Acta
Technica
Jaurinensis

Vol. 11, No. 3, pp. 132-149, 2018
DOI: 10.14513/actatechjaur.v11.n3.457
Available online at acta.sze.hu

Parallel Numerical Creation of Phase-space Diagrams of Nonlinear Systems Using Maple

F. Hajdu, Gy. Molnárka

Széchenyi István University, Faculty of Mechanical Engineering, Informatics and Electrical Engineering, Department of Mechatronics and Machine Design
Egyetem tér 1., 9026 Győr, Hungary
e-mail: hajdfl@sze.hu

Abstract: In this paper the numerical creation of phase-plane diagrams in parallel utilizing Maple is presented. One of the most effective method for studying nonlinear systems is the creation of detailed enough phase-plane diagrams. But in the case of large systems it requires huge amount of numerical calculation, which can be accelerated using parallel computers. Here we show some attempts for this using moderate size known problems. We demonstrate that detailed diagrams can be created fast and efficiently with a SIMD model based algorithm even using simple PC-s. We exhibit that the parallel algorithm taken for one- and two-dimensional problems can be expanded for 3D phase-space diagram creation without any loss of efficiency. In this paper a methodology is showed which can be followed in the study of large dynamical systems as well.

Keywords: parallel computing, SIMD, Maple, numerical analysis, nonlinear system modeling

1. Introduction

One effective way for the study of nonlinear differential system of equations is the construction of the detailed enough phase space diagrams 145. For construction of such diagrams we can construct numerical algorithms with many arithmetic operations. However these algorithms are easily parallelizable, so the map of phase space can be drawn with high resolution and very fast using many processors or supercomputers. In our research work we begin with the program of Parallel Maple because it provides algorithms for solving differential equations and two models for parallelization. The paper first introduces the parallelization possibilities with Maple. In the next part in our paper we briefly describe the sequential and parallel algorithms to create the phase-plane diagram. In the next section the tests and results with the parallel program are described in detail. The paper concludes with further development tasks and the summary of the achieved results.

2. Parallelization with Maple

Maple provides two models for parallelization: the thread-based Task programming model, which enables parallelization by executing multiple tasks within a single process and the Grid programming model with starting multiple processes [2]. The main difference between the two models is the memory access: former uses a shared memory (SMP), latter a distributed memory (DMP) [3].

In [4] both of them was tested and compared. As the problem size increased the Grid programming model operated increasingly better, with surprisingly good results at the highest resolution. For solving larger problems the Task programming model seemed less suitable. Not only memory sharing problems occurred, but the speedup and the efficiency also decreased with increasing resolution. Grid programming model was chosen for further research.

Parallelized Maple has already been used for solving arithmetic problems [5], parallel symbolic computation [6], [7], [8], operations with polynomials[9], [10], solving initial value problems for ordinary differential equations [11],[12] and solving nonlinear algebraic problems [13], [14], .

The development of using Maple in parallel started in the 1990s, there were a lot of approach. First there were interfaces between Maple and another programming environment (C/Linda [5], Strand[6], C++[8], Eden[13]) capable of parallelization. There are examples of extended Maple kernels [9] and grid enabling wrappers [16] too. Distributed Maple was an approach of using Maple parallel without any external interface [14]. Nowadays Maple's Grid Computing Toolbox is used for example for parallel operations with polynomials [17]. Parallelized Maple has been used on massively parallel, distributed memory machine [9] and computer clusters too [16].

The detailed description of parallelized Maple applications can be found in [4]. It can be seen that Maple was used mostly on distributed systems, which also confirms our results, that Maple Grid Programming model suits better for our task

3. Numerical creation of a state-space diagrams

3.1. Sequential algorithm

To construct the phase-plane diagrams the system of equation with many initial states was solved numerically. The initial states were stored in a list. For construction of a phase-plane diagram a sequential algorithm was used, which went through all the elements of a list. For numerical solution Maple's *dsolve* command was used with *rkf45* numerical solver [18].

In this study the sequential algorithm was expanded in a way to create 3D state-space diagrams. The sequential algorithm to construct the 3D state-plane diagram in case of *h* system variables and *h* differential equations can be seen in Figure 1.

