

ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

DOI: 10.18137/RNU.V9I87.26.03.P.105

УДК 519.85

Al-Khalidi Arkan Mokhammed Ali

PhD of Engineering Sciences, Associate Professor, Kazan Federal University, Kazan.

ORCID: 0000-0001-6492-3291.

Email address: AMAl-Khalidi@kpfu.ru

Аль-Халиди Аркан Мохаммед Али

кандидат технических наук, доцент, Казанский (Приволжский) федеральный университет, город Казань. ORCID: 0000-0001-6492-3291.

Электронный адрес: AMAl-Khalidi@kpfu.ru

Al-Shimari Nofl Shaker

PhD of Mathematics, Associate Professor, University of Babylon, Babylon, Iraq.

Email address: noflshaker8@gmail.com

Аль-Шимари Нофл Шейкер

Доктор философии (PhD) в области математики, доцент, Вавилонский университет, город Вавилон, Ирак.

Электронный адрес: noflshaker8@gmail.com

Amer Ismail Falah Omar

PhD of Engineering Sciences, Associate Professor, Kazan Federal University, Kazan. ORCID: 0000-0002-1561-6559.

Email address: IAmer@kpfu.ru

Амер Исмаил Фалах Омар

кандидат технических наук, доцент, Казанский (Приволжский) федеральный университет, город Казань. ORCID: 0000-0002-1561-6559

Электронный адрес: IAmer@kpfu.ru

APPROXIMATE SOLUTION OF A GLOBAL OPTIMIZATION ALGORITHM FOR PROBLEMS WITH ORDINARY DIFFERENTIAL EQUATIONS

Abstract. Ordinary differential equations (ODEs) are widely used to model dynamical systems across diverse domains, ranging from molecular and chemical processes to economic phenomena. In these settings, dynamic optimization is frequently needed to determine operating policies that achieve the best possible performance under changing conditions. A major challenge arises from nonconvexity, which often produces multiple local solutions. Since many numerical methods can only identify local optima, dependable global optimization techniques are essential. This review discusses several formulations of dynamic optimization, including variable discretization and control parameterization, and evaluates their advantages and drawbacks. Deterministic and stochastic approaches to global optimality are also examined, with emphasis on recent advances such as the α BB method. The overall objective is to provide a structured overview of established methods and emerging trends in global optimization for systems governed by ODEs.

Keywords: optimization, ordinary diffraction equations, python software, systems governed by ODEs.

For citation: Al-Khalidi Arkan Mokhammed Ali, Al-Shimari Nofl Shaker, Amer Ismail Falah Omar (2026) Approximate solution of a global optimization algorithm for problems with ordinary differential equations. *Vestnik of the Russian New University. The series: Complex systems: models, analysis, and management*. No. 3. Pp. 105–114. DOI: 10.18137/RNU.V9I87.26.03.P.105 (In Russian).

1. Introduction

Many systems can be represented by ordinary differential equations (ODEs), including physical systems such as molecular dynamics, chemical systems like chemical kinetics, and dynamic economic processes. Utilizing dynamic optimization for these systems allows for the determination of optimal performance under varying conditions. Often, these problems have multiple local solutions. For instance, Luus and Cormack illustrated this in simple cases, while Luus et al. found 25 local optima for a bifunctional catalyst using 100 random initial points, and Esposito and Floudas discovered over 300 local optima with 1000 starting points. Nonconvexities can prevent current numerical methods from finding solutions for feasible problems, and even when they do, they may only provide local optima. This highlights the urgent need for global optimization algorithms to effectively tackle these issues [1-8].

1.1. Dynamic Optimization

There are several numerical methods for addressing dynamic optimization problems. One method involves variable discretization, converting the problem into a finite-dimensional nonlinear programming (NLP) problem, applicable to both controls and state variables. However, due to nonconvexity, most existing NLP solvers can only find local solutions. In the complete discretization or simultaneous approach, solutions are obtained across the entire variable space. Techniques such as collocation and orthogonal collocation have been used, but they often result in large NLP problems with many nonlinear constraints [9-13].

Alternatively, the sequential approach focuses solely on discretizing the controls, separating the dynamic system from the optimization phase. Established integration methods are used to evaluate the objective function and constraints without substantially increasing the size of the NLP problem. Various control parameterization schemes have been proposed to achieve accurate approximations of state variables. Another approach is dynamic programming, which employs stochastic elements, using grids and penalty functions to effectively manage constrained problems [14-19].

