UDK 519.61/.64:536.2
Original scientific article/Izvirni znanstveni članek

ISSN 1580-2949
MTAEC9, 50(2)183(2016)

IMPLICIT NUMERICAL MULTIDIMENSIONAL HEAT-CONDUCTION ALGORITHM PARALLELIZATION AND ACCELERATION ON A GRAPHICS CARD

PARALELIZACIJA IN POSPEŠITEV IMPLICITNEGA NUMERIČNEGA VEČDIMENZIJSKEGA ALGORITMA PREVAJANJA TOPLOTE NA GRAFIČNI KARTICI

Michal Pohanka, Jana Ondroušková

Heat Transfer and Fluid Flow Laboratory, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, 616 69 Brno, Czech Republic
pohanka@fme.vutbr.cz

Prejem rokopisa – received: 2014-07-29; sprejem za objavo – accepted for publication: 2015-03-30

doi:10.17222/mit.2014.128

Analytical solutions are much less computationally intensive than numerical ones, and moreover, they are more accurate because they do not contain numerical errors; however, they can only describe a small group of simple heat-conduction problems. A numerical simulation of heat conduction is often used as it is able to describe complex problems, but its computational time is much longer, especially for unsteady multidimensional models with temperature-dependent material properties. After a discretization using the implicit scheme, the heat-conduction problem can be described with N non-linear equations, where N is the large number of the elements of the discretized model. This set of equations can be efficiently solved with an iteration of the line-by-line method, based on the heat-flux superposition, although the computational procedure is strictly serial. This means that no parallel computation can be done, which is strictly required when a graphics card is used to accelerate the computation. This paper describes a multidimensional numerical model of unsteady heat conduction solved with the line-by-line method and a modification of this method for a highly parallel computation. An enormous increase in the speed is demonstrated for the modified line-by-line method accelerated on the graphics card, and the durations of the computations for various mesh sizes are compared with the original line-by-line method.

Keywords: heat conduction, numerical simulation, multidimensional numerical model algorithm, acceleration, parallelization, graphics card

Analitične rešitve so mnogo manj računsko intenzivne kot numerične, poleg tega pa so bolj natančne, ker ne vsebujejo numeričnih napak, vendar pa lahko opisujejo samo majhno skupino enostavnih problemov prevajanja toplote. Numerična simulacija prevajanja toplote se pogosto uporablja, ker je sposobna opisati kompleksne probleme, vendar pa je čas izračuna mnogo daljši, še posebno pri nestabilnih večdimenzijskih modelih, z lastnostmi materiala, odvisnimi od temperature. Po diskretizaciji z uporabo implicitne sheme je mogoče problem prevajanja toplote opisati z N nelinearnimi enačbami, kjer je N veliko število elementov diskretiziranega modela. Ta sklop enačb je mogoče učinkovito rešiti s približkom metode vrsta za vrsto, ki temelji na predpostavki toka toplote, čeprav izračun poteka serijsko. To pomeni, da ni mogoč vzporedni izračun, kar je striktna zahteva, kadar se uporablja grafično kartico za pohitritev izračuna. Ta članek opisuje večdimenzijski numerični model nestabilnega prevajanja toplote, kar je bilo rešeno z metodo vrsta za vrsto in s pospešitvijo modifikacije te metode na grafični kartici. Trajanje izračuna je primerjano z osnovno metodo vrsta za vrsto pri različnih dimenzijah mreže.

Ključne besede: prevajanje toplote, numerična simulacija, večdimenzijski model algoritma, pospešitev, paralelizacija, grafična kartica

1 INTRODUCTION

Although the computational power of modern computers continues to rapidly increase year by year, numerical simulations can last several hours or days for detailed numerical models where high accuracy is required. The computational time becomes even longer when the models are used for inverse computations, where thousands of direct-heat-conduction computations can be required. Computations of the boundary conditions (such as the surface temperature, the heat flux, and the heat-transfer coefficient) for continuous casting[1–2], heat treatment[3], hot rolling[4–6] and different types of spray cooling[7] using ill-posed inverse methods[8,9] are typical applications.

