

A Modified Algorithm of Porting a Wave Motion

Alexander V. Bogdanov, Vladimir V. Mareev, Nikita Storublevtcev(✉), Victor Smirnov, and Ivan Podsevalov

Faculty of Applied Mathematics and Control Processes, Saint-Petersburg State University, 198504 Petergof, Saint-Petersburg, Russia
{bogdanov,map}@csa.ru, 100.rub@mail.ru

Abstract. The purpose of our work is to identify the key features of algorithms in porting codes for calculating of essentially nonlinear processes to a modern cluster of hybrid architecture. The KPI equation with a source term was numerically modeled to reveal the features of the occurrence of extreme waves. A methodology for the implementation of boundary conditions for the modified KPI equation was also proposed.

Keywords: High performance computing · CPU architectures · GPU · FPGA · Rogue wave · KPI

1 Introduction

One of the main challenges of computational physics is the high computational cost of any simulation sufficiently complex to be practically useful. With the continuous advancement of computer hardware, this problem is slowly but steadily solved. The highly parallel architecture of General-Purpose Graphical Processing Units (GPGPUs) lends itself well to the simulation of various processes, such as fluid dynamics.

Unlike previous works dedicated to porting of applications to a different software-hardware platform [16–18], in this paper we present a modified algorithm of porting, using a wave motion model based on heterogeneous generalized two-dimensional Kadomtsev–Petviashvili equation taking into account the new boundary conditions as an example.

2 The Problem

Extreme waves—frequently called “killer waves” (“rogue waves” or “freak waves”)—are lone waves of at least 20-30 m in height, which unexpectedly arise in the ocean with otherwise little disturbance and have completely unpredictable and uncharacteristic behavior compared to other ocean waves [1–8].

The study of the generation of powerful waves, in particular, solitary, is carried out using numerical simulation. For this purpose, model partial differential equations are used. Some important information is obtained as a result of observations and experiments.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 40–50, 2020.
https://doi.org/10.1007/978-3-030-58817-5_4

In our previous analysis it was shown that the main problems in the numerical simulation of two-dimensional waves with the effects of an avalanche-like increase in amplitude are associated with changes in the type of equations in the transition from a continuum to a discrete analogue of the problem. In particular, the loss of full integrability or reduction in the number of conservation equations in a discrete problem is critical [14, 19]. As a result of this, a number of compensation mechanisms typical for nonlinear waves are destroyed, and effects called artifacts appear. Since these effects are most pronounced in completely integrable systems, the most "unpleasant" model with the maximum violation of the compensation conditions, namely KPI, was chosen for analysis.

Kadomtsev-Petviashvili equation $\left[u_t + 0.5(u^2)_x + \beta u_{xxx}\right]_x = \eta u_{yy}$ in regards to the function $u(x, y, t)$ is usually labeled as "KPI equation" and is considered at $t \geq 0$, $x, y \in (-\infty, \infty)$, $\beta, \eta > 0$. This equation is also called "unstable Kadomtsev-Petviashvili equation" or "focusing effect equation".

Earlier, for the two-dimensional KPI equation with growing solutions, a strategy for creating mesh methods was formulated and analyzed. When solving such problems numerically, the possibility of parallelizing programs using graphic processors was tested.

Using finite-difference approach for KPI equation calculation is not effective due to the presence of the mixed derivative u_{tx}.

Instead of the original equation KPI its integro-differential analogue is considered [16]:

$$u_t + 0.5\left(u^2\right)_x + \beta u_{xxx} = \eta S(x, y, t) + G(x, y) \tag{1}$$

Equation (1) with respect to function $u(x, y, t)$ is considered in the domain $t \geq 0$, $x, y \in (-\infty, \infty)$, $\beta, \eta \geq 0$, $S(x, y, t) = \int_{-\infty}^{x} u_{yy}(x', y, t)dx'$; $G(x, y)$ is external source. Solution of the Eq. (1) is sought for initial distribution $u(x, y, 0)$.

