UDC 519.64:004.032.24

Acceleration of computation of the discrete currents method by modification, which takes into account the architectural features of a modern PCs

VO Mishchenko, BV Patochkin
V. Karazin Kharkiv National University, Ukraine

The authors have modified the popular techniques of modeling the diffraction phenomena in the resonance wavelength range in order to accelerate calculations using the possibilities hidden in the architecture of modern PCs. This modification takes advantage of the CPU vector registers. In particular, the authors have employed the said vector registers in combination with cache memory performance optimization for modification of the discrete currents method (DCM). Their results are clearly illustrated by the following example: solving the diffraction problem (for the wavenumber of 14π) on 80 screens using 4-core CPU was 30 to 40 times as fast as with the single-core processor without vector registers involvement and processor cache optimization.

Key words: *diffraction, discrete currents method, vector registers, concurrency, Discrete Singularities Methods, system of linear algebraic equations, computing experiment.*

Здійснено модифікацію популярного методу моделювання дифракційних явищ у резонансному діапазоні хвиль з метою прискорення обчислень за рахунок використання можливостей, прихованих в архітектурі сучасних ПК. Ця модифікація використовує векторні регістри процесора. В результаті модифікації методу дискретних струмів (МДТ), спрямованої на використання векторних регістрів в поєднанні з оптимізацією роботи з кеш пам'яттю, вдалося отримати результат, який добре ілюструється таким прикладом. При розв'язку за МДТ дифракційної задачі на 80 екранах (хвильове число 14π), на 4-ядерному процесорі спостерігалося прискорення розрахунку в 30-40 разів порівняно з виконанням на одному ядрі без використання векторних регістрів і оптимізації роботи з кешем процесора.

Ключові слова: *дифракція, провідний екран, метод дискретних струмів, векторні регістри, тайлінг, паралелізм, МДО, СЛАР, експеримент, швидкість обчислень.*

We develop a modification of the popular modeling techniques diffraction phenomena in the resonance wavelength range in order to accelerate the calculations at the expense of the opportunities hidden in the architecture of modern PCs. This modification uses the vector registers of the processor. As a result of the modification of the method of discrete currents (DCM), which aims to use the vector registers, combined with the optimization of the work with the cache memory, we got the result, which is well illustrated by the following example. In case of deciding the diffraction problem on 80 screens (wave number 14π) on a 4-core CPU, following the DCM, we got acceleration 30-40 times in comparison with the performance on a single processor core without the use of vector registers and cache processor optimization.

Ключевые слова: *дифракция, проводящий экран, метод дискретных токов, векторные регистры, тайлинг, параллелизм, МДО, СЛАУ, эксперимент, скорость вычислений.*

1 Introduction

There are various methods for solving the problems of the mathematical theory of diffraction. Those of them, which use the boundary integral equations, are the most effective. In the case when one can assume that the scatter (e.g., an antenna) is a set of perfectly conductive and infinitely long cylindrical screens (whose generatrices are

parallel), he obtains the equations over the contours of cross-sections of these screens. The method of discrete currents (DCM) (whose several variations are available depending on the field polarization and selected fundamental solution of the Helmholtz equation) is based on a particular system of equations of the first kind (for the current density functions over the contours) with kernels having singularities such as the Cauchy kernels, the logarithmic type kernels, or hypersingular kernels [1, 2]. A discrete model of such system depends on the corresponding quadrature formulas for integrals (which are treated, if necessary, as generalized functions) [3], and their approximate solution can be reduced to solving a system of linear algebraic equations (further – "linear system"). On this way, the computational difficulties occur in the case of resonant wavelengths, especially if there is a large number of scattering contours. As an example, see the configuration (cross section) shown in Fig. 1.

