TASHKENT UNIVERSITY OF
INFORMATION TECHNOLOGIES
NAMED AFTER MUHAMMAD AL-KHWARIZMI

MUHAMMAD AL-XORAZMIY NOMIDAGI
TOSHKENT AXBOROT TEXNOLOGIYALARI
UNIVERSITETI

BULLETIN OF TUIT:

MANAGEMENT AND COMMUNICATION TECHNOLOGIES



ISSN:2181-1083 DOI: Date:2025-12-02
Sagatov Miraziz Vorisovich № 3s(1)2025

UDC 517.946

Modeling inhomogeneous dynamic systems described by integral-differential equations

Sagatov Miraziz Vorisovich
Information technologies
Tashkent state technical university
named after Islam Karimov
Tashkent, Uzbekistan
informtgtu@mail.ru

Abstract. Models in the form of ordinary differential equations traditionally have a wide range of applications. These models are widely applicable, including various mechanical systems, control systems, moving objects, and so on. Integro-differential models of dynamic systems occupy an intermediate position between differential and integral models. Their analysis remains an under-researched area of mathematical modeling. Thus, identifying classes of problems effectively solved using integro-differential equations, methods for generating nonlinear integro-differential dynamic models, and their numerical implementation remain poorly understood. As system dynamics problems become more complex and the class of dynamic objects studied expands, the need for further development and refinement of mathematical modeling methods becomes evident. Given this, it is often necessary and useful to employ integro-differential equations as models equivalent to other possible types of models. Integro-differential equations enable the use of a wide variety of numerical model implementation methods, both for adapting algorithms to a narrow class of problems and for increasing the model's generality. Algorithms for implementing direct and iterative numerical methods for modeling heterogeneous dynamic systems described by integro-differential equations are presented.

Keywords: *integro-differential equations, modeling of inhomogeneous dynamic systems, direct and iterative numerical methods.*

1. INTRODUCTION

Existing scientific advances in the field of mathematical modeling of dynamic systems are primarily based on the use of differential equations. Numerous computer modeling methods and tools have been developed and are actively used based on this approach, but they have a number of limitations. These include issues of ensuring the stability of the computational process in the presence of significant levels of high-frequency noise in the input data, the widespread manifestation of the Gibbs effect when modeling objects with distributed parameters, and the limited use of iterative algorithms. To address these issues, it is necessary to develop new methods for constructing and numerically implementing mathematical models in addition to existing ones. An effective step in this direction is the use of integro-differential equations. [1-6]

Integro-differential equations are a powerful mathematical tool for describing heterogeneous dynamic systems containing both lumped and distributed parameters. The effectiveness of these dynamic models in applied research directly depends on the effectiveness of the software implementing the models. When developing application programs, it is necessary to consider both the characteristics of the equations being solved and the characteristics of the objects being modeled, such as control systems. These features include: heterogeneity in the methods for discretizing the differential and integral parts of equations, the need for precision compatibility of the component parts of modeling algorithms, and performance requirements corresponding to control modes. These circumstances necessitate the development of a set of modeling algorithms and

programs that provide the necessary alternatives, taking into account the varying levels of complexity and the diversity of modes of the modeled objects. In particular, for the development of specific tools intended for use during the operation of the object, preference should be given to direct numerical methods. For solving laboratory research and design problems, the use of iterative numerical methods can be considered. [7-12]

2. ALGORITHMS FOR IMPLEMENTING DIRECT NUMERICAL METHODS

Direct numerical methods involve reducing the original integro-differential equations to simple computational operations by approximating operators or approximating the desired solutions (projection methods), or by combining both approaches. We will consider the derivation of quadrature-difference algorithms based on the approximation of differential and integral operators. In the most general case, the Cauchy problem is solved for a system of n first-order nonlinear integro-differential equations.

$$\begin{cases} \frac{du_1}{dx} = F_1\left(x, u_1, u_2, \dots, u_m, \int_a^x K_1(x, s, u_1, u_2, \dots, u_m)ds\right), \\ \frac{du_2}{dx} = F_2\left(x, u_1, u_2, \dots, u_m, \int_a^x K_2(x, s, u_1, u_2, \dots, u_m)ds\right), \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \frac{du_n}{dx} = F_n\left(x, u_1, u_2, \dots, u_m, \int_a^x K_n(x, s, u_1, u_2, \dots, u_m)ds\right), n \geq m, \end{cases} \quad (1)$$

with initial conditions $u_i(0) = c_i, \overline{i = 0, n-1}$.