The continuous region of the equation KPI specification is replaced by a finite computational domain $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [0, T]$. A uniform differential grid with time t and spatial coordinates x and y is set in the calculation area:

$$x_j = (j - j_0)\Delta x, j \in [1, M], x_{\min} = x_1, x_{\max} = x_M$$

$$y_k = (k - L/2)\Delta y, k \in [1, L], y_{\min} = y_1, y_{\max} = y_L$$

$$y_k = (k - L/2)\Delta y, k \in [1, L], y_{\min} = y_1, y_{\max} = y_L$$

$t^n = n\Delta t, n = 0, 1, 2, \ldots, T/\Delta t - 1$, with $\Delta x, \Delta y$ being spatial coordinates steps, Δt being time step, j_0 being the index corresponding to $x = 0$, $u[(j - j_0)\Delta x, (k - L/2)\Delta y, n\Delta t] = u^n_{j,k}$.

For (1) the approximation is performed using the central-difference operators.

$$u_{j,k}^{n+1} - u_{j,k}^{n} + \Delta t/(4\Delta x)\left(F_{j+1,k}^{n+1} - F_{j-1,k}^{n+1}\right) + \beta\Delta t/(2\Delta x^3)\left(u_{j+2,k}^{n+1} - 2u_{j+1,k}^{n+1} + 2u_{j-1,k}^{n+1} - u_{j-2,k}^{n+1}\right) = \Delta t\eta S_{j,k}^{n+1} + \Delta t G_{j,k} \tag{2}$$

The integral $S(x, y, t)$ is calculated by the trapezoid method, the integrand derivative u_{yy} is approximated by second order central differences:

$$\int_{-\infty}^{x_j} u_{yy}dx' \approx \int_{x_{\min}}^{x_j} u_{yy}dx' \equiv S_{j,k}^{n}$$

$$S_{j,k}^{n} \approx \frac{\Delta t}{\Delta y^2}\left(u_{j,k-1}^{n+1} - 2u_{jk}^{n+1} + u_{j,k+1}^{n+1}\right).;$$

$$u_{yy} \approx \frac{u_{j,k-1}^{n} - 2u_{j,k}^{n} + u_{j,k+1}^{n}}{\Delta y^2} \tag{3}$$

The resulting system of difference equations (2) is reduced to the form:

$$a_j\Delta u_{j-2,k}^{n+1} + b_j\Delta u_{j-1,k}^{n+1} + c_j\Delta u_{j,k}^{n+1} + d_j\Delta u_{j+1,k}^{n+1} + e_j\Delta u_{j+2,k}^{n+1} = f_{j,k}^{n} \tag{4}$$

with $\Delta u_{j,k}^{n+1} = u_{j,k}^{n+1} - u_{j,k}^{n}$ and $F_{j,k}^{n+1} \equiv (u^2)_{j,k}^{n+1} = (u^2)_{j,k}^{n} + 2u_{j,k}^{n}\Delta u_{j,k}^{n+1} + O(\Delta t^2)$.

The boundary conditions KPI are used: $u_x = u_{xx} = 0$ along boundary lines x_1 and x_M, and $u_y = 0$ along the lines y_1 and $y_L, x_{\min} = x_1, x_{\max} = x_M, y_{\min} = y_1, y_{\max} = y_L$:

$$u_{-1,k}^{n} = u_{0,k}^{n} = u_{1,k}^{n}; u_{M+2,k}^{n} = u_{M+1,k}^{n} = u_{M,k}^{n};$$

$$u_{j,0}^{n} = u_{j,1}^{n}; u_{j,L+1}^{n} = u_{j,L}^{n}$$

KPI solutions dampen proportionally to the square of distance $O[1/(x^2 + y^2)]$, which means that in practical calculations it is impossible to choose a sufficiently large area $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ to use uniform Dirichlet conditions.

The coefficients $a_j, b_j, c_j, d_j, e_j, f_j$ (4) are determined from relations for the derivatives. Difference scheme is an implicit linearized finite-difference scheme. Quasi-one-dimensional system is solved by the five-point sweep method. In some necessary cases, a flow correction procedure (FCT) is used [9].

A series of previous papers [10–16] was devoted to the numerical simulation of nonlinear KdVB and KPI type equations. In these papers, a difference scheme was chosen, its testing was carried out, and numerical results were obtained. Various methods for numerical solution of KPI were discussed in [13].

