JOURNAL OF THEORETICAL
AND APPLIED MECHANICS
57, 4, pp. 1067-1079, Warsaw 2019
DOI: 10.15632/jtam-pl/112463

MULTI-CORE AND MANY-CORE SPMD PARALLEL ALGORITHMS FOR CONSTRUCTION OF BASINS OF ATTRACTION

Marcos Silveira, Paulo J.P. Gonçalves, José M. Balthazar
São Paulo State University – UNESP, Faculty of Engineering, Bauru, Brazil
e-mail: m.silveira@feb.unesp.br

Construction of basins of attraction, used for the analysis of nonlinear dynamical systems which present multistability, are computationaly very expensive. Because of the long runtime needed, in many cases, the construction of basins does not have any practical use. Numerical time integration is currently the bottleneck of algorithms used for the construction of such basins. The integrations related to each set of initial conditions are independent of each other. The assignment of each integration to a separate thread seems very attractive, and parallel algorithms which use this approach to construct the basins are presented here. Two versions are considered, one for multi-core and another for many-core architectures, both based on a SPMD approach. The algorithm is tested on three systems, the classic nonlinear Duffing system, a non-ideal system exhibiting the Sommerfeld effect and an immunodynamic system. The results for all examples demonstrate the versatility of the proposed parallel algorithm, showing that the multi-core parallel algorithm using MPI has nearly an ideal speedup and efficiency.

Keywords: basins of attraction, MPI, CUDA, Duffing equation, Sommerfeld effect, immunodynamics

1. Introduction

Usual tools for numerical analysis of nonlinear dynamical systems which present multistability and chaotic behaviour, such as construction of bifurcation diagrams, construction and visualisation of basins of attraction, and construction of stability maps, are essential for a thorough understanding of the underlying phenomena (Soliman and Thompson, 1989; Thompson and Stewart, 2002; Rega and Lenci, 2005). However, they are computationaly very expensive and, sometimes, these tools may not be of any practical use.

Improvements of processor core raw performance have been decreasing for over a decade, and advances in computer hardware and software have been aimed towards parallel computation (Pacheco, 2011; Brodtkorb *et al.*, 2013). A common technique for parallelisation is the domain partitioning as in the parallel algorithm for mixed time integration for large-scale structural dynamic analysis with nonlinearities, Rao (2002), which is based on the serial algorithm presented by Belytschko *et al.* (1979). Other examples are single step implicit algorithms (Modak and Sotelino, 2002), population dynamics problems (Ramachandramurthi *et al.*, 1997; Michaels and Zubik-Kowal, 2012) and molecular dynamics (Murty and Okunbor, 1999; Trobec *et al.*, 1993; Aktulga *et al.*, 2012). In other types of problems, a modular approach can be used for systems that can be subdivided, such as in vehicle dynamics (Agrawal *et al.*, 1992), large articulated robotic systems (Bhalerao *et al.*, 2012), and contact dynamics (Koziara and Bićanić, 2011). In these cases, however, the communication overhead limits the efficiency and overall gains of parallelisation.

Currently, the time integration is the bottleneck of algorithms used for simulation and dynamical stability analysis of nonlinear dynamical systems which present multistability. Although parallel algorithms for time integration have been sought (Sommeijer, 1993; Cong, 1996), their performance in large scale simulations is usually limited, as they suffer from low scalability and large overhead (Bendtsen, 1996).

The construction of bifurcation diagrams, basins of attraction, stability maps and sensitivity analysis involve numerous time integrations for different combination of values of parameters and initial conditions. In analysis of dynamical integrity (Soliman and Thompson, 1989; Rega and Lenci, 2005; McDonald *et al.*, 1985; Lenci *et al.*, 2013), the total run-time is drastically increased, as the construction of basins of attraction is repeated many times. For a specific case of construction of basins, the integrations related to each set of initial conditions are independent of each other. This way, the assignment of each integration to a separate thread seems very attractive. Each process is computationally intensive, although the memory requirement is relatively low.

In view of the necessity to find faster ways to construct basins of attraction and to perform dynamical stability analysis of nonlinear dynamical systems fully utilising the available computing power, parallel algorithms for construction of basins of attraction are presented here. Two versions are considered, one for multi-core and another for many-core architectures, both based on a SPMD (single program multiple data) approach. The rest of the paper is organised as follows: Section 2 introduces the generic serial algorithm for construction of basins of attraction. Sections 3 and 4 present the multi-core algorithm using MPI and the many-core algorithm using CUDA. Sections 5-7 contain results based on the classical Duffing oscillator, a non-ideal oscillator and an immunodynamics model. These examples of application are used to verify the accuracy, scalability and efficiency while demonstrating the versatility of the proposed parallel algorithms.

2. Tasks for the construction of basins of attraction

The construction of basins of attraction involves finding all attractors of a dynamical system and identifying regions in the state space corresponding to initial conditions that lead to each attractor. Details concerning the search and classification of attractors among static or dynamic equilibria are not the focus of this work, and analytical and numerical methods can be found elsewhere (Thompson and Stewart, 2002; Parker and Chua, 1992; Nayfeh and Balachandran, 1995). Once all the attractors are known, the dynamical equations can be numerically evaluated with one set of initial conditions to determine to which attractor this set leads. This is repeated for all initial conditions, identifying the basins of attraction.

An algorithm to determine the basins consists in cycling search through all desired initial points, integrating a trajectory from each point to determine to which attractor it leads, and grouping the points that belong to the same attractor. As a consequence of the uniqueness of solutions, all points touched by a trajectory can be marked as pertaining to the same attractor. This eliminates the necessity to evaluate a large quantity of grid points. Therefore, as a grid point is selected, it must be checked if this point already belongs, or is close with a certain degree of accuracy, to a trajectory. A serial algorithm to perform such tasks is depicted in Fig. 1a.

