Serhi RENDZINYAK[1], Roman KHOLODNIAK[1], Vasyl KORUD[2], Dmitro TRUSHAKOV[3], Oleksandr KOZLOVSKYI[4]

Faculty of Electronics and Computer Technologies, Ivan Franko-National University of Lviv (1), Institute of Power Engineering and Control System Lviv Polytechnic Natonial University, Lviv (2), Department of Automation and Control in Technical Systems Central Ukrainian National Technical University, Kropyvnytskyi (3), Department of Electrical Systems and Energy Management Central Ukrainian National Technical University, Kropyvnytskyi (4)
ORCID: 1. 0000-0003-4544-4871; 2. 0000-0002-1289-3534; 3. 0000-0003-0326-2383; 4. 0000-0001-6885-5994

doi:10.15199/48.2025.03.56

Research on the efficiency of parallel computations in relaxation methods for the analysisof dynamic systems

Abstract. *Peculiarities of the use of parallel computing for the analysis of dynamic systems are studied using the example of the Lotka-Volterra "predator-prey" model. Such relaxation methods as the Jacobi, Gauss-Seidel, SOR and their parallel implementations using Python libraries and modules are considered. The analysis of the efficiency and accuracy of these methods, and the influence of the model parameters on the calculation results, was carried out. A graphical interface for visualization and study of population dynamics has been developed. The obtained results show the capabilities of parallel computing to accelerate and improve the accuracy of modeling complex dynamic systems that consist of parts with deep connections.*

Streszczenie. *Specyfika wykorzystania obliczeń równoległych do analizy układów dynamicznych jest badana na przykładzie modelu „drapieżnik-ofiara" Lotki-Volterry. Rozważane są takie metody relaksacji jak Jacobiego, Gaussa-Seidela, SOR i ich równoległe implementacje z wykorzystaniem bibliotek i modułów Python. Przeprowadzono analizę efektywności i dokładności tych metod oraz wpływu parametrów modelu na wyniki obliczeń. Opracowano graficzny interfejs do wizualizacji i badania dynamiki populacji. Uzyskane wyniki pokazują możliwości obliczeń równoległych w celu przyspieszenia i poprawy dokładności modelowania złożonych układów dynamicznych, które składają się z części o głębokich połączeniach. (**Badania nad efektywnością obliczeń równoległych w metodach relaksacyjnych do analizy układów dynamicznych**)*

Keywords: parallel computations, relaxation method, dynamic system, Lotka-Volterra equations
Słowa kluczowe: obliczenia równoległe, metoda relaksacji, system dynamiczny, równania Lotka-Volterra

Introduction

In today's rapidly developing world, processing huge amounts of data and modeling complex systems are increasingly important tasks. One of the basic tools for solving them is parallel computing, which allows you to speed up information processing and solve complex problems significantly.

The application of parallel computing for modeling dynamic systems, which describe the interaction of various system components in time, is especially relevant. Such models are used in various scientific disciplines, including physics, electrical engineering, biology, and ecology.

At the same time, there is an important problem of creating such a task, which would be universal, and on the example of which it is possible to check the capabilities of parallel approaches, verify the developed software tools, and compare their effectiveness.

Such a task can be the study of one of the classic models of dynamic systems, namely, the Lotka-Volterra model [1, 2], which describes the interaction between populations of predators and prey:

$$\begin{cases} \dfrac{dx_1}{dt} = x_1(a - bx_2) \\ \dfrac{dx_2}{dt} = x_2(-d + cx_1) \end{cases} \qquad \square\square\square$$

where x_1 is the number of preys; x_2 is number of predators; a is prey reproduction rate; b is the death rate of prey from predators; d is rate of death of predators from starvation; c is a coefficient reflecting whether the predator will have enough food for reproduction.

