Check for updates

First Experiences on Parallelizing Peer Methods for Numerical Solution of a Vegetation Model

Dajana Conte[1], Pasquale De Luca[2(✉)], Ardelio Galletti[3], Giulio Giunta[3], Livia Marcellino[3], Giovanni Pagano[1], and Beatrice Paternoster[1]

[1] Department of Mathematics, University of Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, SA, Italy
{dajconte,gpagano,beapat}@unisa.it
[2] International PhD Programme/UNESCO Chair "Environment, Resources and Sustainable Development", Department of Science and Technology, Parthenope University of Naples, Centro Direzionale, Isola C4, 80143 Naples, Italy
pasquale.deluca@uniparthenope.it
[3] Department of Science and Technology, Parthenope University of Naples, Centro Direzionale, Isola C4, 80143 Naples, Italy
{ardelio.galletti,giulio.giunta,livia.marcellino}@uniparthenope.it
https://www.dipmat.unisa.it

Abstract. The purpose of this paper is to provide a parallel acceleration of peer methods for the numerical solution of systems of Ordinary Differential Equations (ODEs) arising from the space discretization of Partial Differential Equations (PDEs) modeling the growth of vegetation in semi-arid climatic zones. The parallel algorithm is implemented by using the CUDA environment for Graphics Processing Units (GPUs) architectures. Numerical experiments, showing the performance gain of the proposed strategy, are provided.

Keywords: Self-organization in ecology · Vegetation · PDEs · Peer methods · Parallel strategies · GPGPU

1 Introduction

This paper focuses on efficient numerical solution of PDEs arising in vegetation problems, by means of parallel numerical methods employing GPUs. We consider the following system of PDEs [12]:

The authors are members of the GNCS group. This work is supported by GNCS-INDAM project and by the Italian Ministry of University and Research (MUR), through the PRIN 2020 project (No. 2020JLWP23) "Integrated Mathematical Approaches to Socio-Epidemiological Dynamics" (CUP: E15F21005420006) and the PRIN 2017 project (No. 2017JYCLSF) "Structure preserving approximation of evolutionary problems".

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022, LNCS 13376, pp. 384–394, 2022.
https://doi.org/10.1007/978-3-031-10450-3_33

$$\begin{cases} \dfrac{\partial u_1}{\partial t} = \dfrac{\partial^2 u_1}{\partial x^2} + w u_1(u_1 + H u_2) - B_1 u_1 - S u_1 u_2, \\ \dfrac{\partial u_2}{\partial t} = D\dfrac{\partial^2 u_2}{\partial x^2} + F w u_2(u_1 + H u_2) - B_2 u_2, \qquad (x,t) \in [x_0, X] \times [t_0, T], \\ \dfrac{\partial w}{\partial t} = d\dfrac{\partial^2 w}{\partial x^2} + A - w - w(u_1 + u_2)(u_1 + H u_2), \end{cases} \tag{1}$$

with initial conditions:

$$\begin{cases} u_1(x, t_0) = U_{1,0}(x), \\ u_2(x, t_0) = U_{2,0}(x), \qquad x \in [x_0, X]. \\ w(x, t_0) = W_0(x), \end{cases} \tag{2}$$

The PDEs system (1) represents a model of vegetation in semi-arid climatic zones, and constitutes an example of the principle of self-organization in ecology. It was introduced to study how two herbaceous species compete for the same limiting resource (the water) managing to resist. To better understand how the model works, you can imagine that the function u_1 describes the density of grass present on the soil x at the instant t, that the function u_2 represents the same for trees, and that w is the quantity of water available. In [12], the authors analytically study the existence of states in which both species coexist, proving that coexistence is not a stable solution of the system. That is, coexistence occurs as metastable state, i.e. as a state appearing as stable solution of the system for very long times. However, for infinite times one of the two plants dies.

