TWMS synchronization network simulation with parallel computing

M. S. Batista[1], E. A. Maciel[1], E. C. Panzo[1], F. H. Dermendjian[2], C. M. Batistela[3], J. R. C. Piqueira[3], J. M. Balthazar[2] and A. M. Bueno[1]
[1]*Institute of Science and Technology, São Paulo State University - UNESP, Sorocaba - SP, Brazil*
[2]*School of Engineering, São Paulo State University - UNESP, Bauru - SP, Brazil*
[3]*Polytechnique School, São Paulo University - USP, São Paulo - SP, Brazil.*

Summary. Synchronization networks play important roles in Engineering and Physical systems, allowing information distribution and exchange between the components of computation, communication and complex dynamical systems. In the recent decades the growing need for data communication led to the development of a world wide communication and data transmission network. The clock signal distribution network is an important part of these networks. The Phase-Locked Loop (PLL) is the circuit that synchronizes the local oscillator (clock) to a reference signal (phase and frequency scale). Usually, the PLL networks are build in Mutually-Connected or in Master-Slave topologies. In many cases the Master-Slave is used due to its simplicity and low cost. The nonlinear behavior of PLLs makes the design of clock signal distribution networks a difficult task, therefore, numerical simulation is an important tool. In this work, a Two-Way Master-Slave network is build using parallel computation, aiming to develop a more efficient simulation strategy, and to study the effects of the interacting PLLs running in a parallel computation system.

Introduction

Complex systems can be thought as a set of a large number of connected, usually simple, elements, exchanging signal (information), and producing new behaviors difficult to predicted from the components or boundary conditions. The study of dynamical systems arise from the need to understand phenomena, and to build mechanisms or circuits, such as the complex systems. Synchronization networks are complex systems with growing interest given the need for global communication systems [1, 2, 3].
The PLL is the fundamental component of clock signal distribution networks, and consists of a closed-loop control system that synchronizes a local oscillator, or clock, to a reference signal. The PLL block diagram, shown in Fig. 1, is composed of a Phase Detector (PD), of a Low-Pass Filter, and of a Voltage-Controlled Oscillator (VCO) [4].

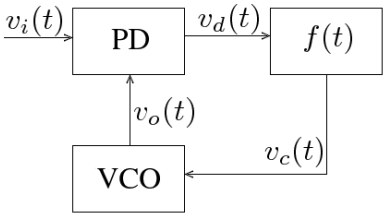


Figure 1: PLL block diagram.

Given the input and output signals in Eqs. 1 and 2, the multiplier type PD generates an error signal $v_d(t)$ with the same sign of the phase difference $\theta_i(t) - \theta_o(t)$.

$$v_i(t) = \mathrm{v}_i \sin(\omega_M t + \theta_i(t)) \tag{1}$$
$$v_i(o) = \mathrm{v}_o \cos(\omega_M t + \theta_o(t)) \tag{2}$$

The LPF controls the VCO phase and frequency, according to the relation in 3, around the free-running frequency ω_M, tracking the input phase $\theta_i(t)$ and filtering noise, jitter and wander frequency fluctuations [2, 5, 6].

$$\frac{d}{dt}\theta_o(t) = k_o v_c(t) \tag{3}$$

Considering the foregoing relations the PLL mathematical model is given by Eq. 4,

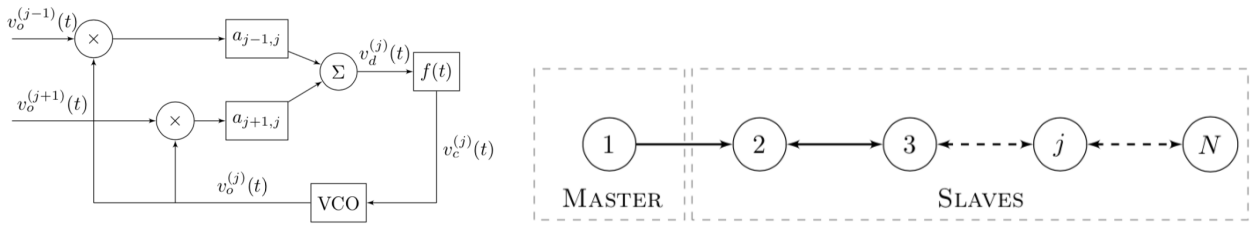
$$L\left[\theta_o(t)\right] + GQ\left[\sin\left(\theta_o(t) - \theta_i(t)\right)\right] = GQ\left[\sin\left(2\omega_M t + \theta_i(t) + \theta_o(t)\right)\right], \tag{4}$$

where $G = \frac{1}{2}k_m k_o \mathrm{v}_i \mathrm{v}_o$ is the loop gain and the operators L and Q depend on the LPF transfer function [1, 2].