This model may be of interest to electrical engineers if accepted x_1 is the number of equipment failures; x_2 is the number of operating personnel; a is damage/failure rate; b is repair of damage by personnel; d is speed of dismissal of personnel from absence from work; c is a coefficient that reflects the need to hire additional personnel.

The Lotka-Volterra model predicts a cyclical relationship between predator and prey populations, since as the number of predators increases, so does the level of prey consumption, which in turn increases the number of predators. However, an increase in the level of consumption causes obvious consequences – a decrease in the population of prey, which leads to a decrease in the number of predators. As the predator population declines, the number of victims will recover. After that, the number of predators can begin to increase, and the cycle begins again. This model is important for understanding ecosystem dynamics and predicting changes in populations of various species.

The Lotka-Wolterra mathematical model differs from the classical system of differential equations in the Cauchy form by the presence of the x_1x_2 component on the right side of the equations. Thus, it is impossible to divide the system (1) into two subsystems or equations with one state variable, which can be connected using two additional matching sources. This was done in works [3, 4], and other well-known works [5-9] in which a diakoptic approach is developed as the partitioning simulation of subcircuits using relaxation methods.

It is not difficult to model system (1) by explicit numerical methods and, accordingly, to analyze each equation in parallel. Since the use of implicit integration methods is assumed here, a false opinion can form that it is impossible to parallelize such a process of simulation of the system (1).

This paper focuses on research and implementation of parallel computations in relaxation methods used to solve the Lotka-Volterra model. Relaxation methods are iterative methods that allow you to find approximate solutions to a system of equations through successive approaches to the result.

Purpose and objectives of the research

The purpose of this work is to develop effective parallel algorithms [10-14] for relaxation methods and their application for modeling the dynamics of populations in the "predator-prey" system. Let's consider different Python

libraries and modules that allow you to implement parallel calculations and compare their efficiency. Also, the relevance of this work is explained by the fact that parallel computing was used for population forecasting according to the Lotka-Volterra model using relaxation methods for the first time.

To achieve the goal, it is necessary:

1) to conduct a study of the effectiveness of relaxation methods (Jacobi, Gauss-Seidel, relaxation method with SOR – Successive Over-Relaxation) for solving the discretized Lotka-Volterra model;

2) to analyze the impact of parallel calculations using various Python libraries and modules (Ray, Dash, Multiprocessing, Joblib, Concurrent.Futures) on the speed and accuracy of the solution;

3) to evaluate the accuracy and convergence of the numerical solutions obtained by relaxation methods, in comparison with the Euler method solution of the problem, which is not divided into subtasks;

4) to demonstrate the capabilities of the developed graphical interface for interactive research of the Lotka-Volterra model and the influence of various parameters on its behavior.

Research methods

A. Discretization of the Lotka-Volterra model by the Euler method

Formally, it is possible to create an equivalence electrical circuit in which parametric resistances R_{12} and R_{21} (Fig. 1) are used instead of matching sources if specialized software has such capabilities,

$$\begin{cases} C_1 \dfrac{du_{C1}}{dt} = -\dfrac{u_{C1}}{R_1} - \dfrac{u_{C1}}{R_{12}} \\ C_2 \dfrac{du_{C2}}{dt} = -\dfrac{u_{C2}}{R_2} - \dfrac{u_{C2}}{R_{21}} \end{cases} \quad \square\square\square$$

where $u_{C1} = x_1$; $R_1 = -1/(aC_1)$; $R_{12} = 1/(bC_1x_2)$; $u_{C2} = x_2$; $R_2 = 1/(dC_2)$; $R_{21} = -1/(cC_2x_1)$. However, for our task, the use of existing software for the simulation of electric circuits is not necessary.