1.2. Global Optimization of Dynamic Problems

Algorithms aimed at achieving global optimality in dynamic issues can be categorized as either stochastic or deterministic. Stochastic optimization methods have been employed to address dynamic optimization problems, with researchers like Rosen and Luus employing control parameterization and line search methods. Deterministic algorithms, including the α BB approach, have also been investigated and show potential in solving NLP problems arising from dynamic optimization.

To conclude, the section surveys established dynamic optimization approaches together with emerging global optimization methods for ODE-based systems [20].

Approximate solution of a global optimization algorithm for problems with ordinary differential equations

2. Dynamic Optimization

The dynamic optimization problem can be formulated as follows:

$$\begin{cases} \text{Minimize: } J(x(t_i, p), p) & \text{for } i = 0, 1, \dots, N_S \\ \text{Subject to:} \\ \quad \dot{x} = f(t, x, p) & \text{for all } t \text{ in } [t_0, t_{\{N_S\}}] \\ \quad x(t_0, p) = x_{0(p)} \\ \quad g_i(x(t_i, p), p) \leq 0, & \text{for } i = 0, 1, \dots, N_S \\ \quad L \leq p \leq U \end{cases}$$

where:

- $t \in \mathbb{R}$ represents time, with t_0 and $t_{\{N_S\}}$ as the initial and final times;
- t_i belongs to the interval $[t_0, t_{\{N_S\}}]$;
- x and $\dot{x} \in \mathbb{R}^n$ denote the state variables and their time derivatives;
- $p \in \mathbb{R}_r$ are time-invariant parameters.

Functions Defined

- $J: \mathbb{R}^n(N_S + 1) \times \mathbb{R}_r \rightarrow \mathbb{R}$
- $f: [t_0, t_{\{N_S\}}] \times \mathbb{R}^n \times \mathbb{R}_r \rightarrow \mathbb{R}^n$
- $x^0: \mathbb{R}_r \rightarrow \mathbb{R}^n$
- $g_i: \mathbb{R}^n \times \mathbb{R}_r \rightarrow \mathbb{R}^{\{s_i\}}$

Systems with time-dependent controls can be reformulated using control parameterization (Vassiliadis et al., 1994a).

Differential Inequalities

The influence of variable bounds on convex relaxations is a common aspect of deterministic global optimization algorithms. Since state variables are included in the nonconvex objective function and constraints, a method is needed to derive precise bounds for these variables at specific times t_i (where $i = 0, 1, \dots, N_S$; $i = 0, 1, \dots, N_C$). This can be achieved by overestimating the solution space of ordinary differential equations (ODEs) [21].

Consider the following ODE:

$$\begin{aligned} \dot{x}(t) &= -x(t)2 + v, \quad \forall t \in [0, 1]; \\ x(0) &= 9. \end{aligned}$$

If the parameter v can vary within the interval:

$$-5 \leq v \leq 5$$

the right-hand side of the ODE represents a set of functions.

Validated or interval methods create an enclosure for the solutions of an ODE system. Reviews of these methods are available in works by Moore (1984), Nickel (1986), and Corliss (1995). Differential inequalities can be used to construct sub- and superfunctions. Nickel and Adams characterized systems of ODEs where differential inequalities can compute tight inclusions. Finite difference approximations combined with interval calculations can also yield interval solutions. Moore applied these methods to overestimate remainder terms in Taylor se-

ries expansions. Nedialkov et al. provided a framework for Taylor series methods, comparing different schemes and identifying challenges [22-27].

Application

1. Consider the following system of differential equations:

$$\begin{cases} \dot{x}^1(t) = \inf(-x^1(t)^2 + [\underline{v}, \bar{v}]) \\ \dot{x}^2(t) = \inf(-2[x^1(t), x^1(t)]x^2(t) + 1) \\ \dot{x}^3(t) = \inf(-2[x^2(t), x^2(t)]^2 - 2[x^1(t), x^1(t)]x^3(t)) \\ \dot{x}^1(t) = \sup(-x^1(t)^2 + [\bar{v}, \underline{v}]) \\ \dot{x}^2(t) = \sup(-2[x^1(t), x^1(t)]x^2(t) + 1) \\ \dot{x}^3(t) = \sup(-2[x^2(t), x^2(t)]^2 - 2[x^1(t), x^1(t)]x^3(t)) \end{cases}$$

for all $t \in [0, 1]$ with initial conditions:

$$x^1(0) = 9, x^2(0) = 0, x^3(0) = 0$$

2. Solution Bounds

The solution of this system provides bounds for the set of solutions of the original system:

$$x_i(t) \leq x_i(t, v) \leq x_i(t) \quad \forall v \in [\underline{v}, \bar{v}], \text{ for all } t \in [0, 1], i = 1, 2, 3$$