To directly solve system (1), one can construct a family of quadrature-difference methods based on a combination of difference methods known from the theory of differential equations and quadrature methods of approximating the integral operator used in the numerical solution of integral equations.

Difference methods consist of calculating the approximate solution $U_{k+1} = (u_1(x_{k+1}), \dots, u_n(x_{k+1}))$ at the point x_{k+1} based on the values of the solution at the previous points $x_{k-j+1}, \dots, x_k$. The general formula for such methods is as follows [13]

$$U_{k+1} = \sum_{i=1}^{j} \alpha_i U_{k+1-j} + h\sum_{i=0}^{j} \beta_i f_{k+1-i}, \quad (2)$$

where $f = (F_1, \dots, F_n)$; α_i, β_i – some coefficients. If *j=1*, then the method is single-step, and if *j>1*, it is multi-step. In this case, the integral operator is replaced by a finite sum based on various quadrature formulas for calculating the definite integral, which in the general case have the form

$$\int_a^b \Psi(x)dx = h\sum_{i=0}^{k} A_i \Psi(x_k) + R[\Psi], \quad (3)$$

where A_j – coefficients of the quadrature formula, $R[\Psi]$- remainder term.

To ensure convergence, it is necessary [5] that for all h the inequality be satisfied

$$\left|\int_a^{kh} \Psi(x)dx - \sum_{i=0}^{k} A_i \Psi(ih)\right| \leq C\omega(\Psi; h) \quad (4)$$

where $\omega(\Psi; h)$ - modulus of continuity of a function

$$\omega(\Psi; h) = sup|\Psi(x_1) - \Psi(x_2)|, x_1, x_2 \epsilon [a, b], |x_1 - x_2| < h$$

A sufficient condition for satisfying inequality (4) is the use of quadrature formulas, the coefficients $w_i = hA_i$ of which are Riemann sums [5]. The Riemann sum for the integral $\int_a^b \Psi(x)dx$ is a sum of the form

$$S(P, Q; \Psi) = \sum_{i=0}^{k} (v_{i+1} - v_i)\Psi(x_i),$$

associated with two partitions: $P = \{a = v_0 \leq v_1 \leq \dots \leq v_k \leq v_{k+1} = b\}$ и $Q = \{x_0 \leq x_1 \leq \dots \leq x_{k-1} \leq x_k\}$, which are defined so that

$$a = v_0 \leq x_0 \leq v_1 \leq y_1 \leq v_2 \leq \dots \leq v_k \leq x_k \leq v_{k+1} = b$$

A quadrature formula is a Riemann sum if the following conditions are satisfied:

$$\sum_{i=0}^{j-1} w_i + a \leq x_j \leq \sum_{i=0}^{j} w_i + a\,, (j = 0,1,2, \dots, k) \text{ and } \sum_{i=0}^{j-1} w_i = (b - a).$$

In particular, these conditions are satisfied by the generalized trapezoid formula, for which

$$A_1 = A_k = 1/2, A_2 = A_3 = \dots = A_{k-1} = 1 \quad (5)$$

or, in case of combination with open-type formulas:

$$A_1 = 1/2, A_2 = A_3 = \dots = A_{k-2} = 1, A_{k-1} = 3/2, \quad (6)$$

and the generalized Simpson formula

$$A_i = \begin{cases} \frac{1}{3}, \frac{4}{3}, \frac{2}{3}, \dots, \frac{4}{3}, \frac{1}{3}\ (i = 2r + 1, r = 1,2, \dots); \\ \frac{1}{3}, \frac{4}{3}, \frac{2}{3}, \dots, \frac{4}{3}, \frac{5}{6}, \frac{1}{2}\ (i = 2r, r = 1,2, \dots). \end{cases} \quad (7)$$

Formulas (6) and (7) have remainders of order $o(h^2)$ and $o(h^4)$, respectively. To improve the approximation accuracy, it is possible to use the Newton-Cotes quadrature formulas; however, this is associated with a significant increase in the amount of calculations. Given the need to repeatedly calculate the integral operator when solving integro-differential equations, this makes the use of these formulas irrational.