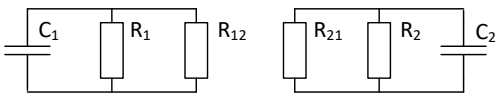

Fig. 1. An equivalence electrical circuit of the Lotka-Volterra model

If the Lotka-Volterra differential equations are discretized by Euler's implicit method, then we obtain a system of nonlinear differential equations (3), which will be solved by relaxation methods

$$\begin{cases} x_1(t_{n+1}) = x_1(t_n) + x_1(t_{n+1})\left(a - bx_2(t_{n+1})\right)dt \\ x_2(t_{n+1}) = x_2(t_n) + x_2(t_{n+1})\left(-d + cx_1(t_{n+1})\right)dt \end{cases} \quad \square\square\square$$

where n is the integration step number.

Thus, a system of difference equations was obtained that describes the change in predator and prey populations at each time step.

B. Implementation of relaxation methods

Python programming codes are developed for the implementation of Jacobi, Gauss-Seidel and SOR relaxation methods, respectively.

1) Jacobi method

$$\begin{cases} x_1\left(t_{n+1}^{i+1}\right) = x_1(t_n) + x_1\left(t_{n+1}^{i}\right)\left(a - bx_2\left(t_{n+1}^{i}\right)\right)dt \\ x_2\left(t_{n+1}^{i+1}\right) = x_2(t_n) + x_2\left(t_{n+1}^{i}\right)\left(-d + cx_1\left(t_{n+1}^{i}\right)\right)dt \end{cases} \quad \square\square\square$$

where i is the iteration number.

In the Jacobi method, the update of the values of the unknowns x_1 and x_2 at each iteration step is independent. This makes it easy to distribute calculations across several processors using such an algorithm:

1. There are divided the calculation into independent tasks for each element x_1 and x_2 at each step of the iteration.
2. Each processor receives a subset of tasks to calculate the new values of x_1 and x_2.
3. The calculated values are then collected and used for the next iteration step.

2) Gauss-Seidel method

$$\begin{cases} x_1\left(t_{n+1}^{i+1}\right) = x_1(t_n) + x_1\left(t_{n+1}^{i}\right)\left(a - bx_2\left(t_{n+1}^{i}\right)\right)dt \\ x_2\left(t_{n+1}^{i+1}\right) = x_2(t_n) + x_2\left(t_{n+1}^{i}\right)\left(-d + cx_1\left(t_{n+1}^{i+1}\right)\right)dt \end{cases} \quad \square\square\square$$

In the Gauss-Seidel method, the values of x_1 and x_2 are updated sequentially. However, it is possible to use parallel computations for each iteration step, computing new values in parallel and then synchronizing the results.

1. There are divided the calculation into parallel tasks for finding x_1 and x_2 at each step of the iteration.
2. Each processor calculates a value for its subset of the data.
3. After the new values are calculated, the results are synchronized and used for the next step.

3) Method SOR

$$\begin{cases} X_1 = x_1(t_n) + x_1\left(t_{n+1}^{i}\right)\left(a - bx_2\left(t_{n+1}^{i}\right)\right)dt \\ X_2 = x_2(t_n) + x_2\left(t_{n+1}^{i}\right)\left(-d + cx_1\left(t_{n+1}^{i+1}\right)\right)dt \end{cases} \quad \square\square\square$$

$$\begin{cases} x_1\left(t_{n+1}^{i+1}\right) = (1-\omega)x_1\left(t_{n+1}^{i}\right) + \omega X_1 \\ x_2\left(t_{n+1}^{i+1}\right) = (1-\omega)x_2\left(t_{n+1}^{i}\right) + \omega X_2 \end{cases}$$

The SOR method involves the use of the relaxation parameter ω, which makes it a bit more difficult to parallelize. However, similar to the Gauss-Seidel method, it is possible to use parallel computations for each iteration step with subsequent synchronization of the results.

1. There are divided the calculation into parallel tasks for calculating x_1 and x_2 at each step of the iteration.
2. Each processor calculates the value for its subset of data considering the relaxation parameter ω.
3. After the new values are calculated, the results are synchronized and used for the next step.

