BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 61, No. 4, 2013
DOI: 10.2478/bpasts-2013-0102

Parallel computations of the step response of a floor heater with the use of a graphics processing unit. Part 2: results and their evaluation

J. GOŁĘBIOWSKI* and J. FORENC

Faculty of Electrical Engineering, Bialystok University of Technology, 45D Wiejska St., 15-351 Białystok, Poland

Abstract. Using models and algorithms presented in the first part of the article, a spatio-temporal distribution of the step response of a floor heater was determined. The results have been presented in the form of heating curves and temperature profiles of the heater in the selected time moments. The computations results were verified through comparing them with the solution obtained with the use of a commercial program - NISA. Additionally, the distribution of the average time constant of thermal processes occurring in the heater was determined. The analysis of the use of a graphics processing unit in numerical computations based on the conjugate gradient method was done. It was proved that the use of a graphics processing unit is profitable in the case of solving linear systems of equations with dense coefficient matrices. In the case of a sparse matrix, the speed-up depends on the number of its non-zero elements.

Key words: air floor heating, step response computation, parallel computations, GPGPU.

1. Computational program parameters, material properties of the model, computing platform

The construction of the analyzed model of the heater was described in the first part of this paper (Fig. 2 in [1], where $L = 0.15$ m). In order to conduct the discrete approximation of the model, an appropriate fragment of the floor slab was covered with a finite difference mesh of an identical step along x, y axes which equal $\Delta x = \Delta y = 2.34375 \cdot 10^{-3}$ m. The mesh has 65×97 nodes. However, the number of nodes taken into consideration (and thus the number of unknown quantities of the obtained system of equations) is smaller and equals to 4289. As 2016 nodes, which are placed in the cross-section of the duct of the given temperature T_H, should be omitted. The step response was determined with the assumed time step $\Delta t = 2$ s and the $N = 27000$ number of steps. As a result, the considered time interval equals to 54 000 s (15 hours from letting the hot air into the ducts).

Denotations of material properties were explained in the first part of the article (chapter 2 in [1]). The following set of data was assumed:

$$T_{amb} = 20^{\circ}\text{C}, \quad T_H = 40^{\circ}\text{C}, \quad \lambda = 1 \text{ W/(m} \cdot \text{K)},$$
$$c = 840 \text{ J/(kg} \cdot \text{K)}, \quad \rho = 2000 \text{ kg/m}^3,$$
$$\alpha_F = 8 \text{ W/(m}^2 \cdot \text{K)}, \quad \alpha_H = 15 \text{ W/(m}^2 \cdot \text{K)}.$$

Computations were done with the use of a personal computer equipped with an Intel Core 2 Quad CPU Q9650 3.00 GHz processor, RAM memory of 4 GB and a 64-bit Windows 7 Professional operating system. A Gigabyte Nvidia GeForce GTX480 graphics card equipped with 480 streaming processors, and 1536 MB of GDDR5 memory were installed in the computer. It also contained the CUDA parallel computing environment [2, 3] in version 3.2 including libraries of numerical linear algebra operations – CUBLAS [4] and CUSPARSE [5].

2. Step responses and their computational verification

The computations of the step response were done according to the algorithm described in the first part of the article [1]. The obtained curves can be presented in a graph for selected points of the analyzed model of a floor heater. Characteristic points (A-J) were marked in Fig. 1, and the obtained graphs of the step response are presented in Fig. 2.