3. Multi-core parallel algorithm using MPI

From the previous Section, it can be seen that the detection of points belonging to calculated trajectories can significantly reduce the run-time. However, this detection can lead to unacceptable communication overhead. In order to minimise communication between threads, information of points already hit and of points hit by the current trajectory are sent at the beginning and end of the process. In a region of the space state with an attractor at its center, trajectories tend

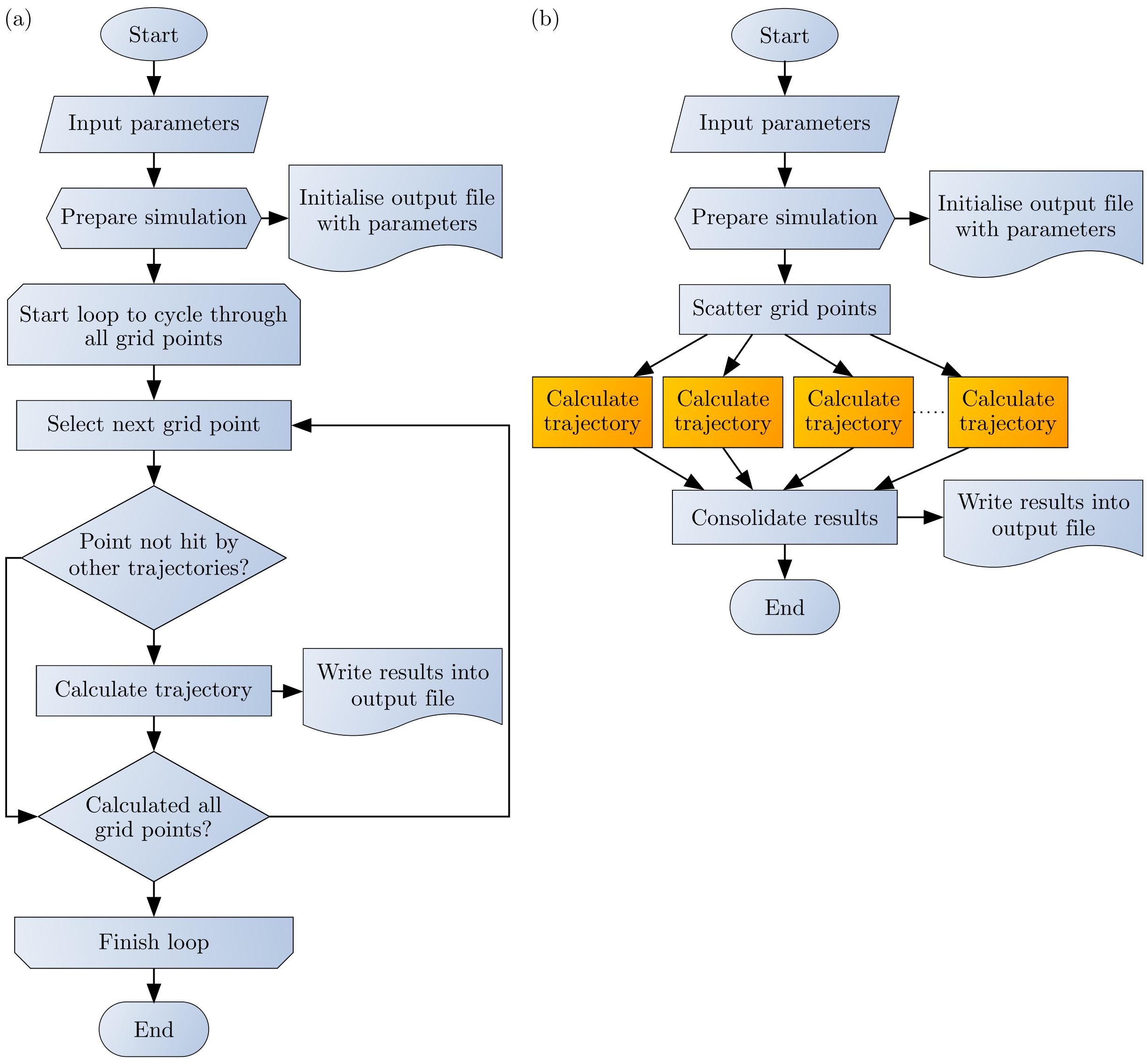

Fig. 1. Serial algorithm used for construction of basins of attraction (a) and the multi-core parallel algorithm using MPI (b). The processes in blue are executed by the master thread, the processes in orange are executed in parallel by all other threads

to flow from the border towards the center. Note that this is only true if the corresponding basin is either unbounded or the boundary is not contained in the considered region. If the points on the border are used first, the trajectories originating from them will fill most of the region, and a large number of grid points can be discarded (Parker and Chua, 1992). The selection of grid points assigned to the computing cores takes this into consideration. As there is no need to wait for any information from other threads, the next task can be started subsequentially, and a good load balancing is achieved.

The serial algorithm for computation of the basins of attraction from Section 2 was modified for use in x86 multi-core processors using MPI (message passing interface), as shown in Fig. 1b. MPI was chosen as it is one of the most used frameworks for parallel scientific computing. The master thread is responsible for broadcasting all parameters to the slave threads, for scattering each subgrid of initial conditions to the respective thread, and for collecting and consolidating the results. As the threads do not communicate with each other, there is no concurrency problem.

When the number of cores c is much lower than the number of grid points p_g, the grid is subdivided in fewer regions. The number of points in each region is given by

$$p_{ra} = \frac{p_g}{c} \tag{3.1}$$

For an increasing number of cores, there is a larger number of regions, although they become smaller, and the amount of checked, and possibly eliminated, grid points diminishes. Usually, one uses from 30 000 to 300 000 grid points, which can be 3 to 4 orders of magnitude greater than the available number of x86 cores. This ensures a good load balancing and that the detection of used points eliminates a satisfactory portion of points. The maximum number of eliminated grid points p_e is given by

$$p_e^{max} = c(p_r - 1) = p_g - c \tag{3.2}$$

The serial run-time T_s is proportional to the number of calculated trajectories. Note that the minimum T_s will be the time of evaluation of one trajectory, and this happens when $p_e = p_e^{max}$. The maximum T_s will be the time of evaluation of p_g trajectories ($p_e = 0$). These relations are given by

$$T_s \propto (p_g - p_e) \qquad\qquad T_p \propto \Big(p_r - \frac{p_e}{c}\Big) \tag{3.3}$$

Although the minimum number of evaluated trajectories equals c in the parallel case, the minimum T_p ($p_e = p_e^{max}$) will be the time of evaluation of one trajectory, and the maximum T_p ($p_e = 0$) corresponds to p_r trajectories. The time for communication must be considered in these relations. When the number of eliminated grid points tends to p_e^{max}, the run-time of the multi-core parallel algorithm tends to be similar to the serial algorithm. When the number of eliminated grid points tends to 0, the run-time of the multi-core parallel algorithm tends to be c times faster than the serial algorithm.

4. Many-core parallel algorithm using CUDA

The use of GPUs for general computing in scientific problems may be very advantageous when the problem can be divided in several light processes. In the problem of construction of basins of attraction, the lightest independent process is the time integration of trajectories. Using the CUDA (compute unified device architecture) platform, the algorithm for computing basins of attraction was modified in order to assign the integration of each trajectory to one computing core of the GPU. The CUDA platform was chosen as it is one of the most frequently used technologies for GPU computing. Within the CUDA programming model, threads are grouped in blocks, and the blocks are grouped in grids. The number of threads in a block and the number of blocks in a grid depend on the device being used.