The initial data for calculation should be specified. They are the initial states, the system parameters, and the system of equations. For defining the initial states sets were given for x1(0)-x2(0)-..-xh(0). Inside the sets x1(0)-x2(0)-..-xh(0) pairs are created, which are stored in a list. With changing the resolution inside sets the number of initial states could be varied.

The next step is solving the system of equations. Maple stores the solution in a procedure (*procrkf45*) [18]. This procedure can calculate the result at a given time. For constructing the phase-plane diagram a sequence is necessary for calculate the results at a given time period (*seq* command). Maple can store the results in a plot. All plots are collected in a list, which can be easily displayed after calculating the results for all of the initial states.

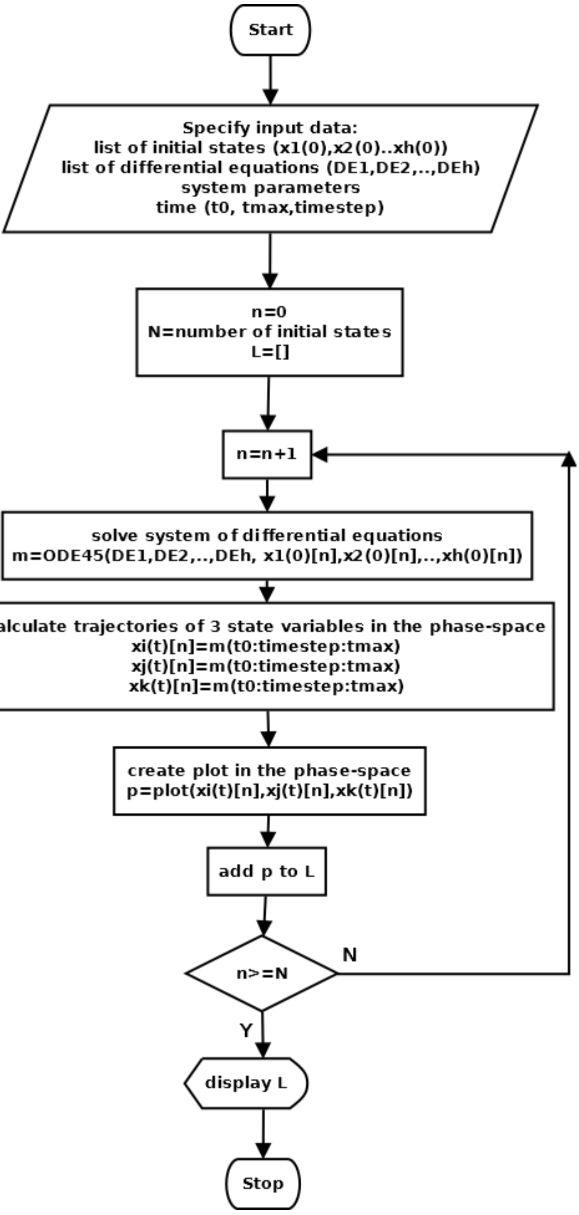
Start
Specify input data:
list of initial states (x1(0),x2(0)..xh(0))
list of differential equations (DE1,DE2,..,DEh)
system parameters
time (t0, tmax,timestep)
n=0
N=number of initial states
L=[]
n=n+1
solve system of differential equations
m=ODE45(DE1,DE2,..,DEh, x1(0)[n],x2(0)[n],..,xh(0)[n])
calculate trajectories of 3 state variables in the phase-space
xi(t)[n]=m(t0:timestep:tmax)
xj(t)[n]=m(t0:timestep:tmax)
xk(t)[n]=m(t0:timestep:tmax)
create plot in the phase-space
p=plot(xi(t)[n],xj(t)[n],xk(t)[n])
add p to L
n>=N
N
Y
display L
Stop

Figure 1. The flow chart of the sequential algorithm

3.2. Parallel algorithm

Constructing phase-plane diagrams in a numerical way can be easily parallelized. There are a lot of different input data (initial states), but the same calculations should be carried out with them. The parallelization therefore can be done according to SIMD model [19].

For parallelization using Maple's Grid Programming model a master node must be created to supervise slave nodes and arrange the data of the results. The slave nodes calculate the results for different initial states and send them to the master node. The flow chart for this model in Maple can be seen in Figure 2. The algorithm was based on [20].