The main features of the difference scheme are illuminated in [12, 13, 15].

As an initial distribution is considered the ellipsoid of rotation:

$$u(x, y, 0) = \sigma \exp\left[-\omega\left(x^2/a_1^2 + y^2/b_1^2\right)\right], \tag{5}$$

where $\sigma > 0$, the volume $V_1 = \pi\sigma a_1 b_1/\omega$, and a_1, b_1 being the half axis.

A distribution of sources as an ellipsoid of rotation is chosen:

$$G(x, y) = c_2\sqrt{1 - (x - x_0)^2/a_2^2 - (y - y_0)^2/b_2^2}, \tag{6}$$

with the volume $V_2 = 2\pi a_2 b_2 c_2/3$, and a_2, b_2, c_2 being the half axis, (x_0, y_0)—center of ellipsoid.

When calculating such an Eq. (1) by a finite-difference method in a bounded rectangular region, the integral S introduces a significant error in the solution. Because of the finite region of calculation, the integral S at the boundaries of the region does not tend to zero. In order to get rid of the distortion of the numerical solution, due to the reasons described above, the following procedure is proposed [19].

From the test results, several conclusions were drawn – the schema has sufficient resolution for areas of high gradient; the schema describes the process of formation and distribution of solitons with characteristic conservation very well; the scheme satisfactorily handles cases with initial distributions that are not entirely integrable; a more powerful hardware is needed to model long-term cases.

When calculating the equation system (4) with boundary conditions (5), the flow condition along the y axis are not implemented correctly, which leads to reflection condition.

To be able to correctly set the boundary condition of the flow type $u_y = 0$ at $y = y_{\min}$, $y = y_{\max}$ in the left part of Eq. (2) a small quantity is introduced, which can be called artificial convection over y: $-\varepsilon \cdot \operatorname{sign}(u_y) u_y$, where the parameter $\varepsilon > 0$ and small enough ($\varepsilon \ll 1$), such as not to introduce significant error in the solution of the corresponding Eq. (1).

Instead of the integral $S(x, y, t)$, the function $P(y)$ is introduced:

$$\tilde{S}(x, y, t) = P(y) \cdot S(x, y, t),$$

where

$$P(y) = \begin{cases} 1, y \in [y_{\min} + \delta, y_{\max} - \delta] \\ h^-(y), y \in [y_{\min}, y_{\min} + \delta] \\ h^+(y), y \in [y_{\max} - \delta, y_{\max}] \end{cases}$$

For functions $h^-(y)$ and $h^+(y)$ fulfil the symmetry condition: $h^-(y_{\min} + y^*) = h^+(y_{\max} - y^*)$ at $y^* \in [0, \delta)$.

When choosing the finite function $P(y)$, and practically the function $h^+(y)$, there are two criteria:

The function $h^+(y)$ should monotonously decrease in the band adjacent to the upper boundary $y_{\max}$ (increase in the lower band for $y_{\min}$).

The derivative of function $h^-(y)$ and $h^+(y)$ at the boundaries $y_{\min}$ and $y_{\max}$ should be as small as possible.

During the choice of finite function $P(y)$ several different $h^+(y)$ functions were considered, in particular, sigmoid function, 3-rd degree spline, parabolas.

3 Porting Challenges

Model in question is represented by 7 matrices of equal size, which are operated upon by 4 main functions. Each matrix holds single-precision 32-bit floating-point values representing the wave height relative to the base ocean level.

Two functions also feature temporary data arrays, size of which can differ depending on the implementation. In the original algorithm for CPU, these arrays were equal to 1–4 rows of the main matrices. Some of this temporary data can be reduced or eliminated completely by usage of additional on-the-spot calculations, but some of is required for parallelization. Reduction in temporary data increases cache localization, which can have a positive influence on performance, but inhibits parallelization, for example, automated loop unwinding.

Matrices U_0 and U_1 represent the previous and current model states respectively. Since each subsequent model state depends on a previous one, we are forced to hold both in memory at all times. Matrix $P_{j,k}$ is responsible for boundary smoothing and contains coefficients from 0 to 1, with edge cells set at zero.