The modified algorithm (Fig. 2) consists of dividing the total number of grid points p_g in b blocks, with t being the number of available threads per block. All blocks are cycled, selecting a subset of grid points, evaluating trajectories from each point, and consolidating the results from the subset. The CPU is the master thread and is responsible for initialising the simulation, selecting the subsets of points, checking for used points, assigning the points to the GPU and consolidating the results. The processes executed in the GPU consist of the actual numerical integrations of each point. If one or more points are detected as used, a new subset must be selected in order to have exactly one point for each thread in the block, with the objective of achieving a good load balancing.

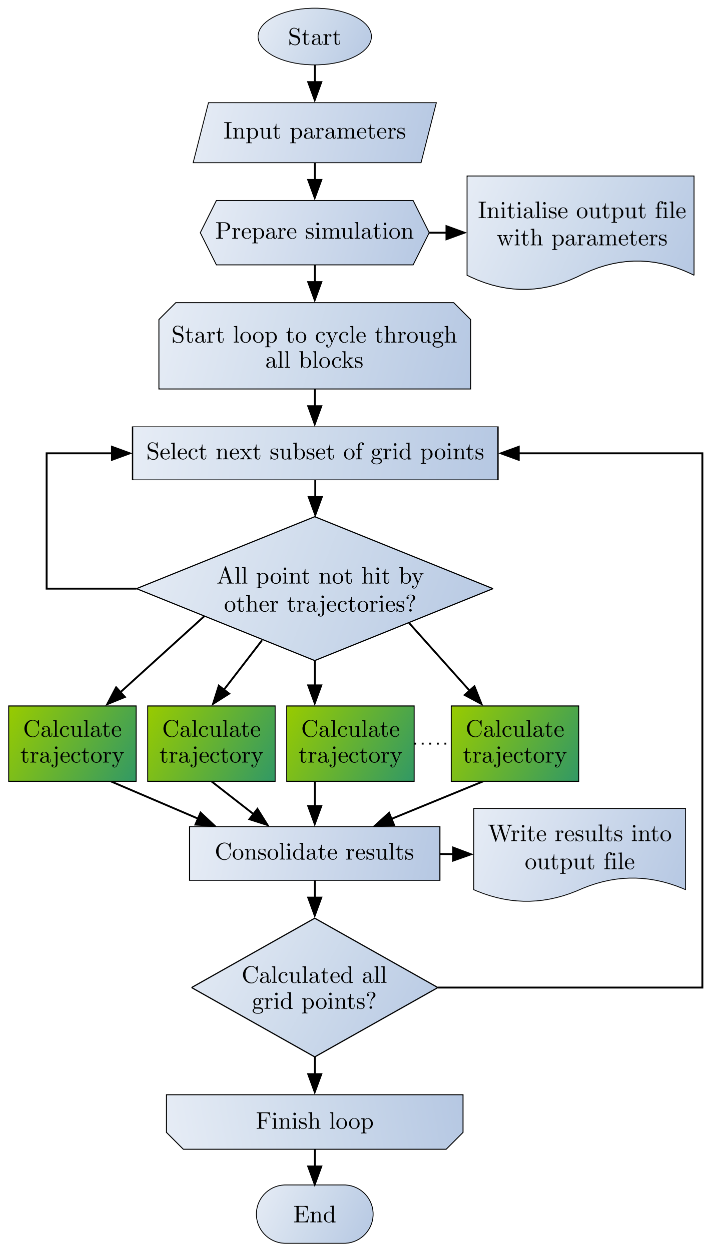
Start
Input parameters
Prepare simulation
Initialise output file with parameters
Start loop to cycle through all blocks
Select next subset of grid points
All point not hit by other trajectories?
Calculate trajectory
Calculate trajectory
Calculate trajectory
Calculate trajectory
Consolidate results
Write results into output file
Calculated all grid points?
Finish loop
End

Fig. 2. Flowchart of the multi-core parallel algorithm used for construction of basins of attraction using CUDA. The processes in blue are executed in the CPU, while the processes in green are executed in parallel in the GPU

The number of blocks and the maximum number of points which can be eliminated are given by

$$b = \frac{p_g}{t} \qquad p_e^{max} = (b-1)t = p_g - t \tag{4.1}$$

in which t is the number of threads per block, p_g is the number of grid points. The check for used points is only made at the beginning and the end of the process. The equivalent of relation (3.3) for the run-time of the many-core parallel algorithm is given by

$$T_p \propto \frac{p_g}{t} - \left\lfloor \frac{p_e}{b} \right\rfloor \tag{4.2}$$

in which $\lfloor \cdot \rfloor$ is the notation for the *floor* function. The minimum number of evaluated trajectories equals to b in this case. The minimum T_p ($p_e = p_e^{max}$) will be the time of integration of one trajectory, and the maximum T_p ($p_e = 0$) corresponds to b trajectories. When p_e tends to p_e^{max}, the run-time of the many-core algorithm tends to be similar to the serial algorithm, and when p_e tends to 0, the run-time of the many-core parallel algorithm tends to be b times faster than the serial algorithm.

5. Multi-core and many-core algorithms used on the Duffing oscillator

In order to verify the accuracy, scalability and efficiency of the proposed parallel algorithm, the Duffing equation is used in this Section as an example of application, given by

$$m\ddot{x} + c\dot{x} + k_1x + k_2x^3 = B\cos(\omega t) \tag{5.1}$$

in which m is mass, k_1 and k_2 are linear and nonlinear coefficients of stiffness, c is damping coefficient, B and ω are amplitude and frequency of the excitation force. The values of the parameters used here are $m = 2.56\,\mathrm{kg}$, $k_1 = 1\,\mathrm{N/m}$, $k_2 = 0.05\,\mathrm{N/m^3}$, $c = 0.32\,\mathrm{Ns/m}$, $B = 2.5\,\mathrm{N}$ and $\omega = 1\,\mathrm{rad/s}$, which result in the existence of two co-existing limit-cycles, one with a large and another with small amplitude (Thompson and Stewart, 2002). Figure 3 shows the basins of the two attractors.