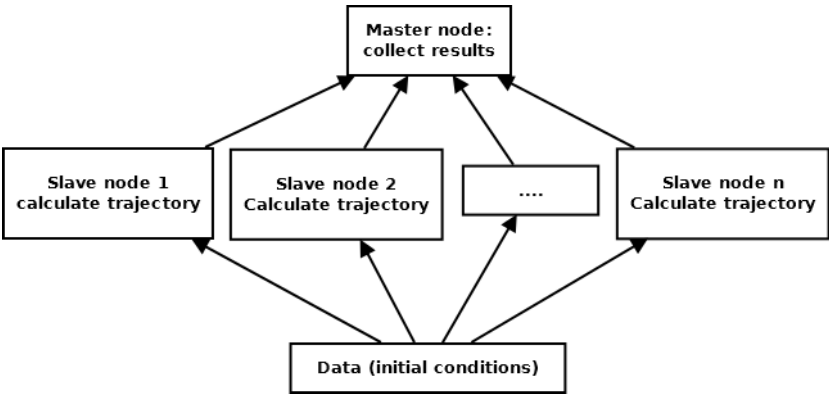
Master node: collect results
Slave node 1 calculate trajectory
Slave node 2 Calculate trajectory
....
Slave node n Calculate trajectory
Data (initial conditions)

Figure 2. Flow chart of the parallel algorithm

4. Tests

The tests were carried out on a PC with an IntelCorei5-4460 @ 3200 Mhz CPU processor and 16 GB RAM. The processor has 4 cores. The equations for calculating the speedup (S) [21] and the relative speedup [22] (R) are:

$$S = \frac{T_s}{T_n} \tag{1}$$

$$R = \frac{S}{n} \tag{2}$$

where T_s is the sequential running time, T_n is the parallel running time and n is the number of cores.

This parallel algorithm was previously tested with the numerical creation of phase-plane diagrams of a Tunnel diode circuit [23]. The results were very promising, a 3 fold speedup could be achieved with a simple PC. In this study the phase-plane diagram of other 2 simple nonlinear systems were created: the Van der Pol oscillator and a biochemical reaction.

Van der Pol oscillator

The Van der Pol oscillator is a simple nonlinear oscillator [1]. From the phase-plane diagram of nonlinear oscillators also a lot of information can be derived, one of the most important one is the limit cycle (can be seen in Figure 1). The differential equation of the system is:

$$\frac{d^2}{dt^2}x(t) = \mu(1 - x(t)^2)\frac{d}{dt}x(t) - x(t) \tag{3}$$

For defining the initial states sets were given for x(0) and dx(0). The sets are [-3; 3] in both cases. This set was chosen in a way that the limit cycle is inside the set.

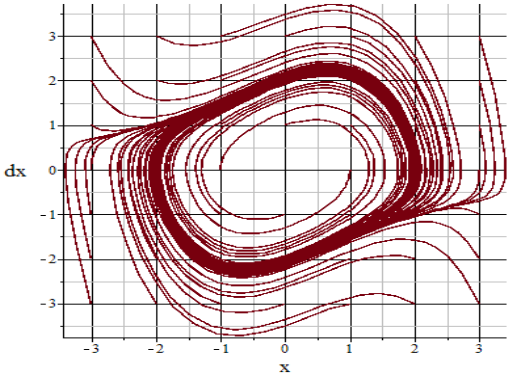

Figure 1. The phase-plane diagram of the Van der Pol oscillator (μ=1)

The results can be seen in Table 1 and in Figure 2.

Table 1. Average calculation times (T), speedup (S) and relative speedup (R) for constructing the phase-plane diagram of the Van der Pol oscillator

Resolution (number of initial states)		Sequential	Parallel (2 cores)	Parallel (4 cores)
0.5 (16)	T	3.439	2.869	1.255
	S		1.199	2.742
	R		0.599	0.685
1 (49)	T	10.686	7.737	3.479
	S		1.381	3.072
	R		0.691	0.768
2 (169)	T	37.19	26.768	11.451
	S		1.389	3.248
	R		0.695	0.812
4 (625)	T	146.422	98.358	42.541
	S		1.489	3.442
	R		0.744	0.860
8 (2401)	T	609.757	384.479	165.828
	S		1.586	3.677
	R		0.793	0.919

In Table 2 it can be seen that around a 3 fold speedup could be achieved with a 0.8 relative speedup in case of 4 cores. As the resolution increased the speedup and the efficiency increased, at the highest resolution over 90% efficiency could be achieved with almost a 4 fold speedup. In case of 2 cores a maximum 1.5 speedup could be achieved with a 0.8 relative speedup. Except the lowest resolution the efficiency was above 60% in all cases. The overhead is less using 4 cores in all cases. Better results could be achieved using 4 cores, therefore the algorithm is scalable. At higher resolution the increase in efficiency was better (>10%) using 4 cores, which means that larger problems are better scalable.