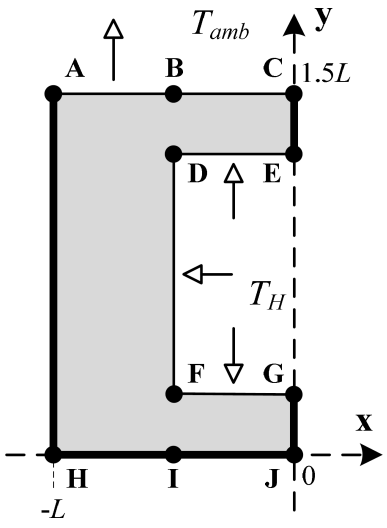

Fig. 1. Analyzed fragment of the heater with the characteristic points and adiabats

*e-mail: j.golebiowski@pb.edu.pl

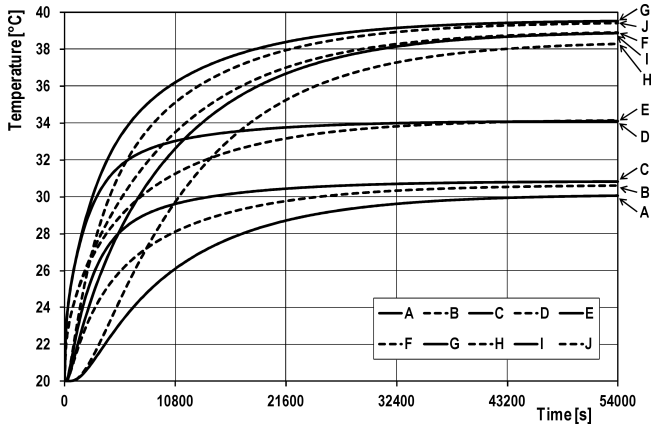

Fig. 2. Step response of the model of a floor heater in characteristic points

Points A, B and C are situated on the top surface of the floor slab, so their temperature is the lowest in comparison with the rest of the characteristic points (especially H, I and J, which are located on the bottom insulation). According to theoretical discussion, presented in the first part of the article [1], temperature $T(x,\ y = 1.5L,\ t)$ reaches its minimum value (point A) in the half distance between ducts ($x = -L$), and the maximum value for $x = 0$ (point C). A corresponding field property can be observed on the $y = 0$ surface in points H and J. Points D and E are situated in the upper part of a heating duct, and points F and G – in its lower part. Temperature in points D and E is lower than in points F and G due to the fact that the heat is transferred into the environment from the top surface of the floor slab. Points A and H are the furthest from the heat source. This results in a slow increase of curves of A and H in the beginning of the transient state (so called: apparent delay effect).

Figure 3 presents the temperature field distribution in the whole cross-section of the heater for two selected time moments (13 500 s and 54 000 s). Analyzing the figures we can notice that together with the lapse of time, the areas of higher temperature also spread. It is especially visible in the bottom part of the floor. This results from the lack of heat transfer through the bottom surface of the heater where the perfect insulation was assumed. As a conclusion, it can be said that the results presented in Figs. 2 and 3 have the appropriate physical interpretation.

In order to verify the results, the analyzed problem was solved again, with the use of a commercial program – NISA [6]. The model of a floor heater built in NISA program was of identical dimensions and material properties as in the case of original computer programs using the finite difference method (FDM). The NISA program uses the finite element method (FEM) to solve thermal problems. The analyzed model was discretized with the use of 4096 quadrilateral elements with 4289 nodes. The location of all nodes of the finite difference mesh and the finite element mesh was exactly the same. In both cases also the same time interval of the transient state analysis was assumed, as well as identical length of time step. On the basis of the obtained values, relative differences (1) of the computation results, obtained by FEM and FDM, were determined

$$\delta_h^n = \frac{h_{FEM}^n - h_{FDM}^n}{h_{FEM}^n} \cdot 100\%. \tag{1}$$

The step response h_{FEM}^n was obtained in NISA program in time moment n, whereas h_{FDM}^n is the step response in the same characteristic point and in the same time moment n, obtained with the use of the finite difference method.