An approach that is usually followed (and that we will use in this paper) to numerically solve a PDEs system is given by the Method Of Lines (MOL), which consists in spatially discretizing the problem, then deriving a large system of ODEs (whose size is proportional to the number of grid points used in each spatial direction). To efficiently solve this ODEs system, it would be preferable to use a cheap method (i.e. a method involving the least possible number of function evaluations, matrix inversions, and so on), equipped with excellent stability properties. The latter request derives from the fact that ODEs coming from PDEs related to real applications are endowed with severe stiffness.

The requirements of cheapness and excellent stability are difficult to be fulfilled at the same time. In fact, the most widely used numerical methods for solving stiff ODEs are implicit Runge-Kutta [1,2], linearly implicit Runge-Kutta (see e.g. [4,8,27]), Rosenbrock [30], or IMEX (see an overview on all these methods, e.g., also in [26]). All these methods are implicit, which means that it is necessary to solve at each integration step, and for each stage, systems of linear or non-linear equations. That is, it is necessary to numerically compute inversions of matrices having dimension proportional to the size of the problem. The implicit methods are therefore expensive, but they are the only ones that enjoy excellent stability properties. On the other hand, using explicit methods does not involve the solution of systems of equations, but requires the use of very small time steps for the numerical solution to not explode, as explicit methods have bounded stability regions. Therefore, their use for the numerical solution of stiff ODEs becomes problematic.

In this paper, we overcome the mentioned difficulties by solving the model (1) making use of a class of explicit methods, known as peer methods, together with a parallel strategy. Peer methods [9–11,14–18,28,29,32] are two-step numerical schemes belonging to the class of general linear methods [3]. They are newer than the Runge-Kuttas, and have two properties whereby their use can be very convenient. Indeed, in these methods, all stages are calculated with the same accuracy as the advancing solution, and this ensures that peer methods do not suffer from order reduction when integrating stiff problems, unlike what happens for Runge-Kutta methods [31]. Furthermore, since in peer methods the next stages just depend on the previous ones, we have been able to provide a new formulation, for the case study (1), which has been implemented in a parallel environment.

Thanks to this approach, a computational gain can be achieved also when a large amount of data have to be processed. We highlight that some initial attempts to parallelize peer methods have been addressed (see [13]), where software, for multicore system is presented. Here, we refine the parallel strategy in order to exploit the GPUs (Graphics Processing Units) architectures powerful by employing the CUDA environment [19]. We remark that nowadays the choice of the GPUs becomes almost mandatory thanks to the possibility they offer to solve large size problems, by means of suitable parallel algorithms, in a reduced execution time [5–7,20–23].

The rest of the paper is organized as follows. Section 2 recalls peer methods and shows the chosen numerical scheme. In Sect. 3, the sequential and parallel algorithms and the related GPU-CUDA implementation are presented. The experiments are discussed in Sect. 4 to confirm the efficiency of the proposed implementation in terms of performance. Finally, conclusions in Sect. 5 close the paper.

2 Numerical Scheme

In this section, numerical difference schemes and peer methods, with their use for discretizing in space and time Eq. (1), are presented. In the first step a spatial semi-discretization, via the MOL method, is described, while second step shows general details about peer methods and how they can be adapted to be used in problem (1).

2.1 Spatial Semi-discretization

First step is based on the MOL technique. Hence, we spatially semi-discretize (1), by dividing the spatial interval $[x_0, X]$ into $M-1$ sub-intervals, using central finite differences to approximate the involved second derivatives, as follows:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u(x-\Delta x,t) - 2u(x,t) + u(x+\Delta x,t)}{\Delta x^2}, \quad u = u_1, u_2, w, \quad \Delta x = \frac{X - x_0}{M-1}.$$

Therefore, the following ODEs system of size $3M$ is obtained:

$$\begin{cases} \dfrac{\partial U_1}{\partial t} = \dfrac{1}{\Delta x^2} L_{Diff} U_1 + F_1, \\ \dfrac{\partial U_2}{\partial t} = \dfrac{D}{\Delta x^2} L_{Diff} U_2 + F_2, \\ \dfrac{\partial W}{\partial t} = \dfrac{d}{\Delta x^2} L_{Diff} W + F_3. \end{cases} \tag{3}$$