Further tests will be necessary in the future using supercomputers with more cores to examine the scalability of the algorithm in more detail.

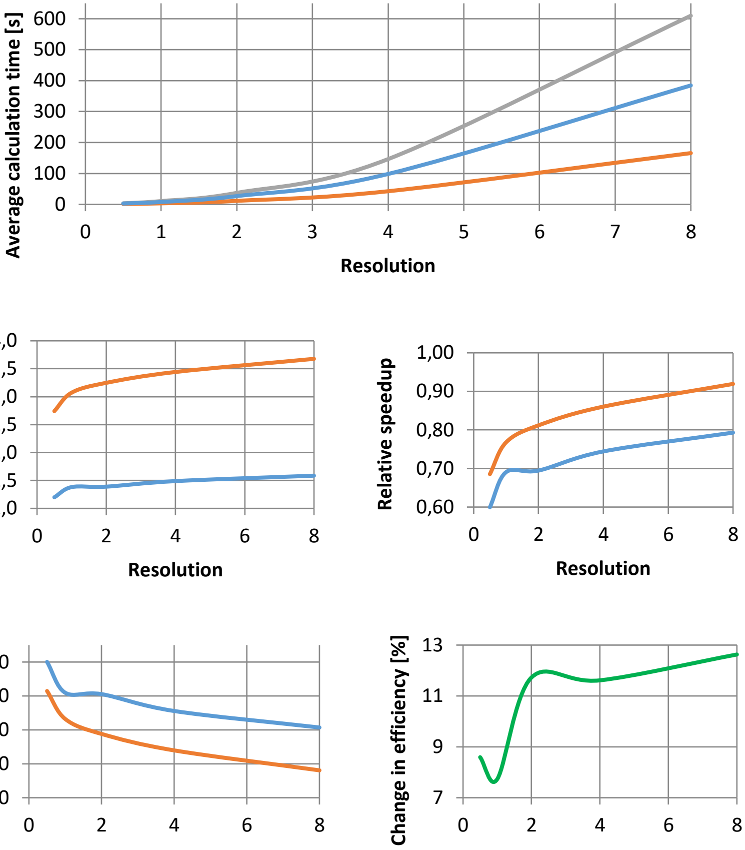

Figure 2. The average calculation time (up, grey: sequential, blue: 2 cores red: 4 cores), speedup (middle left), relative speedup (middle right), overhead (bottom left) and change in efficiency comparing 2 and 4 cores (bottom right) versus the resolution in case of creating the phase-plane diagram of the Van der Pol oscillator

First the increase is faster using both 2 and 4 cores, but after resolution 1 the increase slows down. Next task is to examine the changing in limit cycle as parameter μ is varied in parallel.

Creation of 3D state-space diagram: a biochemical reaction

The described algorithm was expanded for 3D tasks too, which means the creation of the state-space diagram. It was tested with a simple example with 3 variables. It is the model of a biochemical reaction [24]. The differential equation is:

$$\frac{dx(t)}{dt} = \frac{1}{\alpha}\left(x(t) + y(t) - x(t) \cdot y(t) - qx(t)^2\right) \tag{4}$$

$$\frac{dy(t)}{dt} = 2mz(t) - y(t) - x(t) \cdot y(t) \tag{5}$$

$$\frac{dz(t)}{dt} = \frac{1}{r}\left(x(t) - z(t)\right) \tag{6}$$

where x(t), y(t) and z(t) are the species concentrations, *α*, *q*, *m*, *r* are constants. The state-space diagram can be seen in Figure 3. From the state-space diagram it can be seen that a limit cycle exists if α=0.1, q=0.01 , m=0.5 and r=1.