Many very fast algorithms have been developed, but they are often used strictly in a serial fashion[10,11] and cannot be efficiently used by modern multicore processors, as they require parallel processing. It has been shown that a seemingly lesser algorithm can be much more efficient because it allows parallel processing[12–15] that is absolutely necessary for the use of the maximum computational power of modern processors, where thousands of tasks can be computed at the same time. The time when a personal computer's central processing unit (CPU) was able to compute only one thread at a time is over. Today, CPUs often compute from 4 to 16 threads at the same time.[16,17] New graphics processing units (GPUs)[18,19] allow scientific computing in double precision, which is necessary for numerical heat-con-

duction computations and the GPUs can run more than 2800 threads at the same time. When looking at the computational power, a GPU has over 2500 Giga FLoating-point OPerations per Second (GFLOPS) in double precision, while a CPU has only 70 GFLOPS. However, to use all of the computational power of a GPU a highly parallel computational algorithm is required.

After the discretization of a heat-conduction problem (HCP) using the implicit scheme, the HCP can be described with a set of linear equations $A \cdot T = B$ for constant material properties where T is the unknown temperature in the model and A is a very large sparse square matrix for large models. The A matrix has dimensions of $N \cdot N$ where N is the number of the nodes in the entire model. For example, a 3D model with 100 nodes in each dimension has an A matrix of 10^6 times 10^6 consisting of 10^{12} values. It is necessary to compute the huge inverse matrix A^{-1} and store this inverse matrix in the computer memory to compute the T vector. Furthermore, for a temperature-dependent model, the computation must be iterated because matrix A and vector B change with a better estimate of the final T vector. However, there is a different approach, often called the line-by-line method, which requires much less computer memory, although the computation procedure is strictly serial[10] and does not allow for parallel processing.

2 LINE-BY-LINE METHOD FOR HCP

A two-dimensional (2D) problem can be discretized into an orthogonal mesh.[20] One equation is necessary for each control volume represented by one temperature. An explicit, implicit, or Crank-Nicholson differencing scheme for the time domain can be used. The focus here is on the implicit scheme because the explicit one is conditionally stable. When writing the heat-flux equation for each control volume, all the equations can be rearranged into the matrix form of $A \cdot T = B$ where T is the vector of nodal temperatures. The inverse matrix A^{-1} must be computed to solve this set of equations.[21] This approach is very inefficient and unstable for the models with many control volumes. The stability problem arises from the rounding of the values during a large number of operations because the computers use only a limited number of decimal places to store these values.

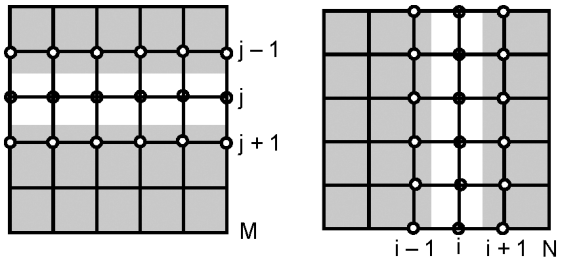

Figure 1: Line-by-line method for a 2D problem
Slika 1: Metoda vrsta za vrsto za 2D problem

Patankar[10] described an approach called the line-by-line method to overcome the problem with an inverse matrix. This method uses the principle of the heat-flux superposition. The domain is solved by iterating two steps for a 2D model and by iterating three steps for a 3D model. The 2D model is partitioned into lines in the first step and into columns in the second step (**Figure 1**). The first line is solved separately, then the second one and so on until all the lines are finished. Each line j (where j = 1..M) is solved using an implicit scheme and the most up-to-date temperatures for lines j–1 and j+1 are used to include vertical heat fluxes for line j to ensure the fastest convergence to the final solution. This means that the temperatures for line j–1 are those only computed for this line and the temperatures for line j+1 are those computed during the previous iteration. A similar process is performed with columns in the second step. This approach leads to M matrices $C_jT_j = D_j$ and N matrices $E_iT_i = F_i$ where C_j and E_i are tridiagonal matrices. This set of equations can be solved very fast using the direct TDMA method.[11] These two steps (computations of rows and columns) are repeated until the desired accuracy of the computed temperature field is attained.