Therefore, to obtain formulas of higher order accuracy, it is proposed to modify the trapezoidal method using the Euler-Maclaurin formula [13].

$$\int_a^b \Psi(x)dx = h\sum_{i=0}^{k} A_i\Psi(x_k) - \sum_{r=1}^{M} h^{2r}\frac{B_{2r}}{(2r)!}\left\{\Psi^{(2r-1)}(b) - \Psi^{(2r-1)}(a)\right\} + E \quad (8)$$

where A_i - coefficients of the trapezoid formula (5)-(6); B_{2r} - Bernoulli numbers; $E = o(h^{(2M+1)})$, if $\Psi^{(2M+1)}(x)$ is a continuous function or $E = o(h^{(2M+2)})$, if $\Psi^{(2M+2)}(x)$ is a continuous function. Based on the Euler-Maclaurin formula, one can obtain Gregory's computational scheme using finite differences.

The *j-th* right difference $\Delta^j\Psi(x_0)$ corresponding to a fixed step *h* is determined by the expression

$$\Delta^j\Psi(x_0) = \Delta^{j-1}\Psi(x_0 + h) - \Delta^{j-1}\Psi(x_0), j \geq 1;$$

The j *-th* left difference $\nabla^j\Psi(x_0)$ is equal to $\Delta^j\Psi(x_0 - jh)$. Thus, the expression is valid

$$\Delta^j\Psi(x_0) = \nabla^j\Psi(x_0 + jh) = \sum_{s=0}^{j}(-1)^{j+s}C_s^j\Psi(x_0 + sh),$$

where $C_s^j = \frac{j!}{(j-s)!s!}$.

In formula (8), we set $M = \frac{1}{2}p$ if p is even and $M = \frac{1}{2}(p+1)$ if p is odd. We approximate each odd-order derivative from $\Psi'(a), \Psi'(b)$ ..., to $\Psi^{p-1}(a), \Psi^{p-1}(b)$ or $\Psi^p(a), \Psi^p(b)$ by finite-difference formulas, using differences up to and including order p. As a result, we obtain the following expressions for the first and *p-th* derivatives, and the corresponding formulas for the intermediate derivatives:

$$h\Psi'(a) = \Delta\Psi(\mathrm{a}) - \frac{1}{2}\Delta^2\Psi(a) + \frac{1}{3}\Delta^3\Psi(a) \ldots - \frac{(-1)^p}{p}\Delta^p\Psi(a) + o(h^{p+1}),$$

$$h\Psi'(b) = \nabla\Psi(\mathrm{b}) - \frac{1}{2}\nabla^2\Psi(b) + \frac{1}{3}\nabla^3\Psi(b) \ldots - \frac{(-1)^p}{p}\nabla^p\Psi(b) + o(h^{p+1}),$$

$$h^p\Psi^{(p)}(a) = \Delta^p\Psi(\mathrm{a}) + (h^{p+1}), h^p\Psi^{(p)}(b) = \nabla^p\Psi(b) + o(h^{p+1}).$$

Substituting the obtained expressions into (8), we obtain a formula that defines Gregory's method [13]

$$\int_a^b \Psi(x)dx = h\sum_{i=0}^{k} A_i\Psi(x_k) + \sum_{s=1}^{p} c_{s+1}^{*}\{\nabla^s\Psi(\mathrm{b}) + (-1)^s\Delta^s\Psi(\mathrm{a})\} + E_*, \quad (9)$$

where $E_* = o(h^{p+2})$. When making calculations, it is more convenient to transform formula (9) into the form

$$\int_a^b \Psi(x)dx = h\sum_{i=0}^{k} A_i\Psi(x_k) + \sum_{j=0}^{p} q_j^{[p]}\{\Psi(\mathrm{a} + \mathrm{jh}) + \Psi(\mathrm{b} - \mathrm{jh})\} + E_*,$$

The values of the coefficients $q_j^{[p]}$ for $p < 3$ are given in Table 1.

Condition (4) is satisfied for Gregory's method if there is a fixed bound for *p* independent of *k*.