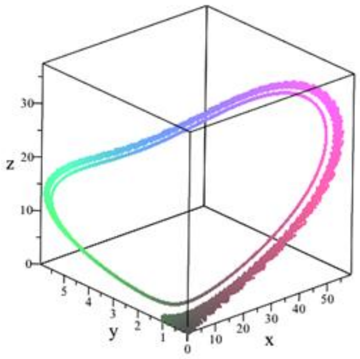

Figure 3. The phase-space diagram of a biochemical reaction

For defining the initial states sets were given for x(0), y(0) and z(0). The sets are [0; 1] for all variables.

The results can be seen in Table 2 and in Figure 4.

Table 2. Average calculation times (T), speedup (S) and relative speedup (R) for constructing the phase-space diagram of a biochemical reaction

Resolution (number of initial states)		Sequential	Parallel (2 cores)	Parallel (4 cores)
0.25 (27)	T	1.353	1.295	0.537
	S		1.045	2.521
	R		0.523	0.630
0.5 (216)	T	10.952	7.347	3.853
	S		1.491	2.842
	R		0.745	0.711
1 (1331)	T	63.478	45.068	18.735
	S		1.408	3.388
	R		0.704	0.847
2 (9261)	T	482.317	-	131.361
	S		-	3.672
	R		-	0.918

It can be seen that around 3 fold speedup could be achieved with 75% efficiency in case of 4 cores. For small resolutions the speedup was less than 3 and the efficiency less than 70%. After resolution 1 an over 3 fold speedup could be achieved with an above 80% efficiency. For the highest resolution a surprisingly good result could be achieved: 3.6 fold speedup with 92% efficiency. In case of 2 cores a maximum 1.5 fold speedup could be achieved with 75% efficiency at resolution 0.5. When the resolution was further increased both of them decreased. At the highest resolution the calculation failed because of lack of memory. The overhead was less using 2 cores at resolution 0.5, which means the algorithm is better scalable at higher resolution. At resolution 1 a 14% increase in efficiency could be achieved using 4 cores.

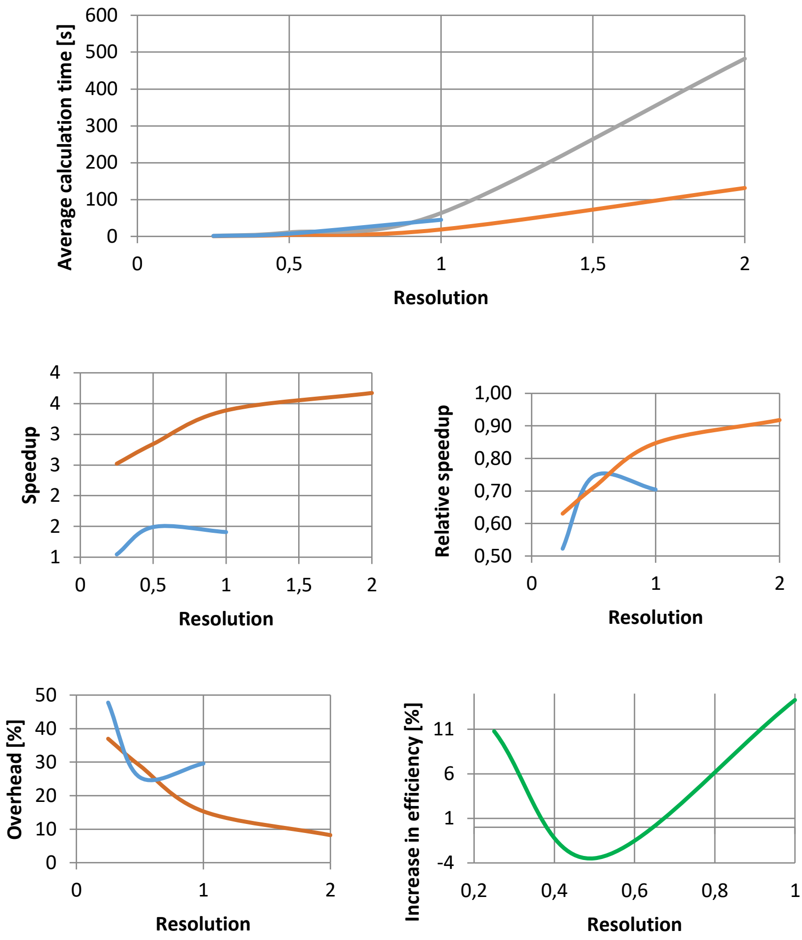

Figure 4. The average calculation time (up, grey: sequential, blue: 2 cores red: 4 cores), speedup (middle left), relative speedup (middle right), overhead (bottom left) and change in efficiency comparing 2 and 4 cores (bottom right) versus the resolution in case of creating the phase-space diagram of a biochemical reaction