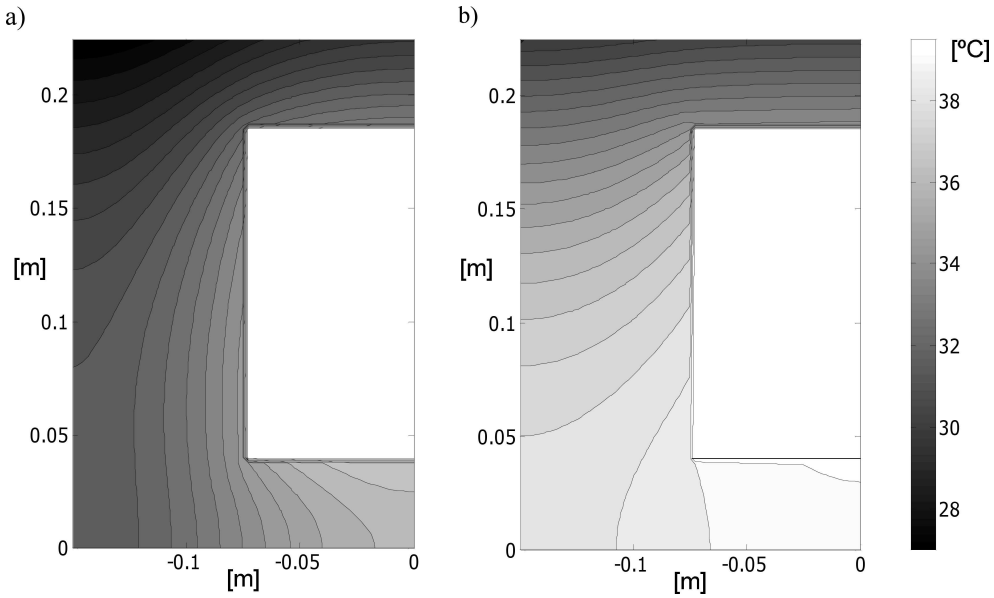

Fig. 3. Temperature field in the cross-section of a floor heater after: a) 13 500 s, b) 54 000 s

The obtained relative differences (in the whole analyzed time interval) are very small – they do not exceed the value of 0.3%. Their greatest values (about 0.29%) can be observable at the beginning of the transient state (Fig. 4). Those values relate to E and G points situated, respectively, on the bottom and top edges of a duct with hot air (on its symmetry axis) – Fig. 1. Similar situation occurs in the case of points D and F, situated on corners of a duct. The maximum value of the relative difference did not exceed 0.2% in this case. After about 160 s relative differences in all characteristic points are not bigger than ±0.025%. Because of that, Fig. 4 shows differences only for limited time interval (from 0 to 240 s). Very small values of relative differences prove that the algorithm presented in [1] is correct. The small differences result from identical location of the mesh nodes in both discrete models and fine mesh density.

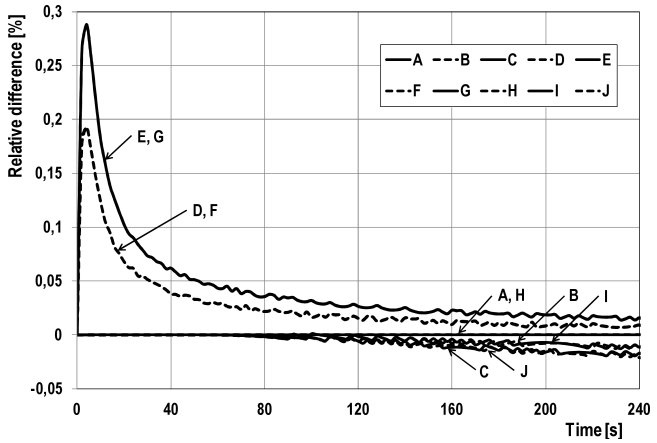

Fig. 4. Relative differences of computational results obtained by the finite element method (NISA program) and the finite difference method (original program)

3. Average time constants

On the basis of the step response of the heater, its average time constant can be computed. It depends on the position of a field point (x, y) and is defined [7, 8] as follows:

$$\tau(x,y) = \int_0^\infty \frac{h(x,y,t) - h_S(x,y)}{h(x,y,t=0) - h_S(x,y)} \mathrm{d}t, \tag{2}$$

where $h(x,y,t)$ is a step response of a heater, $h_S(x,y) = \lim_{t\to\infty} h(x,y,t)$ – stationary component of the step response, and $h(x,y,t=0)$ – the initial value of the step response. Knowing $\tau(x,y)$, we are able to approximate the dynamics of the transient state in each point of the system using the first-order lag.

