Efficient Parallel Hybrid DIRK Solvers for Stiff Complex Critical Infrastructure Models

Iryna Nazarova[1,*,†], Yaroslav Dorohyi[1,*,†], Tetiana Altukhova[1,†], Maksym Kunda[1,†]
Donetsk National Technical University, 76, Sambirska Str., Drohobych, Lviv region, 82111, Ukraine

Abstract
Ensuring the resilience of digitalized critical infrastructure against cyber-physical threats requires high-speed real-time simulation of large-scale dynamic processes. These processes are often characterized by strong stiffness and interconnected components, making traditional sequential simulation approaches ineffective. This paper addresses the problem by improving the computational performance of numerical integration for stiff large-scale systems of differential equations. Singly Diagonally Implicit Runge-Kutta methods are used to reduce computational overhead by reusing the system matrix factorization. Parallel implementations of these schemes are developed and analyzed within a hybrid MPI + OpenMP framework. A comparative evaluation of three iterative solvers is conducted, including simple iteration, the classical Newton method, and the Jacobian-free Newton-Krylov method. Using the Van der Pol oscillator as a benchmark demonstrates that while direct Newton methods excel in uncoupled systems, the Newton-Krylov approach provides superior scalability and reduced memory overhead for highly coupled infrastructure networks. The proposed parallel framework achieves significant speedup, enabling rapid predictive analysis for real-time cyber defense applications. The proposed parallel framework achieves significant speedup, enabling rapid predictive analysis for real-time cyber defense applications.

Keywords
Critical infrastructure, stiff Cauchy problems, DIRK: SDIRK, ESDIRK methods, Jacobian-free Newton-Krylov solver, parallel algorithms, MPI, OpenMP, 1D-torus topology, Van der Pol Oscillator

1. Introduction

Modern critical infrastructure resilience fundamentally depends on the stability of its digital control systems. Cybersecurity for these vital structures demands ultra-fast, real-time modeling of system behavior during cyber-physical incidents. Mathematically, modeling of similar complex dynamic processes is often reduced to solving large-scale systems of ordinary differential equations (SODEs). A significant part of such practical problems involves stiff systems characterized by processes occurring at vastly different rates.

Classical explicit numerical methods are usually either inapplicable or ineffective for such problems due to strict restrictions on the integration step. Fully implicit Runge-Kutta (FIRK) methods offer absolute stability, allowing for larger steps in very stiff systems. Their primary disadvantage, however, is the necessity to solve a system of nonlinear algebraic equations (SNAEs) at each step, which greatly increases the computational complexity of FIRK and limits its use in real-time critical applications. In this regard, diagonally implicit Runge-Kutta (DIRK) methods are of special significance. These methods possess absolute stability properties (A- and L-stability) and permit integration with much larger step sizes. Although DIRK schemes also require solving SNAEs, the Butcher matrix's specific structure makes their implementation considerably less complex, rendering them highly effective for the rapid simulation of critical processes.

Thus, the primary goal of this research is to develop efficient parallel methods to solve stiff, large-scale Cauchy problems for SODEs, enabling the accelerated, real-time simulation of complex

[1]*RS-2026: Resilient Systems: Secure Digital Technologies and Critical Infrastructure, June 30, 2026, Drohobych, Ukraine*
* Corresponding author.
† These authors contributed equally.
iryna.nazarova@donntu.edu.ua (I. Nazarova); yaroslav.dorohui@donntu.edu.ua (Ya. Dorohyi); tetiana.altukhova@donntu.edu.ua (T. Altukhova); maksym.kunda@donntu.kita.edu.ua (M. Kunda)
0000-0003-2141-5697 (I. Nazarova); 0000-0003-3848-9852 (Ya. Dorohyi); 0000-0002-8255-5725 (T. Altukhova); 0009-0000-1491-3503 (M. Kunda)

dynamic processes in critical infrastructure and cybersecurity systems. A hybrid parallel approach using MPI and OpenMP, combined with advanced numerical methods, specifically singly DIRK (SDIRK) with Newton–Krylov solvers, is employed to minimize computation time on modern multi-core and multiprocessor architectures.

To achieve the intended goal, this article addresses the following tasks:

- Development and analysis of parallel numerical schemes of the DIRK family in terms of efficiency and scalability, stability regions, the relationship between the order and number of stages, and adaptive step control.
- Evaluation of the efficiency and convergence of iterative solvers for systems of nonlinear algebraic equations generated by these methods.
- Development and optimization of direct and hybrid parallelization technologies for scalable mapping to high-performance computing architectures.
- Execution of an analytical and experimental analysis of the classical Van der Pol benchmark to evaluate the performance and limitations of the proposed approach under varying stiffness.

2. Review of Related Works

The theoretical foundations for solving ordinary differential equations are well-established. Fundamental results regarding a wide range of numerical methods for solving IVPs, including approaches for stiff systems, are presented in modern monographs [1-4] by renowned experts such as J. Butcher, G. Dahlquist, E. Hairer. These classical sources remain essential for understanding the theoretical basis of research concerning the determination of the approximation order, stability, and convergence.

Regarding the main objective of the study, namely, the efficiency of implementing diagonally implicit Runge–Kutta methods, the work by Kennedy and Carpenter serves as a fundamental modern review [5]. This work provides a survey of sequential DIRK methods applied to first-order ODEs, considering more than 20 critical aspects and conducting a design study of schemes with two to seven implicit stages.

The following two papers [6, 7] represent modern analytical review. The article by Sabawi et al. is dedicated to the family of SDIRK methods applied to stiff IVPs [6]. A comparison of this class of methods with FIRK methods is performed, A-stability characteristics are obtained, and estimates for the order and number of stages are provided. The article [7] by Jørgensen et al. presents a systematic approach to studying the class of explicit singly diagonally implicit Runge-Kutta (ESDIRK) methods. This paper derives and analyzes the properties of ESDIRK integration methods while discussing the construction principles of Runge-Kutta schemes with embedded methods of different orders for error estimation and continuous extensions. These methods are suitable for systems of ordinary differential equations with low or medium accuracy as well as for differential-algebraic equations (DAEs). Many contemporary studies are dedicated to the construction of new IRK methods and the analysis of their properties. These include works [8-10], which present various numerical diagonally implicit schemes of orders 6–8 with A- and L-stability properties that ensure balance between accuracy and computational cost. The final group of reviewed articles [11-15] is dedicated to the comparative analysis of the DIRK, SDIRK, and ESDIRK subclasses.