The speedup and the efficiency up to resolution 1 is linear using 4 cores. After that is still remains linear, but the slope is smaller. With this test it was verified that this algorithm is suitable also for creating 3D state-space diagrams in parallel very efficiently. The scalability of the algorithm should be examined with more cores in the future.

5. Further development

In this section the further development tasks are described. These are the possibilities of using supercomputers in the future and examining other more complex nonlinear systems with creation of phase-plane and other diagrams in parallel.

Using supercomputers

Our forthcoming research is to measure the speedup effect and examine the scalability of the algorithm on a supercomputer using a much more number of processors [25]. Three possibilities were examined

- using Maple
- connecting Maple to another environment
- use a completely different programming language

As we have limited number of Maple licenses it was discarded to use it on supercomputers.

Another possibility is to connect Maple to other parallel programming language. In the literature there are several examples, like Sugarbrush (Maple+C/Linda) [5],||MAPLE|| (Maple+Strand)[6], FoxBox (Maple + C++) [8], PVMaple (Maple + PVM) [11], Maple+Eden [13], OpenMaple+MPI [17]. Most of these solutions did not spread due their special kernel [14] or were not effective enough as format conversion can be very time consuming [13],[17].

The third possibility was chosen, which is to develop a parallel target program in C++. A demo program was already developed for creating the phase-plane diagram of a Tunnel diode circuit in parallel using OpenMp directives. Compared to parallel Maple another 6 fold speedup could be achieved on the same computer. More about the development of the demo program can be read in [26].

Other tasks

Further development task is to speedup the examination of other nonlinear systems as well and to create further high-performance computing workload diagrams like bifurcation diagrams [27] (seen in Figure 5) in parallel. For simple bistable systems, like the Tunnel diode circuit Maple command *implicitplot* is applicable. For fast creation of bifurcation diagrams of oscillators and more complex systems an iterative program based on [28] can be used. The parallelization of 3D bifurcation diagrams using the simple iterative algorithm has already been achieved [29].

Other future task is to create Poincaré sections [28] and frequency spectrum maps [30] also in parallel to facilitate the numerical examination of more complex nonlinear systems as well.

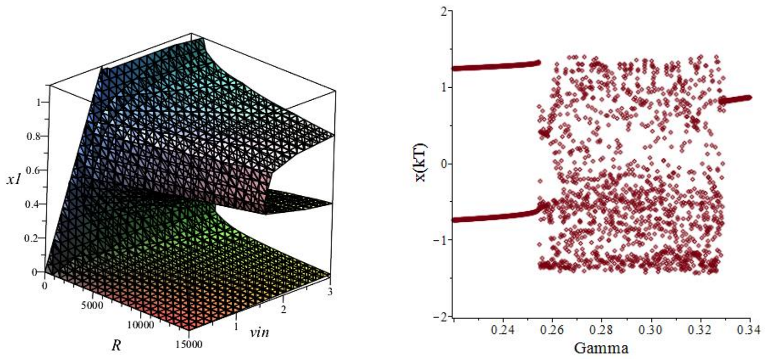

Figure 5. The 2 parametric bifurcation diagram of the Tunnel diode circuit (left)[23] and the bifurcation diagram of the Duffing-Holmes oscillator (right)

6. Conclusion

The numerical creation of the phase-plane diagrams of simple nonlinear systems in parallel was achieved with a SIMD model based algorithm utilizing Maple's Grid programming model. On a 4 core desktop an average 3 fold speedup could be achieved with an average 75-80% efficiency in all test cases. As the problem size increased the parallel program operated increasingly better, with surprisingly good results at the highest resolutions. For simple nonlinear systems a detailed phase-plane and 3D phase space diagrams can be created fast and efficiently even on simple PC-s with limited number of cores. The presented method can be used for numerical examination of more complex dynamical systems in the future.