Matrix $G_{j,k} \equiv G(x_j, y_k)$ is the external perturbation matrix. Matrix $(U_yy)_{j,k}$—is second derivative u_{yy} (3).

Matrix s_eta $\equiv S^n_{jk}$. Matrix $(U_y)_{j,k}$—is first derivatives u_y in boundary conditions.

The original implementation was written in Fortran 95. OpenCL was chosen as the framework for the GPGPU porting process. Main reason for that choice was high degree of portability and low dependency from the host-side code language.

As was mentioned above, heterogeneous architecture of GPGPUs is a good fit for matrix operations, but not all functions of the model allow for complete cell-level parallelism, as each cell's value may be dependent on the adjacent ones. Additionally, some functions feature accumulation calculations on one or both axes, making full parallelization very difficult.

For example, the function implementing five-point tridiagonal matrix algorithm, can be roughly represented like this:

```
for k in 1..N-1{
for j in 2..M-2{
a[j+1] = F1(a[j-1], a[j], b[j-1], b[j], u0[j*, k], u_y[j, k], s_eta[j, k],
g[j, k])
b[j+1] = F2(a[j-1], a[j], b[j-1])
c[j+1] = F3(a[j-1], c[j-1], c[j], , u0[j*, k], u_y[j, k], s_eta[j,k], g[j,
k])
}
for j in M-3..0{
U`[j] = F4(U`[j+1], U`[j+2], a[j+1], b[j+1], c[j+1])
u[j,k] = u0[j,k] + U`[j]
}
}
```

Where j* - neighborhoods j, i.e. j − 1. j. j + 1; a, b, c and U' are temporary buffer matrices, F, G and H are simple linear functions, and U is the result. From this it is easy to see that column calculation is relatively independent, while row calculation includes forward and backward accumulation. In all other main functions, every element's value is dependent on the adjacent one. For example, an element with coordinates $[x, y]$ may rely on one at $[x+/-1, y]$, and in the worst case require a simple additive accumulation.

Taking all of this into account, the main focus was made on the inter-column parallelization, which ended up being surprisingly straightforward, with very little code changes required. Five-point tridiagonal matrix algorithm implementation required buffer arrays for each row, which were allocated using private memory which for datasets of this size is automatically projected onto the global device memory, and as such does not require additional synchronization. This resulted in an expected increase of performance.

Next step was to implement the inter-row parallelism, at least in the trivial cases, such as calculation of U_yy and U_y matrices, which do not feature accumulation. This had significantly improved calculation times. This is most likely due to the memory block read optimization.

When calculating s_eta j-th row element depends on the intermediary results from j − 1-th element. In the original algorithm it was solved by allocating a temporary row s such that s[j] = s[j − 1] * F1(U_yy[j*, k]), after which s_eta was calculated as s_eta [j] = F2(s[j]). This approach is ill-suited for GPU, because cumulative addition is hard to parallelize and intermediary results have to be stored in global memory. Thankfully, it can be avoided by utilizing local memory – breaking the calculation of s into blocks, each calculating F1(U_yy[j*, k]) in parallel, after which the main thread performs the additive accumulation and immediately calculates F2(s[j]), and repeats the same for the next block. Despite the fact that F1 function is very simple in terms of calculation, this approach game a noticeable boost to performance. This is, most likely, due to the GPU architecture details – global memory access is much more efficient when reading a block of data in parallel, inside a single local group.

The last step was to improve the five-point tridiagonal matrix algorithm parallelization. In this case local memory would not help as there is simply not enough of it to contain all the required data. The solution was to create a single buffer matrix and utilize the comparatively large 4 Gb global memory space. Big memory usage is unpleasant, but acceptable if it solves a much larger problem of slow global memory access. Each matrix element is only dependent on the two previous ones, which allows us to store these two values in faster memory. This way, at the first stage of calculations, each iteration writes the data into global memory but does not read it, which allows the optimizer to perform a much faster asynchronous write. For subsequent iterations the same data is stored in private memory, which is fast for both write and read. At the second stage of calculations the intermediary data is only read from global memory, which avoid the need for additional synchronization. There was a risk that the intermediary data would not fit into the private memory, since it is very limited, but in practice there were no problems with it, and it provided a significant boost to performance.