TWMS PLL network

There are many strategies for synchronization of clocks. Each strategy depend on the choice for the network topology. In Mutually-Connected networks all the nodes (PLLs) contribute to the phase and frequency scale of the network, however, in Master-Slave networks, the master clock dictates the phase and frequency scales for the whole network. In most cases the Master-Slave strategy is used due to its simplicity and low cost [2, 7].

In One-Way Maste-Slave network the master node alone dictates the phase and frequency scales for the whole network. On the other hand, in the Two-Way Master-Slave strategy, the slave nodes also contribute to the determination of the phase and frequency scales of the network. In Fig. 2(a) the PLL block diagram for a TWMS network is shown. Differently from Fig. 1, the error signal depends on the average phase between the previous and of following nodes, as it shown in Fig. 2(b).



(a) TWMS PLL. (b) TWMS network.

Figure 2: TWMS Clock signal distribution networks.

Following a procedure similar to the previous section the mathematical model of the slave-nodes in a TWMS network is given by

$$L^{(j)}\left[\theta_o^{(j)}(t)\right] + Q^{(j)}\left[\sum_{\substack{\ell=j-1\\ \ell\neq j}}^{j+1} G^{(j,\ell)}\sin\left(\theta_o^{(j)}(t)-\theta_o^{(\ell)}(t)\right)\right] = Q^{(j)}\left[\sum_{\substack{\ell=j-1\\ \ell\neq j}}^{j+1} G^{(j,\ell)}\sin\left(2\omega_M t+\theta_o^{(\ell)}(t)+\theta_o^{(j)}(t)\right)\right] \tag{5}$$

where $j = 2, 3, \cdots, N$, $G^{(j,\ell)} = \frac{1}{2}a_{\ell,j}k_m^{(j)}k_o^{(j)}\mathrm{v}_o^{(j)}\mathrm{v}_o^{(\ell)}$ is the loop gain and the operators $L^{(j)}$ and $Q^{(j)}$ depend on each slave node LPF transfer function.
As it can be noticed from Fig. 2, the phase and frequency scales for the slave nodes depend on the master clock and on the parallel interaction of the slave nodes. Although this is a MS strategy, there is a mutual connection of the slave nodes. The nonlinear and simultaneous interactions generate complex dynamic behaviors. In this case parallel computing is helpful to study and to build more realistic synchonization networks models.

Parallel Programming

Parallel computation consists of solving parts of the system concurrently, that is, in parallel, and in different processors (or cores), in order to reduce the simulation time. In addition, this technique gives more flexibility to build the network [8].
MATLAB's Parallel Computing Toolbox (PCT) allows the developer to work with memory sharing and distributed memory architecture. Within this tool, the development of the "local" infrastructure is built inside a multi-core computer [9]. Generally speaking, hardware architectures are commonly two-, four-, or eight-core versions.
PCT delivers functions to MATLAB such as parallel loop execution, creation and/or manipulation as well as enabling the construction of animal arrays of parallel logic. PCT allows you to use even the cores of the machine on which they all run as they enable interactive development and debugging of code in parallel logic. As a possibility of expanding the work developed on this platform, the largest developed can scale to a number of cores using the MATLAB Distributed Computing Server [10] . Fig. 3 shows a division of the sections into desktop system and cluster computer.
Using MATLAB's PCT, a program with parallel logic can be expressed in three different ways [11]:

1. by the *parfor* feature, which executes a for loop in parallel;
2. by the *spmd* feature, which creates a synchronous process of cooperation;
3. by the *task* feature, which creates multiple, independent programs.