An analysis of the literature demonstrates that DIRK methods remain a universal class for stiff problems, while SDIRK schemes reduce computational costs through the reuse of factorizations. Furthermore, ESDIRK methods represent the optimal compromise between accuracy, stability, and speed. Current trends include a transition toward high-order schemes (order>6), the development of adaptive methods, and the integration of these approaches into cybersecurity problems and the simulation of complex critical systems.

Most current work investigates the mathematical properties of implicit one-step numerical methods, including convergence, stability, and order. However, practically all classical approaches for solving stiff SODEs were developed for implementation on sequential execution. As a result, there

is a lack of comprehensive studies focusing on the parallel scalability and efficiency of such integration techniques in high-performance computing environments [16-18]. Addressing this gap is essential for the practical simulation of extremely large-scale models where sequential processing becomes a bottleneck. Consequently, it is highly desirable to modify existing methods or develop new computational schemes better suited for the complex parallel architectures of modern supercomputers. The primary aim of this work is to design, adapt, and study the characteristics of algorithms specifically tailored for parallel hybrid hierarchical architectures, that utilize both distributed memory and multithreading. The results presented in the article are based on the authors' previous research [19-25].

3. Mathematical formulation of problem and basic notation

3.1. Description and notation for the stiff IVP

To address the identified challenges, this section describes the mathematical formulation of the problem and the methods. An approximate numerical solution is considered for the Cauchy problem (initial value problem, IVP) of a system of first-order ordinary differential equations with initial conditions:

$$\overline{y}'(t)=F\left(t,\overline{y}(t)\right),\overline{y}(t_0)=\overline{y}_0,\ t\in[t_0,t_{end}], \tag{1}$$

where m is the dimension of the SODE, $\overline{y}(t)=\left(y_1(t),y_2(t),\dots,y_m(t)\right)^T$ is the vector of unknown functions, $\overline{y}(t_0)=\overline{y}_0$ is the vector of initial conditions, and $F(t,\overline{y})=\left(f_1(t,\overline{y}),\right.$ $\left.f_2(t,\overline{y}),\dots,f_m(t,\overline{y})\right)^T$ represents the right-hand side of (1). In its general form F is a vector of nonlinear functions defining the mapping: $R\times R^m\longrightarrow R^m$.

The integration of SODE arises in many practical dynamical applications and often encounters the following phenomenon: despite the slow change of some, sometimes most, unknown functions, the calculation must be performed with a very small step size. Systems of equations with this property are called stiff. It is difficult to give a mathematically precise definition of stiffness, since the problems that fall into this class are very diverse. Most often, differential equations are called stiff when the solution contains a smoothly varying component, as well as rapidly decaying perturbations. A stiff Cauchy problem is an initial value problem for which numerical solution by standard explicit methods is inefficient or unstable due to the presence of vastly different time scales [1-2]. Since a strict mathematical definition of a stiff SODE is difficult to establish, the stiffness number is typically used to characterize the level of stiffness. Under the condition that the eigenvalues of the Jacobian matrix of the SODE have negative real parts for any t, the stiffness number is defined as:

$$g=\frac{\max\limits_i(|\mathrm{Re}(\lambda_i)|)}{\min\limits_i(|\mathrm{Re}(\lambda_i)|)}, \tag{2}$$

where $\max\limits_i(|\mathrm{Re}(\lambda_i)|)$ corresponds to the fastest processes in the system, $\min\limits_i(|\mathrm{Re}(\lambda_i)|)$ corresponds to the slowest processes in the system, their large ratio indicates the presence of very different scales.

3.2. Description of methods and schemes

The primary purpose of this section is to describe one-step multi-stage Runge-Kutta methods, which constitute a family of IVP integration methods with diverse and often contrasting properties. The fundamental formal difference between them lies in the structure of the Butcher table [4], which defines the coefficients of a specific method (Figure 1).

The general form of an s-stage Runge-Kutta method of order r is given by:

$$\overline{y}_{n+1}=\overline{y}_n+h\sum_{i=1}^{s} b_i\,\overline{k}_i,\ \overline{k}_i=\overline{f}\left(t_n+c_ih;\overline{y}_n+h\sum_{j=1}^{s} a_{ij}\,\overline{k}_j\right), \quad (3)$$

where the matrix $A=\left(a_{ij}\right)$ and vectors $b^T=(b_i)^T$, $c^T=(c_i)^T$, $i,j=\overline{1,s}$, define a unique method derived from convergence and stability conditions (Figure 1).

Simple ODEs are typically integrated using explicit Runge-Kutta (ERK) methods. Their main structural constraint is the strict lower triangularity of the matrix: $A=(a_{ij})_{i,j=1}^{s}$, $\forall i,j=\overline{1,s},\ j\geq i,\ a_{ij}=0$. However, ERK methods cannot successfully integrate stiff systems. The applicability of these methods to a particular SODE depends on non-trivial characteristics of its spectrum, making their determination computationally expensive and often leading to reliance on an empirical approach. The conditional stability of ERK schemes significantly limits their scope of application. Moreover, they exhibit low potential for internal parallelism. Consequently, implicit schemes are of great interest. Despite their higher computational complexity, they have no alternative among one-step methods for solving stiff IVPs and problems with singularities. Fully implicit Runge-Kutta (FIRK) methods are characterised by a dense Butcher matrix [1-2, 24]. For non-stiff problems, FIRK methods are rarely used due to high computational costs. Moreover, even for stiff problems, the total overhead of these methods often turns out to be unacceptable. At each step, they require solving a system of $m\times s$ nonlinear algebraic equations, where m is the number of differential equations and s is the number of stages. In many cases, the computational scheme can be significantly simplified by requiring the coefficient matrix A to be lower triangular. Under this condition, instead of a system of $m\times s$ equations, it is necessary to solve s successive systems of m equations at each step, which is computationally much simpler. DIRK methods are a subclass of implicit methods in which the matrix A is lower triangular with a non-zero main diagonal: $A=(a_{ij})_{i,j=1}^{s}$, $\forall i,j=\overline{1,s},\ j>i,\ a_{ij}=0$ and $a_{ii}\neq 0$. Scheme: $\overline{k}_i=\overline{f}\left(t_n+c_ih;\overline{y}_n+h\sum_{j=1}^{i} a_{ij}\,\overline{k}_j\right)$. Diagonally implicit methods combine the advantages of both approaches: they utilize a lower triangular coefficient matrix with a non-zero diagonal, ensuring the stability characteristics of implicit methods while requiring the solution of only one system of equations per stage. This renders them numerically more efficient than fully implicit methods.