C. Parallelization of calculations

To speed up calculations, parallel calculations using Python libraries and modules (Ray, Dash, Multiprocessing, Joblib, Concurrent. Futures) were applied for each of the relaxation methods [3, 4].

Below are examples of the implementation of parallel calculations for each method.

D. Graphical program interface

To obtain the necessary research results and ease of visualization, a graphical interface has been developed that displays phase and time diagrams, changes in calculation errors at each step, and a field with algorithm parameters.

Results of experimental research

With standard model parameters
α (alpha): 1.0 (Growth rate of prey),
β (beta): 0.1 (Rate predators consume prey),
δ (delta): 0.074 (Growth rate of predators),
γ (gamma): 1.5 (Rate predators die),
X_{10}: 40 (Initial prey population),
X_{20}: 9 (Initial predator population),
dt: 0.01 (Time step),
Steps: 1000 (Number of steps),
the Jacobi method shows almost one hundred percent convergence (Fig. 2).

The Gauss-Seidel method shows minimal deviations (Fig. 3).

The SOR method shows deviations that directly depend on the value of ω, for example, if ω = 1.5 (Fig. 4), then there is a shift to the left; at ω = 0.5, the curve shifts to the right and at ω = 1.0, the values are identical according to the Gauss-Seidel method, which is not surprising, because the SOR method is a modified version of this method.

Overall, after setting the various model parameters, it is safe to say that at small integration step (*dt* = 0.01 or less) and with the number of steps <10000, the Jacobi method and the Gauss-Seidel method show very accurate predictions with small deviations for Gauss-Seidel method and with minimum deviations for Jacobi method, although at large integration step (*dt* > 0.01) or large number of steps >10000, these methods lose their accuracy and show very unstable results. The SOR method, in turn, either shows the same values as the Gauss-Seidel method, or a value with a shift that is directly proportional to the value of ω.

In the context of parallel computing, the situation is somewhat different (Table I). The table of results is given below (ω = 1.5 for SOR method).