Calculations of the improper integral (2) were replaced with numerical integration by Simpson's rule [9, 10]. The method is based on approximation of integrand through interpolation by a polynomial of second degree. Such a polynomial is built on three, equally-spaced, following points. For the sake of the accuracy of numerical integration, it was assumed that the time step Δt should be shortened twice. This means that the number of steps in the average time constant calculation increased twice. In this case, the use of the Simpson's rule enables us to show formula (2) in a discrete form:

$$\tau_{k,l} \approx \frac{1}{3}\Delta t^* \left(1 + 4\frac{h_{k,l}^{N^*-1} - h_{k,l}^{N^*}}{h_{k,l}^0 - h_{k,l}^{N^*}} + 2\sum_{n=1}^{\frac{N^*}{2}-1} \frac{h_{k,l}^{2n} - h_{k,l}^{N^*}}{h_{k,l}^0 - h_{k,l}^{N^*}} + 4\sum_{n=1}^{\frac{N^*}{2}-1} \frac{h_{k,l}^{2n-1} - h_{k,l}^{N^*}}{h_{k,l}^0 - h_{k,l}^{N^*}} \right), \tag{3}$$

where $h_{k,l}^{2n}$ and $h_{k,l}^{2n-1}$ are the step response values in node k, l, defined for even $2n$ and uneven $2n-1$ time moment, $h_{k,l}^0$ is the step response value in k, l node, defined in the initial time moment, $h_{k,l}^{N^*}$ – the response value in k, l node in the steady state, $\Delta t^* = 0.5 \cdot \Delta t$ – a shortened time step, $N^* = 2 \cdot N$ – the enlarged number of time steps, N – the number of time steps used in computations of the step response of a floor heater.

An exact determination of the average time constant requires calculating an improper integral (2) in the whole integration interval. Doing numerical integration, the interval should be appropriately limited. Determining the upper limit, it was assumed that enlarging the integration interval by 50% should not cause any change of the average time constant more than by 1%. Therefore, the values of the average time constant for enlarging integration interval were calculated. The obtained results for two characteristic points E and H are presented in Table 1. The choice of points mentioned above results from the smallest (E) and the largest (H) value of time constant in the system, respectively.

Table 1
Values of the average time constant in E and H characteristic points at enlarging integration interval

The upper limit of integration interval		Average time constant	
[h]	[s]	point E [s]	point H [s]
20	72 000	3436	13 215
22.5	81 000	3445	13 269
25	90 000	3450	13 296
27.5	99 000	3452	13 309
30	108 000	3453	13 315

Analyzing Table 1, we can notice that, the longer the integration interval, the bigger the values of the average time constant approaching a certain limit. The condition relating the limitation of the integration interval, which is mentioned above, is fulfilled when the upper limit is changed from 20 to 30 hours. In this case the average time constant changes in point E by 0.48%, and in point H – by 0.75%. Taking this into consideration, an average time constant was determined for the upper limit of integration interval equaling 30 hours, which is 108 000 s.

The obtained values of average time constant in the cross-section of a heater are presented in Fig. 5. Moreover, Table 2 collects the constants in characteristic points. The largest average time constant occurs in point H, which is situated furthest from the heat source. It can be observed that the closer to a heating duct, the smaller the average time constants. The

smallest time constant occurs in point E, which is situated in the middle of the top edge of a duct with hot air. Described directions of change of averaged time constant correspond with the heating curves presented in Fig. 2.

