Acceleration of calculations using block algorithms for the difference solution of the heat equation

D L Golovashkin¹,², L V Yablokova² and I D Reznik²

¹Image Processing Systems Institute of RAS - Branch of the FSRC "Crystallography and Photonics" RAS, Molodogvardejskaya street 151, Samara, Russia, 443001
²Samara National Research University, Moskovskoe Shosse 34, Samara, Russia, 443086

e-mail: golovashkin2010@yandex.ru, lyablokova@gmail.com

Abstract. In this paper, we propose taking into account the architectural features of the processor at the stage of constructing the numerical method itself. This idea is illustrated by the example of the synthesis of a new difference scheme for the heat conduction equation, which has traditionally been the object of testing innovations in the theory of difference schemes. The architectural feature hierarchical structure of the computer memory chosen led to considerable communication costs even when a single hardware computational flow was used for organising the calculations. This feature is accounted for in computational linear algebra by using block algorithms, and in the theory of difference schemes, by using the technique of programming ‘tiling’. However, for the two-layer difference schemes of block algorithms for solving grid equations, prior to the proposed work, it was not known because of the impossibility of organising block calculations by using the existing schemes. Here, we propose a new method of constructing two-layer difference schemes and a mixed scheme with a shift as an example of the application of this method. In the course of the experiments, a five-fold acceleration of calculations according to this scheme was demonstrated relative to the traditional explicit model, with the same computational complexity.

1. Introduction

Mathematical modelling, as an independent subject area, gained recognition in the middle of the last century because of the need to implement major scientific and technical projects that gave impetus to the development of mathematical physics, computational mathematics and computer technology. The classical understanding of mathematical modelling [1] is connected with the triad of scientific directions of academician A. A. Samarskii with the model, numerical method and software package. For a long time, the second and the third components of the triad were considered minor; they were started after the completion of the mathematical model. In turn, the creation of a numerical method, as a rule, preceded the choice of the computing system on which it was later implemented. The situation changed when the director of the Computing Centre of the Siberian Branch of the USSR Academy of Sciences, G. I. Marchuk, set the task of mapping the numerical method onto the architecture of the computing system [2]; the corresponding solution implied taking into account architectural features while synthesising the method.

The proposed work is an application of this problem to the theory of difference schemes. Among the mentioned features, the hierarchical structure of computer memory is chosen as the least studied
from the viewpoint of the theory of difference schemes [3] on the one hand, and the most important for this theory, according to the authors of a number of publications [4–8], on the other hand. In the adjacent subject domain, computational linear algebra, the method of compiling block algorithms has long been considered classical [9, 10] and has proven to be a reliable tool for reducing the computational time by minimising the communication between the RAM and the CPU cache or graphics memory. From the viewpoint of software developers, this technique is referred to as “tiling” [11] and is used to optimise the code irrespective of its purpose [12, 13].

The most general idea of block algorithms [10] is consistent with the requirement of intensifying the use of a local fragment of the common data loaded on the upper hierarchical level of computer memory before it is unloaded. The larger the number of arithmetic operations performed on this fragment, the less often one will have to load it in the future. Consequently, the duration of communications between different levels of memory often decreases, often determining the total duration of the computations by the algorithm. Programmatically, such a strategy is implemented by doubling the cyclic structures (“tiling”), which allow the redistribution of the total number of iterations between different levels of nesting with a constant number of arithmetic operations in the algorithm.

With respect to the theory of difference schemes, the discussed method of reducing the duration of calculations have been applied recently [6] and has not had time to be used widely. In contrast to the trivial approach, in which the calculations at the next time layer of the grid area begin after their completion at the previous one, the block algorithms are characterised by the localisation of the computational process within a given sub-region of the grid area (block) intersecting several (from units to hundreds) layers in time. In the block, calculations are performed in layers, and the blocks are sorted in the prescribed manner. The parallelepiped (“Diamond Torre” [4, 5]) is recognised as a popular block shape.

The known block algorithms [4–7] are intended for organising calculations by using explicit three-layer difference schemes. Calculations using implicit schemes are accompanied by solving systems of linear equations (multiple complication of the problem), and explicit two-layer ones are not suitable for block algorithms. Traditionally, with their software implementation, the values of the grid function on the new layer are written over the values on the previous layer in order to save memory. The introduction of an additional array to store the old values increases the communication within the hierarchical structure of the computer memory, which contradicts the very idea of block algorithms. Therefore, we developed a new technique for constructing difference schemes that lack the indicated drawback.