The problem with the original line-by-line method is that line j can only be computed when line j-1 has already been finished. Neither does the TDMA method[11] allow parallel processing, which is strictly required by a GPU. However, parallel processing can be enabled with a simple modification. The temperatures from the previous iterations for both line j–1 and j+1 can be used

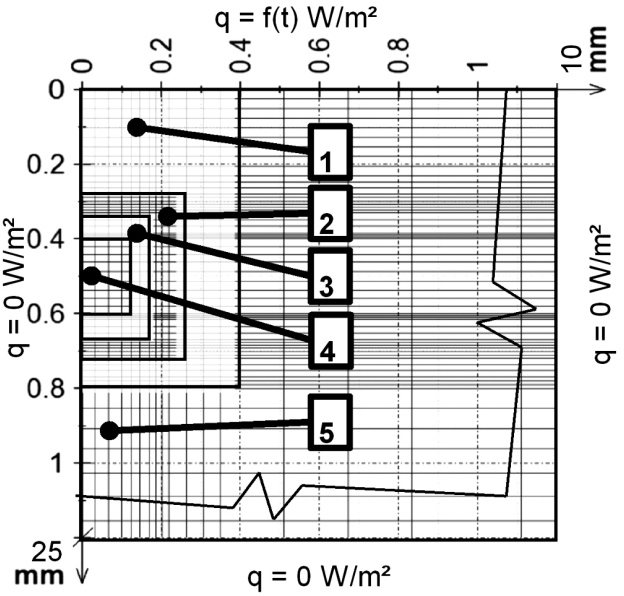

Figure 2: Structure of the 2D model used for testing and non-uniform discretization: 1) silver solder Ag40, 2) thermocouple shield Inconel 600, 3) eletrical insulation MgO, 4) K-thermocouple, 5) stainless steel 1.4301

Slika 2: Zgradba 2D modela, uporabljenega za preizkus in spremenljiva diskretizacija: 1) srebrova spajka Ag40, 2) termoelement Inconel 600, 3) električna izolacija MgO, 4) K-termoelement, 5) nerjavno jeklo 1.4301

when line *j* is being computed. However, this modification significantly slows down the speed of convergence.

3 EXAMPLE OF PARALLEL PROCESSING

GPU acceleration was tested on a 2D computational model (**Figure 2**) which is often used for the inverse computation of boundary conditions (the cooling intensity of a spraying header on a work roll during hot rolling).[6] This represents a cut of a shielded K-thermocouple soldered with a silver solder in a slot made on the surface of stainless steel. The actual circular geometry of the shielded thermocouple is simplified in this example to the equivalent square geometry so that the model can be easily refined by dividing the control volumes. Because the geometry is symmetrical, only one half is used for the model. All sides are insulated except for the upper one, where a time-dependent heat flux is applied (**Figure 3**). The applied heat flux represents the heat flux from the real cooling measurement during one rotation of a heated roll. The roll is hollow with an outer perimeter of 2 m and a wall thickness of 25 mm. The velocity of the outer surface of the roll during the rotation is 2 m/s. One rotation lasts for 1 s. The sampling frequency is 300 Hz. The starting temperature is 400 °C. The temperatures computed on the surface and in the center of the thermocouple are shown in **Figure 3**.