3. Interval Arithmetic Properties

Using the relevant equations and interval arithmetic properties, we have:

$$\nabla^2 x^1(1, v) \in [x^3(1), x^3(1)] \quad \forall v \in [\underline{v}, \bar{v}]$$

$$\nabla^2(-x^1)(1, v) \in [-x^3(1), -x^3(1)] \quad \forall v \in [\underline{v}, \bar{v}]$$

4. Optimization Problem

Applying the scaled Gerschgorin method allows the calculation of parameters α^+ , α^- . The objective function can be underestimated using the secant underestimator, with bounds defined by:

$$\begin{cases} \min\{-[x^1(1) + x^1(1)]x + x^1(1)x^1(1)\} , x, v \\ \text{subject to:} \\ x^1(1) \leq x \leq x^1(1) \\ x^1(1, v) + \alpha^+(\underline{v} - v)(\bar{v} - v) - x \leq 0 \\ x - x^1(1, v) + \alpha^-(\underline{v} - v)(\bar{v} - v) \leq 0 \\ \underline{v} \leq v \leq \bar{v} \end{cases}$$

5. Solution Approach

The value of $x_1(1, v)$ is found by solving the ODE system. This involves:

- Numerical integration of the ODE system from $t = 0$ to $t = 1$;
- Evaluation of the constraints at $t = 1$;
- Solving the optimization problem to find the minimum value.

6. Next Steps

7. Implement a numerical ODE solver (e.g., Runge-Kutta method).

8. Develop an optimization algorithm to solve the constrained minimization problem.

Approximate solution of a global optimization algorithm
for problems with ordinary differential equations

9. Analyze the results and interpret their physical or practical meaning in the context of the original problem.

Note: The specific numerical methods and algorithms used may depend on the desired accuracy and computational resources available

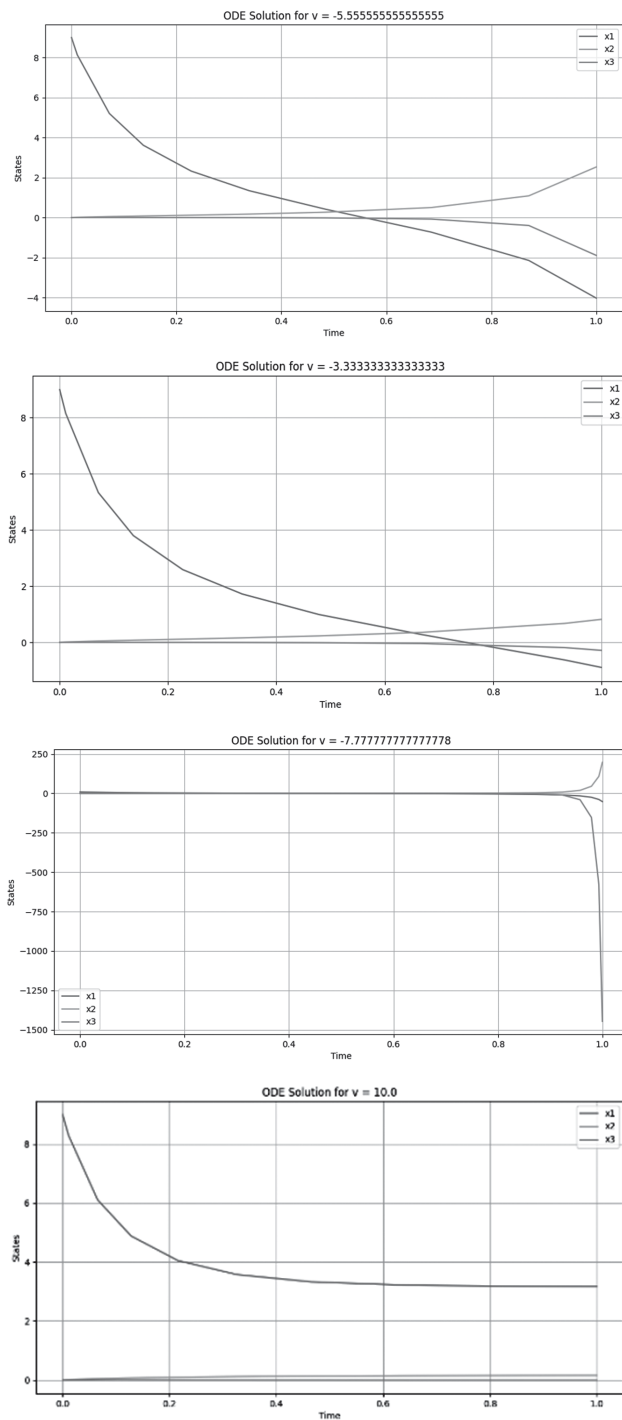
Iteration	v	x1_final	x2_final	x3_final
1	-10.0	-3.87066e + 14	7.39309e + 27	-2.82507e + 41
2	-7.778	-52.2548	197.215	-1446.59
3	-5.556	-4.02885	2.52693	-1.89273
4	-3.333	-0.891805	0.819	-0.284777
5	-1.111	0.44636	0.453239	-0.0925144
6	1.111	1.27823	0.313643	-0.0422151
7	3.333	1.88986	0.243694	-0.023385
8	5.556	2.3821	0.202501	-0.0146811
9	7.778	2.8002	0.175508	-0.0100507
10	10.0	3.16787	0.156443	-0.00732648