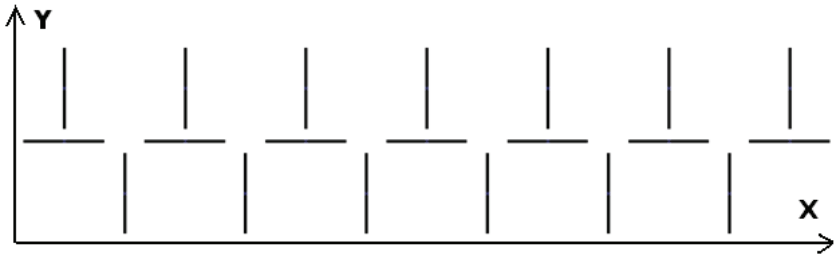

Fig.1 Example of a grid consisting of 20 metal strips

If an electrodynamic system includes dozens of elements (for example strips) and the wavenumbers are chosen in the range from 10 to 20, then, to achieve the needed accuracy, the discretization parameter value should be very large, as well as the dimension of linear system (for problems of such kind, up to several thousands). The memory of modern PCs usually has enough room to accommodate matrices of such systems, but the solution process can last for hours.

For a period of time, parallel computing in approximate numerical methods (discrete current methods among them), had been available only by using improvised clusters made of several PCs (e.g. [4-6]); this could give acceleration somewhat smaller than the number of machines or processor cores involved. Another way, based on sophisticated operating with the cache and luckily used granularity of calculations [7-9], could result in acceleration equal or even somewhat larger than the number of processor cores.

This work is aimed at such further acceleration of computations implementing DCM, which will significantly exceed the number of processor cores due to the fact that the PC architecture provides the possibility to use vector operations.

This approach presumes the maximum possible for this application type parallelization of calculations with the help of some additional specialized device with a large number of processor cores, whose the most common example is the CUDA-card [10]. However, putting aside the economic aspect, one has to keep in mind that "In course of optimization or modification of a program fragment, aimed at taking advantage of specific hardware architecture, one faces a problem of the algorithm performance limitation due to specific architectural features of an open heterogeneous system. This issue is particularly relevant in the case of specific computing devices used, such as graphics processors, signal processors, programmable logic integrated

circuits, application-oriented processors" [11, P.2] (quotation translated into English by Yu. Kolomiets).

Some issues closely related to said above and appearing in practical implementation of CUDA technology in complicated algorithms are the following:

- this requires additional equipment;
- more knowledge is needed:
  - understanding of the CUDA device principles;
  - familiarity with the special programming language for the CUDA device;
- for practically useful classes, the parallel calculation algorithm complexity often exceeds their own one.

Specifically, the following features are known [11] to complicate the block algorithms implementation in an open heterogeneous computing system based on the GPU architecture (cited by [11 P.4]).

1. The high data-level concurrency, not the code-level.
2. Structured GPU memory, whose different types have specific operation features.
3. Small volume of "fast" memory available to one computing thread at a time.
4. The specificity of logic expression processing.
5. The specificity of conditional branching and loops processing.

In this connection, it becomes relevant to check whether some possibilities for the MDO algorithms acceleration can be found directly in the architecture of modern PC main processors. Our attention was drawn to the vector registers [12]. The first vector registers appeared in the early 90s, but at that time their size was only 64 bits and they could process nothing but integers. Later, with the introduction of SSE instructions and register expansion to 128 bits, they became capable of operating with a float-point double-precision numbers. Next after SSE instructions, the so-called AVX ones appeared, and the size of vector registers had raised up to 256 bits. The latest set of instructions version for vector registers is AVX512; its name implies that their size is 512 bits. The future expansion is planned up to 1024 bits, which indicates that the technology is relevant and developing.

The real objective of integration of vector registers in the PC processor (same as in the case of CUDA) was to accelerate graphic information processing. However (again as in the case of CUDA), the programmers, upon necessity, began to use the vector registers to accelerate computation, including operating with complex numbers, which is the feature required in the DCM. The examples of complex multiplication implemented with the help of 128-bit SSE instructions and resulting gain in calculation speed were published in the WEB in 2010-11 mailing [13]. Employment of the vector registers to speed up the inversion of sparse matrices built of "small" blocks have been implemented and studied in [14]. At this point, it is necessary to note that the DCM matrices are entirely filled, and their division into blocks makes sense only in connection with development of parallel processing algorithms [5].