For simplicity of notation, we have omitted the time dependence of all the involved functions. We have used the following notation:

$$x_m = x_0 + m\Delta x; \quad m = 0, \ldots, M-1; \quad x_{M-1} = X, \tag{4}$$

and:

$$u_1^m = u_1(x_m, t), \qquad u_2^m = u_2(x_m, t), \qquad w^m = w(x_m, t),$$

$$U_1 = \big(u_1^m(t)\big)_{m=0}^{M-1}, \quad U_2 = \big(u_2^m(t)\big)_{m=0}^{M-1}, \quad W = \big(w^m(t)\big)_{m=0}^{M-1},$$

$$\frac{\partial U_1}{\partial t} = \Big(\frac{\partial u_1^m}{\partial t}\Big)_{m=0}^{M-1}, \quad \frac{\partial U_2}{\partial t} = \Big(\frac{\partial u_2^m}{\partial t}\Big)_{m=0}^{M-1}, \quad \frac{\partial W}{\partial t} = \Big(\frac{\partial w^m}{\partial t}\Big)_{m=0}^{M-1},$$

$$F_1 = (w^m u_1^m (u_1^m + H u_2^m) - B_1 u_1^m - S u_1^m u_2^m)_{m=0}^{M-1},$$

$$F_2 = (F w^m u_2^m (u_1^m + H u_2^m) - B_2 u_2^m)_{m=0}^{M-1},$$

$$F_3 = \big(A - w^m - w^m (u_1^m + u_2^m)(u_1^m + H u_2^m)\big)_{m=0}^{M-1}.$$

If u_1, u_2, w are assumed to be zero outside their domain, it turns out that matrix L_{Diff} is tridiagonal and it is given by:

$$L_{Diff} = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{M,M}. \tag{5}$$

While, by using periodic boundary conditions, the matrix L_{Diff} must actually be slightly modified, setting $L_{Diff}(M,1) = L_{Diff}(1,M) = 1$. Finally, by further compacting the semi-discrete ODEs system (3), we get:

$$y'(t) = L \cdot y(t) + NL\big(y(t)\big), \tag{6}$$

where

$$y(t) = [U_1, U_2, W]^T \in \mathbb{R}^{3M} \qquad \text{and} \qquad NL\big(y(t)\big) = [F_1, F_2, F_3]^T \in \mathbb{R}^{3M}, \tag{7}$$

while matrix L takes the form:

$$L = \frac{1}{\Delta x^2} \begin{pmatrix} L_{Diff} & 0 & 0 \\ 0 & D L_{Diff} & 0 \\ 0 & 0 & d L_{Diff} \end{pmatrix} \in \mathbb{R}^{3M,3M}. \tag{8}$$

Next step aims to discretize the ODEs system (6) by means of peer methods.

2.2 Time Discretization by Peer Methods

Peer methods are a class of two-step numerical schemes that solve first order ODEs systems, in the general form $y'(t) = f(t, y(t))$ $(t \in [t_0, T])$ with initial condition $y_0 = y(t_0) \in \mathbb{R}^d$. Let consider the time discretization:

$$t_n = t_0 + nh; \quad n = 0, \ldots, N; \quad t_N = T, \tag{9}$$

of the integration interval $[t_0, T]$ related to (1), and therefore to (6). Explicit s-stage two-step peer methods, with fixed step-size h, are of the following form:

$$\begin{aligned} &Y_{n,i} = \sum_{j=1}^{s} b_{ij} Y_{n-1,j} + h \sum_{j=1}^{s} a_{ij} f(t_{n-1,j}, Y_{n-1,j}), \qquad n = 0, ..., N-1, \\ &Y_{n,i} \approx y(t_{n,i}), \quad t_{n,i} = t_n + hc_i, \qquad i = 1, ..., s. \end{aligned} \tag{10}$$

