially a multistep extrapolation method for solving a boundary value problem, the exact solution $y=y(t)$ – provided that an “exact” solution was also obtained in the previous calculation steps—will be determined as follows: yf

$$\begin{aligned} y(t_{i+1}) = & \sum_{p=0}^{p_1} a_p y(t_{i-p}) + h \sum_{s=1}^{s_1} b_s f[Y(t_{i-s}), u_{i-s}, t_{i-s}] + \\ & + h b_0 f[Y(t_{i-1}), u_{i-1}, t_{i-1}] + r_i \end{aligned} \quad (24)$$

where r_i is the residual term.

By choosing the coefficients $a_p, p = \overline{0, p_1}$, and $b_0; b_s, s = \overline{1, s_1}$, one should ensure that $r_{(i)}$ is small; for simplicity, one may assume that $r_i \rightarrow 0$ (or even $r_i \equiv 0$, if the solution $y=y(t)$ can be represented as a high-degree algebraic polynomial). Then, by specifying the initial values a_0, b_0 , and b_1 , the first approximation for formula (24) is determined, and the condition

$$y(t) = y(t_{i+1}) + r_i \Big|_{r_i \rightarrow 0}, \quad (25)$$

and, if (25) is not satisfied, then by sequentially iterating over the values of the coefficients $a_p, p = \overline{1, p_1}$ and $b_s, s = \overline{2, s_1}$, the above condition is satisfied (to simplify the iteration, the coefficient $a_{(p)}$ is varied first, i.e., $a_p = \text{var} \Big|_{p=1, p_1}$, $b_s = \text{const}$; if (24) is not satisfied, a further search is performed over the values $a_{p_1} = \text{const}$, $b_s = \text{var} \Big|_{s=2, s_1}$. As a result, the values of a_p and b_s that ensure the smallest value of r_i are selected.

If the solution to equation (1) is a polynomial of degree q , then to obtain an “exact” solution using expression (24), it is necessary to choose the coefficients $a_p, p = \overline{0, p_1}$, and $b_0; b_s, s = \overline{1, s_1}$ such that $r_i \equiv 0$, where $y(t)=1, t, \dots, t^q$.

To do this, we must solve the following system of algebraic equations:

$$(t_i + h)^m = \sum_{p=0}^{p_1} a_p (t_i - ph)^m + h \left\{ \sum_{\substack{s=1 \\ s \neq l}}^{s_1} b_s (t_i - sh)^m + b_0 (t_i - lh)^m \right\}', \quad m = \overline{0, q} \quad (26)$$

The condition $s=1$ is introduced so that, when $s = 1$, the same term is not repeated twice.

After shifting the origin to the point t_i and dividing both sides of (26) by h^m , we obtain:

$$1 = \sum_{p=0}^{p_1} a_p (-p)^m + m \left[\sum_{\substack{s=1 \\ s \neq l}}^{s_1} b_s (-s)^{m-1} + b_0 (l)^{m-1} \right] m = \overline{0, q}, \quad (27)$$

Under condition (24), the value of l varies for different numerical methods on the interval $0 \leq l \leq 1$ depending on which estimate of the components of the solution (1), obtained by the main algorithm A_0 , can be used in the control algorithm A_k . If, for example, the Adams extrapolation method is chosen as the algorithm A_0 , then $l=0$ and expression (27) reduces to (24).

As an example, we present a control formula designed to monitor the results obtained using algorithm (23), where the discretization of the nodal points is performed according to the expression $t_{i+1} = t_i + \lambda h$, where $\lambda = 2/3$:

$$\bar{y}_{i+1} = y_i + h y_{i-1+\lambda}, \quad (28)$$

For the n -th-order matrix equation (1), formula (22) takes the form:

$$\bar{y}_{i+1} = \sum_{p=0}^{p_1} a_{j,p} y_{j,p-1} + h \left[\sum_{s=1}^{s_1} b_{j,s} f_j(\mathbf{Y}, U, t_{i-s}) + b_{j,0} f_j(\mathbf{Y}, U, t_{i+l}) \right], \quad j = \overline{1, n}$$

(29)

In this work, a numerical experiment was conducted to determine the qualitative characteristics of the proposed extrapolation control method.