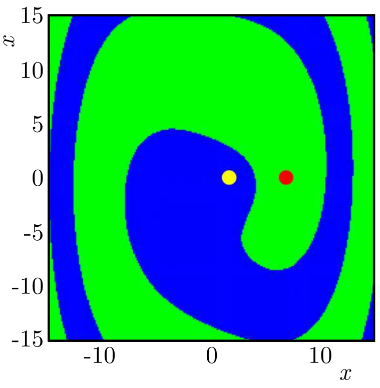

Fig. 3. Basins of attraction of the Duffing oscillator, Eq. (5.1). The red dot is the small amplitude limit-cycle, with its basin in green, and the yellow dot is the large amplitude limit-cycle, with its basin in blue. The attractors are represented by their Poincaré sections

The multi-core algorithm was executed on a x86 processor with 16 physical cores. The many--core algorithm was executed on a GPU with 448 CUDA cores, limited to a maximum of 512 threads per block. All timings shown are the arithmetic mean of at least 5 runs. The desired resolution of the figure of basins of attraction is from 256×256 to 512×512 points, although lower resolution basins were calculated for comparison of different problem sizes. The numerical algorithm used to calculate the time response is the 4-th order Runge-Kutta method with a fixed step. The fixed step is used to enable direct comparisons between different architectures.

The performance is measured by the run-time, speedup and efficiency. The run-time of an ideal parallel implementation T_p, the speedup S and efficiency E are given by

$$T_p^{ideal} = \frac{T_s}{c} \qquad S = \frac{T_s}{T_p} \qquad E = \frac{S}{c} = \frac{T_s}{cT_p} \tag{5.2}$$

This is only reached when no overhead is incurred by parallelisation. In an ideal parallel implementation, $S_{ideal} = c$.

Remaining serial sections and inter-thread communication overhead are the contributing factors for lower efficiency of real implementations. Note that the speedup and efficiency are calculated based on the reference T_s, which can be obtained by the fastest known algorithm to solve the problem, or by the parallel algorithm run on a single core (adopted here). The reference T_s for the many-core algorithm is taken from the run on a single CUDA core.

Figure 4a shows the run-time as a function of the number of used cores. The number of trajectories is given as the size of the grid of initial conditions. Note that there is a rapid decrease in the run-time and fast saturation. However, incremental gains continue to be observed. Figure 4b shows the run-time as a function of the number of calculated trajectories. An increase in the run-time is linearly proportional to the number of calculated trajectories, indicating excellent scalability. Figure 4c shows that the speedup is very close to the ideal line (in red). Figure 4d confirms that the efficiency is nearly ideal. A small variation is seen for different grid sizes, indicating that this is not a limiting factor. Besides that, the excellent scalability shown demonstrates that the techniques applied to the algorithm are appropriate for the problem being studied.

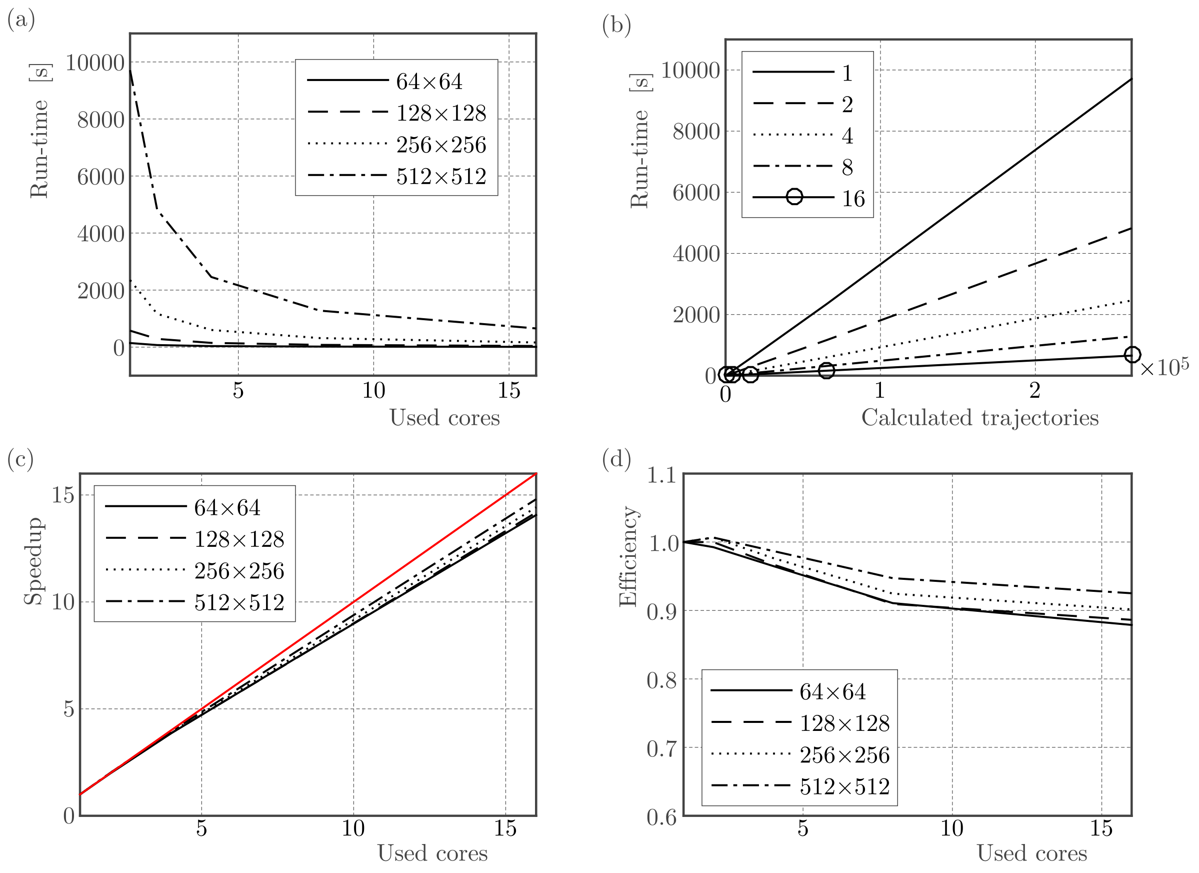

Fig. 4. Run-time as a function of the number of used cores (a) and as a function of the number of calculated trajectories (b), speedup (c) and efficiency (d) as a function of the number of applied cores using the multi-core algorithm with MPI for system (5.1)

The results of computation of basins of attraction using the many-core parallel algorithm show that GPU technology also provides good time gains. Figure 5a shows a decrease in the run-time as the number of used cores increases. Again, the saturation tendency of the run-time is observed. For example, using a grid of initial conditions of 128×128 points, the difference betweeen using 128 or 256 cores is less than 1%. Hence, it is possible to identify an optimal number of cores to be used for a given number of trajectories. Figure 5b shows that the run--time is also linearly proportional to the number of calculated trajectories. In terms of the speedup and efficiency (Fig. 5), the many-core algorithm is shown to be inferior to the multi--core version. Although there are substantial run-time gains compared to the performance on a single core, they are obtained via the much larger number of used cores, resulting in a low efficiency. For direct comparison, the run-time of a 256×256 grid using the multi-core algorithm is 1.87 faster than the many-core version.