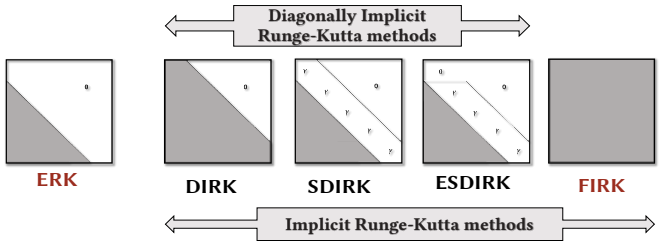

Figure 1: Structure of the Butcher table for various classes of Runge-Kutta methods: Explicit (ERK), Fully Implicit (FIRK), and Diagonally Implicit (DIRK/SDIRK/ESDIRK).

The Singly Diagonally Implicit Runge-Kutta is a subclass of DIRK methods in which all diagonal elements of the matrix A are identical: $a_{ii}=\gamma$. This is the most computationally effective approach for practical implementation, since the system's Jacobian matrix remain the same across all stages, which saves resources for its manipulation: $J(Y,t)=\left(\frac{\partial f_i}{\partial y_j}\right)_{i,j=\overline{1,m}}$, $J(Y,t)\in R^{m\times m}$. Consequently, this property significantly reduces the computational cost of matrix factorizations, making SDIRK methods much more performant than general DIRK schemes, especially for large-scale systems, while maintaining A-stability or L-stability through an appropriate choice of the parameter γ. The ESDIRK method is a special case of SDIRK where $a_{11}=0$ (making the first stage explicit), while the remaining diagonal elements are equal to γ. This specific structure facilitates the embedding of error

estimates for adaptive step-size control. Despite the rapid growth of sequential computing power, conventional single-threaded architectures often lack the capability to model highly complex dynamic systems described by large-scale ODEs. One effective way to address this limitation is to transition the computation to parallel computing structures, which are capable of solving high-dimensional problems far more rapidly than sequential systems.

4. Parallel diagonally implicit methods for stiff Cauchy problems

Advanced research into the efficiency of parallelizing stiff large-scale Cauchy problems using implicit numerical schemes has led to the following findings [19–25]:

- Implicit Runge-Kutta methods demonstrate significantly better dynamic parallelism characteristics compared to explicit methods.
- Block or multi-point implicit methods provide stability characteristics comparable to those of FIRK methods, yet exhibit lower computational complexity in both sequential (by a factor of s) and parallel (by a factor of $2s$ implementations).
- Diagonally implicit methods of all types possess lower computational complexity than FIRK methods, as they do not require solving a system of stage coefficients of dimension $m \times s$. At the same time, different sub classes within the DIRK family feature distinct structures, resulting in varying degrees of potential parallelism.

The primary advantage of SDIRK methods lies in the identity of all diagonal elements $a_{ii}=\gamma$, which ensures that the system matrix $(I\text{-}h\gamma J)$ remains constant for all stages of a single integration step. In a parallel environment, this allows the matrix to be constructed and factorized across multiple nodes only once per step, with the resulting factors reused for all stages. This significantly reduces computational overhead compared to general DIRK methods, which require matrix factorization at each stage. The ESDIRK method inherits these advantages while offering enhanced efficiency in error control due to its explicit first stage. This structure facilitates faster step-size adjustments and prevents redundant parallel computations that might occur with excessively small integration steps. This paper presents the developed parallel methods for solving stiff IVPs based on schemes with different solvers of stage systems and under conditions of high-performance computers (Figure 2).

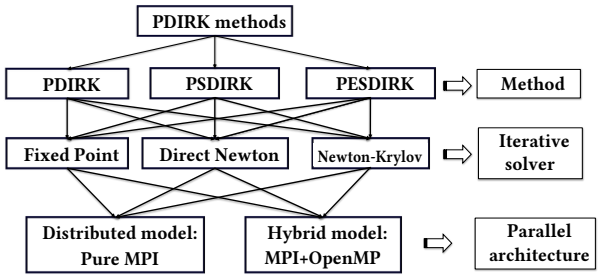

Figure 2: Parallel numerical methods: PDIRK, PESDIRK, PSDIRK with iterative solvers: fixed point, direct Newton, Newton-Krylov for MIMD-systems with distributed or hybrid memory.

It should be noted that implementing such methods requires solving systems of algebraic equations to determine the stage coefficients.

Consequently, parallel efficiency and overall performance depend heavily on the choice of the iterative solver.

Parallelization for most implicit methods is focused within a single integration step, with the exception of block methods. The development of the parallel algorithm utilizes a decomposition technique carried out in several stages. Initially, the theoretical potential for parallelizing the sequential method is assessed using the concept of unlimited parallelism (where the number of processors is considered unlimited and the interprocessor connection is a complete graph).

Subsequently, a mapping of the resulting computational scheme onto the selected processor network topology is constructed. The final stage involves an assessment of the actual parallel performance based on standard metrics: speedup (S_p), efficiency (E_p), overhead (T_o), and scalability (f_E). The communication time component for different data exchange operations and network topologies is estimated based on a new complexity estimator developed for hybrid architectures using the linear Hockney model [16-18] in the message or packed passing mode.

4.1. Parallel SDIRK methods for distributed memory

The implementation of the *s*-stage SDIRK method is considered within the framework of a parallel system with distributed memory and a 1D-torus (ring) topology consisting of p processors. A distinctive characteristic of SDIRK methods is the structure of the staged system of equations:

$$\overline{k}_i=\overline{f}\left(t_n+c_ih;\overline{y}_n+h\sum_{j=1}^{i}a_{ij}\,\overline{k}_j\right),\ i=\overline{1,s}, \tag{4}$$

$$k_{ij}=f_j[x_n+c_ih;\ y_{n,1}+h\sum_{l=1}^{i\text{-}1}a_{il}\,k_{l1}+h\gamma k_{i1},\ \ldots,\ y_{n,m}+h\sum_{l=1}^{i\text{-}1}a_{il}\,k_{lm}+h\gamma k_{\text{im}},] \tag{5}$$

where i- is the number of stage coefficient, j- is the number of component.