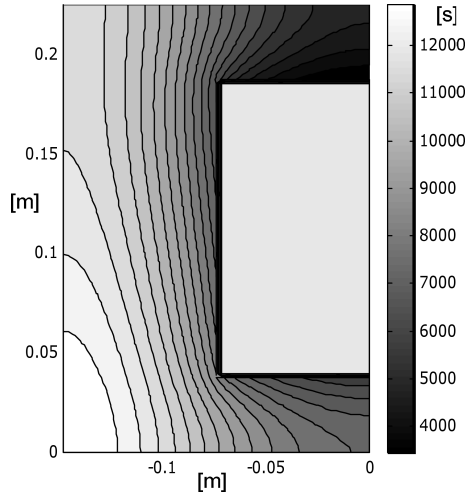

Fig. 5. Average time constant in cross-section of a floor heater

Table 2
Values of the average time constant in characteristic points of the heater model

Point	Time constant [s]	Point	Time constant [s]
A	11782	F	8663
B	7905	G	5919
C	4966	H	13315
D	6657	I	10125
E	3453	J	7645

The knowledge of the time constant is practically significant. It is then possible to estimate the duration of the transient state in $(x,\ y)$ position as $4\tau(x,\ y)$. It is also easy to approximate field dynamics in any point of the system, using responses of the first-order lag

$$h(x,y,t) = h_S(x,y)\left\{1 - e^{\frac{-t}{\tau(x,y)}}\right\} + h(x,y,t=0)\, e^{\frac{-t}{\tau(x,y)}}. \tag{4}$$

The values $h_S(x,\ y)$ could be read from Fig. 3b, while values $\tau(x,\ y)$ from Fig. 5. Graphs of functions (4) present approximated heating curves.

4. Efficiency of computations conducted on a graphics processing unit

The calculations of step response of a floor heater were conducted with the use of a PC and a graphics processing unit, whose parameters were given at the beginning of this article. To examine how useful the graphics processing unit can be in the analysis of problems based on solving a linear algebraic system of equations with the use of the conjugate gradient method [11], two programs, which implement algorithms adapted to dense and sparse matrices, were created. The programs were described in detail in the article [1].

Evaluation of the efficiency of calculations done on the graphics processing unit consists in comparing the execution time of a program implemented only on a standard processor (CPU) with the execution time of a program using also a graphics processing unit (GPU). On the basis of the results, the speed-up S is determined. It is assigned according to the following equation [12]:

$$S = \frac{t_{CPU}}{t_{CPU+GPU}}, \tag{5}$$

where t_{CPU} is the time of calculations done without the graphics processing unit, and $t_{CPU+GPU}$ – the time of calculations with additional use of the graphics processing unit. The speed-up defines, then, how many times quicker the program can be executed with the use of a supporting graphics processing unit.

In the first examined program, adapted to dense matrices, only a matrix-vector multiplication was implemented on the graphics processing unit. In this case cublasSgemv(...) function from CUBLAS library was used. The obtained execution times and calculated speed-ups are presented in Table 3.

Table 3
Execution times and speed-ups of programs for dense matrices (CPU – author's own program, CPU+GPU – CUBLAS library)

Mesh	Execution time		Speed-up
	CPU [s]	CPU+GPU [s]	
65 × 97	3223.912	97.344	33.1

The use of a graphics processing unit enabled us to speed up calculations by 33.1 times. However, it should be stressed that in the program using only CPU a simple method of matrix-vector multiplication (without any optimization) was applied. Taking this in consideration, a modification was introduced into the program. The matrix-vector multiplication was carried out with the use of the cblas_sgemv(...) function from the Intel Math Kernel Library (MKL) [13]. The library contains implementations of all BLAS (Basic Linear Algebra Subprograms) functions [14] optimized for Intel processors. The execution times obtained with the use of this library (CPU/MKL) and with additional use of a graphics processing unit (CPU+GPU) are presented in Table 4.