For the applied problem of studying aircraft flight dynamics [17], Fig. 1 and Fig. 2, respectively, show the dependence of the expected value and the root mean square deviation of the quantity

$$\xi = \sum_{j=1}^n \frac{|y_{j,i} - \bar{y}_{j,i}|}{\max_t |y_j|}, \quad j = \overline{1,6}, \quad (30)$$

on the step size h during the numerical solution of equation (1). The curves in the figures were obtained by using expression (28) for control with different values of the parameter λ .

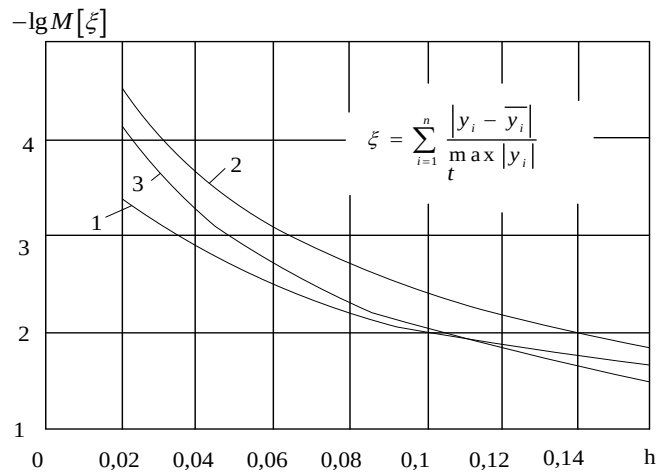


Figure 1: Dependence of the mathematical expectation $M[\xi]$ of the variable ξ on the step size h

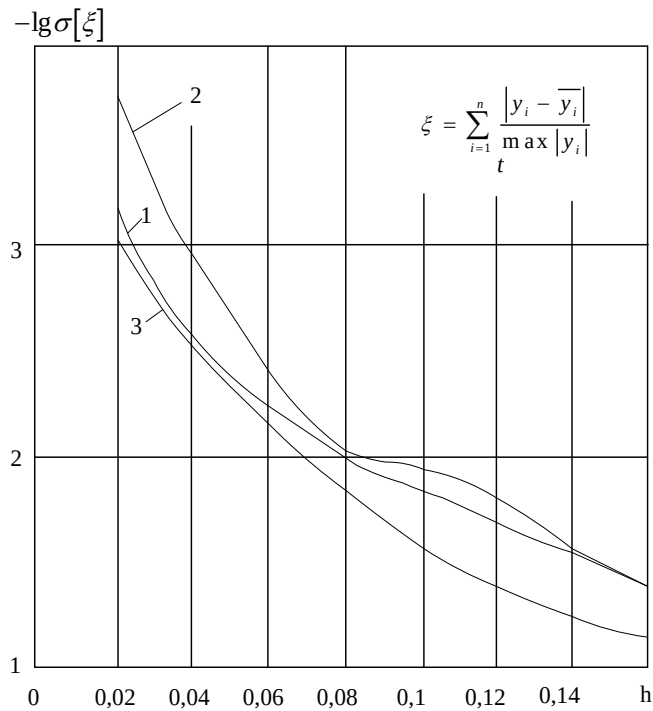


Figure 2: Dependence of the mean square error $\sigma[\xi]$ of the variable ξ on the step size h

4.3 Interpolation-Based Control Method

When using the extrapolation control method, it is possible to correct the solution not by repeating the calculations, but by replacing y_{i+1} with $\bar{y}_{i+1}$, which is essential for modeling and control on a real-time (or accelerated) time scale. However, the accuracy of the control in this case depends on the accuracy of the solution extrapolation using Equation (22) and may prove unsatisfactory.

For this reason, from the perspective of improving control accuracy, the method based on the concept of *interpolation* should be considered superior to the extrapolation method [17]. Note that, unlike extrapolation formulas, interpolation control formulas account for *the change in the solution at the controlled calculation step*, which leads to a *reduction in the difference* between the values y and $\bar{y}$, which result from the application of algorithms A_0 and A_k , respectively. The accuracy of the interpolation can be improved by incorporating the results of calculations using the main algorithm A_b up to the calculation step being controlled. To increase *the sensitivity* of error control caused by disruptions, one should use a control interpolation formula to determine the function value obtained at the calculation step closest to the controlled calculation step.

In light of the above, we propose a control-based multi-step interpolation formula that can be used to monitor the computation process at the $(i+1)$ -th step:

$$\bar{y}_i = \sum_{p=-1}^{p_1} a_p y_{i-p} + h \sum_{s=-1}^{s_1} b_s f(\mathbf{Y}, \mathbf{U}, t_{i-s}), \quad (31)$$

where a_p and b_s are real coefficients; $p \neq 0$ (this is a degenerate case in which $a_0=1$ and all other coefficients are zero, and the identity $\bar{y}_i \equiv y_i$ holds).