2 Formulation of the problem

Let, in the problem of E-polarized electromagnetic waves diffraction against a perfectly conductive contours, the time dependence is given by the amplitude factor $e^{i\omega t}$ Let the total field amplitudes are represented in this way:

$$\vec{E}^{tot} = \vec{E} + \vec{E}_0, \qquad \vec{H}^{tot} = \vec{H} + \vec{H}_0 \ , \tag{1}$$

where ($\vec{E}_0, \vec{H}_0$) - is an incident field, (*E, H*) – is a scattered field, and where

$$E_0 = (0,0,U_0), \quad H_0 = \left(\frac{1}{i\omega\mu}\frac{\partial U_0}{\partial y}, \frac{1}{i\omega\mu}\frac{\partial U_0}{\partial x}, 0 \right) \tag{2}$$

Then both the scattered and the total fields will be also E-polarized [e.g. 1],

$$E^{tot} = \left(0,0,U^{tot}\right), \; U^{tot}(X) = U(X) + U_0(X), \; X \in R^2 . \tag{3}$$

Under the above assumptions, the scattered field U satisfies the Helmholtz equation everywhere outside the set $C = \bigcup_{n=0}^{N-1} C_n$, where C_n – n^{th} contour, and *N* is the total contours number:

$$\begin{aligned} &\Delta U(X) + k^2 U(X) = 0, \\ &x \in R^2 / \overline{C}, X = (x, y) \end{aligned} \tag{4}$$

For any contour C_n the Dirichlet condition is valid

$$U(X)\big|_c = -U_0(X)\big|_c \tag{5}$$

The condition at infinity is the Sommerfeld radiation condition

$$U(X) \underset{|X|\to\infty}{=} O\left(\frac{1}{\sqrt{r}}\right), r = |X| \, , \; \frac{\partial U(X)}{\partial r} - ikU(X) \underset{x\to\infty}{=} O\left(\frac{1}{r^{3/2}}\right), \tag{6}$$

and for all contour ends the so-called "on the edge" condition is fulfilled

$$\int_{\Omega} \left(|U(X)|^2 + |\nabla U(X)|^2 \right) d\Omega < \infty \tag{7}$$

where Ω – is any bounded neighborhood of a contour end.

Let a computer solution of this problem uses the method of discrete singularities (DSM). In this paper, the version of DSM is applied, which uses the boundary integral equations with the logarithmic type kernels [1,2]. The advantages of DSM in numerical modeling of physical processes are well known [15]:

- The matrix of the process discrete model is very easy to form;
- It is possible to solve a linear system using the Gauss scheme without permutations.

However, the data exchange between running in parallel processes, needed in the course of solving the linear system obtained by problem (1)-(6) discretization (with the help of DCM or other method that uses the boundary equations), requires their systematic interaction via shared memory. This explains the non-triviality of parallel algorithms of the Gauss method. For example, the classical approach to parallelization in linear algebra problems, such as solving the linear system with an entirely filled matrix, is to divide the matrix into blocks with concurrent processing these blocks. As

it was explained above, in this case, the potential benefits of the large number of processor cores used in CUDA device cannot be implemented to the full extent.

We had in mind the task to accelerate the DCM-calculations by employment of the vector registers. For result comparability, calculations were carried out until the minimal standard numerical solution (MSNS) of diffraction problem was obtained. It means that "discrete currents" (that approximates the current density functions) were evaluated based on the discrete model of the system of boundary equations and mapping of the field was built for the far-field region (field strength directivity diagram with the pitch of 1˚ in accordance with given polarization).