First, the direct Newton method is chosen as the method for solving the system of stage coefficients. Since the SDIRK-methods - diagonally implicit, then each stage is executed sequentially, but the decision of the SNAE at each of the stages is distributed between the processors. General scheme of parallel algorithm consists of the following steps. Division of data by processors (initialization): vector of unknowns Y is divided into p blocks, each processor represents a subsystem the size of $\lceil m/p \rceil$, Jacob's matrix (or its proximity) is distributed according to the structure of connections. Iterative cycle over time ($t_n \to t_{n+1}$), for each stage $i=\overline{1,s}$, the following is performed: calculation of the right side - each processor calculates its part of the vector of stage sums $\overline{y}_n+h\sum_{j=1}^{i}a_{ij}\,\overline{k}_j$. Next, the implicit equations are solved using Newton's method, which involves calculating residuals and solving a system of linear algebraic equations. In this case, the following data exchange operations are carried out: processors transmit data to neighbors to calculate the function f_i. For calculating the residual, as well as two operations: block cyclic reduction and all-to-all transmission in a ring. Next, each processor calculates new values of the stage coefficients and solutions. The advantages of the method include high scalability in terms of system size. The disadvantages determine a weak degree of time parallelization. The fact that the Jacobian often has a block-diagonal or strictly banded structure should be taken into account when considering ring topology. Note that all stages of the method yield the same matrix (I-$h\gamma J$), and matrix factorization (LU-decomposition) is performed once per cycle, which also saves an hour of data transfer time.

4.2. Iterative Parallel Solvers

The primary computational load in the implementation of implicit numerical methods lies in solving the systems of stage coefficients, $\overline{k}_i, i=1,\ldots,s$.

Typically, these are systems of nonlinear algebraic equations solved either by the fixed-point iterations or by methods of the Newton family.

It should be understood that the comparison of parallel implementations of these two iterative methods in each specific case significantly depends on the problem being solved, the properties of both the numerical method and the parallel architecture. Nevertheless, the theoretical analysis of

parallel algorithms and experimental studies on test problems allow us to generalize the results and draw the following conclusions. The fixed-point (FP) method is generally inefficient for solving the nonlinear systems arising in DIRK/SDIRK methods applied to stiff problems, even in parallel computing environments.

The FP-method converges only when the step h is very small (the mapping compression condition), which is unacceptably small for stiff systems. This completely negates the advantages of implicit methods (DIRK/SDIRK), which are designed to perform larger steps in stiff domains. As a result, the increased number of integration steps required by the FP method often leads to higher overall computational costs than Newton-type methods, which use large step sizes. The advantages of simple iteration include a simple, straightforward implementation and the ease of incorporating integration step control mechanisms based on various local error estimation methods. Furthermore, parallelization of a simple iterative method can be effective for non-stiff or moderately stiff systems of extremely high dimensionality and parallel architectures with distributed memory.

The direct classic Newton-SDIRK algorithm consists of the following key steps:

- Initial approximation. Typically: $k_{ij}^{(0)}=f_j(t_n, Y_n) \vee \overline{0}$ or extrapolation from the previous step.
- Jacobian formation. Calculate of the system matrix $J_i=I-ha_{ii}\frac{\partial F}{\partial y}$.
- Linear system solution. Determination of the correction vector ΔK (e.g., via LU-decomposition).
- Update $K^{(l+1)}=K^{(l)}-\Delta K^{(l)}$.
- Convergence check. Iterations terminate when $\|\Delta K\|<\varepsilon$.

Newton's method offers clear advantages, provides quadratic convergence (compared to linear convergence of simply iteration) and maintains stability in stiff modes, which substantially reduces overall computational costs. Despite its advantages, the direct Newton method possesses high computational complexity, primarily due to the matrix factorization (LU decomposition) stage. In the parallel implementations, this stage also incurs high communication overhead due to the large volume of data synchronization required. Therefore, for solving SODEs of extremely large dimensions, the Jacobian-Free Newton-Krylov (JFNK) method is preferred.

The Newton-Krylov method is a powerful hybrid approach for solving large systems of nonlinear equations that arise at each stage of the SDIRK/ESDIRK methods. Instead of working with the large Jacobian matrix directly, it uses only the results of its multiplication by a vector. Using the Newton-Krylov method: at each stage of the SDIRK, a nonlinear equation is solved. The outer iteration is performed using Newton's method; the residual is found and the solution is updated. The inner iteration, using the Krylov method, solves a system of linear algebraic equations: $Ax=b$ in the Krylov subspace $\{b, Ab, A^2b, \dots, A^kb\}$ instead of inverting the Jacobi matrix [16]. The main feature of the solution method is that there is no need to store the Jacobian matrix J, you just need to be able to multiply it by a vector v, and this is approximated by a finite difference:$Jv \approx [f(y+v\epsilon)-f(y)]/\epsilon$.

From a parallelization perspective, JFNK is highly scalable. Its core operations consist of vector-vector dot products and matrix-vector multiplications, which are significantly more efficient on distributed-memory architectures than LU decomposition. In terms of memory efficiency, JFNK is superior for ultra-large systems (e.g., $m>10^7$), where the Jacobian cannot fit into memory; the method only requires storage for a few Krylov basis vectors.

An analysis of the areas of application of these two methods gives grounds to assert the following:

- Direct Newton is most effective when m is relatively small or when the Jacobian possesses a block-banded structure. In SDIRK, the LU decomposition can be performed once at the first stage and reused across subsequent stages, minimizing cost.
- JFNK is the optimal choice for very large-scale problems (e.g., complex cybersecurity systems), where the Jacobian is too memory-intensive. On a ring topology, JFNK exhibits

superior performance because communication is primarily required only for function evaluations and vector synchronization.

4.3. Hybrid programming model MPI + OpenMP

The current stage of development of parallel high-performance systems is characterized by the presence of hierarchical hybrid architectures based on different types of memory. One of the key features of such systems is the coexistence of both distributed memory (cluster nodes exchange information via the MPI interface) and shared memory (multi-core architecture: cores inside and outside the node communicate via OpenMP). For extremely large-scale problems (for example, complex critical infrastructure models), this is the only way to overcome memory limitations and obtain solutions. This part of the article considers the application of a two-level hybrid approach to improve the quality of parallelization of SDIRK methods. Analysis of the sequential algorithms for such methods allows us to distinguish two levels of potential parallelization. The first level is the calculation of parts of the system ODE. Uniform distribution of the system of equations, the right-hand sides, arguments, and Jacobian into parts speeds up the processing of the system as a whole and reduces the execution time. The second level is the calculation of different stages of the method. Since stage i depends only on stages $j < i$, but not on subsequent ones, different stages can be processed in parallel provided that the results of previous stages are transferred to the corresponding processes, i.e., functional decomposition is applied. In the algorithmic and software implementation, a hybrid approach was chosen combining the first and second levels of system decomposition and functional decomposition by stages.

This decision is justified by the fact that the number of stages s is usually small, but commensurate with the typical number of threads that can be allocated for a problem of this class. However, with a large system size, this approach does not give significant results. Parallelization of applied methods and linear algebra operations is inefficient for small m due to significant communication overhead. Therefore, a hybrid approach to parallelization is chosen: at the first level, MPI is used to divide the system into parts into processes (processors) and the OpenMP functionality is used to parallelize stages into processor threads. For communication between processes, the “ring” topology is chosen, since it provides the minimum number of connections (each process is connected to only two neighbors) and is naturally suitable for chain data transfer, as well as for ring reduction when calculating the global norm of the residual.