where c_i are nodes in $[0, 1]$ and $c_s = 1$. Then, the advancing solution $Y_{n,s}$ is the computed approximation of $y(t_n + h)$, i.e. the last stage computes the numerical solution at the grid points. Coefficients in matrices $A = (a_{ij})_{i,j=1}^{s}$ and $B = (b_{ij})_{i,j=1}^{s}$ characterize the peer method used. In particular, to choose the methods free coefficients, we need to recall some theoretical results: firstly, peer methods are said to be optimally zero-stable by setting:

$$(b_{ij})_{i,j=1}^{s-1} = 0, \qquad \text{and} \qquad b_{is} = 1, \forall i = 1, \ldots, s.$$

This choice is linked to the stability of peer methods near the origin [32]. Furthermore, the A matrix coefficients are assigned by imposing the consistency order of peer methods [32], i.e. by annihilating the necessary number of residuals, defined as

$$h\Delta_i := y(t_{n,i}) - \sum_{j=1}^{s} b_{ij} y(t_{n-1,j}) - h \sum_{j=1}^{s} a_{ij} y'(t_{n-1,j}), \quad i = 1, ..., s.$$

where y spans over suitable polynomial bases.
Note that the i-th residual measures the error between the i-th stage and its exact value.

Hence, assuming s-stages peer methods of order $p = s$, the coefficients $(a_{ij})_{i,j=1}^{s}$ have to satisfy [32]

$$A = (CV_0 D^{-1}) V_1^{-1} - B(C - I_s) V_1 D^{-1} V_1^{-1},$$

where $V_0 = (c_i^{j-1})_{i,j=1}^{s}$, $V_1 = \left((c_i - 1)^{j-1}\right)_{i,j=1}^{s}$, $C = diag(c_i)$, $D = diag(1, ..., s)$, and I_s is the Identity matrix of dimension s. In particular, in our scheme, we choose to use both values $s = 2$ and $s = 3$ for stages. In these settings, we get:

$$A = \begin{pmatrix} 0 & 0 \\ -1/2 & 3/2 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \quad (c_1, c_2) = (0, 1), \tag{11}$$

for $s = 2$, and:

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 5/24 & -2/3 & 23/24 \\ 7/6 & -10/3 & 19/6 \end{pmatrix}, \quad B = \begin{pmatrix} 0\,0\,1 \\ 0\,0\,1 \\ 0\,0\,1 \end{pmatrix}, \quad (c_1, c_2, c_3) = (0, 1/2, 1), \quad (12)$$

for $s = 3$. Notice that therefore that all the stages $Y^{[n]} = (Y_{n,i})_{i=1}^{s}$ in the interval $[t_n, t_n + h]$ are functions of all the stages in the interval $[t_n - h, t_n]$ only, which are totally known at the current time step. Then, by denoting by $\mathcal{F}$ the function evaluations of the stages at each discrete point $t_{n,i}$, i.e.,

$$\mathcal{F}(Y^{[n]}) = \big(f(t_{n,i}, Y_{n,i})\big)_{i=1}^{s}$$

peer methods (10) can take the more compact vector representation:

$$Y^{[n]} = (B \otimes I_d)Y^{[n-1]} + h(A \otimes I_d)\mathcal{F}(Y^{[n-1]}), \quad (13)$$

being d the solution dimension and I_d is the identity matrix of order d. Finally, we can derive the whole numerical procedure by setting in formulation (13) $d = 3M$ and:

$$\mathcal{F}(Y^{[n-1]}) = \big(LY_{n-1,i} + NL(Y_{n-1,i})\big)_{i=1}^{s}, \quad (14)$$

where L and NL are as in (7) and (8), and by applying the scheme to the vegetation problem in the ODE form (6).