The research methods and materials

The todays compilers, it seems, "know" to use the vector registers. But adequacy of their results is clearly seen comparing the source code in C++ with the corresponding code in assembler created by the compiler GCC-4.9.

Table 1 A C++ function translated into assembler code

C++ code	***The code generated by the compiler***

double VecMul(double* a, double* b, int size)
{
double sum = 0;
for (int i=0; i < size ; i++)
{
sum += a[i] * b[i];
}
return sum;
}
	VecMul(double*, double*, int):
	vxorpd %xmm0, %xmm0, %xmm0
	testl %edx, %edx
	jle .L4
	xorl %eax, %eax
	.L3:
	Vmovsd (%rdi,%rax,8), %xmm1
	vfmadd231sd (%rsi,%rax,8), %xmm1, %xmm0
	addq$1, %rax
	cmpl %eax, %edx
	jg .L3
	ret
	.L4:
	ret

An attentive study of the text representing multiplying of two vectors reveals (see Table 1) that only one value was sent to a vector register. As it was a double float number, only 64 bits of 256 available were used.

Note that the function vfmadd231sd() in the right column of the table is the special assembler instruction, which corresponds to the expression "sum += a[i] * b[i];".

As one can see from this example, today the compiler uses the vector registers just to increase the number of available registers, which yields only a few percent acceleration of an algorithm execution. To achieve more significant acceleration using such instructions, one can include in his code patches written manually in assembler. However, preferred are the so-called intrinsic-functions (i.e. the special system-dependent functions that implement some commonly used operations), which are much more efficient than the standard ones because they use known features of computer architecture at the level of assembler code [16].

In problems being solved by DCM, the most time-consuming steps on the way to MSNS are filling the linear system matrix and solving the linear system itself. Since calculation of various matrix elements follows some common algorithm, the vector registers being employed allow substituting the arithmetic operations by the vector ones. However, there are certain difficulties with the use of vector registers, if for some reasons we chose Gaussian compact scheme, as in [5-9]. This has sense, for example, when we want to use some other reserves to accelerate solution of linear system having a complex matrix. The difficulties mentioned above are the following:

- buckets allocated for triangular matrix strings are not consistent with the natural requirement of their length being proportional to the size of vector registers;
- there are no ready-made built-in operations whose operands are complex numbers stored in the vector registers, so, if necessary, they are to be coded by a developer himself.

Representation of a matrix as a set of square blocks, whose size (optimal in the sense of [2]) is a multiple of vector register size, helps to cope with the first of these complications, except the case of diagonal blocks.

Calculations implementing the compact scheme of Gauss method require operations of multiplication and subtraction of numbers (in our case, of complex numbers). Let's start with the first of them, which is more time-consuming. It consists of known steps involving the pairs of real numbers. One of these steps is swapping the numbers contained in a register. This is the known problem among software developers [13,14], however, we could not find any published results of its constructive solution for AVX registers . Here we offer our own solution.

The standard for C++ compiler representation of a complex number in the memory of PC is consecutive allocation of two double precision numbers (which takes 16 bytes). The same 16 bytes representing our complex number we write into a vector register. Calling to mind multiplication and subtraction operations, it is appropriate to define uniting them operation mulAndSub(), which will process three given complex numbers in the same sequence as in the expression A = A - B * C.

The requirement to optimize execution time of this operation makes its implementation nontrivial. We have taken into account times of each assembler instruction of those described in [16], which potentially could be involved, understanding that the instruction sequence is not unique.