Among the PCT functions, *parfor* (parallel for)-loop stands out, which is very useful in situations that require several iterations in a loop, with a simple function being executed between these iterations, such as Monte Carlo simulations and image processing. In order to run the *parfor*, it is necessary to use PCT, adapting the original code to the new commands of that toolbox [12].
The iterations that run in *parfor* are independent and run in independent instances of MATLAB, running them separately from the operating system process. That way, the function that will be executed at each iteration cannot depend on results that are found in that loop. This is due to the construction of *parfor* inside the PCT, which does not share memory during execution. The application of this solution is feasible for different computers connected via the network because the PCT is able to transform each core of this network into a processing unit of *parfor* iterations [9, 12, 13].

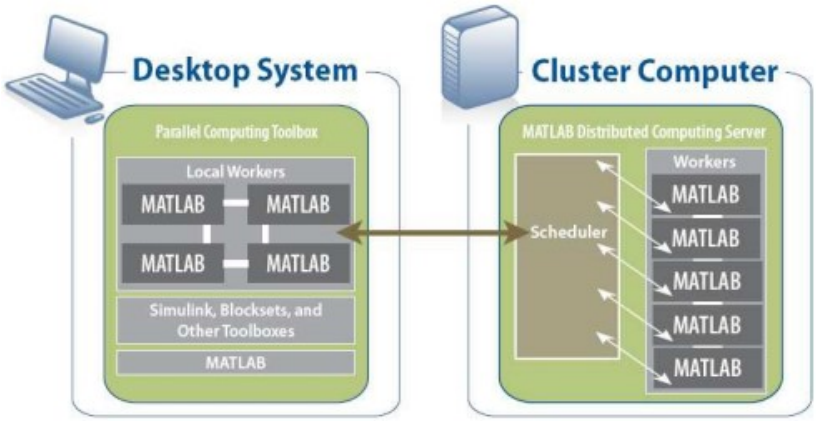


Figure 3: The Parallel Computing Toolbox and MATLAB Distributed Computing Server [10].

Proposed Algorithm

For the development of the work, the necessary calculation steps for the execution of the PLL were followed. Thus, in the first moment, the phase comparison of the input signal and the output signal of the VCO of the PLL itself was performed. This signal is called VD and it feeds the LPF filter of the PLL, in which the fourth-order Runge-Kutta integrative method is used, in which the implementation was based on [14]. Finally, the output signal from the LPF, called VC, is directed to the VCO, closing the loop. The proposed algorithm for running the simulations has three main sections:

1. Definition of initial and simulation parameters – the following questions are answered: What is the input signal; What is the size of the network; What is the start time, end time and sample rate of the simulation; What are the filter parameters;

2. TWMS network simulation – the calculations referring to the constructed TWMS network are performed, in which the calculation of the LPF output is performed in parallel;

3. Construction and presentation of the results obtained – creation of an output file with the simulation results and graphs of the system's behavior.

In this work, the development of parallelism via TPC was chosen through the function *parfor*. Basically in the PLL integration loop, the repetitive structure that advances in time according to the sample rate is a *for* structure. This structure directs the "round" the system is in, that is, it manages the integration step. In this case, we call it the main program.
Within the main program, due to the LPF operation, we need an integration, so that the control signal for the VCO (which in this case is the LPF output) is found. In this step, the *parfor* was applied, that is, the calculation of all PLLS of the TWMS network is performed simultaneously.

Simulation features

The simulated TWMS network is formed by a PLL Master, a PLL Slave, called Slave 1, whose input signal $v_i(t)$ depends on both the Master and Slave 2, and another PLL Slave, called Slave 2, whose signal depends only on the output signal $v_o(t)$ of Slave 1,as described in [15].

Results to step

The simulated TWMS network contains a PLL Master, Slave 1 (having its input signal dependent on the output signals of the PLL Master and Slave 2) and Slave 2 (having its input signal dependent only on Slave 1). The simulations followed the models found in [4, 15, 16, 17].
When analyzing Fig. 4, we see that the PLL Master and Slave 2 are in sync with the input signal because the phase error of θ_o with respect to θ_i is zero. This fact is confirmed by observing Fig. 5 which are results of the steady state of the network. In this case, the Lissajous figure in the post-transient regime demonstrates that the PD input ($v_i(t)$) and the VCO output ($v_o(t)$) of Master and Slave 2 are signals periodic, with the same frequency and quadrature phase, implying synchronization.
Slave 1, in the studied interval, does not reset the phase error of the output in relation to the input. This fact evidenced by Fig. 4 and 5. The signal of Slave 1, when looking at the Lissajous graph, the behavior of the PD input ($v_i(t)$) and the VCO output ($v_o(t)$) Slave 1 has $\theta_o(t)$ equal to 180º, thus being out of phase in relation to $\theta_i(t)$, implying non-synchronization.