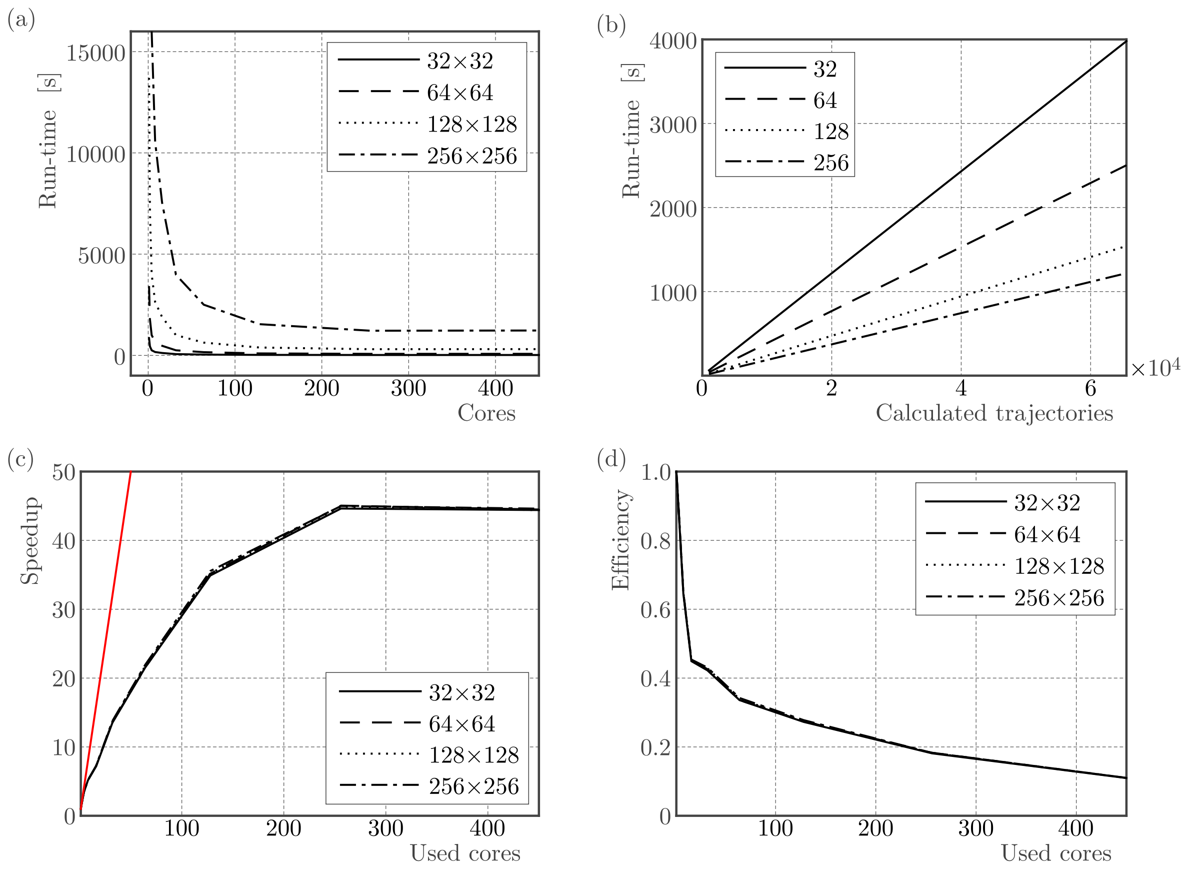

Fig. 5. Run-time as a function of the number of used cores (a) and as a function of the number of calculated trajectories (b), speedup (c) and efficiency (d) as a function of the number of applied cores using the many-core algorithm with CUDA for system (5.1)

6. Application to a non-ideal system exhibiting the Sommerfeld effect

A simplified spring-mass-damper oscillator model with a non-ideal excitation source was defined by Gonçalves *et al.* (2014, 2015), Balthazar *et al.* (2018), in which a cart moves horizontally without friction, with a non-ideal DC motor fixed to it. The motor has an unbalanced mass that provides excitation to the system. It was found that the Sommerfeld effect depends on some system parameters and the motor operational procedures. The dynamic equations for motion of the cart and of the motor shaft are written as

$$
\begin{aligned}
&\ddot{x}(M + m_r) + kx + c\dot{x} = m_r r(\dot{\phi}^2 \cos\phi + \ddot{\phi}\sin\phi) \\
&\ddot{\phi}(J_0 + m_r r^2) = \ddot{x} m_r r \sin\phi + M_0\Big(1 - \frac{\dot{\phi}}{\Omega_0}\Big)
\end{aligned}
\tag{6.1}
$$

The values used for the parameters are given in Gonçalves *et al.* (2014). Two distinct responses of the system are related to two co-existent period-1 limit-cycle attractors, with a large amplitude limit-cycle related to the resonance capture. With low Ω_0, only the large amplitude limit-cycle exists. This indicates the presence of the Sommerfeld effect, as the motor in this configuration is not able to provide enough energy to the system to escape from the resonance. Increasing Ω_0, two limit-cycle attractors co-exist, with the state space divided in two regions: one with initial conditions leading to the large amplitude limit-cycle (Sommerfeld efect), and another leading to the small amplitude limit-cycle (Fig. 6).

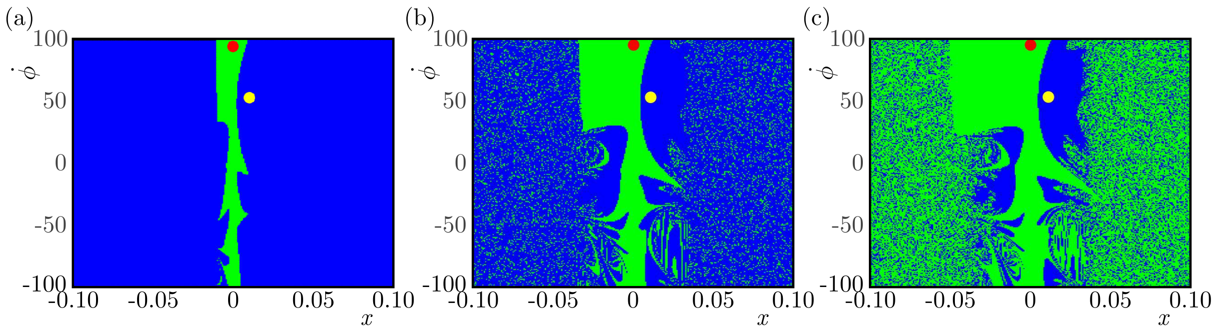

Fig. 6. Basins of attraction of the non-ideal oscillator, Eq. (6.1), using $\Omega_0 = 1.625\omega_0$ (a), $\Omega_0 = 2.167\omega_0$ (b) and $\Omega_0 = 2.438\omega_0$ (c). The red dot represents the small amplitude limit-cycle, with its basin in green, and the yellow dot represents the large amplitude limit-cycle, with its basin in blue