To make the right decision, we needed to pick them in such a way that the total execution time (which can be estimated with the help of the table [16]) was the shortest. Eventually, the following code, which uses the vector registers, was developed:

```
void mulAndSub(cmpx* to, cmpx* from1, cmpx* from2)
void mulAndSub(cmpx* to, cmpx* from1, cmpx* from2)
 {
    __m128d* v1 = (__m128d*)from1;
    double* v2 = (double*)from2;
    double* v3 = (double*)to;
    __m256d ymm0_v1;
    __m256d ymm1_v2;
```

```
    __m256d ymm2_axa;
    __m256d ymm3_perm_v2;
    __m256d ymm4_axb;
    __m256d ymm5_bb_xch_ab;
    __m256d ymm6_aa_xch_ab;
    __m256d ymm7_res;
    ymm0_v1 = _mm256_broadcast_pd(v1);
    ymm1_v2 = _mm256_loadu_pd(v2);
    //mul
    ymm2_axa = _mm256_mul_pd(ymm0_v1, ymm1_v2);
    ymm3_perm_v2 = _mm256_permute_pd (ymm1_v2, 0b0101);
    ymm4_axb = _mm256_mul_pd(ymm0_v1, ymm3_perm_v2);
    ymm5_bb_xch_ab = _mm256_unpackhi_pd(ymm2_axa, ymm4_axb);
    ymm6_aa_xch_ab = _mm256_unpacklo_pd(ymm2_axa, ymm4_axb);
    ymm7_res=_mm256_addsub_pd(ymm6_aa_xch_ab, ymm5_bb_xch_ab);
    //sub
    __m256d ymm9_v3 = _mm256_loadu_pd(v3);
    ymm9_v3 = _mm256_sub_pd(ymm9_v3, ymm7_res);
    _mm256_storeu_pd((double*)to, ymm9_v3);
}
```

4 Computational experiments

In numerical experiments conducted to estimate the count rate, the processor Intel core i5-4430 was used. The size of its vector registers is 256-bit that allows fitting in two complex numbers of the type double. Three C++ implementations of obtaining MSNS using DCM with the core of the logarithm type were compared: serial processing (basic), optimized relatively grain size [17, 18] and other parameters in accordance with [7-9], and its modification intended to use vector registers.

Consider the case of the E-polarized wave falling against the electrodynamic structure shown in Fig. 1 in direction of the axis Y at the angle of π to it with the wave number $\kappa = 4\pi,\ 14\pi$. To obtain MSNS, we calculated (see example in fig. 2-3) the

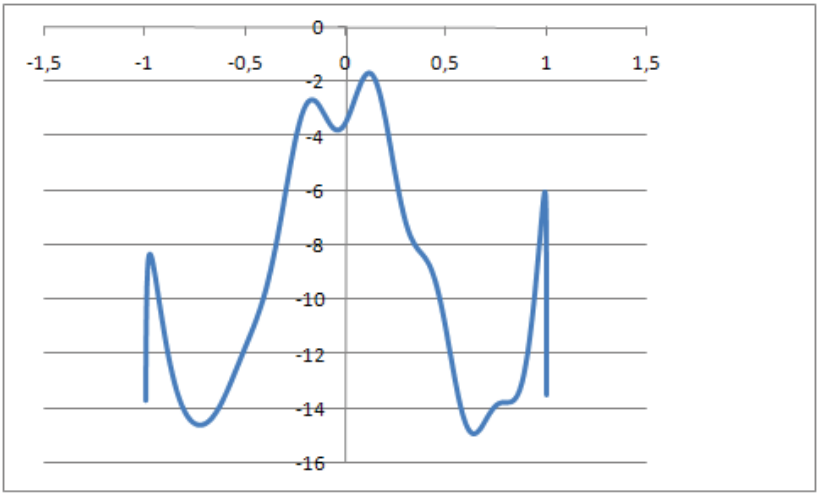

Fig. 2 The real part of the current density function of the first horizontal contour, the wavelength is 1/7 of strip width, totally 20 contours (fig. 1)