3 Sequential and Parallel Implementation

Previous discussions allow us to introduce an algorithm which recalls main numerical steps described above.

Algorithm 1 shows the main operations needed to find numerical solution of problem (1) based on discretization discussed in Sect. 2. This algorithm has been implemented, in a first version, through a sequential C code whose details are below described.

In particular, the algorithm initializes the necessary parameters for time and space discretization (lines 1–4). Then, to perform the spatial semi-discretization, in lines 5–6 the procedure builds the tridiagonal matrix L_{Diff}, as in Eq. (5), and the tridiagonal block matrix L, as in (8). Now, to apply the peer method, it is mandatory to prime it, that is for time step $n = 0$ we need precompute the values of $Y_{0,i}$ at all stages (lines 7–8). To this aim, in the algorithm we used the explicit one-step four order Runge-Kutta that guarantees the peer method maintains the same order of accuracy when computing next values $Y_{n,i}$. Finally, lines 9–12 illustrate the main core of the procedure, i.e. the *loop-for* on time steps $(n = 1, \ldots, N)$.

We remark that, when small step sizes are chosen large space-time discretized grids occur, generating a remarkably increase of the procedure time complexity, which in turn implies possibly huge execution times of the whole procedure. To address this issue, here, we propose an accelerated implementation that parallelizes the sequential Algorithm 1, by exploiting the powerful parallel nature of GPUs. The idea comes from taking advantages of large number of threads equipped by a GPU and decomposing the problem among them.

Algorithm 1. Sequential pseudo-algorithm for problem (1)

`Input:` $s, x_0, X, N, t_0, T, M, y_0, NL_0, a, B1, B2, H, F, S, d, D,$ `Output:` Y

% Initialization:
1: `set` $\Delta x = {}^{(X-x_0)}/_{(M-1)}$
2: `set` x_m $(m = 0, \ldots, M-1)$ as in (4)
3: `set` $h = {}^{(T-t_0)}/_{N}$
4: `set` t_n $(n = 0, \ldots, N)$ as in (9)
% Spatial semi-discretization:
5: `build` L_{Diff} as in (5)
6: `build` L as in (8)
% Peer method
% initialization: time step $n = 0$
7: `set` $t_{0,i} = t_0 + h \cdot c_i, \quad (i = 1, \ldots, s)$
8: `compute` $Y_{0,i}$ `for` $i = 1, \ldots, s$
% loop on time steps
9: `for` $n = 1, \ldots, N$ `do`
10: `set` $t_{n,i} = t_n + h \cdot c_i, \quad (i = 1, \ldots, s)$
11: `compute` $Y_{n,i}$ as in (10), `for` $i = 1, \ldots, s$.
12: `end for`

3.1 Parallel Approach

Due to the high computational time required by the procedure on finding solution of Eq. (10), we chose to parallelize the majority of the overall work. The parallel pseudo-code Algorithm 2 highlights the main features to be addressed to implement the numerical procedure in a GPU environment.

In this work, we have implemented Algorithm 2 in a GPU parallel software by means of suitably built parallel functions. Details and main operations are listed in following steps:

`Line 1:` The discrete space-time grids are defined by using the *loop-unrolling* strategy, in order to optimize the computation and memory accesses.

`Lines 2,8:` These instructions are related to the data transfer from host-to-device and vice-versa. The procedures are carried out only during the starting and final phases of the GPU-parallel code, in order to improve the software performance.