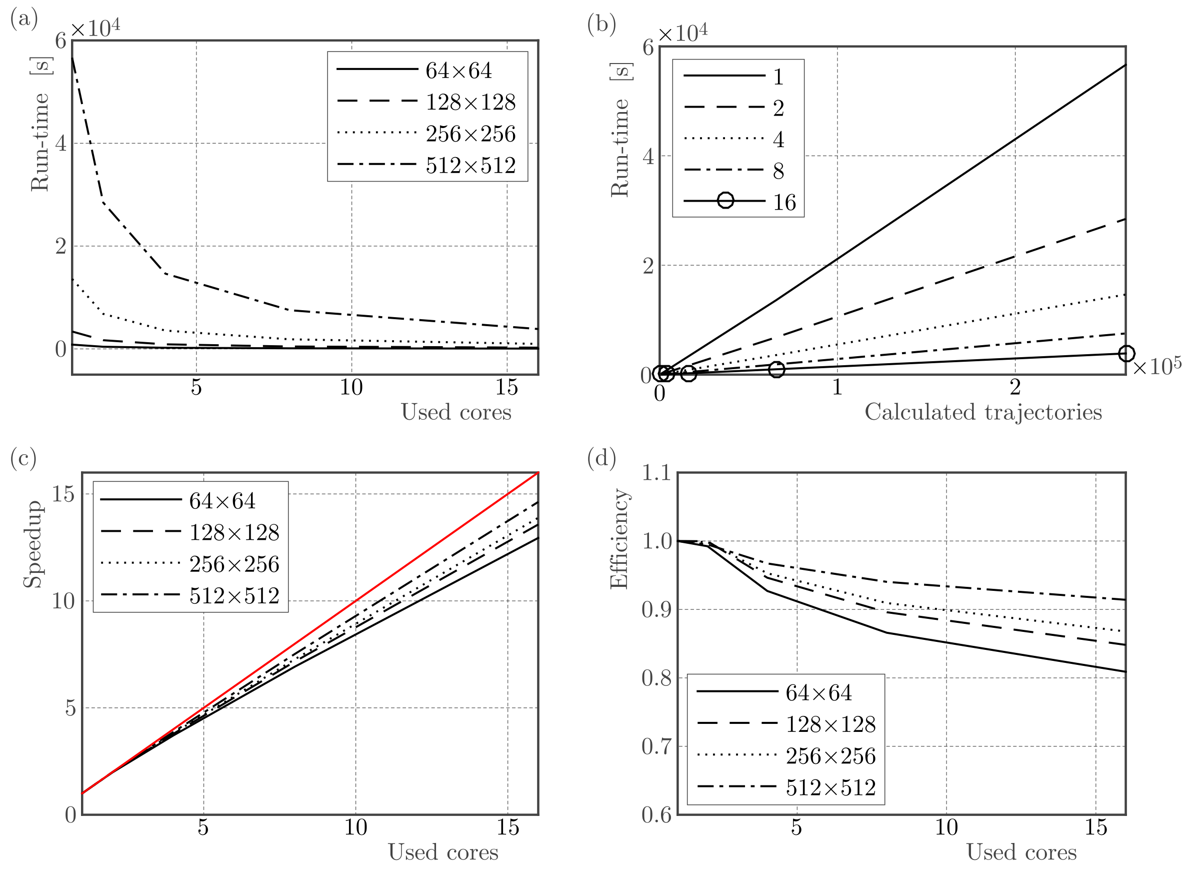

Fig. 7. Run-time as a function of the number of used cores (a) and as a function of the number of calculated trajectories (b), speedup (c) and efficiency (d) as a function of the number of applied cores using the multi-core algorithm with MPI for system (6.1)

Figure 7 shows that the run-time for this system is considerably longer than for the Duffing system, as this non-ideal oscillator takes longer to settle into steady-state behaviour. The algorithm provides the same level of time gains, the speedup remains very close to the ideal line, and the efficiency is maintained closer to ideal than in the Duffing system. The results of computation of basins of attraction using the many-core parallel algorithm with CUDA present the same tendencies as shown in the previous Section with reasonable speedups at the cost of lower efficiency. However, the run-time is more than 2.1 times slower when compared to the multi-core version.

7. Application to an immunodynamics system

Lang and Li (2012) studied the presence of different types of equilibria on a model of Human T cell leukemia virus type I (HTLV-I) infection. The model describes uninfected CD4$^+$ T cells (x), infected CD4$^+$ T cells (y) and CTL (z), and is given by

$$\begin{aligned} \dot{x} &= \lambda - \beta xy - \mu_1 x \qquad \dot{y} = \sigma\beta xy - \gamma yz - \mu_2 y \\ \dot{z} &= \nu y \frac{z^n}{z^n + a} - \mu_3 z \end{aligned} \tag{7.1}$$

This system has three types of equilibria (Lang and Li, 2012), with the first related to an organism without infection, the second related to an asymptomatic carrier state, and the third related to the development of HAM/TSP. Using the parameters related to CD4$^+$ T cell production rate λ and CTL production rate ν as bifurcation parameters, the system undergoes Hopf bifurcations, and regions with co-existing attractors arise. In the context of immune dynamics, the basins represent biological conditions at the time of infection which will result in one specific manifestation of a disease and, therefore, indicates the long-term health state of an individual. With the base set of parameters given in Lang and Li (2012), the system presents one equilibrium point, representing evolution of the infection in which the levels of healthy and infected cells reach fixed levels, and a limit-cycle, representing a condition in which the levels of cells oscillate over time.

The two attractors are shown in Fig. 8. On the left, the equilibrium point attractor is in yellow, and the limit-cycle attractor is in red. The blue region represents the basin of the equilibrium point, and the green region is the basin of the limit-cycle. On the right, the CTL effectiveness is decreased. As a result, the red attractor becomes an equilibrium point, and its basin (green region) is enlarged. The blue region is now the basin of an equilibrium point at another region of the state space, not shown in this figure.