4 Numerical Tests

The benchmarking platform specifications are as follows:

CPU AMD FX 8550 ~4 GHz

GPU AMD R9 380 ~970 MHz

For the testing purposes, the model size was limited to 960 by 640 cells, with 200 iterations, which equates to 0.01 s of simulation time. Benchmark results of different versions are presented in Table 1.

Table 1. Benchmark results

Version	Time (s)
CPU	28.683
GPU	1.355
GPU async	1.338
GPU 2d	0.304
GPU 2d async	0.302

As seen from the above Table 1, the simple inter-column parallelism increased the computation speed by ~20 times despite significantly lower GPU clock speed. Utilization of asynchronous kernel calls did provide an additional small boost to performance.

Inter-row parallelization of trivial functions cut down the total computation time by ~40%, while s_eta optimization improved it to ~50%. The last step brought it to a total of ~75% less compute time than a simple inter-column version, or about 90 times faster than the CPU version.

Unfortunately, it was impossible to showcase performance increase per core, as profiling software provided incoherent results.

5 Porting Results

An analysis of the results calculated with various finite functions showed that the most suitable for this initial distribution (5) are two functions: a spline of the 3rd order $h^+(\bar{y}) = 2\bar{y}^3 - 3\bar{y}^2 + 1$ and a parabola $h^+(\bar{y}) = \bar{y}^2 - 2\bar{y} + 1$, $\varepsilon = 0.01$.

Figure 1 shows visible effects of the interaction of propagating waves with the boundary $y = y_{\min}$ and $y = y_{\max}$. For this picture and for all furthers, the main numerical values of the parameters are the same: mesh is $800 \times 600 (x \times y)$, $\Delta x = \Delta y = 0.1$, $\Delta t = 5 \cdot 10^{-5}$, $\beta = \eta = 1$. In our case it is $y_{\min} = -31.9, y_{\max} = 48$.

The Gaussian distribution (5) is taken with $\sigma = 10$, $a_1 = 2$, $b_1 = 3$, $\omega = 1.570796$.

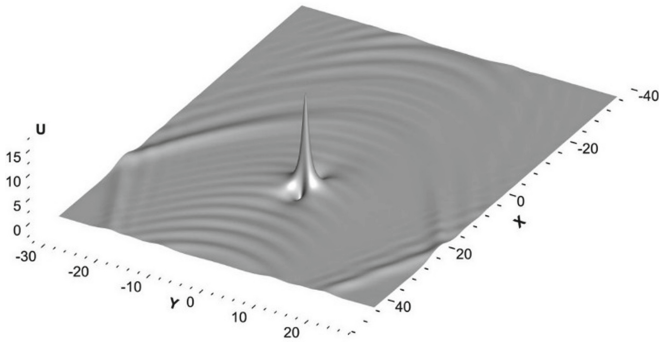

Fig. 1. The effects of the interaction of propagating waves without smoothing at $t = 8$.

Figure 2 gives a comparison of the smoothing effect for functions $h^+(\bar{y})$ for upper band: sigmoid function $h^+(\bar{y}) = 0.5[1 \pm \tanh(\bar{y}/2)]$. The width $\delta = 0,11 \cdot |y_{\max} - y_{\min}| = 6.6$ (11% from the width of the computational domain $|y_{\max} - y_{\min}|$).

The testing (Fig. 2) was conducted for the case with a rotation ellipsoid (6) source where $a_2 = 2, b_2 = 3, c_2 = 5$, i.e. $V_2 = 20\pi$. Grid: 800×600, $\Delta x = \Delta y = 0.1$, $\Delta t = 5 \cdot 10^{-5}$. $t = 14$, $\varepsilon = 0.01$ (Fig. 3).