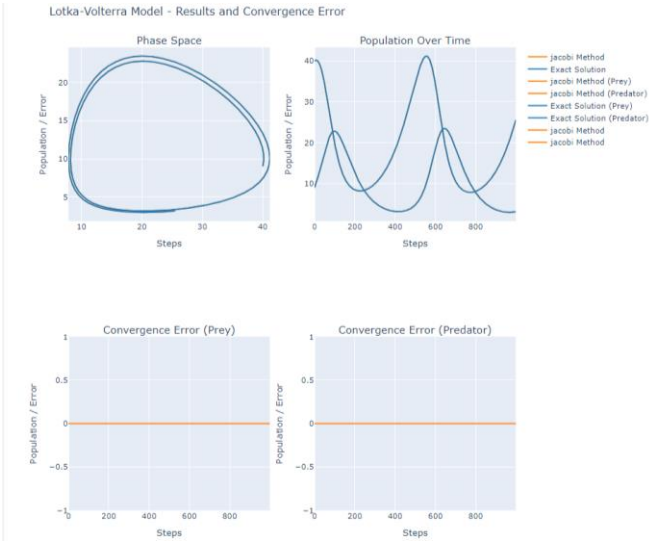

Fig. 2. Results of population analysis by the Jacobian method

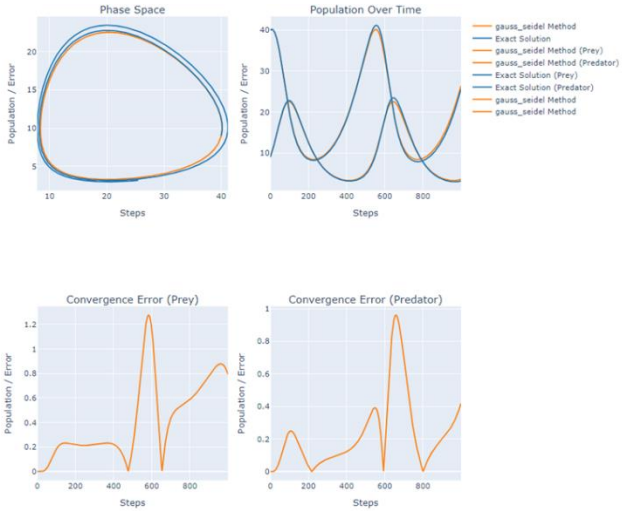

Fig. 3. Results of the population analysis using the Gauss-Seidel method

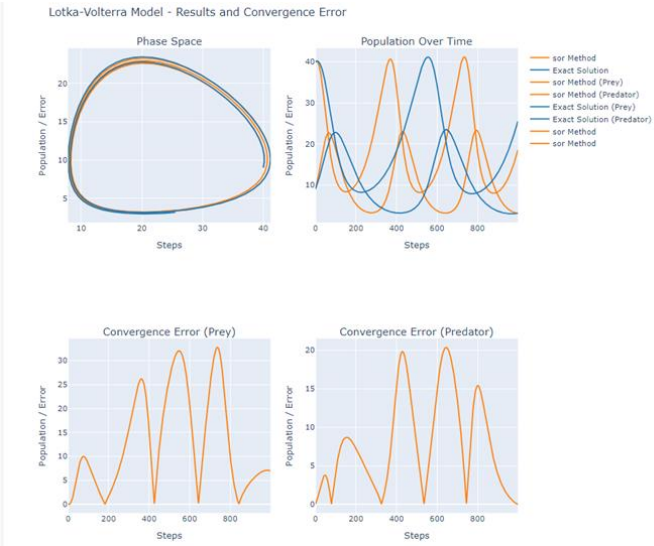

Fig. 4. Results of the population analysis by the SOR method

When comparing different methods of parallel calculations, the following was found: the Dask library shows the same values as the relaxation methods without parallel calculations (None), the Ray library has a significantly increased running time, compared to Dask and None, which is not a good symptom. The Joblib library, when using all processors, shows results 10 times worse than other methods, which indicates its complete unsuitability for relaxation methods in the Lotka-Volterra model. In turn, the Concurrent.Futures module shows a phenomenal reduction in the time required for data processing, which, with the increase in the number of steps and the increase in the values of the steps, does not exceed 1 second, while None and other modules can spend more than 5 seconds on this. There was also an attempt to implement the Multiprocessing module, but it showed results even worse than Joblib, spending consistently more than 2 minutes to process the results, when other methods take no more than 10 seconds. That is why it was decided not to enter the value for Multiprocessing in the table, but to warn that the module is probably the worst way to implement relaxation methods in our model. Therefore, an effective approach to the use of parallel computations has been investigated and shown, as the duration of computations, which can vary by orders of magnitude, depends on it, as well as their importance in the

context of accelerating the analysis of complex dynamic systems.

Table 1. Statistical results of parallelization of algorithms

	Dask	Joblib	concurrent. futures	Ray	None
dt=0.01 steps=1000 method = Jacobi	0.65s	16 s	0.039 s	2 s	0.61s
dt=0.01 steps=1000 method = Gauss	1.1 s	30 s	0.05 s	3.2 s	1.1 s
dt=0.01 steps=1000 method = SOR	1.1 s	31 s	0.05 s	3.2 s	1.1 s
dt=0.01 steps=5000 method = Jacobi	3.2 s	77 s	0.2 s	9.3 s	3.2 s
dt=0.01 steps=5000 method = Gauss	5.4 s	155 s	0.25 s	15.5s	5.4 s
dt=0.01 steps=5000 method = SOR	8.9 s	155 s	0.35 s	16.6s	7.7 s
dt=0.001 steps=2500 method = Jacobi	1.76s	38.4 s	0.13 s	5 s	1.71s
dt=0.001 steps=2500 method = Gauss	3.2 s	77 s	0.15 s	11 s	3 s
dt=0.001 steps=2500 method = SOR	3 s	83 s	0.15 s	8.7 s	3.1 s

Conclusions

In the paper, a deep study and verification of parallel computations using Jacobi, Gauss-Seidel and SOR relaxation methods for the analysis of dynamic systems using the example of the Lotka-Volterra model, which describes the "predator-prey" interaction, is carried out.