When the computation is accelerated, the GPU is used as a co-processor. Only the most computationally intensive sections that can be processed in parallel are run on the GPU, rather than the whole program. The main program is run on the CPU and the most computationally intensive tasks are sent to the GPU using Open Computing Language (OpenCL).[22] In this case, three procedures are accelerated on the GPU: 1) the computing-material properties for the actual temperature; 2) the TDMA method for solving a set of equations; 3) the estimation of the attained accuracy of the computed temperature field. The first and the third functions are well suited for parallelization. Each node can be computed in a separate thread. This means that the total number of nodes determines the maximum number of parallel threads. Function 2 is not as well suited for parallelization. The maximum number of parallel threads is the number of rows *M* in the first stage and the number of columns *N* in the second stage.

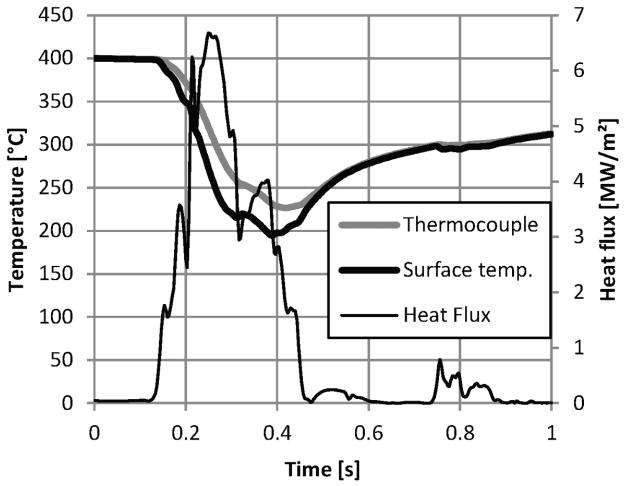

Figure 3: Computed temperature history for the thermocouple, for the surface above the thermocouple (corner (0;0)) and the used heat flux
Slika 3: Izračunana zgodovina temperature v termoelementu, na površini nad termoelementom (vogal (0;0)) in uporabljeni tok toplote

The durations of the computations for functions 1 and 2 are listed in **Table 1** for different numbers of nodes. These are the net values without the time required for preparing the tasks, data, and for launching. **Table 2** includes the results from the entire simulation of one rotation during the roll cooling. The values for the CPU were obtained using the original line-by-line method, computed in one thread, while the values for the GPU were obtained using the modified line-by-line method, accelerated on the GPU. The desired accuracy was 1 °C for the whole simulation in all the nodes. An Intel Core i5-2500K[23] was used for the CPU and an AMD Radeon HD 7970[24] was used for the GPU.

Table 1: Computations on CPU and GPU for one iteration
Tabela 1: Izračunana CPU in GPU za eno ponovitev

Nodes		Mat. properties (μs)		TDMA *M*+*N*(ms)	
X	Y	CPU	GPU	CPU	GPU
32	107	51	19	5	1.1
92	107	147	20	17	1.5
182	212	605	24	72	3.1
362	422	2532	57	293	6.4
722	842	10220	191	1152	12.3
1442	1682	41479	726	4611	16.0

4 DISCUSSION

The test example showed that the GPU acceleration makes sense for the models with many nodes (the more nodes the better). For a very fine mesh (800 rows and 800 columns), results can be obtained almost 11 times faster when using the GPU acceleration. For a very small number of nodes, the original method used only by the CPU can be even faster. This is caused by a relatively slow communication between the CPU and GPU, which is necessary for the acceleration on the GPU. An explicit scheme was also tested because it is more suitable for parallel processing. However, it was found that it is 164 times slower on the CPU than an implicit scheme due to its conditional stability, which requires a 6000× higher sampling frequency.

The numbers of iterations for steps 1 and 2 of the line-by-line method for 2D are listed in **Table 2**. It is clear that the original method (the CPU column) needs approximately half the number of iterations required by the modified method (the GPU column) to reach the same 1 °C accuracy. However, the model introduced here has a very slow convergence rate because the silver

Table 2: Computing times on CPU and with GPU acceleration for the increasing number of nodes. *M* is the number of rows and *N* is the number of columns.
Tabela 2: Časi izračunov na CPU in z GPU pospešitvijo pri naraščajočem številu vozlišč. *M* je število vrst in *N* je število stolpcev.