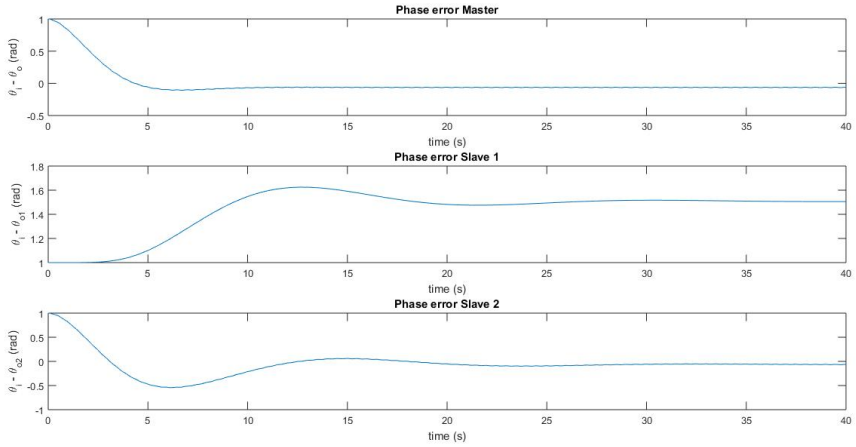


Figure 4: TWMS: Phase error of the step response.

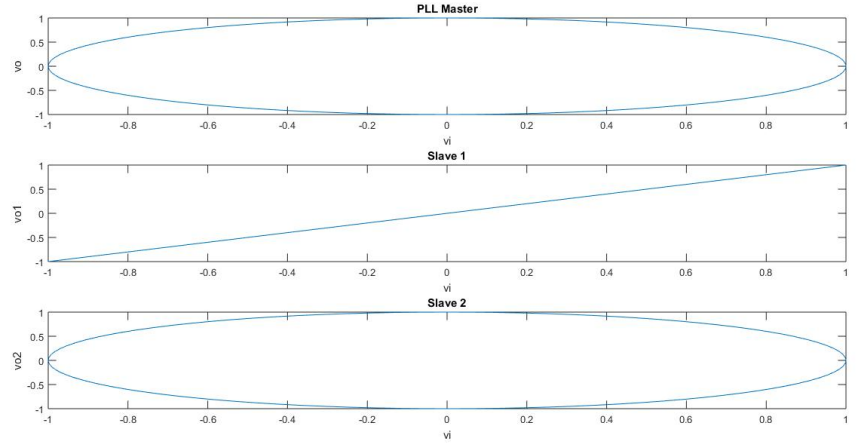


Figure 5: TWMS: Lissajous of the step response.

Results to ramp

When we apply a ramp to the network and observe the behavior of the network, we see that in the TWMS model that the phase of θ_o in relation to θ_i follows the ramp. In Fig 9, we notice that the Master, Slave 1 and Slave 2 replicate the ramp applied to the PD ($v_i(t)$), causing the VCO output ($v_o(t)$) has the same behavior, not synchronizing any of the network PLLs.

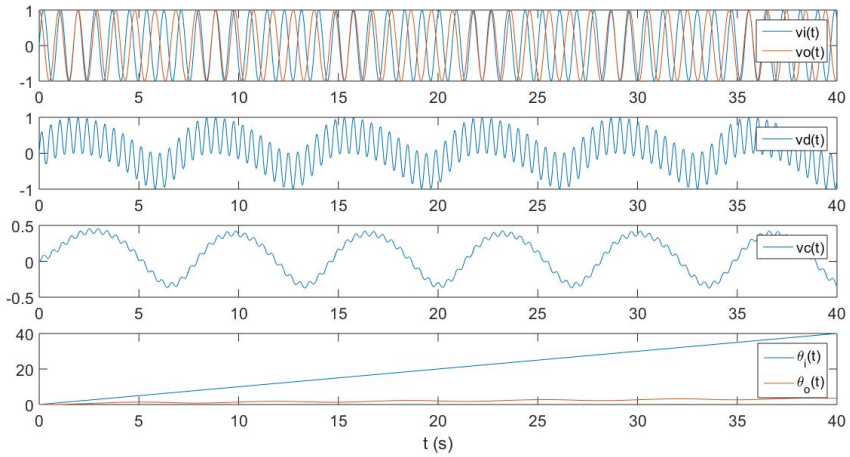


Figure 6: TWMS: Master's response to ramp.