Acknowledgement

The publishing of this paper was supported by EFOP-3.6.1-16-2016-00017 project.

References

[1] H.K. Khalil, Nonlinear Systems, Prentice Hall, (1999)

[2] Maple help, 2016, [cited 01.10.2016] https://www.maplesoft.com/support/help/Maple/view.aspx?path=ProgrammingGuide/Chapter15

[3] S. Jin, S.E. Oh, J.W. Hong, Performance Evaluation of a Finite Element Code Parallelized with OpenMP, in P. Iványi, B.H.V. Topping, G. Várady, (Eds), Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, Stirlingshire, UK, Paper 36, (2017).
DOI:10.4203/ccp.111.36

[4] F. Hajdu, Gy. Molnárka, Parallelization of Numerical Examination of Nonlinear Systems using Maple, in P. Iványi, B.H.V. Topping, G. Várady, (Eds), Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Civil-Comp Press, Stirlingshire, UK, Paper 30, (2017).
DOI:10.4203/ccp.111.30

[5] B. Char, J. Johnson, Some experiments with Parallel Bignum Aritmetic, in H. Hong (Ed) PASCO'94 Proceedings of First International Symposiumon Parallel Symbolic Computation, World Scientific Publishing Co., Inc. River Edge, NJ, USA, (1994)

[6] K. Siegl, Parallelizing Algorithms for Symbolic Computation using ||MA PLE||, in Proceedings of the fourth ACM SIGPLAN symposium on Principles

and practice of parallel programming, ACM New York, NY, USA, (1993), pp. 179–186.
DOI: 10.1145/155332.155351

[7] K.C. Chan, A. Diaz, E. Kaltofeln, A Distributed Approach to Problem Solving in Maple, in R.J. Lopez (Ed) Maple V: Mathematics and its Applications, Birkhäuser, Boston, (1994), pp. 13-21.

[8] A. Diaz, E. Kaltofeln, FOXBOX: A System for Manipulating Symbolic Objects in Black Box Representation, in "Proceedings of the 1998 international symposium on Symbolic and algebraic computation, ACM New York, NY, USA,(1998), pp. 30-37.

[9] L. Bernardin, Maple on a massively parallel, distributed memory machine, in M. Hitz (Ed) PASCO '97 Proceedings of the second international symposium on Parallel symbolic computation, ACM New York, NY, USA, (1997), pp. 217-222.
DOI: 10.1145/266670.266732

[10] J. Liu, S. Wu, H. Li, X. Yao, F. Du, Parallelization of Characteristic Series, in W. Wang (Ed) Mechatronics and Automatic Control Systems: Proceedings of the 2013 International Conference on Mechatronics and Automatic Control Systems (ICMS2013) , Springer International Publishing, (2013) pp. 363-372.
DOI: 10.1007/978-3-319-01273-5

[11] D. Petcu, PVMaple: A Distributed Approach to Cooperative Work of Maple Processes, in J. Dongara et al (Eds) Recent Advances in Parallel Virtual Machine and Message Passing Interface: Seventh European PVM/MPI Users' Group Meeting, Balatonfüred, Hungary, September 10-13, 2000 Proceedings, Springer-Verlag Berlin Heidelberg, (2000), pp. 216–224.

[12] D. Petcu, Numerical Solution of ODEs with Distributed Maple", in L. Vulkov et al (Eds), Numerical Analysis and Its Applications: Second International Conference, NAA 2000 Rousse, Bulgaria, June 11-15, Revised Papers, (2000), pp. 666-674.

[13] R. Martinez, R. Pena: Building an Interface Between Eden and Maple: A way of Parallelizing Computer Algebra Algorithms, in P. Trinder et al (Eds) Implementation of Functional Languages 15th International Workshop IFL

2003 Edinburgh, UK, September 8-11, 2003. Revised Papers, Springer-Verlag Berlin Heidelberg, (2005), pp. 135-151.
DOI: 10.1007/978-3-540-27861-0_9

[14] W. Schreiner, C. Mittermaier, K. Bosa, Distributed Maple: parallel computer algebra in networked environments, in H Hong (editor) Journal of Symbolic Computation, 35 (3) (2003), pp. 305–347.
DOI: 10.1016/S0747-7171(02)00137-2

[15] D. Petcu, Numerical Solution of ODEs with Distributed Maple", in L. Vulkov et al (Eds), Numerical Analysis and Its Applications: Second International Conference, NAA 2000 Rousse, Bulgaria, June 11-15, Revised Papers, (2000), pp. 666-674.