Table 1

Values of the coefficients $q_j^{[p]}$

p	j			
	0	1	2	3
1	-(1/12)	1/12	-	-
2	-(1/8)	1/6	-(1/24)	-
3	-	177/7	-	19/72

The final choice of quadrature formula depends on the specific problem, in particular on the properties of the integral operator kernel. For example, if the integral operator has a singular kernel of the form

$$K\big(x, s, y(s)\big) = \frac{K_1(x, s, y(s))}{(x-s)^\alpha},$$

it is proposed to use open-type quadrature formulas [13]:

$$\int_a^{a+h} \varphi(s)ds = h\varphi(a) + \frac{h^2}{2}\varphi'(\xi), a < \xi < a + 2h,$$

$$\int_a^{a+2h} \varphi(s)ds = 2h\varphi(a) + \frac{h^3}{3}\varphi''(\xi), a < \xi < a + 2h,$$

which, when combined with closed-type formulas, allow us to construct efficient algorithms for approximating the integral operator when solving integro-differential equations. Thus, by combining the simplest open-type formula with the

generalized trapezoidal formula, we obtain the coefficients of the quadrature formula, which can be used to construct quadrature-difference algorithms for solving integro-differential equations with weakly singular kernels.

The choice of quadrature formula is also influenced by the expected nature of the desired solution and the number of steps. The latter is especially relevant when simulating a dynamic system in real time, when the integration interval may be unknown in advance or very large. Furthermore, when constructing numerical simulation algorithms, it is important to consider the Volterra integral operator's ability to reuse each value of the integrand, which significantly reduces computational time.

Most single-step methods are based on the Taylor series expansion.

$$U(x_{k+1}) = U(x_k) + hU'(x_k) + \frac{h^2}{2}U''(x_k) + \cdots.$$

Using the first two terms of the series, we obtain a formula for the simplest one-step method, according to which the algorithm for finding a solution has the form

$$\begin{gathered}\Delta u_i(x_k) = h \cdot F_i \\ \left(x_k, u_1(x_k), \ldots, u_m(x_k), \sum_{j=0}^{k} A_j \cdot K_i\left(x_k, s_j, u_1(x_j), \ldots, u_m(x_j)\right)\right), \\ u_i(x_{k+1}) = u_i(x_k) + \Delta u_i(x_k), \quad i = \overline{1,n}\ ,\end{gathered} \tag{10}$$

where A_j – coefficients of the quadrature formula.

This algorithm is simple to implement on a machine and does not require any preliminary calculations to begin the calculation, but the accuracy of the solution is low. To improve accuracy, it is possible to construct algorithms using three or more terms of the Taylor series; however, in practice, this approach is often impractical due to the need to numerically calculate higher-order derivatives.

This problem is solved using Runge-Kutta methods, which are designed to approximate methods based on Taylor series but without explicitly calculating derivatives, with the exception of the first. The most efficient, from a machine-implementation perspective, is the four-stage Runge-Kutta method, whose algorithm has the following form.

$$\begin{cases} k_{1i} = h F_i\left(x_k, U(x_k), \sum_{j=0}^{k} A_j \cdot K_i\left(x_k, s_j, U(s_j)\right)\right), \\ k_{2i} = h F_i \\ \left(x_k + \frac{h}{2}, U(x_k) + \frac{k_{1i}}{2}, \sum_{j=0}^{k} A_j \cdot K_i\left(x_k + \frac{h}{2}, s_j, U(s_j)\right) + \frac{k_{1i}}{2}\right), \\ k_{3i} = h F_i \\ \left(x_k + \frac{h}{2}, U(x_k) + \frac{k_{2i}}{2}, \sum_{j=0}^{k} A_j \cdot K_i\left(x_k + \frac{h}{2}, s_j, U(s_j)\right) + \frac{k_{2i}}{2}\right), \\ k_{4i} = h F_i \\ \left(x_k + h, U(x_k) + k_{3i}, \sum_{j=0}^{k} A_j \cdot K_i\left(x_k + h, s_j, U(s_j) + k_{3i}\right)\right), \end{cases} \tag{11}$$

$$\begin{gathered} U(x) = \left(u_1(x), \ldots, u_m(x)\right), \\ \Delta u_i(x_k) = \frac{1}{6}\left(k_{1i} + 2k_{2i} + 2k_{3i} + k_{4i}\right), \\ u_i(x_{k+1}) = u_i(x_k) + \Delta u_i(x_k), \quad i = \overline{1,n}\ , \end{gathered} \tag{12}$$

where A_j - coefficients of the quadrature formula.

A disadvantage of this method is the need to repeatedly calculate the integral at each step, which significantly slows down the computation time and makes calculations on a larger number of nodes difficult.

By analogy with differential equations, multi-step algorithms can be used to obtain high accuracy, the most effective of which are algorithms based on multi-step methods that have high stability [14].