2. Mixed difference scheme with a shift
Choosing an example illustrating the proposed technique, we settled on a one-dimensional homogeneous linear non-stationary heat conduction equation. This equation is a traditional object for the demonstration of methods for compiling grid equations in the theory of difference schemes [3]. In addition to the mentioned methodological value, the unconditional scientific value was characterised by a generalisation of reception for the case of large dimensions [14]. However, it was first necessary to attest to its effectiveness for the chosen example, as the simplest. In addition, the conduction problem is of relevant [15], including in the one-dimensional case [16].

For the equation

\[ \frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} \]  

(1)

the known explicit

\[ \frac{U^n_i - U^{n-1}_i}{\tau} = \frac{U^{n-1}_{i-1} - 2U^{n-1}_i + U^{n-1}_{i+1}}{h^2} \]  

(2)

and the implicit

\[ \frac{U^n_i - U^{n-1}_i}{\tau} = \frac{U^n_{i-1} - 2U^n_i + U^n_{i+1}}{h^2} \]  

(3)

difference scheme, the function \( U \) was defined on the computing area \( \omega = \{(t, x): 0 \leq t \leq T; 0 \leq x \leq L\} \), and its net counterpart \( U^n_i \) defined on the grid area \( \omega_h = \{(t_n, x_i): t_n = \tau \cdot n; n = 0, .., N; \tau = T/N; x_i = h \cdot i; \)
\[ i = 0, \ldots, I + 1; \quad \tau = L/(I + 1) \]. The boundary conditions of the first type, the initial condition and their discretisation complemented the differential problem and the corresponding difference schemes.

As noted earlier, the well-known methods of compiling block algorithms [4–7] were not suitable for working with (2) and (3) for the reasons mentioned above. However, the combination of (2) and (3) in one difference scheme changed the situation.

Suppose for definiteness, \( I \) was odd. Then, we assumed that at the time layer \( n = 1 \) in its even nodes \((i = 2, 4, 6, \ldots, I - 1)\), the values of the grid function were calculated using an explicit differential pattern and formula (2). The values on the previous layer with \( n = 0 \) were known (discretisation of the initial condition):

\[
U_i^n = U_i^{n-1} + \frac{\tau}{h^2}\left( U_{i-1}^{n-1} - 2U_i^{n-1} + U_{i+1}^{n-1} \right) \quad (4)
\]

Thus, the values found \( U_i^n \) were placed in a one-dimensional array on top of the values \( U_i^0 \) for even \( i \). Then, we performed calculations on the same layer to determine the values of the grid function in odd nodes \( i = 1, 3, 5, \ldots, I \) by using the implicit differential pattern and formula (3). Thus, the calculations for the implicit part of the new difference scheme were performed explicitly:

\[
U_i^n = \left( U_i^{n-1} + \frac{\tau}{h^2}\left( U_{i+1}^{n-1} + U_{i-1}^{n-1} \right) \right) \left( 1 + 2\frac{\tau}{h^2} \right)^{-1} \quad (5)
\]

The values found \( U_i^n \) were placed in the same array on the top \( U_i^0 \) for odd \( i \); as a result, this array stored the values of the grid functions only on layer 1.

The explicit expression of the sought grid function in the calculations by the implicit scheme is not new, the running account schemes are known, including for equation (1) [17]. However, unlike them, here, different differential patterns were applied on one time layer and when moving to the next node in space, the pattern did not shift by one position, but ‘jumped’ through the node. Therefore, we called the proposed scheme mixed.

The transition to the next temporal layer \( n = 2 \) is accompanied by a shift of the patterns. Now, in odd nodes, calculations are performed on (4), then on even ones, on (5). The use of different templates on different layers is also accepted in the already mentioned running account scheme from [14]; the novelty proposed here is due to the shift of the templates used on the previous layer instead of introducing a new one. Given this circumstance, the full name of the proposed scheme was decided as mixed with a shift.

As a result, on odd time layers, first, calculations were performed using an explicit pattern for even-spaced nodes (step 1), then using an implicit one for odd numbers (step 2). On even nodes, the opposite: by the explicit pattern for odd nodes (step 3) and by implicit pattern for even nodes (step 4). In any case, when moving to the next temporary layer, an explicit differential pattern was applied first. The case of even \( I \) differed from that considered only by the type of pattern used for the calculations in node \( I \).

3. Block algorithm

Let us explain the idea of the block algorithm with an example in Figure 1 and in Table 1.