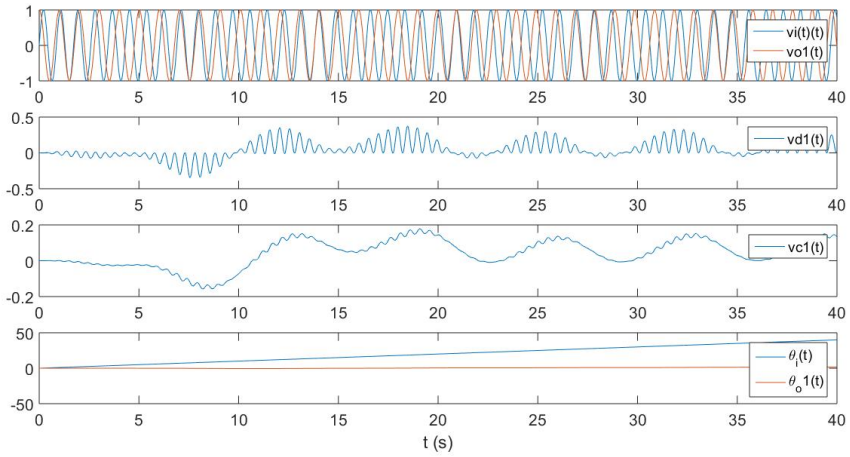


Figure 7: TWMS: Slave 1's response to ramp.

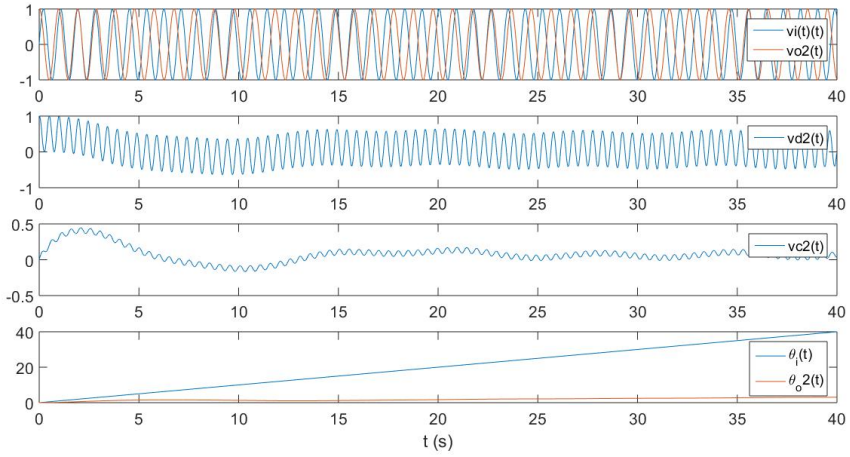


Figure 8: TWMS: Slave 2's response to ramp.

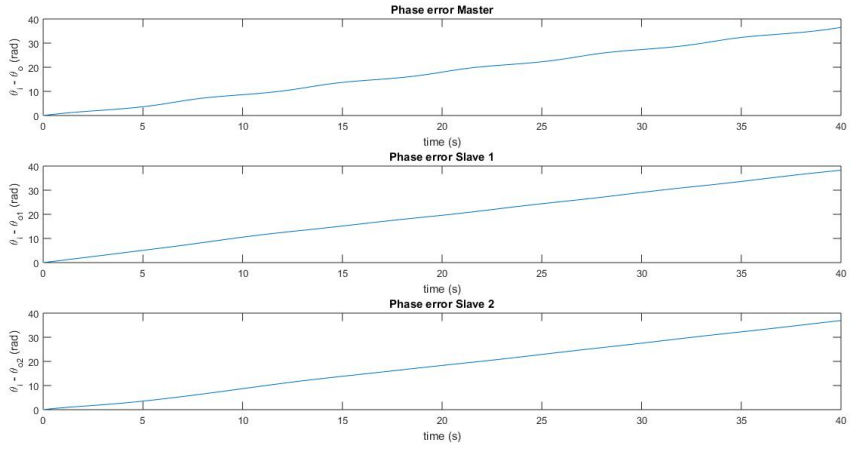


Figure 9: TWMS: Phase error of the ramp response.