Let's consider the computational flowchart of an algorithm based on a fourth-order multi-step method:

1)the first approximation to $U_{k+1} = \left(u_1(x_{k+1}), \ldots, u_n(x_{k+1})\right)$ is calculated and then

$$U_{k+1} = U_k + \frac{h}{24} \cdot (55 f_k - 59 f_{k-1} + 37 f_{k-2} - 9 f_{k-3}); \tag{13}$$

2) using the obtained approximation, the right-hand sides at the point x_{k+1} are calculated;

3) the final approximate solution is calculated at point x_{k+1}:

$$U_{k+1} = U_k + \frac{h}{24} \cdot (9 f_{k+1} + 19 f_k - 5 f_{k-1} + f_{k-2}) \tag{14}$$

where $f_k = \begin{Bmatrix} F_1\left(x_k, U_k, \sum_{j=0}^{k} A_j K_1\left(x_k, s_j, U_j\right)\right), \ldots, \\ F_n\left(x_k, U_k, \sum_{j=0}^{k} A_j K_n\left(x_k, s_j, U_j\right)\right) \end{Bmatrix}$

,

A_j – coefficients of the quadrature formula.

To begin the calculation according to this algorithm, it is necessary to first find the initial segment of the solution (in the first four points), for the calculation of which the one-step methods described above or the first-order multi-step method can be used, the algorithm of which has the form:

1) the first approximation to $U_{k+1} = (u_1(x_{k+1}), \ldots, u_n(x_{k+1}))$ is calculated

$$U_{k+1}^0 = U_k + h\, f_k; \qquad (15)$$

2) using the resulting approximation, the right-hand sides are calculated at point x_{k+1};

3) the final approximate solution is calculated at point x_{k+1}:

$$U_{k+1} = U_{k+1}^0 + \frac{h}{2} \cdot (f_{k+1} - f_k) \qquad (16)$$

All of the presented algorithms can also be used to solve the Cauchy problem for various types of integro-differential equations.

3. ALGORITHMS FOR THE NUMERICAL IMPLEMENTATION OF ITERATIVE METHODS

A distinctive feature of iterative methods is their ease of computer implementation, which enables their effective use not only as standalone methods, but also as auxiliary methods for refining results previously obtained by direct methods. It is important that the iterative process converges at high speed. One effective method for numerically solving the equation

$$y(x) = \varphi(x) + \sum_0^x k[x, s, y(s)]ds \qquad (17)$$

is modified Newton-Kantorovich method [1] which allows for significantly faster convergence compared to other iterative methods. This method is based on solving a linear integral equation

$$\psi_{k-1}(x) = \varepsilon_{k-1}(x) + \int_a^x \widetilde{K}'_y[x, s, y_{k-1}(s)]\psi_{k-1}(s)\,ds \qquad (18)$$

where

$$\varepsilon_{k-1}(x) = \varphi(x) + \int_a^x \widetilde{K}[x, s, y_{k-1}(s)]\,ds - y_{k-1}(x) \qquad (19)$$

Then the *k-th* approximation is calculated using the formula

$$y_k(x) = y_{k-1}(x) + \psi_{k-1}(x), \quad k = 1,2,3,\ldots \qquad (20)$$

Thus, for the numerical solution of the integro-differential equation

$$\sum_{ix}^{n} y^{(i)}(x) = y(x) + \int_0^x k[x, s, y(s)]ds$$

the following iterative algorithm based on the Newton-Kantorovich method is proposed.

1. Implementation of the transition to an integral mathematical model using equivalent transformations.

2. Finding the initial approximation using the quadrature method, using open-type quadrature formulas [13]:

$$y(x_1) = \psi(x_1),$$
$$y(x_2) = \psi(x_2) + h\widetilde{K}(x_2, s_1, 0),$$
$$y(x_i) = \psi(x_i) + A_{i1}\widetilde{K}(x_i, s_1, 0) + \sum_{j=2}^{i-1} A_{ij}\widetilde{K}(x_i, s_j, y(s_j)),\ i = \overline{3,m},$$

where $A_{31} = \frac{1}{2}h,\ A_{32} = \frac{3}{2}h,\ A_{i1} = \frac{1}{2}h,$

$$A_{ij} = h,\ j = \overline{2, i-2},\ A_{i,i-1} = \frac{3}{2}h,\ i = \overline{4,m}, \qquad (22)$$

(the order of accuracy of the method is $o(h^2)$).