The numerical experiment was performed on the HPE Apollo 9000 Hawk supercomputer, which belongs to the hybrid architecture systems [24]. Despite the apparently better parallelization results, the use of the MPI + OpenMP approach carries certain risks. A parallel distributed solver (forward/backward substitution) is used to solve the system of linear equations. The main load here falls on the interprocessor network, since substitution is a high-level operation. Here, OpenMP helps to optimize vector operations, but threads may be idle, waiting for data from MPI.

4.4. Comparison of parallel implementations of DIRK methods

This section of the article analyzes the quality of the resulting parallel algorithms and corresponding software applications for the DIRK family of methods and the efficiency of their mapping to parallel architectures. It should be noted that even for large systems of equations, parallelization of classical DIRK/SDIRK/ESDIRK methods is not ideal, since parallelism characteristics largely depend on the specific parameters of the problem and method.

On the other hand, the influence of the parallel architecture, especially the structure of the connection between processors and memory elements, cannot be ignored. In this study, theoretical estimates of dynamic parallelism metrics were determined under the assumption of a distributed/hybrid architecture. A 1D-torus was used as the topology for two reasons: the structure of interaction between the DIRK-methods' processes corresponds to the connections in a ring topology, using Gray code, it is possible to construct a logical mapping of the ring to other topologies (mesh, hypercube). Clearly, an efficient parallel solution IVP requires careful consistency between

the numerical solution method and the problem properties. However, analytical estimates of the complexity of subclasses of methods and the conducted experiment allow us to draw some general conclusions.

The main computational overhead in DIRK methods falls on the LU-decomposition (or preconditioner preparation) of the Jacobian matrix. Since the diagonal elements of SDIRK methods are equal to γ, the system matrix $(I - h\gamma J)$ remains unchanged across all stages of a single step. In the parallel version, the matrix is calculated and factorized once on multiple threads/processors and then reused across all s steps. This significantly saves time compared to traditional DIRK methods, which would require rebuilding and factorizing a new matrix at each step. Clearly, ESDIRK methods have the same advantages as SDIRK (same diagonal), but due to the explicit first step, they are slightly more efficient in terms of error control. ESDIRK methods allow for faster step changes and avoid "unnecessary" parallel computations on too small steps. The comparative data presented in Table 1 highlight the distinct advantages of each subclass within the diagonally implicit Runge-Kutta family.

Table 1
Comparative Analysis of PDIRK, PSDIRK, and PESDIRK methods

Characteristic	PDIRK (1)	PSDIRK (2)	PESDIRK (3)
Computational complexity	High	Medium	Medium
Suitability for stiff problems	+	+	+
Adaptive step size	+	+	++
Communication costs	High	Optimal	Optimal
Scalability	Low	High	High
Speedup (efficience)	$\underset{i=1,3}{min} S_i/E_i$	$\underset{i=1,3}{max} S_i/E_i$	$\underset{i=1,3}{medium} S_i/E_i$

While DIRK methods offer maximum flexibility through arbitrary diagonal coefficients, they require multiple LU-factorizations per step, which significantly increases computational complexity. In contrast, SDIRK and ESDIRK schemes achieve superior efficiency by reusing a single factorization across all stages. ESDIRK methods are combine the benefits of an explicit first stage with identical diagonal elements, making them the most balanced choice for adaptive integration and systems requiring high performance. Thus, SDIRK methods with well-chosen iterative and linear solvers are the most efficient parallel methods in DIRK-class methods (Figure 3).

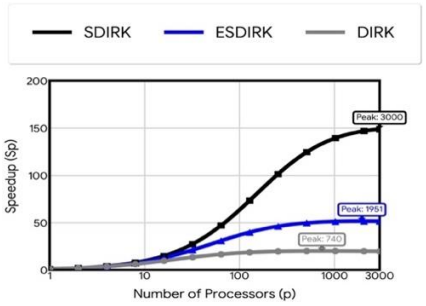

Figure 3: Dependence of the speedup from p for parallel DIRK, ESDIRK, SDIRK methods, order r=4, for MIMD-systems with distributed memory 1D-torus topology.

Especially if parallelize not the method stages but the linear solver within SDIRK (sparse parallel solvers or iterative methods with a parallel preconditioner).

Summary: the SDIRK method minimizes the number of resource-intensive matrix operations, which are the most difficult to scale with increasing the number of nodes.

5. Numerical Results and Discussion

The efficiency and scalability of parallel numerical solution of the initial value problem using DIRK-type methods were investigated in the implementation of the Van der Pol equations [1-2]. The system of m oscillators is a classical model both in the theory of dynamic processes for complex cybersecurity systems and simultaneously in parallel computing.

Note that this problem is not trivial, as it requires considering not only the parameters of the problem and integration method, but also the characteristics of the parallel architecture. The complexity is further enhanced by the fact that all three groups of influences are not independent. First, let's evaluate the influence of the problem parameters, which obviously include the number of oscillators in the system (m), the type of relationship between them (k), and the degree of nonlinearity and stiffness (μ). Then, the influence of different properties of the applied DIRK-methods is considered: type (DIRK/SDIRK/ESDIRK), order (r)/number of stages (s), convergence and stability, the SLAE solver used to determine the stage coefficients. Regarding parallel systems, the study considered modern architectures with both distributed parts and connection topologies, as well as those that take advantage of multithreading.

5.1. Mathematical formulations of the Van Der Pol problem

The Van der Pol equation in the general case has the following form (scalar form):

$$\ddot{x}+\mu(1-x^2)\dot{x}+x=0, \tag{6}$$

and for sufficiently large values of the parameter $\mu, \mu \gg 1$ has varying degrees of stiffness.
For a system of m oscillators, $\forall i=1, \dots, m$, the equations are:

$$\ddot{x}_i-\mu\left(1-x_i^2\right)\dot{x}_i+x_i-\underbrace{k(x_{i-1}-2x_i+x_{i+1})}_{Coupling}=0, \tag{7}$$

where x_i- is the displacement of the i-th oscillator, μ is the nonlinear damping parameter (intensity of self-oscillations), k is the elastic coupling coefficient between adjacent nodes, x_{i-1}, x_{i+1} are the displacements of the left and right neighbors.