`Line 3:` Here, the spatial semi-discretization operation is considered. A specific CUDA kernel has been designed for building, firstly the L_{Diff} matrix, and then the L matrix in a parallel way. We use the structural properties of matrix L, in order to perform a fair work-load balancing among domain problem and threads number. In other words, we set as threads number $\mathcal{T} = 3$, then each thread computes own part of L matrix (each thread is related to

Algorithm 2. GPU pseudo-algorithm for problem (1)

`Input:` $s, x_0, X, N, t_0, T, M, y_0, NL_0, a, B1, B2, H, F, S, d, D.$ `Output:` Y

% initialization, as at lines 1–4 of Algorithm 1

1: `generate` x_m, t_n $(m = 0, \ldots, M-1, \quad n = 0, \ldots, N)$

% GPU computation starting

2: `transfer Input data Host-To-Device`

% Spatial semi-discretization

3: `dynamic parallel building of` L_{Diff} and L, as in (5), (8)

% Parallel peer method

% initialization: time step $n = 0$

4: `dynamic parallel computing of` $Y_{0,\tau}$ `for` $\tau = 1, \ldots, s$

% loop on time steps

5: `for` $n = 1, \ldots, N$ `do`

6: `parallel computing` of $Y_{n,i}$ `for` $i = 1, \ldots, s$.

7: `end for`

8: `transfer Output data Device-To-Host`

% end GPU computation

one of variables u_1, u_2,w) by performing operations concurrently. This parallel step is implemented by using the *dynamic parallelism* which characterizes the CUDA environment and allows us to design a nested parallelization strategy [24].

`Line 4:` In this step, a kind of nested parallelism is adopted. More specifically, each threads τ $(\tau = 1, \ldots, s)$ computes concurrently $Y_{0,\tau}$ by means the Runge-Kutta four order method, which in turn has been implemented parallelized, by distributing basic linear algebra operating among further threads.

`Lines 5-7:` At each time step $n = 1, \ldots, N$, similarly in previous step, a nested kind of parallelism is used for solving the basic linear algebraic operations. Also in this step, we need to implement three specific CUDA kernels in order to perform evaluation of linear and nonlinear terms in (14), and generic loop iteration required in (13). Moreover, the gathering process of the local results is done by guaranteeing a coalescent memory access. This final step is implemented by ensuring any memory contention and a fair work-load balancing among threads.

4 Experimental Tests

In this section, we propose tests and experiments to prove the reliability and the main features of our implementation. In the whole section, we assume to consider

the problem (1) with parameters (see [12]) $A = 1.5$, $B_1 = 0.45$, $B_2 = 0.3611$, $F = 0.802$, $H = 0.802$, $S = 0.0002$, $d = 500$, $D = 0.802$, and space and time intervals $[-50, 50]$ and $[0, 50]$, respectively. According to what has been done in [12], we also set randomly the solution initial values in the interval $[1.4, 2.1]$ for u_1 and u_2, and in the interval $[0.14, 0.21]$ for w. Furthermore, the experiments are carried out in the following high-performance computer with technical specifications:

- 1 Intel(R) Xeon(R) Gold 5218 CPU @ 2.30 GHz, 16 cores, 2 threads per core, 192 GB of RAM, 4 channels 51 Gb/s memory bandwidth
- 1 NVIDIA V100-SXM2, 5120 CUDA cores, 1370 MHz clock speed for core, 16 GB DDR5, 900.1 Gb/s as bandwidth.

Table 1. CPU vs. GPU execution times. `block` $\times$ `threads`: 128×3125

	Execution times (s)	
N	CPU	GPU
2.0×10^5	2 055.23	63.21
4.0×10^5	6 410.54	191.46
8.1×10^5	10 293.11	318.14
1.6×10^6	36 322.95	701.84
3.2×10^6	$\sim$	1 293.19
1.3×10^7	$\sim$	1 820.95

Table 1 shows an execution time comparison between sequential and parallel implementation for solving problem (1). The test is conducted by varying N, and fixing $M = 64$. A standard GPU-CUDA computation provides to preliminarly set the grid threads through the `block` $\times$ `threads` and here, after testing the algorithm with different configurations, we used `block` $\times$ `threads` = `128` $\times$ `3125` that gave best performance. First column of the Table refers to sequential implementation of Algorithm 1 which runs on a single core machine. Last column exhibits the execution times of GPU implementation of Algorithm 2. We observe a significant gain with respect to the serial version. The time of the sequential version grows considerably as N increases. However a noticeable speed-up is observed in the GPU-accelerated version. In detail, due to high resource hardware limits of CPU, the tests are stopped to 1.6×10^6. Thanks to different memory layer equipped by the GPU, and the specific parallel approach employed, we pushed the boundaries from CPU. This means that a fair analysis in accuracy and efficiency terms can be done.