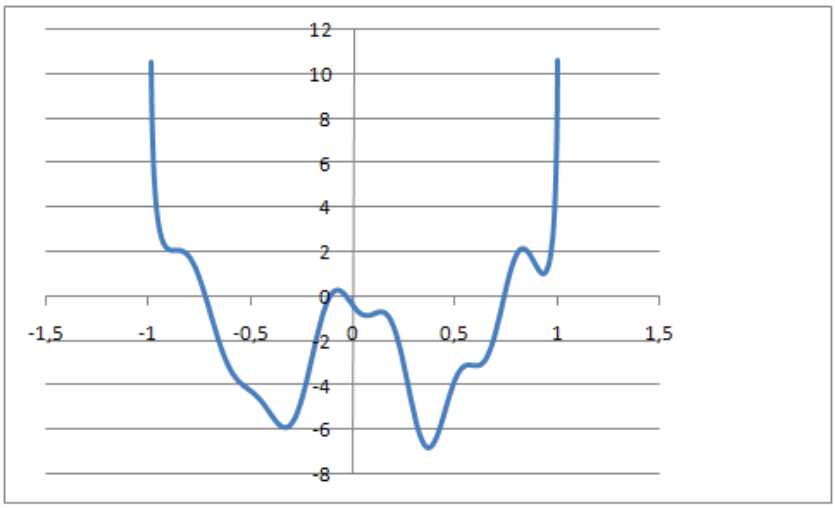

Fig. 3 Imaginary part of the current density function of the first horizontal contour, the wavelength is 1/7 of strip width, totally 20 contours (fig. 1)

DCM-approximation of current density for all strips and built the radiation pattern. The example in Fig. 4 illustrates a diagram of the amplitude absolute value of current obtained for the same electrodynamic structure (fig. 1), but for the shorter wavelength (½ the width of the strip) to provide smaller overlapping of minor lobes in a small figure.

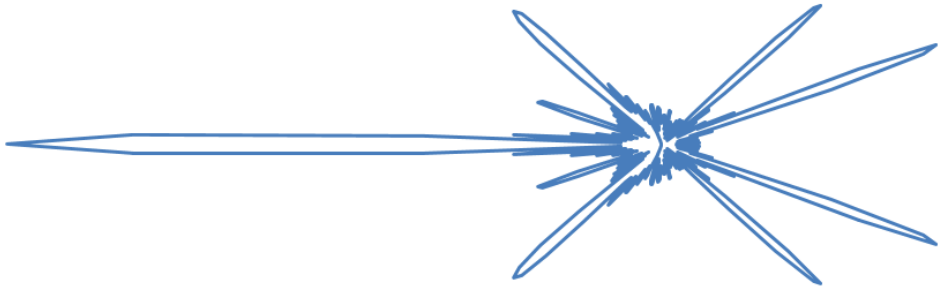

Fig. 4 Field strength directivity diagram (is rotated by π/2), the wavelength of ½, 20 contours

At $\kappa = 14\pi$, the dimension of the discrete model was 10^3 (there were about seven discrete singularities for the wavelength); to verify achieved accuracy, the discretization parameter each time was multiplied approximately by $\sqrt{3}$. Table 1 contains timing of DCM calculations. Similar calculations, at $\kappa = 4\pi, 14\pi$, were performed also for 80 contours, which in addition were arcuate. Note that in numerical experiments with electrodynamic structures, whose screen cross section contours had different forms, but matrices were equidimensional, duration of MSNS computing by DCM varied not more than for 1%. Therefore, two first digits in the index of achieved acceleration, by Table 1, may, in any case, serve as a guide.

Table 1 Three implementations of MSNS calculation compared by their execution time

N: the problem dimension	**t_1:** the base implementation	**t_2:** tiling and multithreading	**t_3:** + vector registers	**t_1/t_3 :** index of total acceleration
1000	5.17	0.88	0.33	15.7
1731	28.7	3.97	1.25	23.0
2996	182	20.1	5.70	32.0
5186	1180	101	28.3	41.5
8978	6500	526	143	45.3

5 Conclusions

We have created a modifications of DCM, which uses the vector registers of modern PC processors in combination with tiling effect. It is experimentally proved that this modification accelerates in dozens of times computing MSNS for problems of electromagnetic waves diffraction against a perfectly conducting screen (for 2D case). Note that an important role in this result plays more fast execution of the algorithm for solving the linear algebraic system (using the Gaussian compact scheme). Our subroutine can be used in other application areas apart from computational problems of electrodynamics, particularly, when linear system having a complex matrix is the most time-consuming part of computational work and it is necessary to accelerate by an order the solution process, but all obvious reserves, such as algorithm improvement or simple parallelization of the processes, have been exhausted.