As in the case of the extrapolation formula (22) in (29), the upper limits of summation must be chosen depending on the required order of control accuracy, and the coefficients a_p and b_s must be chosen based on the accuracy of the representation of polynomials of the corresponding degree.

Thus, expression (29) defines the procedure for *the interpolation method of calculation control* [12].

The fundamental difference between the interpolation expression (29) and the extrapolation expression (22) as control algorithms lies in the following. In the case of extrapolation formulas, the directly controlled quantity is the function value obtained using the main algorithm A_b at the controlled calculation step, and the control value is the result of applying expression (22). In the case of interpolation control formulas, the value $\bar{y}_i$ is calculated using expression (21), and the reference value is the function value obtained using the main algorithm A_b at the previous $(i-p)$ -th calculation step; that is, *indirect* control is essentially performed. The error in calculating the value $\bar{y}_{i+1}$ using the main algorithm A_b is transformed into the error $\Delta\bar{y}_i$ in determining $\bar{y}_i$ according to formula (29). The deviation in $\Delta\bar{y}_i$ in this case, within the framework of linear accuracy theory, will be defined as

$$\Delta\bar{y}_i = L\eta, \quad (32)$$

where $L = \partial\bar{y}_i / \partial y_{i+1}$.

When applied to expression (29), L is determined by the formula

$$L = a_* - b_* h \frac{\partial f(\mathbf{Y}, \mathbf{U}, t_{i+1})}{\partial y_{i+1}}, \quad (33)$$

where a_* and b_* are coefficients, with y_{i+1} in (29).

Fig. 3 and Fig. 4 show the dependencies of the expected value and the standard deviation of the quantity

$$\xi_i = \left\{ \sum_{j=1}^p \left[(y_{i,j} - \bar{y}_{i,j}) / \left(\max_i |y_j| \right) \right]^2 \right\}^{1/2}, \quad (34)$$

on the discretization step size h in Equation (1), which describes the flight dynamics of an aircraft [17]. Curves 1 and 2 were obtained using first- and second-order interpolation formulas, respectively, at the $(i+1)$ -th calculation step according to the basic algorithm A_b :

$$y_i = y_{i+1} - h \dot{y}_{i+1}, \quad (35)$$

$$\bar{y}_i = y_{i+1} - 0.5h(y_{i+1} + y_i), \quad (36)$$

For comparison, curve 3 is shown, corresponding to the use of the extrapolation formula

$$\bar{y}_{i+1} = y_i + h(2.25y_{i-1} - 1.25y_{i-2}), \quad (37)$$

It can be seen that as the discretization step size h increases, the accuracy of the interpolation control method increases compared to the extrapolation method.

The interpolation control method allows for increased accuracy in obtaining reference values. However, when organizing the control of a computational process, it should be borne in mind that when using the interpolation control method, it is not possible to replace an erroneous value obtained using the main algorithm A_b with a value obtained using the control algorithm A_k (which is possible with the extrapolation method).

The paper [16] notes that the control algorithm A_k takes approximately 5–7 times less time than the main algorithm A_b .

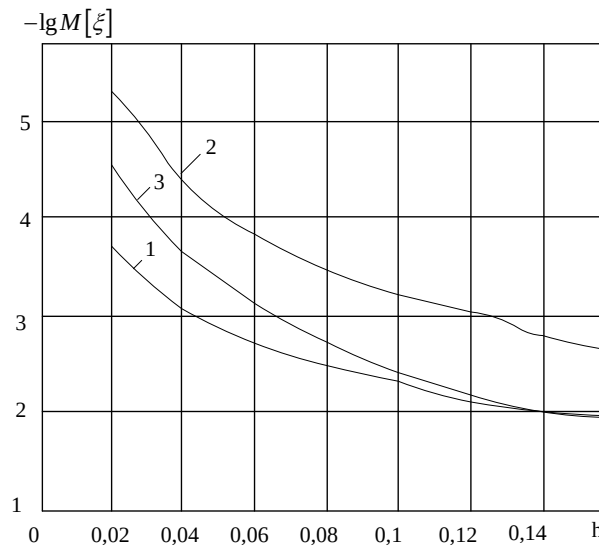


Figure 3: The dependence of the expected value $M[\xi]$ of a value ξ on the step size h