Table 4
Execution times and speed-ups of programs for dense matrices (CPU/MKL – author's own program, CPU+GPU – CUBLAS library)

Mesh	Execution time		Speed-up
	CPU/MKL [s]	CPU+GPU [s]	
65 × 97	1289.437	97.344	13.2

The use of the optimized MKL library shortened the time of the program execution on a CPU, which also caused the decrease of speed-up to the value of 13.2.

In the other examined program, algorithms adapted to sparse matrices were implemented. Non-zero elements of matrices were stored in computer memory with the use of the CSR (Compressed Sparse Row) method [15]. The operation

of matrix-vector multiplication was implemented on a graphics processing unit with use of the cusparseScsrmv(...) function from CUSPARSE library. The obtained execution times of the program and the calculated speed-ups are presented in Table 5. Unfortunately, in this case the execution time was not shortened (speed-up under 1). Enlarging density of the finite difference mesh caused only slight improvement.

Table 5
Execution times and speed-ups of programs for sparse matrices (CPU – author's own program, CPU+GPU – CUSPARSE library)

Mesh	Execution time		Speed-up
	CPU [s]	CPU+GPU [s]	
65 × 97	12.001	23.587	0.51
129 × 193	72.223	76.731	0.94
257 × 385	595.765	460.268	1.29

In the program working only on a CPU a modification was introduced. The matrix-vector multiplication was carried out with the use of mkl_cspblas_scsrgemv(...) function from MKL library. The obtained execution times for 3 densities of the finite difference mesh are presented in Table 6.

Table 6
Execution times and speed-ups of programs for sparse matrices (CPU/MKL – author's own program, CPU+GPU – CUSPARSE library)

Mesh	Execution time		Speed-up
	CPU/MKL [s]	CPU+GPU [s]	
65 × 97	7.654	23.587	0.32
129 × 193	50.679	76.731	0.66
257 × 385	404.233	460.268	0.88

In this case the use of MKL library did not result in a significant shortening of execution time in relation with a simple algorithm. However, it should be stressed that in all cases the time of calculations was not shortened.

5. Conclusions

The article presents parallel computations of a step response of a floor heater with the use of a graphics processing unit. The obtained heating curves are correct, which is proved by their physical interpretation and calculating verification. The approximation (4) of the curves is possible due to the assigned distribution of time constant $\tau(x, y)$. The conducted analysis also showed that the use of a graphics processing unit in the algorithm of the conjugate gradient method is useful when a coefficient matrix of the solved system of equations is dense. In the case of sparse matrices the speed-up was either not obtained, or it was very small. This is due to a small number (21057) of non-zero elements in the examined medium-sized sparse matrix with dimensions of 4289 × 4289. The GPU processor is not sufficiently loaded, then. Too small number of threads prevents also the masking of latencies in accessing the graphics card memory. It results in unfavourable relation of relative times: the duration of computations (54%) and communication between the graphics card memory and PC memory (46%), where the time of program execution is the reference. Tables 5 and 6 show that the speed-up increases with the size of the system of equations (and thus the number of non-zero elements). The solution of the problem of "medium-sized sparse matrices" could be the replacement the data division with the time decomposition, which was previously used in clusters of PCs [16, 17]. In the case of very large sparse matrices, problems concerning the speed-up are considerably smaller [18]. Resigning from the standard libraries of universal functions [5] and introducing programs directly adapted to the analysis of the given problem [19-23] can additionally increase the efficiency of parallel computations which use a GPU.

In the case of dense matrices, the number of threads is greater. The GPU processor is much better loaded. Therefore, for the dense matrices a good relation of relative computations time (88%) and communication time (12%) was obtained. As a result the speed-up is satisfactory (Tables 3 and 4).

Acknowledgements. The paper was prepared at the Bialystok University of Technology within the framework of the S/WE/1/2013 project sponsored by the Ministry of Science and Higher Education.