For coupled oscillators, where each oscillator interacts with its nearest neighbours, the system of first-order differential equations is written as follows: coordinate equations: $\dot{x}_i=y_i, i \in \{1, \dots, m\}$, velocity equations: $\dot{y}_i=\mu(1-x_i^2)y_i-x_i+Coupling_i$, where the coupling is usually defined as in a chain: $Coupling_i=k(x_{i-1}-2x_i+x_{i+1})$, $i=\overline{1,m}$ at the edges (fixed ends): $Coupling_1=k(x_2-x_1)$, $Coupling_m=k(x_{m-1}-x_m)$. If the oscillators are not coupled ($k=0$), the system decomposes into m independent tasks. This is an ideal case for parallelization, since there are no interactions between the equations. It should be noted that DIRK-methods are used to solve the Van der Pol equation, because they have sufficient stability conditions at moderate integration steps. Figure 4 shows the numerical solution of the Van der Pol system for 5 oscillators. A comparison of two cases is performed: $k=0$ (no coupling) and $k=2$ (strong coupling) for $\mu=4$, that is, for a moderately stiff problem that requires the use of an implicit method, in our case SDIRK4.

For the SDIRK method with $s=4$ stages and order of accuracy $r=4$, the Hairer & Wanner scheme [5] is considered the most efficient. It is L-stable, which is critical for stiff problems like Van der Pol. Note that for $k=0$ the connection between the oscillators is completely absent, each of oscillators evolves only based on its initial values. Oscillators perform identical oscillations in form, but with phase shifts, the distance between the peaks is constant over time. While for $k=2$ the graphs demonstrate the effect of collective synchronization, by the second cycle, the lines begin to merge, because the coupling strength is high enough to overcome the difference in initial states and make the system work as a single macro-oscillator.

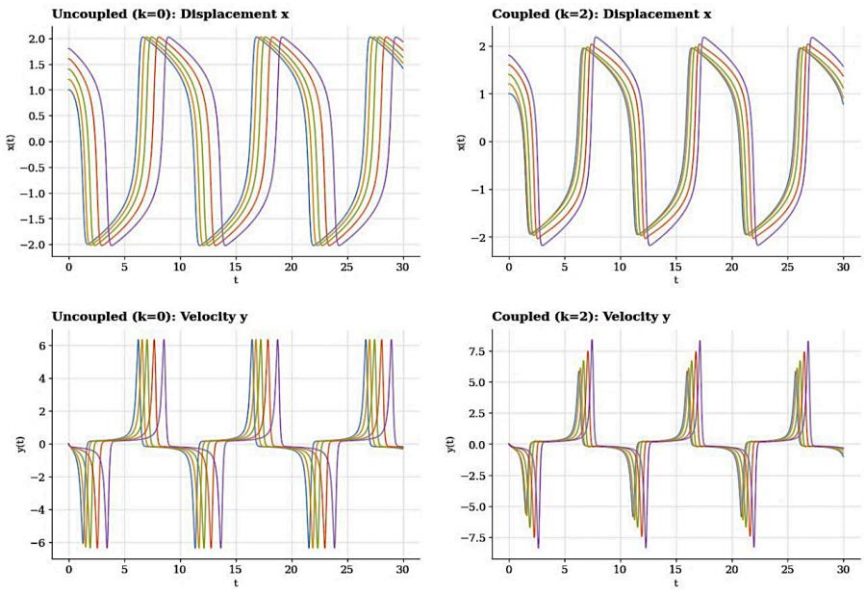

Figure 4: Demonstration of the numerical solution of the Van der Pol problem for 5 oscillators with μ=4 for k=0 and k=2 using the SDIRK4 method [5]. Integration interval: [0.30], initial step size: 0.005, error tolerance per step: 10E−06.

5.2. Efficiency of parallelization of the solution of the Van der Pol problem by SDIRK-methods

The quality of the obtained parallel algorithms is demonstrated using a 4-stage method of the 4th order of the DIRK family, namely SDIRK4 with an iterative solver based on the Newton approach. For analyzing parallel speedup with an arbitrary number of oscillators, the key factor is information connectivity (Figure 5). Uncoupled oscillators demonstrate nearly linear speedup.

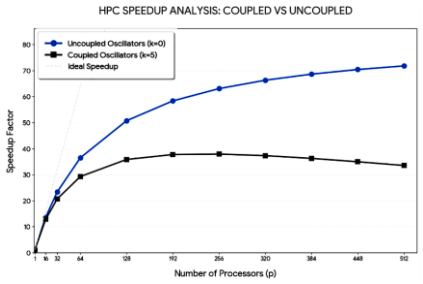

Figure 5: Speedup S_p of the numerical solution of the Van der Pol problem for m=5000 oscillators with μ=4 for k=0 and k=5 using the MPI-realizations of the SDIRK4-method. Integration interval: [0,30], initial step size: 0.005, error tolerance per step: 10E−06.

The computational load is large enough to utilize up to 512 processors. The decline at the end is due to Amdahl's law (1.5% sequential code). The coupled oscillators have a clearly defined peak $(p \approx 128)$, after which the addition of new processors leads to a slow-down in the process of parallel modeling, since the time for data transfer begins to exceed the calculation time. In this case, there is an optimal number of processors, p_{opt}, that directly depends on the ratio of computational complexity to network latency (t_s). Plotting the speedup graph for a system of coupled and uncoupled Van der Pol oscillators allows one to visually evaluate the scalability of the method (SDIRK4+Newton-Krylov)0 and the impact of communication costs.

The overhead on parallelism and costs of managing processors are practically unnoticeable (128 processors provide a 100-fold increase in speedup). The next group of experiments concerns the study of the dependence of the speedup of parallel algorithms for the Van der Pol problem on the

number of oscillators and the number of processors p for different values of k (Figure 6). Based on the experimental results and analytical estimates of speedup, the following conclusions can be drawn for larger scales: uncoupled oscillators demonstrate almost linear scalability.

As for coupled oscillators, for small m, adding processors after 40-60 stops having an effect and even slows down the calculation, it's like a synchronization wall. Only for very large values of m does the computational surface area increase (up to 128 processors), which confirms the fact that the more complex the connection, the greater the processor load must be to justify parallelism. Also, at such scales the implicitness of the method requires solving large-scale SLAE. In parallel mode, this requires efficient iterative distributed solvers (MPI interface), otherwise the «connected» surface will fall earlier.