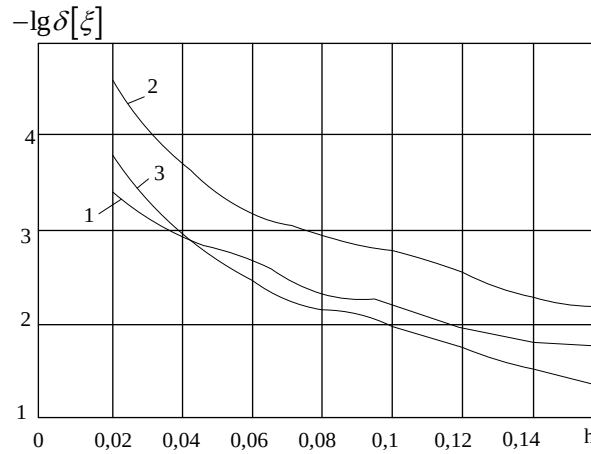


Figure 4: The dependence of the root mean square deviation $\sigma[\xi]$ of a value ξ on the step size h

The test cases are designed such that the models of dynamic objects are described by well-known differential and integral equations, the numerical implementation of which is carried out using standard algorithms and computational procedures common to the MATLAB/SIMULINK packages. The software, algorithmic, and numerical implementation of more complex models of dynamic objects is described in [19].

Conclusion

The possibilities for verifying the accuracy of the computational implementation of dynamic systems' mathematical models during their modeling and control processes have been identified. A method with increasing computational accuracy is proposed, which, unlike the basic and modified methods of prediction and correction, allows for the sequential (in particular, at individual integration steps) determination of the values of the sought state function of dynamic systems with an increasing order of accuracy of the local error. Moreover, the order of accuracy can be specified in advance, prior to the execution of the main stages of the numerical implementation of the mathematical models for a dynamic system.

Algorithmic control methods are proposed that can be applied to control the computation process for any method of numerically solving equations of dynamics, namely: extrapolation and interpolation control methods. These methods are based on the use of a control algorithm that is executed during the numerical solution of the given (main) problem, but which is simpler than the main problem itself. The difference between these methods lies in the fact that the extrapolation method allows for an increase in the accuracy of the calculations by using, within the control algorithm, information that has already been verified in previous steps of the main algorithm, while the interpolation method enables the implementation of implicit control, which reduces the range of deviation of the controlled variable from the reference value—that is, it reduces the noise level.

Criteria are proposed that allow one to conclude whether an error is present (or absent) in the main algorithm of the numerical implementation of a dynamic system's mathematical models, thereby enabling the detection of malfunctions in the main computational algorithm. This has made it possible to develop adaptive methods for verifying the accuracy of the numerical implementation of dynamic systems mathematical models; these methods ensure the determination of the control algorithm's parameters based on the condition of minimizing a selected measure of proximity between the results obtained using the control algorithm and the main algorithm for the numerical implementation of dynamic systems mathematical models.

To perform practical calculations when organizing the control of the computation process, a decision-making method is proposed whereby the presence or absence of a failure is determined and

a decision is made regarding the further course of the computational process. The method is based on taking into account the degree of divergence (or discrepancy) between the values obtained using the main and control algorithms, as well as setting a tolerance for this degree of divergence and subsequently selecting a strategy for continuing the computational process.

Declaration on Generative AI

While preparing this paper, the authors used Gemini 3.5 Flash to check grammar and to format the reference list according to the journal's requirements. After using these services, the authors reviewed and edited the content as necessary and assume full responsibility for the content of the publication.