Caption: Table 1: Final values of state variables x_1 , x_2 , and x_3 for different control inputs v . The table shows how the system's final state changes as the control parameter v varies from -10 to 10.

Additional Results:

- Optimal v : 10.0
- Optimal objective value: 10.0354
- Second-order sensitivity bounds for x_1 : $[-0.00732648, -0.00732648]$
- Second-order sensitivity bounds for $-x_1$: $[0.00732648, 0.00732648]$
- Optimal x_{hat} : 6.08394
- Relaxation objective value: 9.23772.

Figures 1–4 illustrate the dynamic behavior of the system under different control inputs, while Figure 4 presents the objective function as a function of the control parameter v and indicates the optimal control value.



Figures 1-4. Time-dependent behavior of the state variables x_1 , x_2 , and x_3 for selected values of v . Each plot corresponds to a single value of v within the interval $\bar{v}$ and $\underline{v}$.

Approximate solution of a global optimization algorithm
for problems with ordinary differential equations

```

Appendix: Python Code
import numpy as np
from scipy.integrate import solve_ivp
from scipy.optimize import minimize
import matplotlib.pyplot as plt

# Define the ODE system
def ode_system(t, x, v):
    x1, x2, x3 = x
    dx1dt = -x1 ** 2 + v
    dx2dt = -2 * x1 * x2 + 1
    dx3dt = -2 * x2 ** 2 - 2 * x1 * x3
    return [dx1dt, dx2dt, dx3dt]

# Solve the ODE system
def solve_ode(v, t_span=[0, 1], x0=[9, 0, 0]):
    sol = solve_ivp(ode_system, t_span, x0, args=(v,), dense_output=True)
    return sol

# Objective function to minimize
def objective(v, x0=[9, 0, 0]):
    sol = solve_ode(v[0], x0=x0) # Extract scalar v from array
    x1_final = sol.y[0, -1]
    return -x1_final ** 2

# Define bounds for v
vL, vU = -10, 10 # Example bounds, adjust as necessary

# Perform the optimization
result = minimize(objective, x0=[0], bounds=[(vL, vU)])
optimal_v = result.x[0]
optimal_value = -result.fun
print(f"Optimal v: {optimal_v}")
print(f"Optimal objective value: {optimal_value}")

# Sensitivity Analysis
def sensitivity_analysis(v, t_span=[0, 1], x0=[9, 0, 0]):
    sol = solve_ode(v, t_span, x0)
    x1_final = sol.y[0, -1]
    x2_final = sol.y[1, -1]
    x3_final = sol.y[2, -1]
    # Compute second-order sensitivity bounds
    grad2_x1 = [x3_final, x3_final]
    grad2_neg_x1 = [-x3_final, -x3_final]
    return grad2_x1, grad2_neg_x1

grad2_x1, grad2_neg_x1 = sensitivity_analysis(optimal_v)
print(f"Second-order sensitivity bounds for x1: {grad2_x1}")
print(f"Second-order sensitivity bounds for -x1: {grad2_neg_x1}")

# Convex Relaxation
def convex_relaxation(v, alpha_plus, alpha_minus, vL, vU):