A Python application has been developed and implemented that efficiently simulates the Lotka-Volterra system using parallel computations to increase speed and accuracy of the obtained results. Namely, the Jacobi method and the Gauss-Seidel method provide high accuracy at small values of the sampling step and a small number of iterations. However, when these parameters increase, the accuracy of these methods decreases. The SOR method, although more efficient under some conditions, has shown sensitivity to the relaxation parameter, requiring careful tuning to achieve optimal results.

When investigating parallel computations, it was found that the Concurrent.Futures module provides the best computation acceleration, especially as the problem complexity increases. The Dask and Ray libraries also performed well, although Ray requires more resources to configure. The Joblib and Multiprocessing modules proved to be less efficient for this particular task.

REFERENCES

[1] V. Volterra, Lessons on the Mathematical Theory of Struggle for Life. Paris, France: Gauthier-Villars, 1931 (original work).

[2] T. R. Malthus, An assay on the principle of population, as it affects the future improvement of society. London, UK: Science, 1798 (original work).

[3] Rendzinyak Serhiy. Simulation of Spatial Coordinate of Movable Ferromagnetic Solid Object in Induced Magnetic Field by Diakoptic Methods // Przegl d elektrotechniczny, Vol. 2011, No. 5, pp. 146-148.

[4] Modeling of electric power systems based on diakoptic approach and parallel algorithms in modern computer tools // Stakhiv, P., Rendzinyak, S., Hoholyuk, O. Przeglad Elektrotechniczny, 2010, 86(1), pp. 115–117.

[5] Newton A.R., Sangiovanni-Vincentelli A.L. Relaxation-based electrical simulation // IEEE Trans. on Computer-Aided Design. – 1984. Vol. 3, Is. 10. – P. 308-331.

[6] Shima T., Kamatani Y. A circuit simulator based on the waveform-relaxation method using selective overlapped partition and classified latencies // IEEE International Symposium on Circuits and Systems, 7-9 June 1988. – 1988. – Vol. 2. – P. 1651-1654.

[7] Saviz P., Wing O. Circuit simulation by hierarchical waveform relaxation // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – June 1993. – Vol. 12, Is. 6. – P. 845-860.

[8] Peterson L., Mattisson S. Tradeoffs in partitioning for waveform relaxation on multicomputers // IEEE International Symposium on Circuits and Systems, 1-3 May 1990. – 1990. – Vol. 2. – P. 1581-1584.

[9] Lau F.C.M. Waveform relaxation analysis of lossy coupled transmission line sets in cascade // IEE Proc. Circuits, Devices and Systems. – Dec. 1995. – Vol. 142, Is. 6. – P. 373-378.

[10] Robey R., Zamora Y. Parallel and High Performance Computing. New York, NY: Manning Publications, 2021. 704 p.

[11] Foster I. Designing and Building Parallel Programs. London: Pearson Plc, 2019. 408 p.

[12] Mattson T. G., He Y., Koniges A. E. The OpenMP Common Core: Making OpenMP Simple Again. Cambridge, MA: The MIT Press, 2019. 320 p.

[13] Pacheco P., Malensek M. An Introduction to Parallel Programming. Burlington, MA: Morgan Kaufmann Publishers, 2020. 496 p. 15

[14] Chapman B., Jost G., Van Der Pas R. Using OpenMP: Portable Shared Memory Parallel Programming. Cambridge, MA: The MIT Press, 2007. 384 p