References

- [1] R. Castro, M. Bergonzi, E. Pecker Marcosig, J. Fernández, E. Kofman, Discrete-event simulation of continuous-time systems: Evolution and state of the art of quantized state system methods, *Simulation* 100 (2024) 613–638. doi:10.1177/00375497241230985.
- [2] N. Nouaime, B. Després, M.A. Puscas, C. Fiorini, Stability of a continuous/discrete sensitivity model for the Navier–Stokes equations, *International Journal for Numerical Methods in Fluids* 96 (2024) 1883–1909. doi:10.1002/fld.5324.
- [3] R. Gani, M.R. Eden, J.D. Perkins, et al., Models, modeling, and model-based systems in the era of computers, machine learning, and AI, *Computers & Chemical Engineering* 194 (2025) 108957. doi:10.1016/j.compchemeng.2024.108957.
- [4] M. Yavuz, I. Dassios, Significance of Mathematical Modeling and Control in Real-World Problems: New Developments and Applications, *Mathematical and Computational Applications* 29 (2024) 82. doi:10.3390/mca29050082.
- [5] F. Meng, X. Shen, et al., Emerging Methodologies in Stability and Optimization Problems of Learning-Based Nonlinear Model Predictive Control: A Survey, *International Journal of Circuit Theory and Applications* 50 (2022) 3134–3170. doi:10.1002/cta.3370.
- [6] P. Yu, H. Wan, B. Zhang, Q. Wu, B. Zhao, C. Xu, S. Yang, Review on System Identification, Control, and Optimization Based on Artificial Intelligence, *Mathematics* 13 (2025) 952. doi:10.3390/math13060952.
- [7] Y. Lu, J.-B. Liu, et al., Approximate solutions to several classes of Volterra and Fredholm integral equations using a neural network algorithm based on sine-cosine basis functions and an extreme learning machine, *Frontiers in Computational Neuroscience* 17 (2023) 1120516. doi:10.3389/fncom.2023.1120516.
- [8] S.M. Sivalingam, P. Kumar, V. Govindaraj, A neural networks-based numerical method for generalized Caputo-type fractional differential equations, *Mathematics and Computers in Simulation* 213 (2023) 302–323. doi:10.1016/j.matcom.2023.06.012.
- [9] M. Saleh, E. Kovács, Á. Nagy, New stable, explicit, second-order hopscotch methods for diffusion-type problems, *Mathematics and Computers in Simulation* 208 (2023) 301–325. doi:10.1016/j.matcom.2023.01.029.

- [10] K. Taneja, K. Deswal, D. Kumar, A robust higher-order numerical technique with graded and harmonic meshes for the time-fractional diffusion–advection–reaction equation, *Mathematics and Computers in Simulation* 213 (2023) 348–373. doi:10.1016/j.matcom.2023.06.010.
- [11] X. Yang, Z. Yang, C. Zhang, Numerical analysis of the linearly implicit Euler method with a truncated Wiener process for the stochastic SIR model, *Mathematics and Computers in Simulation* 208 (2023) 1–14. doi:10.1016/j.matcom.2023.01.010.
- [12] V. Kumar, H. Pham (Eds.), *Predictive Analytics in System Reliability*, Springer Series in Reliability Engineering, Springer (2022). doi:10.1007/978-3-031-05347-4.
- [13] J. Wang, C. Zhang, Reliability model of open source software considering fault introduction with generalized inflection S-shaped distribution, *SN Applied Sciences* 4 (2022) 244. doi:10.1007/s42452-022-05125-6.
- [14] S. Bernardi, A. Gómez, J. Merseguer, D. Perez-Palacin, J.I. Requeno, DICE simulation: a tool for software performance assessment at the design stage, *Automated Software Engineering* 29 (2022) 36. doi:10.1007/s10515-022-00335-z.
- [15] J. Cabot, A. Vallecillo, Modeling should be an independent scientific discipline, *Software and Systems Modeling* 21 (2022) 2101–2107. doi:10.1007/s10270-022-01035-8.
- [16] A.Yu. Prokofiev, Precision calibration method in the implementation of mathematical models of dynamic systems in modelling and control problems, *Mathematical and Computer Modelling. Series: Technical Sciences: Collection of Scientific Works* 23 (2022) 116–129. doi:10.326226/2308-5916.2022-23.116-129.
- [17] S.A. Polozhaenko, A.Yu. Prokofiev, Methods for controlling the reliability of computational implementation of dynamic systems models based on the use of control algorithms, *Bulletin of the Kherson National Technical University* 1 (92), Part 2 (2025) 185–193. doi:10.35546/kntu2078-4481.2025.1.2.25.
- [18] O.O. Fomin, V.O. Speransky, A.A. Verlan, O.V. Tataryn, A.M. Chmelevskyi, Long Short-Term Memory-based reference model method for nonlinear dynamic system identification, *Herald of Advanced Information Technology* 9 (2026) 307–320. doi:10.15276/hait.09.2026.20.
- [19] S.A. Polozhaenko, A.Yu. Prokofiev, Parameters of the Runge-Kutta method with different orders of accuracy when integrating dynamics equations in modeling problems of non-stationary systems, *Computer Science and Mathematical Methods in Modelling* 14 (2024) 85–96. doi:10.15276/imms.v14.no1-2.85.