3. Determination of the residual $\varepsilon_{k-1}(x)$; in this case, the integral in expression (19) is calculated using the quadrature formula, which is a combination of Newton's formula with the generalized Simpson formula

$$\int_a^{x_i} \widetilde{K}[x, s, y_{k-1}(s)]ds \cong \sum_{j=1}^{i} A_{ij}\widetilde{K}\left[x_i, s_j, y_{k-1}(s_j)\right],$$

in which, for *i* even:

$$A_{41} = A_{44} = \tfrac{3}{8}h,\ A_{42} = A_{43} = \tfrac{9}{8}h,\ A_{61} = \tfrac{1}{3}h,\ A_{62} = \tfrac{4}{3}h,$$
$$A_{63} = \tfrac{17}{24}h,\ A_{64} = A_{65} = \tfrac{9}{8}h,\ A_{66} = \tfrac{3}{8}h,$$
$$\left.\begin{array}{l} A_{i1} = A_{ii} = \frac{3}{8}h,\ A_{i,i-3} = \frac{17}{24}h,\ A_{i,i-2} = A_{i,i-1} = \frac{9}{8}h, \\ A_{i,2m+2} = \frac{4}{3}h,\ m = \overline{0, \frac{i-6}{2}},\ A_{i,2m+1} = \frac{2}{3}h,\ m = \overline{1, \frac{i-6}{2}}, \end{array}\right\} i = 8,10,12,\ldots$$

and if *i* is odd:

$$\left.\begin{array}{l} A_{31} = A_{33} = \frac{1}{3}h,\ A_{32} = \frac{4}{3}h, \\ A_{i1} = A_{ii} = \frac{1}{3}h, \\ A_{i,2m+2} = \frac{4}{3}h,\ m = \overline{0, \frac{i-3}{2}},\ A_{i,2m+1} = \frac{2}{3}h,\ m = \overline{1, \frac{i-3}{2}}. \end{array}\right\} i = 5,7,9,\ldots$$

4. Solution of the linear integral equation (18) by the quadrature method based on the trapezoidal formula:

$$\Psi_{k-1}(x_1) = \varepsilon_{k-1}(x_1),$$
$$\Psi_{k-1}(x_2) = \frac{\varepsilon_{k-1}(x_2) + \frac{h}{2}\widetilde{K}'_y(x_2, s_1, y_{k-1}, (s_1))\varphi_{k-1}(x_1)}{1 - \frac{h}{2}\widetilde{K}'_y(x_2, s_2, y_{k-1}, (s_2))},$$

$\Psi_{k-1}(x_i)$

$$= \frac{\varepsilon_{k-1}(x_i) + \frac{h}{2}\widetilde{K}'_y\left(x_2, s_1, y_{k-1}, (s_1)\right)\varphi_{k-1}(x_1) + \sum_{j=2}^{i-1} h\widetilde{K}'_y\left(x_i, s_j, y_{k-1}, (s_j)\right)\Psi_{k-1}(x_j)}{1 - \frac{h}{2}\widetilde{K}'_y(x_i, s_i, y_{k-1}, (s_i))},$$

5. Calculate the *k-th* approximation of the solution y_k using formula (20).

6. Repeat steps 3-5 if the specified accuracy is not achieved.

7. End the iterative process and calculate the solution *u(x)* of the original integro-differential equation (21).

Applying the Newton-Kantorovich method, as well as other methods used for solving integral equations, to solving integro-differential equations requires complex preliminary equivalent transformations of the original model. Clearly, the efficiency of these methods can be significantly increased by implementing these transformations programmatically. Therefore, a computer implementation of integro-differential models based on the automatic adaptation of the quadrature method used for solving integral equations to solving integro-differential equations of the form

$$u^{(n)}(x) + \sum_{i=0}^{n-1} a_i u^{(i)}(x) = f(x) + \int_a^x K\left(x, s, u(s), u'(s), \dots, u^{(m)}(s)\right) ds, m \leq \quad (23)$$

by software implementation of equivalent dynamic transformations.