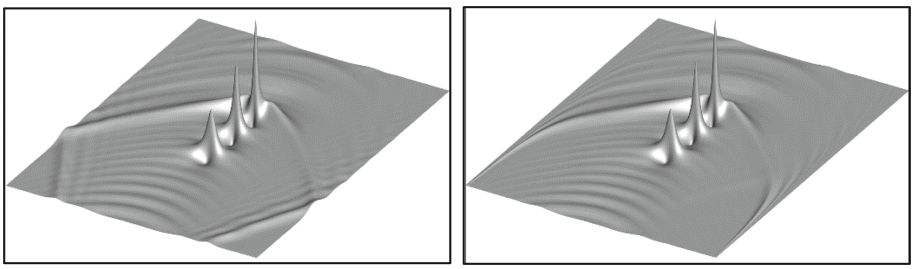

Fig. 2. Comparing the smoothing effect for functions $h^+(\bar{y})$ for upper band: sigmoid function (left) and parabola (right).

It is obvious that both functions give almost identical results (Fig. 4), but parabola gives better ones.

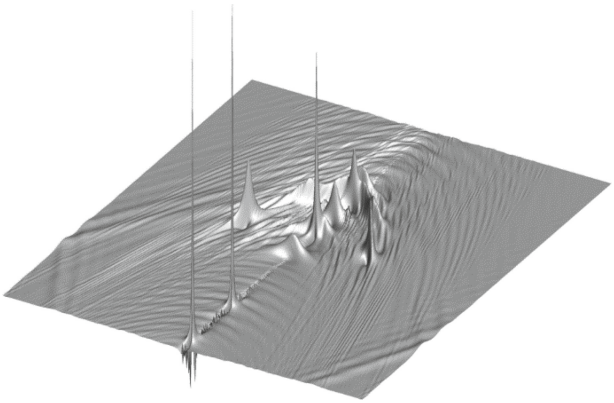

Fig. 3. External source. The effects of the interaction of propagating waves without smoothing at $t = 14$.

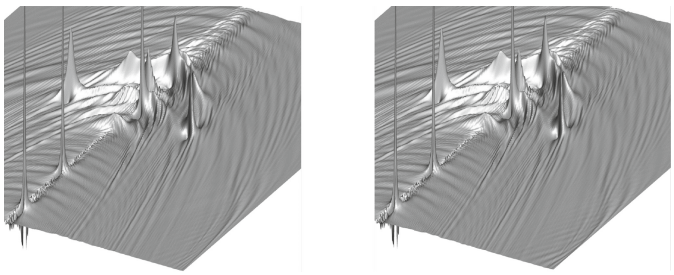

Fig. 4. External source. Comparing the smoothing effect for functions $h^+(\bar{y})$ for upper band: spline $h_3^+(\bar{y}) = 2\bar{y}^3 - 3\bar{y}^2 + 1$ (left) and parabola $h_4^+(\bar{y}) = \bar{y}^2 - 2\bar{y} + 1$ (right) at $t = 14$ and $\varepsilon = 0.01$.

6 Conclusion

The analysis of various numerical procedures for nonlinear equations describing the evolution of strong waves is carried out. An equation is chosen that is a generalization of the Kadomtsev-Petviashvili-I (KPI) equation, which shows the main part of the problems of the evolution of ocean waves and at the same time is the most difficult from the point of view of stability of the numerical algorithm. The solution of such problems reveals the features of various types of discretization.

Numerical analysis shows that the proposed approach solves at least those problems that are related to the complete integrability of the original model. A calculation method is proposed by introducing a smooth finite function and artificial convection. It

works quite efficiently, at least for nonlinear equations with an infinite evolution domain. The proposed procedure showed acceptable accuracy.

We managed not only to clearly demonstrate the effects of an avalanche-like increase in the amplitude of two-dimensional waves, but also to solve all the main problems leading to numerical errors, namely, the shift of the wave crest and the appearance of artifacts. Since the transition to more realistic models of "killer waves" is associated with a decrease in the number of restrictions in the transition to a discrete model, there is every reason to hope that our method there will be even more effective.