5 Conclusions

In this work we presented a parallel strategy, and a related implementation in a GPU environment, for computing approximate solution of a vegetation model.

The underlying numerical procedure combines a spatial semi-discretization and the use of Peer methods for the numerical solution of systems of Ordinary Differential Equations. Numerical results and test are provided and show the gain in terms of performance of the proposed algorithm.

Acknowledgements. The authors are members of the GNCS group. This work is supported by GNCS-INDAM project and by the Italian Ministry of University and Research (MUR), through the PRIN 2020 project (No. 2020JLWP23) "Integrated Mathematical Approaches to Socio-Epidemiological Dynamics" (CUP: E15F21005420006) and the PRIN 2017 project (No. 2017JYCLSF) "Structure preserving approximation of evolutionary problems".

References

1. Butcher, J.C.: Implicit Runge-Kutta processes. Math. Comp. **18**, 50–64 (1964)
2. Butcher, J.C.: Numerical Methods for Ordinary Differential Equations, 2nd edn. Wiley, Chichester (2008)
3. Butcher, J.C.: General linear methods. Acta Numer. **15**, 157–256 (2006)
4. Calvo, M.P., Gerisch, A.: Linearly implicit Runge-Kutta methods and approximate matrix factorization. Appl. Math. **53**(2–4), 183–200 (2005)
5. Conte, D., D'Ambrosio, R., Paternoster, B.: GPU-acceleration of waveform relaxation methods for large differential systems. Numer. Algorithms **71**(2), 293–310 (2015). https://doi.org/10.1007/s11075-015-9993-6
6. Conte, D., Paternoster, B.: Parallel methods for weakly singular Volterra integral equations on GPUs. Appl. Numer. Math. **114**, 30–37 (2016)
7. Cuomo, S., De Michele, P., Galletti, A., Marcellino, L.: A GPU-parallel algorithm for ECG signal denoising based on the NLM method. In: 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 35–39, March 2016
8. Conte, D., D'Ambrosio, R., Pagano, G., Paternoster, B.: Jacobian-dependent vs. Jacobian-free discretizations for nonlinear differential problems. Comput. Appl. Math. **39**(3), 1–12 (2020). https://doi.org/10.1007/s40314-020-01200-z
9. Conte, D., Mohammadi, F., Moradi, L., Paternoster, B.: Exponentially fitted two-step peer methods for oscillatory problems. Comput. Appl. Math. **39**(3), 1–19 (2020). https://doi.org/10.1007/s40314-020-01202-x
10. Conte, D., Pagano, G., Paternoster, B.: Jacobian-dependent two-stage peer method for ordinary differential equations. In: Gervasi, O., et al. (eds.) ICCSA 2021, Part I. LNCS, vol. 12949, pp. 309–324. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86653-2_23
11. Conte, D., Pagano, G., Paternoster, B.: Two-step peer methods with equation-dependent coefficients. Comput. Appl. Math. **41**(4), 140 (2022)
12. Eigentler, L., Sherratt, J.A.: Metastability as a coexistence mechanism in a model for dryland vegetation patterns. Bull. Math. Biol. **81**, 2290–2322 (2019). https://doi.org/10.1007/s11538-019-00606-z
13. Schmitt, B.A., Weiner, R.: Parallel start for explicit parallel two-step peer methods. Numer. Algorithms **53**(2), 363–381 (2010). https://doi.org/10.1007/s11075-009-9267-2
14. Schmitt, B.A., Weiner, R., Jebens, S.: Parameter optimization for explicit parallel peer two-step methods. Appl. Numer. Math. **59**(3–4), 769–782 (2009)