REFERENCES

1. Гандель Ю. В. Математические вопросы метода дискретных токов. Обоснование численного метода дискретных особенностей решения двумерных задач дифракции электромагнитных волн. Учебное пособие. Ч. 2 / Ю. В. Гандель, С. В. Еременко, Т. С. Полянская. – Харьков : ХГУ, 1992. – 145 с. – *Gandel' Yu. V.* Mathematical issues of the method of discrete currents. Justification of the numerical method of discrete singularities for solving two-dimensional problems of electromagnetic waves diffraction. – 1992. – in Russian.
2. Дмитриев В. И. Метод интегральных уравнений в вычислительной электродинамике / Дмитриев В И., Захаров Е. В. - М.: МАКС Пресс, 2008. - 316 с. – *Dmitriev V. I.* The method of integral equations in computational electrodynamics. – 2008. – in Russian.
3. Гандель Ю. В. Введение в методы вычисления сингулярных и гиперсингулярных интегралов [учебное пособие] / Ю. В. Гандель. – Х.: ХНУ, 2001. – 92 с. – *Yu. V. Gandel,* Introduction to methods of evaluation of singular and hypersingular integrals. – 2001. – in Russian.
4. Мищенко В.О. К моделированию электромагнитных явлений на базе использования методов дискретных особенностей для решения гиперсингулярных интегральных уравнений / В. О. Мищенко // Труды

международной конференции по вычислительной математике МКВМ–2004. Ч. II., Новосибирск: ИВМиМГ РАН, 2004. – С. 555–560. – *Mishchenko V. O.* To the modeling of electromagnetic phenomena based on application of discrete singularities methods for solving hypersingular integral equations. – in Proceedings of ICCM-2004. – 2001. – in Russian.

5. Mishchenko V. O. Ada programming language and specifying, modeling and distributed computing for the implementation of the discrete singularities methods / V. O. Mishchenko // Proceedings of SCALNET–2004. – Kremenchug, 2004. – P. 110-112. – у збірнику праць конференції. – Кременчук, 2004. – in English.
6. Gahov A. V. Parallelism for difraction processes modeling on the base of discrete singularities methods / A. V. Gahov, V. O. Mishchenko // Труды Научно–технической конф. с международным участием «Компьютерное моделирование в наукоемких технологиях» (КМНТ–2010). Ч. 2. – Х.: ХНУ, 2010. – С. 50–53. – in Proceedings of CMHT. – Kharkiv, 2010. – in English.
7. Мищенко В. О. Оптимизация компактной схемы Гаусса для многоядерных процессоров / В. О Мищенко, Б. В Паточкин. // Вестник ХНУ. – 2001. – № 981. – сер. МІА, вып. 18. –С. 70–81. – *Mishchenko V. O.* Gaussian compact scheme optimization for multicore processors. – Bulletin of KNU. — 2001. – in Russian.
8. Parallel Gaussian compact scheme [Электронный ресурс] / V. O. Mishchenko, B. V. Patochkin. – Режим доступу : http://www.mediascan.by/index.files /parallel_gaussian_compscheme_win32.zip. – Дата доступу : 18.03.2015. – Назва з экрану.
9. Б. В. Паточкин Минимизация времени компьютерного моделирования задачи дифракции на экранах методом оптимизации локальности памяти / Б. В. Паточкин // Вісник Кременчуцького національного університету імені Михайла Остроградського. – 2014. – Випуск 6 (89). – 2014. – С. 58-64. – *B. V. Patochkin* Minimization of computer modeling time by local memory optimization for a problem of diffraction against the screens. – Bulletin of Kremenchuk National University. – 2014. – Vol. 6 (89). – in Russian.
10. Боресков А. В. Основы работы с технологией CUDA / А. В. Боресков, А. А. Харламов. – М. : ДМК Пресс, 2010. – 232 с. – *Boreskov A. V.* Basics of using the CUDA technology. – 2010. – in Russian.
11. Андрианова Е. Г. Метод оценки эффективности реализации блочного алгоритма на основе графического процессора в открытой гетерогенной системе / Е. Г. Андрианова, Д. И. Мирзоян, А. Б. Петров // Журнал радиоэлектроники. – 2014. – N3. – С. 1-15. – *Andrianova E. G.* The method of efficiency assessment of block algorithm implementation based on GPU in an open heterogeneous system. – Radio Electronics Magazine, N3. — 2014. – in Russian.
12. Паточкін Б. В. Модифікація методу дискретних токів для використання векторних регістрів процессору ПК при числовому моделюванні дифракції на екранах / Б.В. Паточкін, В.О. Міщенко // Матеріали конференції «Сучасні проблеми моделювання та обчислювальних методів». – Рівне, 2015. С. 130. – *Patochkin B. V.* Modification of discrete currents method aimed at vector registers of processor PC employment in numerical simulation of diffraction against the