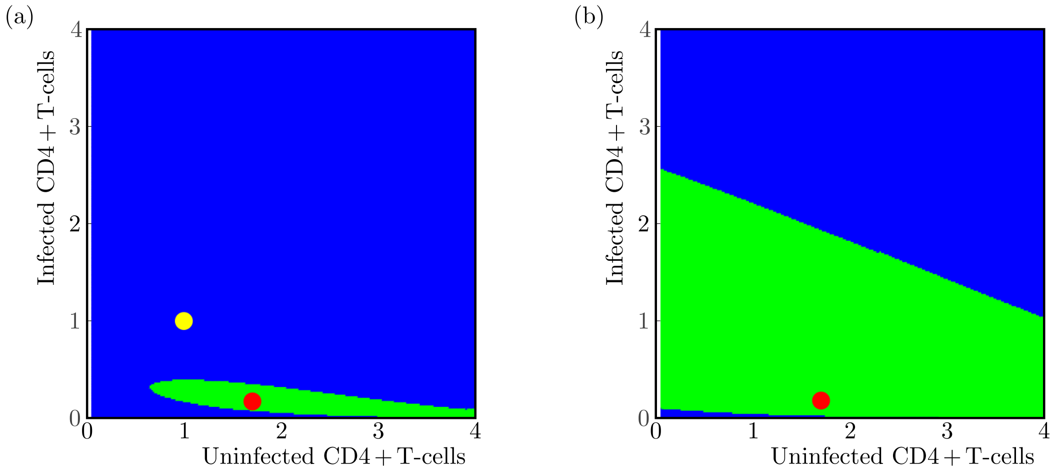

Fig. 8. Co-existing attractors and respective basins of model (7.1). With base parameters (a), there is one equilibrium point (yellow), with its basin in blue and one limit-cycle (red), with its basin in green. With $\gamma = 1$ (b), there is one equilibrium point (red)

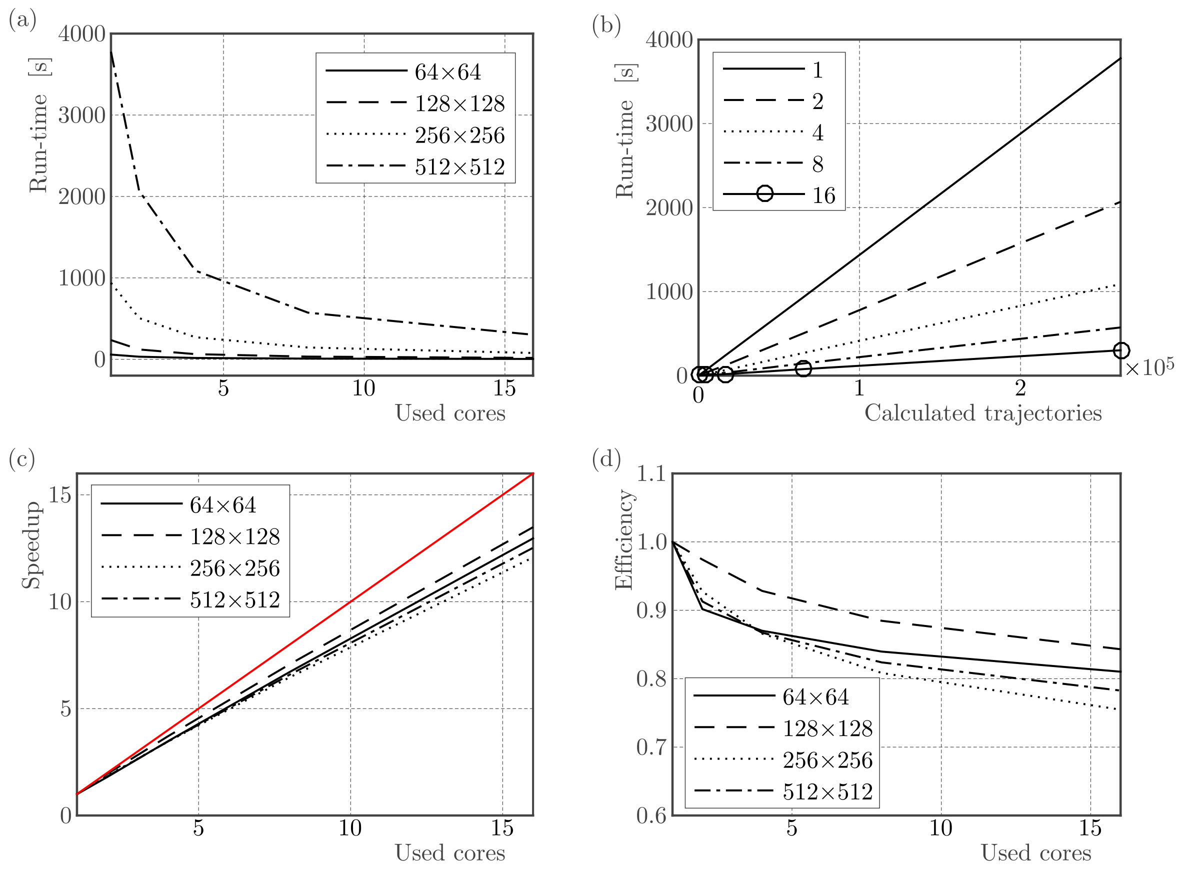

Fig. 9. Run-time as a function of the number of used cores (a) and as a function of the number of calculated trajectories (b), speedup (c) and efficiency (d) as a function of the number of applied cores using the multi-core algorithm with MPI for system (7.1)

Figure 9 shows that the run-time is consistently lower using the multi-core algorithm incorporating MPI. Intra-function timings show that the lack of trigonometric functions contribute to this. The algorithm provides the same level of time gains as observed in the previous Sections. The speedup and efficiency are less close to ideal. The run-time of the many-core algorithm with CUDA is more than 1.7 times slower when compared to the multi-core version.

8. Concluding remarks

Examples of application of the algorithm to the classic nonlinear Duffing system, to a non-ideal system exhibiting the Sommerfeld effect and to a immunodynamics system are shown. The results for all examples indicate that the multi-core algorithm using MPI has nearly an ideal speedup and efficiency, and perform consistently on different types of systems.

The use of GPUs for general computing in scientific problems may be very advantageous when the problem can be divided in several light processes. Although each integration process has a very small memory footprint, the complexity of the dynamical system being analysed may result in a heavy processing load, which hinders the advantage of using a GPU. In processing intensive cases, the execution in the CPU is usually more advantageous, even considering a lower number of computing cores compared to GPUs. This happens because of different purposes and architectures of GPUs (massive number of light threads) and CPUs (small number of intensive threads). Consequently, the use of a many-core GPU proved to be less satisfactory for constructing basins of attraction, with the multi-core algorithm run in the CPU shown to be faster, more efficient and more scalable.