References

1. Kundu, A.: Tsunami and Nonlinear Wave, 316 pp. Springer Science & Business Media, Berlin (2007)
2. Todorov, M.D.: Nonlinear Waves: Theory, Computer Simulation, Experiment, 184 pp. Morgan & Claypool Publishers, San Rafael (2018)
3. Infeld, E., Rowlands, G.: Nonlinear Waves, Solitons and Chaos, 2nd edn, p. 406. Cambridge University Press, Cambridge (2000)
4. Zeytounian, R.K.: Theory and Applications of Nonviscous Fluid Flows, 295 pp. Springer Science & Business Media, Berlin (2012)
5. Yang, J.: Nonlinear Waves in Integrable and Non-integrable Systems, 430 pp. Society for Industrial and Applied Mathematics, Philadelphia (2010)
6. Feng, B.F., Mitsui, T.: A finite difference method for the Korteweg-de Vries and the Kadomtsev-Petviashvili equations. J. Comput. Appl. Math. **90**, 95–116 (1998)
7. Popov, S.P.: Numerical implementation of two-soliton solutions to the Kadomtsev-Petviashvili equation. Comput. Math. Math. Phys. **40**(10), 1447–1455 (2000)
8. Ayadi, M.: Numerical simulation of Kadomtsev–Petviashvili–Benjamin–Bona-Mahony equations using finite difference method. Appl. Math. Comput. **219**(24), 11214–11222 (2013)
9. Boris, J.P., Book, D.L.: Flux corrected transport I, SHASTA, a fluid transport algorithm that works. J. Comput. Phys. **11**, 38–69 (1973)
10. Bogdanov, A., Stankova, E., Mareev, V.: High performance algorithms for multiphase and multicomponent media. In: 14th Ship Stability Workshop, pp. 242–245, UTMSPACE, Malaysia (2014)
11. Bogdanov, Alexander V., Mareev, Vladimir V., Stankova, Elena N.: Hybrid approach perturbed KdVB equation. In: Gervasi, O., et al. (eds.) ICCSA 2015. LNCS, vol. 9158, pp. 331–341. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21410-8_26
12. Bogdanov, A., Mareev, V.: Numerical simulation perturbed KdVB equation. In: EPJ Web of Conferences [MMCP 2015 Stará Lesná, Slovakia, 2015], vol. 108, p. 02014 (2016)
13. Bogdanov, A.V., Mareev, V.V.: Numerical simulation KPI equation. In: Proceedings of the 15th International Ship Stability Workshop, 13–15 June, Stockholm, Sweden, pp. 115-117 (2016)
14. Bogdanov, A.V., Mareev, V.V.: Hybrid approach for nonlinear wave equation. In: AIP Conference Proceedings, Published by the American Institute of Physics (ICNAAM 2016), vol. 1863, 9–25 September 2016, Rhodes, Greece, p. 110005 (2017). 10.1063/1.4992290
15. Bogdanov, A.V., Mareev, V.V., Kulabukhova, N.V., Degtyarev, A.B., Shchegoleva, N.L.: Influence of external source on KPI equation. In: Gervasi, O. (ed.) ICCSA 2018. LNCS, vol. 10963, pp. 123–135. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95171-3_11

16. Stepanov, E., Khmel, D., Mareev, V., Storublevtcev, N., Bogdanov, A.: Porting the algorithm for calculating an asian option to a new processing architecture. In: Gervasi, O. (ed.) ICCSA 2018. LNCS, vol. 10963, pp. 113–122. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95171-3_10
17. Bogdanov, A., Storublevtcev, N., Mareev, V.: On porting of applications to new heterogeneous systems. In: Proceedings of the VIII International Conference "Distributed Computing and Grid-technologies in Science and Education" (GRID 2018), pp. 328–332 (2018)
18. Bogdanov, A.V., Mareev, V.V., Storublevtcev, N.: Algorithms for the calculation of nonlinear processes on hybrid architecture clusters. In: Proceedings of the VIII International Conference "Distributed Computing and Grid technologies in Science and Education" (GRID 2018), September 10-14 Dubna, Moscow region, Russia, RWTH Aahen University, pp. 333–336, ISSN 1613-0073 (2018)
19. Bogdanov, A.V., Mareev, V.V.: On some problems of the numerical implementation of nonlinear systems on example of KPI equation. In: EPJ Web of Conferences. Mathematical Modeling and Computational Physics 2019, vol. 226, Stara Lesha, Slovakia, p. 01003 (2020). https://doi.org/10.1051/epjconf/202022601003