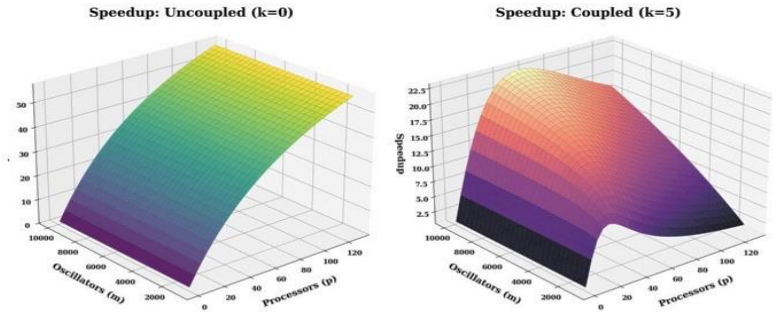

Figure 6: Demonstration speedup S_p of the numerical solution of the Van der Pol problem for m oscillators and p processors with μ=4 for k=0 and k=5 for SDIRK4+Newton-method. Integration interval: [0.30], initial step size: 0.005, error tolerance per step: 10E−06.

To visualize the comparison of Newton and Newton-Krylov methods when solving the Van der Pol problem (SDIRK4, m=10000 equations), graphs of the dependence of the speedup on the number of processors are constructed (Figure 7). For uncoupled oscillators direct Newton method shows the best results. For coupled oscillators direct Newton method loses efficiency sharply as p increases (complexity of the LU-decomposition on the "ring" topology). JFNK method demonstrates stable scaling in both cases. Its main advantage is the absence of the need to store and factorize the Jacobi matrix, which is critical for coupled systems of high dimension. Efficiency for connected systems: the right graph shows that the hybrid model is significantly more stable as p increases. OpenMP processes connections between oscillators through shared memory within a node, without loading the network. From the point of view of scalability, the hybrid approach makes it possible to use a larger number of cores (up to 64 and higher) while maintaining an efficiency of more than 70%, while pure MPI begins to "saturate" due to network latency.

Experimental results show that the Jacobian-Free Newton-Krylov solver is the most efficient candidate for a hybrid MPI + OpenMP implementation. In a ring topology, the scalability of the direct Newton method is fundamentally limited by the LU factorization phase, which requires extensive communication. In contrast, the JFNK approach shifts the bulk of the computational burden to the computation of the F-function, which is highly localized. By using OpenMP threads for these computations within each node, the hybrid model achieves a significant reduction in the total number of MPI messages and global synchronization points. This is especially noticeable when increasing the number of threads to 8 per process, where reduced network latency and efficient use of shared-memory vector operations enable the JFNK solver to outperform the direct method by approximately 35–50% in parallel efficiency for strong coupled systems.

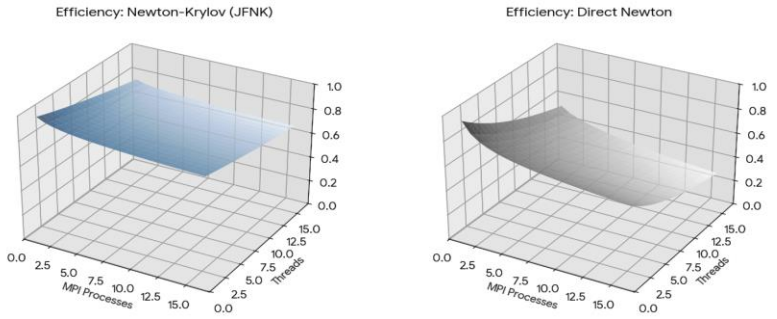

Figure 7: Efficiency $E(p)$ of the numerical solution of the Van der Pol problem for m oscillators from number of processors and threads with μ=4 for k=5 using SDIRK4 based on the Newton or Newton-Krylov methods.

6. Conclusions and Future Research

This paper provides a comprehensive analysis of parallel implementations of modern DIRK methods, focusing on SDIRK and ESDIRK subclasses for parallel systems. The study substantiates the efficiency of SDIRK schemes for solving systems with varying degrees of stiffness, demonstrating an optimal trade-off between absolute stability and computational complexity. Within the DIRK framework, SDIRK methods exhibit superior parallel efficiency when integrated with the classical direct Newton method, primarily due to enhanced load-balancing. ESDIRK methods offer comparable performance. Their hybrid structure, including both explicit and implicit stages, contributes to better step size control, but at the expense of increased interprocessor communication overhead.

One of the main results of this study is the comparative evaluation of iterative solvers: simple iteration, the classical Newton method, and the Jacobian-free Newton-Krylov approach**,** implemented within a hybrid MPI+OpenMP framework. Experimental validation was conducted using the Van der Pol oscillator as a benchmark, considering both uncoupled and coupled oscillator configurations**.** For uncoupled oscillators**,** where the system's Jacobian is block-diagonal, the direct Newton method demonstrated peak performance due to the minimal cost of localized LU-factorization. However, for coupled oscillators**,** which introduce complex off-diagonal dependencies and increased communication requirements in a ring topology, the JFNK method showed significant advantages. By avoiding the explicit construction of the Jacobian and reducing memory overhead, JFNK achieved superior scalability as the coupling density and system size increased.

In contrast, the simple iteration method proved highly inefficient for the Van der Pol problem. Despite its inherent parallelism, it exhibited poor convergence rates, and in cases of high stiffness, complete divergence, making it unsuitable for such benchmarks. Analytical models of time complexity were established and validated through experimental benchmarking, demonstrating high correlation between theoretical estimates and empirical results. The findings confirm that for highly stiff, coupled systems, the hybrid MPI+OpenMP implementation of the JFNK+SDIRK method provides the most robust and scalable solution, ensuring accelerated convergence and numerical stability. The architecture of the parallel algorithm SDIRK has been developed, which is based on a combined decomposition: distribution of the system of equations between computing nodes (MPI level) and parallel processing of the internal stages of the method (OpenMP level) with a ring communication topology.

Comprehensive testing of the created parallel software has been performed on the high-performance HPE APOLLO 9000 architecture (HLRS supercomputer centre, Stuttgart) and in emulation mode. The constructed dependences of speedup and efficiency on the number of allocated processes and threads clearly demonstrated the limits of scalability of the algorithms and confirmed the high performance and computational efficiency of the proposed hybrid approach when solving high-dimensional systems.

Scientific Novelty:

- Hybrid parallelization model. A new computational model integrating intra-step functional parallelization with spatial decomposition on a ring topology, specifically optimized for SDIRK/ESDIRK methods, enables efficient modeling of large-scale cyber-physical systems in real time.
- Optimization of parallel iterative solvers. Development of an analytical adaptive switching mechanism between direct Newton and Newton-Krylov solvers during the integration process, based on real-time monitoring of the system's stiffness and coupling density for minimizing total overhead.
- Advanced complexity modeling. Analytical time-complexity models for SDIRK/ESDIRK algorithms that account for multi-level parallelism (MPI + OpenMP) and iterative convergence rates under dynamic, unpredictable workloads.