Nodes		Iterations		Time (s)		Speed-up	Time/ (Iter.*(*M*+*N*)) (μs)		Time/ (Iter.**M***N*) (μs)	
Rows	Col.	CPU	GPU	CPU	GPU		CPU	GPU	CPU	GPU
107	32	33596	101445	22	53	0.4	4.71	3.76	0.19	0.15
107	92	128624	241864	253	175	1.4	9.88	3.64	0.20	0.07
212	182	492610	970922	3873	1451	2.7	19.95	3.79	0.20	0.04
422	362	1928762	3804033	61256	11826	5.2	40.51	3.97	0.21	0.02
842	722	7664187	14338003	988219	90617	10.9	82.44	4.04	0.21	0.01

solder used has an approximately five-time higher thermal conductivity than stainless steel. In the models where materials with similar thermal conductivities are used, the number of the necessary iterations is much smaller.

The computational time required for one node in one iteration for the CPU is almost independent of the number of nodes and lasts for about 0.2 μs. The situation is completely different for the acceleration using the GPU (it decreases). However, the total time divided by the product of the number of iterations and the sum of lines and columns is almost constant (3.8 μs). A similar situation exists for the TDMA *M*+*N* function in **Table 1** because this function is the longest during the computation. The time/(iter.*(*M*+*N*)) is almost constant because there are not enough parallel threads to fully utilize the GPU. There are only about 800 threads and the GPU can process over 2000 threads. The increase in TDMA *M*+*N* for a higher number of nodes is caused by a larger tridiagonal matrix to be solved but not by a higher number of tridiagonal matrices to be solved.

5 CONCLUSION

It was found that for the models with a high number of computational elements the acceleration on a GPU card can speed up 2D heat-conduction calculations by almost eleven times. For a model with a small number of nodes, it is better to use a CPU and the original line-by-line method. The AMD 7970 starts to be fully utilized when there are more than 32,000 threads. The GPU acceleration will be even more useful for 3D models where there may be more parallel threads for the TDMA function.

Acknowledgement

This work is an output of the research and scientific activities of the NETME Centre, regional R&D centre built with the financial support from Operational Programme Research and Development for Innovations within the NETME Centre project (New Technologies for Mechanical Engineering), Reg. No. CZ.1.05/2.1.00/01.0002 and, in the follow-up sustainability stage, supported through NETME CENTRE PLUS (LO1202) with the financial means from the Ministry of Education, Youth and Sports under the National Sustainability Programme.

6 REFERENCES

[1] M. Pohanka, K. A. Woodbury, A downhill simplex method for computation of interfacial heat transfer coefficients in alloy casting, Inverse Problems in Engineering, 11 (**2003**) 5, 409–424, doi:10.1080/1068276031000109899

[2] J. Horský, M. Raudenský, M. Pohanka, Experimental study of heat transfer in hot rolling and continuous casting, Materials Science Forum, 473–474 (**2005**), 347–354, doi:10.4028/www.scientific.net/MSF.473-474.347

[3] M. Raudenský, J. Horský, P. Kotrbáček, M. Hnízdil, M. Pohanka, In-Line Heat Treatment and Hot Rolling, Proc. of the International Conference on Advances in Materials and Processing Technologies (AMPT2010), 1315 (**2011**), 563–568, doi:10.1063/1.3552506

[4] M. Pohanka, M. Raudenský, J. Horský, Attainment of more precise parameters of a mathematical model for cooling flat and cylindrical hot surfaces by nozzles, In: C. A. Brebbia, B. Suncten (Ed.), Advanced Computational Methods in Heat Transfer VI, WIT Press, 2000, 627–635