Acknowledgements
The authors thank FAPESP (grant #2012/19774-3) and CNPq (grant #448560/2014-3) for financial support during this project.

References

1. Agrawal O.P., Danhof K.J., Kumar R., 1992, A superelement model based parallel algorithm for vehicle dynamics, *Computers and Structures*, **51**, 411423, DOI: 10.1016/0045-7949(94)90326-3
2. Aktulga H.M., Fogarty J.C., Pandit S.A., Grama A.Y., 2012, Parallel reactive molecular dynamics: Numerical methods and algorithmic techniques, *Parallel Computing*, **38**, 245259, DOI: 10.1016/j.parco.2011.08.005
3. Balthazar J.M., Tusset A.M., Brasil R.M.L.R.F., Felix J.L.P., Rocha R.T., Janzen F.C., Nabarrete A., Oliveira C., 2018, An overview on the appearance of the Sommerfeld effect and saturation phenomenon in non-ideal vibrating systems (NIS) in macro and MEMS scales, *Nonlinear Dynamics*, 122, DOI: 10.1007/s11071-018-4126-0
4. Belytschko T., Yen H.J., Mullen R., 1979, Mixed methods for time integration, *Computer Methods in Applied Mechanics and Engineering*, **1718**, 259275, DOI: 10.1016/0045-7825(79)90022-7
5. Bendtsen C., 1996, Highly stable parallel Runge-Kutta methods, *Applied Numerical Mathematics*, **21**, 18, DOI: 10.1016/0168-9274(96)00003-7
6. Bhalerao K.D., Crithcley J., Anderson K., 2012, An efficient parallel dynamics algorithm for simulation of large articulated robotics system, *Mechanism and Machine Theory*, **53**, 8698, DOI: 10.1016/j.mechmachtheory.2012.03.001
7. Brodtkorb A.R., Hagen T.R., Satra M.L., 2013, Graphics processing unit GPU programming strategies and trends in GPU computing, *Journal of Parallel and Distributed Computing*, **73**, 413, DOI: 10.1016/j.jpdc.2012.04.003
8. Cong N.H., 1996, Explicit symmetric Runge-Kutta-Nyström methods for parallel computers, *Computers and Mathematics with Applications*, **31**, 111121, DOI: 10.1016/0898-1221(95)00198-0
9. Gonçalves P.J.P., Silveira M., Pontes B.R. Jr., Balthazar J.M., 2014, Model analogy, numerical and experimental analysis of a cantilever beam with a coupled unbalanced non-ideal motor, *Journal of Sound and Vibration*, **333**, 51155129, DOI: 10.1016/j.jsv.2014.05.039
10. Gonçalves P.J.P., Silveira M., Petrocino E.A., Balthazar J.M., 2015, Double resonance capture of a two-degree-of-freedom oscillator coupled to a non-ideal motor, *Meccanica*, DOI: 10.1007/s11012-015-0349-z
11. Koziara T., Bićanić N., 2011, A distributed memory parallel multibody contact dynamics code, *International Journal for Numerical Methods in Engineering*, **87**, 437456, DOI: 10.1002/nme.3158
12. Lang J., Li M.Y., 2012, Stable and transient periodic oscillations in a mathematical model for CTL response to HTLV-I infection, *Journal of Mathematical Biology*, **65**, 181199, DOI: 10.1007/s00285-011-0455-z
13. Lenci S., Rega G., Ruzziconi L., 2013, The dynamical integrity concept for interpreting/prediction experimental behaviour: from macro- to nano-mechanics, *Philosophical Transactions of the Royal Society A*, **371**, DOI: 10.1098/rsta.2012.0423
14. McDonald S.W., Grebogi C., Ott E., Yorke J.A., 1985, Fractal basin boundaries, *Physica D*, **17**, 125153, DOI: 10.1016/0167-2789(85)90001-6
15. Michaels P., Zubik-Kowal B., 2012, Parallel computations and numerical simulations for nonlinear systems of Volterra integro-differential equations, *Communications in Nonlinear Science and Numerical Simulation*, **17**, 30223030, DOI: 10.1016/j.cnsns.2011.11.006
16. Modak S., Sotelino E.D., 2002, An object-oriented programming framework for the parallel dynamic analysis of structures, *Computers and Structures*, **80**, 7784, DOI: 10.1016/S0045-7949(01)00154-7

17. Murty R., Okunbor D., 1999, Efficient parallel algorithms for molecular dynamics simulations, *Parallel Computing*, **25**, 217230, DOI: 10.1016/S0167-8191(98)00114-8

18. Nayfeh A.H., Balachandran B., 1995, *Applied Nonlinear Dynamics: Analytical, Computational, and Experimental Methods*, Wiley

19. Pacheco P., 2011, *An Introduction to Parallel Programming*, Morgan Kaufmann

20. Parker T.S., Chua L.O., 1992, *Practical Numerical Algorithms for Chaotic Systems*, Springer--Verlag

21. Ramachandramurthi S., Hallam T.G., Nichols J.A., 1997, Parallel simulation of individual-based, physiologically structured population models, *Mathematical and Computer Modelling*, **25**, 5570, DOI: 10.1016/S0895-7177(97)00094-0

22. Rao A.R.M., 2002, A parallel mixed time integration algorithm for nonlinear dynamic analysis, *Advances in Engineering Software*, **33**, 261271, DOI: 10.1016/S0965-9978(02)00021-2

23. Rega G., Lenci S., 2005, Identifying, evaluating, and controlling dynamical integrity measures in non-linear mechanical oscillators, *Numerical Analysis*, **63**, 902914, DOI: 10.1016/j.na.2005.01.084

24. Soliman M.S., Thompson J.M.T., 1989, Integrity measures quantifying the erosion of smooth and fractal basins of attraction, *Journal of Sound and Vibration*, **135**, 453475, DOI: 10.1016/0022-460X(89)90699-8

25. Sommeijer B.P., 1993, Explicit, high-order Runge-Kutta-Nyström methods for parallel computers, *Applied Numerical Mathematics*, **13**, 221240, DOI: 10.1016/0168-9274(93)90145-H

26. Thompson J.M.T., Stewart H.B., 2002, *Nonlinear Dynamics and Chaos*, John Wiley & Sons, 2nd edition

27. Trobec R., Jerebic I., Janežič D., 1993, Parallel algorithm for molecular dynamics integration, *Parallel Computing*, **19**, 10291039, DOI: 10.1016/0167-8191(93)90095-3

Manuscript received March 6, 2018; accepted for print June 25, 2019