[16] D. Petcu, M. Paprzycki, D. Dubu, Design and implementation of a grid extension for Maple, Scientific Programming, 13 (2) IOS Press, (2005) pp. 137–149.
DOI: 10.1155/2005/653638

[17] G. M. Díaz-Toca, A. L. Murica, Berkowitz Algorithm in parallel with the Maple Grid Computing Toolbox [cited 28.12.2017] http://dis.um.es/~domingo/08/CD/27May/Posters/ALopez/poster.pdf [accessed 28.12.2017]

[18] Gy. Molnárka, L. Gergó, F. Wettl, A. Horváth, G. Kallós, Maple V and its applications, Springer Hungarica, Budapest, (1996) in Hungarian

[19] R. Cypher, J. L.C. Sanz, The SIMD Model of Parallel Computation, Springer-Verlag New York, (1994), pp. 1-3.
DOI: 10.1007/978-1-4612-2612-3

[20] W.Auzinger, Demonstration of a parallel computation in Maple using the Grid package, TU Wien:, lecture notes, 2016, [cited 25.10.2016] http://www.asc.tuwien.ac.at/compmath/2016/Grid-Demo.pdf

[21] L. Környei, G. Kallós, D. Fülep: Parallel Computations on the Blade Server at Széchenyi István University, Acta Technica Jaurinensis, 3 (1) (2010) pp. 111-126.

[22] G. Lencse, I. Derka: Testing the Speed-up of Parallel Discrete Event Simulation in Heterogeneous Execution Environments In: Veronique Limère , El-Houssaine Aghezzaf (Eds.) Proceedings of the ISC'2013, 11th Annual Industrial Simulation Conference, (2013) pp. 101-107.

[23] Hajdu, Gy. Molnárka: Numerical examination of a system model with a nonlinear component, in TEAM 2016: Proceedings of the 8th International Scientific and Expert Conference, AlumniPress, (2016), pp. 12-16.

[24] E.Miletics, Gy. Molnárka: Taylor series method with numerical derivatives for initial value problems, Journal of Computational Methods in Sciences and Engineering, 4 (1-2) (2004) pp. 105-114.

[25] G.A. Gravvanis, B.E. Moutafis, C.K. Filelis-Papadopoulos, H.G. Theodosiou: Parallel Semi-Aggregation Techniques for Solving Parabolic Partial Differential Equations, in P. Iványi, B.H.V. Topping and G. Várady, (Eds), Advances in Parallel, Distributed, Grid and Cloud Computing for Engineering, Saxe-Coburg Publications, Stirlingshire, UK, Chapter 8, (2017) pp 157-182.
DOI:10.4203/csets.40.8

[26] F.Hajdu, Cs. Hajdu: Parallelization of numerical creation of phase-plane diagram of a simple nonlinear system, (in Hungarian), in G. Keresztes (Ed) Proceeding of spring wind conference 2, Miskolc, (2017), pp.547-557.

[27] P. L. Simon, H. Farkas, M. Wittmann, Constructing global bifurcation diagrams by the parametric representation method, Journal of Computational and Applied Mathematics, 108 (1-2) (1999) pp.157-176.
DOI: 10.1016/S0377-0427(99)00108-9

[28] S. Lynch, Dynamical Systems with Applications using Maple, Birkhäuser Boston, (2010), pp. 143-165.
DOI: 10.1007/978-1-4899-2849-8

[29] F. Hajdu, Parallel numerical creation of 2-parametric bifurcation diagram of nonlinear oscillators, Acta Technica Jaurinensis, 11 (2) (2018) pp. 61-83.
DOI: 10.14513/actatechjaur.v11.n2.453

[30] S. A. Billings, O.M. Boaghe, The response spectrum map, a frequency domain equivalent to the bifurcation diagram, International Journal of

Bifurcation and Chaos, 11 (7) (2001) pp. 1961-1975.
DOI: 10.1142/S0218127401003164