Conclusion

In the TWMS network, composed of 3 PLLs, when we apply a step to the input signal phase, we notice that through the phase error and the Lissajous figures that the Master and Slave 2 synchronize with this signal, having a jitter due to the

characteristics of the simulated system. Slave 1 has as a response the non-synchronization with the input signal, a fact observed in the phase error and also by its Lissajous, whose arrangement indicates a phase delay in the order of 180º, as shown in [17].
When applying a ramp to the input phase, it is noted that both the Master and Slaves do not find a control voltage capable of tuning the output signal with the input applied. As shown in [15, 17], the ramp signal causes synchronization failures in all PLLs.

References

[1] W. C. Lindsey, F. Ghazvinian, W. C. Hagmann, and K. Dessouky. Network synchronization. *Proceedings of the IEEE*, 73(10):1445–1467, 1985.

[2] A. M. Bueno, A. A. Ferreira, and J. R. C. Piqueira. Modeling and filtering double-frequency jitter in one-way master slave chain networks. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 57(12):3104–3111, Dec. 2010.

[3] R. S. Sánchez, J. C. Piqueira, and Á. M. Bueno. Comparing phase detectors in analog phase-locked loops. *The European Physical Journal Special Topics*, 230(18):3609–3614, 2021.

[4] A. M. Bueno, A. G. Rigon, A. A. Ferreira, and J. R. C. Piqueira. Design constraints for third-order pll nodes in master-slave clock distribution networks. *Communications in Nonlinear Science and Numerical Simulation*, 15(9):2565–2574, 2010.

[5] ITU-T. *Timing Characteristics of Primary Clocks - Recomendation G.811 ITU-T*, 1997.

[6] ITU-T. *Timing Requirements of Slave Clocks Suitable for Use as Node Clocks in Synchronization networks - Recomendation G.812 ITU-T*, 1997.

[7] A. M. Bueno, A. A. Ferreira, and J. R. C. Piqueira. Fully connected pll networks: How filter determines the number of nodes. *Mathematical Problems In Engineering*, 2009(Article ID 256765):13 pages, 2009.

[8] P. Pacheco. *An Introduction to Parallel Programming*. Elsevier Science, 2011.

[9] D. N. Varsamis, C. Talagkozis, P. A. Mastorocostas, E. Outsios and N. P. Karampetakis. *The performance of the MATLAB Parallel Computing Toolbox in specific problems*. In: Federated Conference on Computer Science and Information Systems (FedCSIS), Wroclaw, Poland. 2012. p. 587-593.

[10] A. Krishnamurthy, S. Samsi and V. Gadepally. *Parallel MATLAB Techniques*. InTech, 2009.

[11] M.Szymczyk and P. Szymczyk. *Matlab and parallel computing.* Image Processing & Communications, v. 17, n. 4, p. 207, 2012.

[12] P. Luszczek. *Parallel programming in MATLAB*. The International Journal of High Performance Computing Applications, v. 23, n. 3, p. 277-283, 2009.

[13] Mathworks. *Parallel Computing Fundamentals web site,* https://www.mathworks.com/help/parallel-computing/parallel-computing- fundamentals.html

[14] D. Tan and Z. Chen. *On a general formula of fourth order Runge-Kutta method*. Journal of Mathematical Science & Mathematics Education, v. 7, n. 2, p. 1-10, 2012.

[15] J. R. C.Piqueira, S. A. Castillo-Vargas, L. H. A. MONTEIRO. *Two-way master-slave double-chain networks: limitations imposed by linear master drift for second order PLLs as slave nodes.* IEEE Communications Letters, v. 9, n. 9, p. 829-831, 2005.

[16] J. R. C.Piqueira. *Master-Slave Topologies with Phase-Locked Loops.* Wireless Communications and Mobile Computing, v. 2020, 2020.

[17] J. R. C.Piqueira and A. C. B. de Godoi. *Clock signal distribution with second order nodes: Design hints.* ISA transactions, v. 115, p. 124-142, 2021.