The algorithm for this method is as follows:

1. Using the Lagrange identity, the kernel and right-hand side of the equivalent integral equation are generated programmatically:

$$y_n(x) = \varphi(x) + \int_0^x \widetilde{K}[x, s, y(s)]ds,$$

where

$$\widetilde{K}\big(x, s, y(s)\big) = K\big(x, s, y_0(s), \dots, y_m(s)\big) - \sum_{i=0}^{n-1} a_i y_{i+1},\ \varphi(x) = f(x) - \sum_{i=0}^{n-1} a_i y_i(a),$$

$$y_n(x) = u^{(n)}(x), y_j(x) = \int_a^x \frac{(x-s)^{n-j-1}}{(n-j-1)!} y_n(s)ds + \sum_{i=1}^{n-j} \frac{(x-a)^{n-j-1}}{(n-j-i)!}\, u^{(n-i)}(a), j = \overline{0, n-1}.$$

2. The resulting integral equation is solved using the quadrature method according to the formulas

$$y(x_i) = \varphi(x_i) + \sum_{k=0}^{i} A_k \widetilde{K}(x_i, s_k, y(s_k)),$$

$$y_i(x_i) = \sum_{k=0}^{i} A_k \frac{(x_i - s_k)^{n-j-1}}{(n-j-1)!} y(s_k) + \sum_{i=1}^{n-j} \frac{(x_i - a)^{n-j-1}}{(n-j-i)!}\, u^{(n-i)}(a), j = \overline{0, n-1}.$$

At each step, a nonlinear algebraic equation is solved to find the solution to the integral equation at the current point.

3. Using the resulting solution to the integral equation, the solution to the original integro-differential equation is calculated.

4. CONCLUSION

Thus, the use of integro-differential equations increases the versatility of developed computer models through the natural combination of differential and integral operators in a single model. The differential part of the equation is formed in the traditional manner, while the integral part is formed either through analytical transformations or by converting transfer functions to their original forms in the form of a convolution integral. The proposed algorithms can also be used to solve systems of first-order nonlinear integro-differential equations. However, as the results of numerical experiments show, the convergence rate is often slow, or the iterative process diverges altogether. Given the above, when modeling control problems for solving systems of nonlinear integro-differential equations, we find it more effective to use direct algorithms.

REFERENCES

[1]Верлань А.Ф., Москалюк С.С. Математическое моделирование непрерывных динамических систем. - К.: Наукова думка, 1988. - 287 с.

[2]Бахвалов Н. С. Численные методы / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков — М.: Лаборатория базовых знаний, 2002. — 632 с.

[3]Формалев В. Ф. Численные методы / В. Ф. Формалев, Д. Л. Ревизников. — М. : Физматлит, 2006. — 398 с.

[4]Демидович Б.П. Численные методы анализа. Приближение функций, дифференциальные и интегральные уравнения / Демидович Б.П., Марон И.А., Шувалова Э.З. 2008. – 400 с.

[5]Калиткин Н. Н. Численные методы: учебное пособие / Н. Н. Калиткин. – 2-е изд. – СПб. : БХВ-Петербург, 2011. – 592 с.

[6]Young T. Introduction to Numerical Methods and Matlab Programming for Engineers / T. Young, M. Mohlenkamp. — Athens, 2015. — 180 p.

[7]Volterra V. Theory of functionals and of integral and integro–differential equations. New York, Dover Publ., 2005, 288 p

[8]А.А. Верлань, В.А. Федорчук, М.В. Сагатов. Интегральные динамические модели электромеханических объектов: монография. ISBN 978-9943-13-539-0, “IQTISOD-MOLIYA”, 2015, стр. 328.

[9]Самарский А.А. Численные методы решения обратных задач математической физики / А.А. Самарский, П.Н. Вабищевич. – ЛКИ, 2009. – 480 с.

[10] Годлевский В. С., Годлевский В. В. Вопросы точности при обработке сигналов. – Киев: Альфа реклама, 2020. – 407 с.

[11] Shuvra Das. Mechatronic Modeling and Simulation Using Bond Graphs / D. Shuvra.— Oxford: CRC Press. Taylor & Francis publishing group, 2009. — 504 p.

[12] Самарский А. А. Математическое моделирование: Идеи. Методы. Примеры / А. А. Самарский, А. П. Михайлов. — М. : Физматлит, 2005. — 320 с.

[13] Марчук Г.И. Методы вычислительной математики / Г.И. Марчук. – Лань, 2009. – 608 с.

[14] Сизиков В.С. Математические методы обработки результатов измерений / В.С. Сизиков. — Санкт-Петербург: Политехника, 2001. — 240 с.