REFERENCES

[1] J. Gołębiowski and J. Forenc, "Parallel computations of the step response of a floor heater with the use of a graphics processing unit. Part 1: models and algorithms", *Bull. Pol. Ac.: Tech.* 61 (4), 943–948 (2013).
[2] Compute Unified Device Architecture (CUDA), http://www.nvidia.com/object/cuda_home_new.html (2012).
[3] D.B. Kirk and W.W. Hwu, *Programming Massively Parallel Processors: a Hands-on Approach*, Morgan Kaufmann, Burlington, 2010.
[4] CUDA CUBLAS Library, NVIDIA Corporation, Santa Clara, CA, 2010.
[5] CUDA CUSPARSE Library, NVIDIA Corporation, Santa Clara, CA, 2010.
[6] *Manuals for NISA v. 16. NISA Suite of FEA Software (CD-ROM)*, Cranes Software, Inc., Troy, Michigan, 2008.
[7] J. Gołębiowski and S. Kwiećkowski, "Dynamics of three-dimensional temperature field in electrical system of floor heating", *Int. J. Heat Mass Tran.* 45 (12), 2611–2622 (2002).
[8] W. Lipiński and J. Gołębiowski, "Modelling of electromagnetic shield dynamics", *IEEE Trans. on Magnetics* 16 (6), 1419–1422 (1980).
[9] J.D. Hoffman, *Numerical Methods for Engineers and Scientists*, Marcel Dekker Inc., New York, 2001.
[10] S. Rosłoniec, *Fundamental Numerical Methods for Electrical Engineering*, Springer-Verlag, Berlin, 2008.
[11] J.R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain", in *Technical Report*, Carnegie Mellon University, Pittsburgh, 1994.
[12] V. Jalili-Marandi and V. Dinavahi, "SIMD-based large-scale transient stability simulation on the graphics processing unit", *IEEE Trans. on Power Systems* 25 (3), 1589–1599 (2010).
[13] Intel Math Kernel Library. Reference Manual, MKL 10.3 Update 10, Intel Corporation, 2012.
[14] J. Dongarra, "Basic linear algebra subprograms technical forum standard", *Int. J. High Performance Computing Applications* 16 (1), 1–111 (2002).

[15] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, Ch. Romine, and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.

[16] J. Forenc, "Determination of the initial conditions in the parallel method for the state equation solving", *Electronics and Telecommunications Q.* 54 (3), 277–288 (2008).

[17] J. Gołębiowski and R.P. Bycul, "Thermal analysis of short-circuit and cooling states in a DC cable with the use of parallel computations", *Electrical Review* 85 (10), 95–100 (2009).

[18] M. Naumov, "Incomplete-LU and Cholesky preconditioned iterative methods using CUSPARSE and CUBLAS". *White Paper*, NVIDIA Corporation, London, 2011.

[19] W.A. Wiggers, V. Bakker, A.B.J. Kokkeler, and G.J.M. Smit, "Implementing the conjugate gradient algorithm on multi-core systems", *Proc. Int. Symp. on System-on-Chip* 1, 11–14 (2007).

[20] L. Buatois, G. Caumon, and B. Levy, "Concurrent number cruncher: a GPU implementation of a general sparse linear solver", *Int. J. Parallel Emerg. Distrib. Syst.* 24 (3), 205–223 (2009).

[21] A. Cevahir, A. Nukada, and S. Matsuoka, "Fast conjugate gradients with multiple GPUs", *Lecture Notes in Computer Science* 5544, 893–903 (2009).

[22] M. Ament, G. Knittel, D. Weiskopf, and W. Straßer, "A parallel preconditioned conjugate gradient solver for the poisson problem on a Multi-GPU platform", *Proc. Euromicro Int. Conf. on Parallel, Distributed and Network-Based Computing* 1, 583–592 (2010).

[23] R. Helfenstein and J. Koko, "Parallel preconditioned conjugate gradient algorithm on GPU, *J. Comp. Appl. Math.* 236, 3584–3590 (2012).