[5] M. Raudenský, M. Pohanka, J. Horský, Combined inverse heat conduction method for highly transient processes, WIT Transactions on Engineering Sciences, Advanced Computational Methods in Heat Transfer VII, 35 (**2002**), 35–42, doi:10.2495/HT020041

[6] J. Ondroušková, M. Pohanka, B. Vervaet, Heat-Flux Computation from Measured-Temperature Histories during Hot Rolling, Mater. Tehnol., 47 (**2013**) 1, 85–87

[7] A. A. Tseng, H. Bellerová, M. Pohanka, M. Raudenský, Effects of titania nanoparticles on heat transfer performance of spray cooling with full cone nozzle, Applied Thermal Engineering, 62 (**2013**) 1, 20–27, doi:10.1016/j.applthermaleng.2013.07.023

[8] J. V. Beck, B. Blackwell, C. R. St. Clair, Inverse Heat Conduction: Ill-posed Problems, Wiley, New York 1985, 308

[9] M. Pohanka, P. Kotrbaček, Design of Cooling Units for Heat Treatment, Heat Treatment Conventional and Novel Applications, InTech, 2012, 1–20, doi:10.5772/50492

[10] S. V. Patankar, Numerical Heat Transfer and Fluid Flow, Hemisphere Publishing Corporation, New York 1980, 214

[11] H. P. William, A. T. Saul, T. V. William, P. F. Brian, Numerical Recipes in C, 2nd ed., Cambridge University Press, New York 1992, 994

[12] J. Lee, J. C. Wright, A block-tridiagonal solver with two-level parallelization for finite element-spectral codes, Computer Physics Communication, 185 (**2014**) 10, 2598–2608, doi:10.1016/j.cpc.2014.06.006

[13] E. S. Quintana, G. Quintana, X. Sun, R. Van de Geijn, A note on parallel matrix inversion, SiAM J. Sci. Comput., 22 (**2001**) 5, 1762–1771, doi:10.1137/ S1064827598345679

[14] L. Karlsson, B. Kgström, Parallel two-stage reduction to Hessenberg form using dynamic scheduling on shared-memory architectures, Parallel Computing, 37 (**2011**) 12, 771–782, doi:10.1016/j.parco. 2011.05.001

[15] R. Zheng, Q. Zhang, H. Jin, Z. Shao, X. Feng, A GPU-based parallel method for evolutionary tree construction, Computers & Electrical Engineering, 40 (**2014**) 5, 1580–1591, doi:10.1016/j.compeleceng. 2014.04.013

[16] AMD Opteron™ 6300 Series Processors, http://www.amd.com/ en-us/products/server/opteron/6000/ 6300#

[17] Intel® Xeon® Processor E7 v2 Family, http://ark.intel.com/products/ family/78584/Intel-Xeon-Processor-E7-v2-Family#@Server

[18] AMD FirePro™ S9150 Server GPU, http://www.amd.com/ en-us/products/graphics/server/s9150#

[19] Tesla Server GPUs, http://www.nvidia.co.uk/object/tesla-server-gpus-uk.html

[20] M. Pohanka, Ph.D. Thesis: Technical Experiment Based Inverse Tasks in Mechanics, Brno 2006, 202, http://lab4.fme.vutbr.cz/ Pohanka/PDF/PhDThesisPohankaM. pdf

[21] F. P. Incropera, D. P. DeWitt, Fundamentals of Heat and Mass Transfer, 4th ed., Wiley, New York 1996, 912

[22] OpenCL – The open standard for parallel programming of heterogeneous systems, https://www.khronos.org/opencl/

[23] Intel® Core™ i5-2500K Processor, http://ark.intel.com/products/ 52210/Intel-Core-i5-2500K-Processor-6M-Cache-up-to-3_70-GHz

[24] AMD Radeon™ HD 7900 Series Graphics Cards, http://www.amd.com/en-us/products/graphics/ desktop/7000/7900#