**Figure 1.** Example of block division of the grid area. Nodes, the calculations in which were performed on an explicit pattern are marked with the ‘E’ symbol; on an implicit pattern, they are marked with the ‘I’ symbol.
Table 1. Changes in the contents of array A depending on the algorithm stage.

| Algorithm stage                  | Contents of array A                      |
|----------------------------------|------------------------------------------|
| Before calculations start        | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0       |
| After calculations in the first block | 4 3 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0    |
| After calculations in the second block | 4 4 4 4 3 2 1 0 0 0 0 0 0 0 0 0 0 0 |
| After calculations in the third block | 4 4 4 4 4 4 4 3 2 1 0 0 0 0 0 0 0 0 |
| After calculations in the fourth block | 4 4 4 4 4 4 4 4 4 4 3 2 1 0          |
| After calculations in the fifth block | 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 |

We placed in \( \omega \), \( I = 17 \) and \( N = 4 \) and divided the grid area into five blocks: the left (first) was in the form of a triangle, the right (fifth) was a triangle that complemented the first square, and the remaining three (second, third and fourth), the central parallelogram. The boundaries between the blocks were called internal. Then, on the left inner border of each block, there were knots associated with calculations solely on an implicit template, and on the right inner border of each block, there were knots connected with the calculations solely on an explicit template. This was due to the previously mentioned shift of the differential pattern when moving to the next temporary layer and provided, as discussed later, the ability to be satisfied with one one-dimensional array of length \( I + 2 \) when storing the values of the grid function (including the boundary ones) during the calculations using the block algorithm.

In contrast to the trivial approach, when in the course of a computational process, the nodes of the grid area were bypassed in layers, block algorithms involved searching the nodes inside a block that crossed several temporary layers, and then proceeded to another block. In the example, the corresponding computational process began with a layer-by-layer search of the nodes of the grid area inside the left triangle. Let us follow the description of the algorithm by demonstrating the contents of the one-dimensional array A (Table 1) of length \( I \) (not to be confused with the main array of the same length storing the values of the grid function), the \( i^{th} \) cell of which will contain the index value \( n \) from \( U^n \) the main array. Therefore, after the completion of the calculation of the values of grid functions in the nodes located inside the first block (Figure 1, Table 1), the first four elements of array A were different: for \( i = 1 \), the grid function was defined on the fourth layer, for \( i = 2 \) on the third layer, \( i = 3 \) on the second layer, \( i = 4 \) on the first layer, and for \( i > 4 \), calculations have not yet been performed.

In the second, third, and fourth blocks, the calculations were performed sequentially by using a mixed scheme with a shift, layer by layer from the top to the bottom (Figure 1) and shifted one space node to the left when moving to the next temporary layer. In all the previously calculated blocks, the values of the grid function were determined on the last time layer, and in the newly calculated blocks, all the time layers except the initial one were defined; in blocks with large numbers, they were defined on the initial layer (Table 1).

The actions of the algorithm were completed by calculations in the fifth block of a triangular form, after which the desired values of the grid function on the last layer were considered to be known. Here, in contrast to the first block, as the number of layers increased, the number of arithmetic operations increased.

An important characteristic of the block algorithm is the obviousness of the vectorisation of the calculations on it. Indeed, the actions in each time layer of any block can be divided into two vector operations of the saxpy type [8], associated with calculations using explicit and implicit differential patterns.

Generalising the algorithm to the case of arbitrary \( N \) and \( I \), we introduced two parameters: the width (\( m \)) and the height (\( d \)) of the block, defining the number of blocks in one block time layer as \( \lceil I/m \rceil + 1 \), and the number of such layers as \( \lceil N/d \rceil \), where \( \lceil \cdot \rceil \) and \( \lfloor \cdot \rfloor \) are rounding operations for the smallest and the largest integer, respectively.
4. Another section of your paper: Acceleration of calculations by block algorithm

The detailed study of the proposed difference scheme and the corresponding block algorithm is decreased in a separate publication; here, we limit ourselves to a demonstration of the superiority of the mathematical and algorithmic apparatus over the classical explicit scheme and the trivial algorithm.

The experiments were conducted on the following hardware and system base: processor: Intel Core i5-4460, operating system: Linux Ubuntu 16.04, and compiler: gfortran 5.4.

In the first series of experiments, the computational durations were compared using the classical explicit and the proposed mixed with shift difference schemes with a trivial approach to the organisation of the computational process. An equal time of calculations or a slight acceleration of the classical method compared with the proposed method was expected because of the larger amount of code in the implementation of the latter. However, in a wide range of variations of the grid region discretisation parameters, the proposed numerical method invariably demonstrated double acceleration (2.3 times) as compared with the classical one. The observation of the amount of allocated memory during the calculations led to the conclusion about the influence of this parameter on the result obtained. Despite the use of only one array in both the software implementations to store the values of the grid function, in calculations using the classical explicit scheme, such an array actually doubled (although this was not provided in the program) to prevent the premature mashing of the grid function values on the previous layer over time. In the proposed scheme, such doubling was not necessary. A valuable conclusion from this is the decisive influence of the duration of communications between different levels of computer memory as compared to the duration of the production of arithmetic operations (the number of which is the same in both the methods) on the total computational time. This implied that it was expedient to use block algorithms designed to reduce communication costs.