screens. – in Proceedings of "Modern Problems of the conferrence ‘Mathematical Modeling and Computational Methods’. – Rivne, 2015. – in Russian.
13. Complex Mul and Div using sse Instructions [Электронный ресурс]. – Режим доступу : http://stackoverflow.com/questions/3211346/complex-mul-and-div-using-sse-instructions. – Дата останнього оновлення 2015.6.19.
14. Применение векторных инструкций в алгоритмах блочных операций линейной алгебры / А. Е. Андреев, В. А. Егунов, А. А. Насонов, А. А. Новокщёнов // ИЗВЕСТИЯ ВолгГТУ сер. Актуальные проблемы управления, вычислительной техники и информатики в технических системах. – 2014. – Выпуск 21, № 12 (139). – С. 5-11. – Appliction of vector instructions in the block operation algorithms of linear algebra / *A. E. Andreev* and others. — News of VSTU, Volgograd, Russia. – 2014. – in Russian.
15. Белоцерковский С. М. Численные методы в сингулярных интегральных уравнениях / С. М. Белоцерковский, И. К. Лифанов. – М.: Наука, 1985. – 256 с. – *Belotserkovsky S. M.* Numerical methods in singular integral equations. – 1985. – in Russian.
16. Intel Intrinsics Guidehttps [Электронный ресурс]. – Режим доступа : //software.intel.com/sites/landingpage/IntrinsicsGuide/. – Назва з экрану.
17. Лиходед Н. А. О выборе зерна вычислений при реализации алгоритмов на параллельных компьютерах с распределенной памятью / Н. А. Лиходед, А. К Пашкович // Весці НАН Беларусі. Сер. фіз.-мат. навук. – 2008. – № 2. –С. 121—123. – *Likhoded N. A.*. On the choice of the grain in computing algorithms implementation on parallel computers with distributed memory. – Proceedings of the National Academy of Sciences of Belarus. Series of Physical-Mathematical Sciences. — 2008. – in Russian.
18. Киркорова Л. С. Параллельные алгоритмы математиеских моделей: исследование локальности и применение языка Ада / Л. С. Киркорова, С.И, Киркоров // Вестник Харк. нац. ун-та. – № 863. Сер. «Математическое моделирование. Информационные технологии. Автоматизированные системы управления». – 2009. – вып. 12. – С. 129-142. – *Kirkorova L. S.* Concurrent algorithms in mathematics models: Ada locality study and application. – Bulletin of KNU. — 2009. – in Russian.

Надійшла у першій редакції 16.05.2015, в останній - 28.05.2015.