```

```

sol = solve_ode(v)
x1_final = sol.y[0, -1]
x1_inf = min(sol.y[0])
x1_sup = max(sol.y[0])
def relaxation_objective(x_hat):
    return -x1_inf * x_hat + x1_sup * x1_inf
def constraint1(x_hat):
    return x1_inf - x_hat
def constraint2(x_hat):
    return x_hat - x1_sup
def constraint3(x_hat):
    return x1_final + alpha_plus * (vL - v) * (vU - v) - x_hat
def constraint4(x_hat):
    return x_hat - x1_final + alpha_minus * (vL - v) * (vU - v)
constraints = [{ 'type': 'ineq', 'fun': constraint1 },
               { 'type': 'ineq', 'fun': constraint2 },
               { 'type': 'ineq', 'fun': constraint3 },
               { 'type': 'ineq', 'fun': constraint4 }]
x_hat_initial = (x1_inf + x1_sup) / 2
res = minimize(relaxation_objective, x_hat_initial, constraints=constraints)
return res.x[0], res.fun
# Example values for alpha_plus and alpha_minus, adjust as necessary
alpha_plus, alpha_minus = 0.1, 0.1
optimal_x_hat, relaxation_value = convex_relaxation(optimal_v, alpha_plus, alpha_minus, vL, vU)
print(f"Optimal x_hat: {optimal_x_hat}")
print(f"Relaxation objective value: {relaxation_value}")
# Loop through different values of v and print iterations
v_values = np.linspace(vL, vU, 10) # 10 sample points between vL and vU
for i, v in enumerate(v_values):
    sol = solve_ode(v)
    t = sol.t
    x1, x2, x3 = sol.y
    print(f"Iteration {i + 1}: v = {v}")
    print(f"x1_final = {x1[-1]}")
    print(f"x2_final = {x2[-1]}")
    print(f"x3_final = {x3[-1]}")
    print("-" * 30)
    plt.figure(figsize=(12, 6))
    plt.plot(t, x1, label='x1')
    plt.plot(t, x2, label='x2')
    plt.plot(t, x3, label='x3')
    plt.xlabel('Time')
    plt.ylabel('States')
    plt.title(f'ODE Solution for v = {v}')

```

Approximate solution of a global optimization algorithm
for problems with ordinary differential equations

```
plt.legend()
plt.grid(True)
plt.show()
# Plot the objective function
objective_values = [objective([v]) for v in v_values]
plt.figure(figsize=(12, 6))
plt.plot(v_values, objective_values, marker='o')
plt.xlabel('v')
plt.ylabel('Objective Function Value')
plt.title('Objective Function over')
```

Conclusion

Systems governed by ordinary differential equations (ODEs) pose substantial difficulties when optimal solutions are sought, largely because nonconvexity can produce numerous local optima. Dynamic optimization is therefore indispensable for attaining optimal performance in areas such as physics, chemistry, and economics. Because many existing numerical schemes yield only local solutions, the need for dependable global optimization algorithms is evident. Examining both variable discretization and control parameterization shows how difficult it is to recast dynamic optimization problems into tractable forms, since each strategy has its own drawbacks. In addition, progress in stochastic and deterministic methods, including the α BB approach, offers encouraging routes toward global optimality.

As this area of research advances, further refinement of these techniques will be necessary to handle the complex challenges presented by ODE-based dynamic systems. Future studies should aim to improve the reliability of global optimization methods and expand their use across a wider variety of demanding problems.

References

1. Bras M., Cardone A. (2012). Construction of Efficient General Linear Methods for Non-Stiff Differential Systems. *Mathematical Modelling and Analysis*. No. 17(2). Pp. 171-189.
2. Bras M., Cardone A., D'Ambrosio R. (2013). Implementation of Explicit Nordsieck Methods with Inherent Quadratic Stability. *Mathematical Modelling and Analysis*. No. 18(2). Pp. 289-307.
3. Bras M., Cardone A., Jackiewicz Z., Welfert B. (2015). Order Reduction Phenomenon for Methods. *Journal of Computational and Applied Mathematics*. No. 290. Pp. 64.
4. Antonov N.A., Ishmukhametov S.T., Halidi Al Arkan M, Maiorova M.E. (2017) A Study Of Euclidean K-Ary Gcd Algorithm. *Revista Publicando*. Vol. 4. No. 13(2).
5. Arkan M., Ishmukhametov Sh.T. (2020) An effective programming of GCD Algorithms for natural numbers. *News of universities. Mathematics*. Vol. 64. Is. 6. Pp. 1-5.
6. Amer I., Alkhalidi A.M., Ishmukhametov T. (2020) Comparing different GCD algorithms. *International Journal of Advanced Science and Technology*. Vol. 29. No. 4. Pp.1278-1283.
7. Amer I.F., Ishmukhametov S.T., Alkhalidi A.M. (2021) Programming of an approximating K-ary GCD algorithm. *J. Phys.: Conf. Ser.* 2096 012097.
8. Burnashev R.A., Amer I., Enikeev A.I. (2019) Expert system building tools based on dynamically updated knowledge. *Journal of Physics: Conference Series*. Vol. 1352. No. 1. Pp. 012008.
9. Burrage K., Cardone A., D'Ambrosio R., Paternoster B. (2017). Numerical Solution of Time Fractional Diffusion Systems. *Applied Numerical Mathematics*. Vol. 116. Pp. 82–94.