In the second series of experiments, the acceleration of the block algorithm was studied with a fixed discretisation of the grid area and variable blocking parameters (height and width of the block). Two U-shaped plots of the duration of the calculations on each of the listed parameters were obtained. Indeed, when the block size was significantly less or more than the optimal, the processor’s cache memory was not rationally used, which led to an increase in the duration of communications. The maximum acceleration of the computations as compared to the trivial algorithm for the proposed scheme was 2.2 times; as compared to the classical explicit scheme, the highest acceleration of calculations was 5 times.

5. Conclusions

In this paper, we discussed the prospects of developing two-layer explicit difference schemes with respect to the hierarchical structure of the computer memory by using the example of a mixed scheme with a shift for a homogeneous one-dimensional non-stationary linear heat conduction equation.

We consider the following to be promising areas for the development of this topic: an analytical study of the properties of the new scheme (approximation and stability), the transition to cases of large dimensions (two-dimensional and three-dimensional), the application of the proposed approach to the synthesis of difference schemes for other parabolic equations of mathematical physics and grid equations for the case of implicit difference schemes.

6. References

[1] Samarskii A A, Mikhailov A P 2002 Mathematical Modeling: Ideas. Methods. Examples (Moscow: “Fizmatlit” Publisher) p 320 (in Russian)
[2] Voevodin V V 1986 Mathematical models and methods in parallel processes (Moscow: “Nauka” Publisher) p 296 (in Russian)
[3] Samarskii A A 1977 Theory of difference schemes (Moscow: “Nauka” Publisher) p 656 (in Russian)
[4] Zakirov A V, Levchenko V D, Perepelkina A Yu and Zempo Y 2016 Diamond Torre algorithm and high-performance implementation of the FDTD method for supercomputers with graphics accelerators Proc “Supercomputer days in Russia” 80-94
[5]  Perepelkina A Yu, Levchenko V D 2015 Diamond Torre Algorithm for High-Performance Wave Modeling Keldysh Institute Preprints 18 20
[6]  Orozco D A, Gao G R 2009 Mapping the FDTD Application to Many-Core Chip Architectures Parallel Processing 309-316
[7]  Takeshi M 2014 Automatic Parameter Tuning of Three-Dimensional Tiled FDTD Kernel High Performance Computing for Computational Science – VECPAR 8969 284-297
[8]  Yablokova L V, Golovashkin D L 2018 Block algorithms of a simultaneous difference solution of d’Alembert's and Maxwell's equations Computer Optics 42(2) 320-327 DOI:10.18287/2412-6179-2018-42-2-320-327
[9]  Golub G H, Van Loan C F 2012 Matrix calculations (Johns Hopkins University Press) p 790
[10]  Demmel J W 1997 Applied Numerical Linear Algebra Society for Industrial and Applied Mathematics p 410
[11]  Gallivan K, Jalby W, Meier U and Sameh A H 1988 Impact of Hierarchical Memory Systemson Linear Algebra Algorithm Design International Journal of High Performance Computing Applications 2(1) 12-48
[12]  Wolfe M 1989 More Iteration Space Tiling IEEE Conference on Supercomputing 655-664
[13]  Wolfe M 1986 Loops skewing: The wave front method revisited International Journal of Parallel Programming 15(4) 279-293
[14]  Samarskii A A, Vabischevich P N 2003 Computational heat transfer (Moscow: “Fizmatlit” Publisher) p 784 (in Russian)
[15]  Jacimovic B M, Genic S and Lelea D 2018 Calculation of the Heat Transfer Coefficient for Laminar Flow in Pipes in Practical Engineering Applications Heat Transfer Engineering 39(20) 1790-1796 DOI: 10.1080/01457632.2017.1388949
[16]  Chhabra A, Talukdar P 2019 Investigation of heat transfer in one-dimensional models of polymeric foams by using a ray-splitting and tracing technique International Journal of Numerical Methods for Heat and Fluid Flow 29(1) 146-164 DOI: 10.1108/HFF-07-2017-0276
[17]  Kalitkin N N 1987 Numerical methods (Moscow: “Nauka” Publisher) p 512 (in Russian)

Acknowledgments
This work was supported by the Russian Foundation for Basic Research grant 19-07-00423-A and by the Ministry of Science and Higher Education within the State assignment to the FSRC “Crystallography and Photonics” RAS No. 007-GZ/Ch3363/26.