The practical significance of the research findings is that:

- The developed parallel algorithmic and software toolkit provides a robust framework for simulating nonlinear chaotic dynamics within critical infrastructures.
- Parallel diagonally implicit methods combined with adaptable iterative solvers enable the software to effectively integrate not only problems of different stiffness but also those with singularities.
- The practical viability and computational stability of this approach were validated using the classic Van der Pol stiff benchmark problem, confirming its suitability for modeling systems of critical infrastructure and risk reduction.

The main directions of future research consist of extending the hybrid MPI+OpenMP model to heterogeneous CPU/GPU clusters; the investigation of specialized distributed preconditions (such as Schwarz) within the different typologies to further accelerate the convergence of the Newton-Krylov method; the development and study of parallel stabilized explicit Runge-Kutta methods with extended stability regions.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT-4 and Gemini in order to: Grammar and spelling check, Improve writing style. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication’s content.

References

[1] E. Hairer, S. P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, Springer Science & Business Media, 2008.

[2] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems, SpringerVerlag, 2010.

[3] T. Mitsui, Guang-Da Hu, Introduction in Numerical Analysis of Ordinary and Delay Differential Equations, 2023. URL: https://doi.org/10.1007/978-981-19-9263-6_1.

[4] J.C. Butcher, Numerical Methods for Ordinary Differential Equations, Wiley, 2016. URL: https://doi.org/10.1002/9781119121534.

[5] C. A. Kennedy, M. H. Carpenter, Diagonally implicit Runge–Kutta methods for ordinary differential equations: A review. NASA TM-2016-219173, 2016. URL: https://ntrs.nasa.gov/citations/20160005923.

[6] Y.A. Sabawi, M.A. Pirdawood, A.D. Khalaf, Singly diagonally implicit Runge–Kutta methods for solving stiff ordinary problems. AIP Conference Proceedings. 2023. Vol. 2457. DOI: https://doi.org/10.1063/5.0118644.

[7] J. B. Jørgensen, M. R. Kristensen, P. G. Thomsen, A family of ESDIRK integration methods, 2018. URL: https://arxiv.org/abs/1803.01613.

[8] Y. Alamri, D. I. Ketcheson, Very high-order A-stable stiffly accurate diagonally implicit Runge-Kutta methods with error estimators. Journal of Scientific Computing, 2024. URL: https://link.springer.com/article/10.1007/s10915-024-02627-w.

[9] M. H. Carpenter, C. A. Kennedy, J. M. Derlaga, Intrastep stage-value predictors for DIRK methods. NASA TM, 2024.

[10] N. Ghawadri, N. Senu, F.A. Fawzi, F. Ismail, Z.B. Ibrahim. Diagonally Implicit Runge–Kutta Type Method for Directly Solving Special Fourth-Order Ordinary Differential Equations with Ill-Posed Problem of a Beam on Elastic Foundation. Algorithms, 2019, 12(1), 10. URL: https://doi.org/10.3390/a12010010.

[11] A. H. D. Andersen, J. B. Jørgensen, A comparative study of ESDIRK methods in optimal control, 2024. URL: https://doi.org/10.48550/arXiv.2402.04667.

[12] D. S. Blom, P. Birken, H. Bijl, et al. A comparison of Rosenbrock and ESDIRK methods combined with iterative solvers for unsteady compressible flows. Adv Comput Math 42, 1401–1426 (2016). URL: https://doi.org/10.1007/s10444-016-9468-x.

[13] L. Wang, M. Yu, A comparative study of implicit Jacobian-free Rosenbrock-Wanner, ESDIRK and BDF methods for unsteady flow simulation with high-order flux reconstruction formulations, 2019. URL: https://doi.org/10.13016/M2FAHX-23V1.

[14] L. Wang, M. Yu, Comparison of ROW, ESDIRK, and BDF2 for Unsteady Flows with the High-Order Flux Reconstruction Formulation, 2020. URL: https://doi.org/10.1007/s10915-020-01222-z.

[15] D. A. Knoll, D. E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, 2004. URL: https://www.inf.ufes.br/~luciac/comcie/JFNK-Knoll.pdf.

[16] A. Grama, A. Gupta, V. Kumar, G. Karypis, Introduction to Parallel Computing, 2nd. ed., Addison Wesley, 2003.

[17] X. Sun, Scalability versus Execution Time in Scalable Systems, Journal of Parallel and Distributed Computing, 62.2 (2022): 173–192. URL: https://doi.org/10.1006/jpdc.2001.1773.

[18] Donald Ene, V. I. Anireh, Performance Evaluation of Parallel Algorithms, SSRG International Journal of Computer Science and Engineering, 9.6(2022): 10-14.

[19] L. P. Feldman, I. A. Nazarova, Modern parallel methods for numerical solution of the Cauchy problem: monograph, Donetsk: DonNTU, 2013. ISBN 978-966-377-106-9.

[20] I. A. Nazarova, O. A. Dmitrieva, Parallel calculations, Pokrovsk: DonNTU, 2020. ISBN 978-966-377-237-0.

[21] O.A. Dmitrieva, N.G. Guskova, E.O. Bashkov, I.A. Nazarova, Numerical methods for modeling dynamic objects in multiprocessor systems, Pokrovsk: Donetsk National Technical University, 2020. ISBN 978-966-377-228-8.

[22] I. A. Nazarova, Features of parallel implementation of embedded implicit one-step numerical methods for solving multidimensional stiff Cauchy problem, Scientific works of DonNTU. Series: IKOT, 2 (31): Pokrovsk, DonNTU, 2020. P. 83-93.

[23] I.A. Nazarova, I.V. Yarosh, Control of the integration step in parallel solution of multidimensional stiff Cauchy problems // Scientific works of Donetsk National Technical University. Series: Informatics, Cybernetics and Computer Science, Lutsk: DonNTU, 1.38 (2024): 19-28. URL: https://doi.org/10.31474/1996-1588-2024-1-38-19-28.

[24] I. A. Nazarova, A. O. Chub, Implementation of multi-point methods for solving stiff Cauchy tasks for HPE APOLLO 9000 HAWK, Telecommunications, automation, computer-integrated technologies: collection of all-Ukrainian reports scientific and practical conference, 2024 / Drohobych: DonNTU, 2024. P.94-97.

[25] I. A. Nazarova, T. V. Altukhova, M. S. Kunda. Efficiency analysis of parallel implementations of one-step embedded methods for solving large-scale Cauchy problems with adaptive step size // Scientific works of DonNTU. Series: IKOT. 1(42), 2026. P. 5-18. URL: https://doi.org/10.31474/1996-1588-2026-1-42-5-18.