15. Schmitt, B.A., Weiner, R., Podhaisky, H.: Multi-implicit peer two-step W-methods for parallel time integration. BIT Numer. Math. **45**(1), 197–217 (2005). https://doi.org/10.1007/s10543-005-2635-y
16. Schmitt, B.A., Weiner, R., Erdmann, K.: Implicit parallel peer methods for stiff initial value problems. Appl. Numer. Math. **53**(2–4), 457–470 (2005)
17. Weiner, R., Schmitt, B.A., Podhaisky, H.: Parallel "Peer" two-step W-methods and their application to MOL-systems. Appl. Numer. Math., **48**(3–4), 425–439 (2004)
18. Schmitt, B.A., Weiner, R.: Parallel two-step W-methods with peer variables. SIAM J. Numer. Anal. **42**(1), 265–282 (2004)
19. https://developer.nvidia.com/cuda-zone
20. De Luca, P., Galletti, A., Marcellino, L.: A Gaussian recursive filter parallel implementation with overlapping. In: 2019 15th International Conference on Signal-Image Technology and Internet-Based systems (SITIS), pp. 641–648 (2019)
21. De Luca, P., Galletti, A., Giunta, G., Marcellino, L.: Accelerated Gaussian convolution in a data assimilation scenario. In: Krzhizhanovskaya, V.V., et al. (eds.) ICCS 2020, Part VI. LNCS, vol. 12142, pp. 199–211. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50433-5_16
22. De Luca, P., Galletti, A., Ghehsareh, H.R., Marcellino, L., Raei, M.: A GPU-CUDA framework for solving a two-dimensional inverse anomalous diffusion problem. In: Foster, I., Joubert, G.R., Kučera, L., Nagel, W.E., Peters, F. (eds.) Parallel Computing: Technology Trends, Advances in Parallel Computing, vol. 36, pp. 311–320 (2020)
23. De Luca, P., Galletti, A., Giunta, G., Marcellino, L.: Recursive filter based GPU algorithms in a data assimilation scenario. J. Comput. Sci. **53**, 101339 (2021)
24. Jones, S.: Introduction to dynamic parallelism. In: GPU Technology Conference Presentation, vol. 338 (2012)
25. Hairer, E., Norsett, S.P., Wanner, G.: Solving Ordinary Differential Equations I: Nonstiff Problems, 2nd edn. Springer, Berlin (1993). https://doi.org/10.1007/978-3-540-78862-1
26. Hundsdorfer, W., Verwer, J.: Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations. Springer Series in Computational Mathematics, Berlin (2003). https://doi.org/10.1007/978-3-662-09017-6
27. Ixaru, L.G.: Runge-Kutta methods with equation dependent coefficients. Comput. Phys. Commun. **183**(1), 63–69 (2012)
28. Jebens, S., Weiner, R., Podhaisky, H., Schmitt, B.: Explicit multi-step peer methods for special second-order differential equations. Appl. Math. Comput. **202**(2), 803–813 (2008)
29. Klinge, M., Weiner, R., Podhaisky, H.: Optimally zero stable explicit peer methods with variable nodes. BIT Numer. Math. **58**(2), 331–345 (2017). https://doi.org/10.1007/s10543-017-0691-8
30. Rosenbrock, H.H.: Some general implicit processes for the numerical solution of differential equations. Comput. J. **5**(4), 329–330 (1963)
31. Sanz-Serna, J.M., Verwer, J.G., Hundsdorfer, W.H.: Convergence and order reduction of Runge-Kutta schemes applied to evolutionary problems in partial differential equations. Numer. Math. **50**, 405–418 (1986). https://doi.org/10.1007/BF01396661
32. Weiner, R., Biermann, K., Schmitt, B., Podhaisky, H.: Explicit two-step peer methods. Comput. Math. Appl. **55**(4), 609–619 (2008)