10. Butcher J.C. (1987). The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods. John Wiley & Sons.
11. Coleman Order (2003) Conditions for a class of two-step methods for $y'' = f(x,y)$. *IMA Journal of Numerical Analysis*. Vol. 23. Pp. 97-220.
12. Coleman and Ixaru (1996) P-stability and exponential-fitting methods for $y'' = f(x,y)$. *IMA Journal of Numerical Analysis*. Vol. 16. No. 2. Pp. 179-199.
13. Coleman. (1992) Rational approximations for the cosine function; P-acceptability and order. *Numerical Algorithms*. Vol. 3. No. 1-4. Pp. 143-158.
14. Butcher J.C. (2008). Numerical Methods for Ordinary Differential Equations (2nd Edition). John Wiley & Sons Publ.
15. Butcher J., D'Ambrosio R. (2017) Partitioned General Linear Methods for Separable Hamiltonian Problems. *Applied Numerical Mathematics*. Vol. 117. Pp. 69-86.
16. Cong (1993) Note on the performance of direct and indirect Runge-Kutta-Nystrom methods. *Journal of Computational and Applied Mathematics*. Vol. 45. No. 3. Pp. 347-355.
17. Cong (1996) Explicit parallel two-step Runge-Kutta-Nystrom methods. *Computers & Mathematics with Applications*. Vol. 32. No. 2. Pp. 119-130.
18. Cong (2002) Parallel-iterated pseudo two-step Runge-Kutta-Nystrom methods for nonstiff second-order IVPs. *Computers & Mathematics with Applications*. Vol. 44. No. 1-2. Pp. 143-155.
19. Conte et al. (2017) Construction and implementation of two-step continuous methods for Volterra Integral Equations. *Applied Numerical Mathematics*. Vol. 119. Pp. 239-247.
20. Butcher J.C., Jackiewicz Z. (2004). Construction of General Linear Methods with Runge-Kutta Stability Properties. *Numerical Algorithms*. Vol. 36. Pp. 53-72.
21. Butcher J.C., Wright W.M. (2003). The Construction of Practical General Linear Methods. *BIT Numerical Mathematics*. Vol. 43. Pp. 695-721.
22. Capobianco G., Conte D. (2006). An Efficient and Fast Parallel Method for Volterra Integral Equations of Abel Type. *Journal of Computational and Applied Mathematics*. Vol. 89. No. 1-2. Pp. 481-493.
23. Capobianco G., Conte D., Del Prete I. (2009). High Performance Numerical Methods for Volterra Equations with Weakly Singular Kernels. *Journal of Computational and Applied Mathematics*. Vol. 228. Pp. 571-579.
24. Cardone et al. (2014) Extrapolation-based implicit-explicit general linear methods. *Numerical Algorithms*. Vol. 65. No. 3. Pp. 377-399.
25. Cardone et al. (2014) Extrapolated Implicit-Explicit Runge-Kutta Methods. *Mathematical Modelling and Analysis*. Vol. 19. No. 1. Pp. 18-43 (2014).
26. Cardone et al. (2015) Construction of highly stable implicit-explicit methods. *Discrete and Continuous Dynamical Systems, AIMS Proceedings*. Pp. 185-194.
27. Cardone et al. (2017) Order conditions for general linear methods. *Applied Numerical Mathematics*. Vol. 119. Pp. 94-114.

Поступила в редакцию: 10.04.2026

Поступила после рецензирования: 01.06.2026

Принята к публикации: 14.06.2026

Received: 10.04.2026

Revised: 01.06.2026

Accepted: 14.06.2026